TRAMP Reference Manual

Generated by Doxygen 1.2.13.1

Thu Feb 7 11:58:09 2002

Contents

Chapter 1

TRAMP Package List

1.1 TRAMP Package List

Here are the packages with brief descriptions (if available):

Chapter 2

TRAMP Namespace Index

2.1 TRAMP Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

TRAMP Hierarchical Index

3.1 TRAMP Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

TRAMP Compound Index

4.1 TRAMP Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

TRAMP File Index

5.1 TRAMP File List

Here is a list of all documented files with brief descriptions:

Chapter 6

TRAMP Page Index

6.1 TRAMP Related Pages

Here is a list of all related documentation pages:

Chapter 7

TRAMP Package Documentation

7.1 Package de.arvika.workflow

Interfaces

- interface **Command**

  This interface is part of a CommandPattern that is used in the WorkflowEngine context to represent actions to be performed on a workflow.

- interface **DocumentListener**

  This interface defines the method a class has to implement so that it may be registered as a listener for DocumentEvents.

- interface **MultipleDisplayDocument**

  A **Document** (p. 109) *that implements this interface allows a multiple display user interface to decide if the document is suited for the primary (usually head mounted) display.*

- interface **Renderer**
- interface **TaskListener**
- interface **WorkflowApplication**
- interface **WorkflowCommand**
- interface **WorkflowListener**

Classes

- class **Comment**

  This class stub represents comments that users may add to a Task or Workflow.

- class **Dialog**

  Some workflows require that complex dialogs like checklists are presented to the user - these should be represented by this class stub.

- class **Document**

  This class should include the information necessary to describe a document.

- class **DocumentEvent**

This class is used to provide a DocumentEventListener *with information about avilable or requested documents.*

- class **DocumentListenerAdapter**

  This class provides empty bodies for all the methods of the **DocumentListener** (p. 113) *interface.*

- class **FsaWorkflowWebRenderer**
- class **Information**

  The WorkflowEngine's main purpose is to deliver information to the client and thus we encapsulate this need in this information.

- class **InformationList**

  Utility class for easier access to an InformationList.

- class **LabeledIdTable**

  This helper class associates ids with labels that may be used to represent the entity behind the id.

- class **LocalizedLabels**

  This class enables us to have a single object for each element in the workflow.

- class **SimpleWorkflowWebRenderer**
- class **Task**
- class **TaskEvent**

  This Event delivers a task to the TaskListener.

- class **UnknownCommandException**
- class **Workflow**
- class **WorkflowController**
- class **WorkflowEngine**
- class **WorkflowEngineEvent**
- class **WorkflowEngineLogger**
- class **WorkflowEvent**
- class **WorkflowListenerAdapter**
- class **WorkflowServlet**
- class **Workspace**
- class **WorkspaceManager**

7.1.1 Detailed Description

Author:
Stefan Ri"s

Version:

7.2 Package de.arvika.workflow.fsaWorkflow

This is the central class of the de.arvika.workflow.fsaWorkflow package.

7.2.1 Detailed Description

This is the central class of the de.arvika.workflow.fsaWorkflow package.

It implements the workflow's DTD as specified in my thesis and provides mechanisms for event handling and variable handling. As we use an event mechanism similar to the Java AWT's event model this class implements a Runnnable's **run**() (p. 325) method that takes Commands (incoming events) from a queue and executes them. The addCommand method simply adds Commands to the end of the queue. Other methods deal with the information handling and variable mechansim of the workflow.

7.3 Package DWARF

This file is used to catch the command to run a service through a parser.

Classes

- class **DemoStatusExporter**

  This is a Demo Service that can be used as a template for **DWARF** (p. 59) *services that want to provide object references to other service as a* **DWARF** (p. 59) *ability.*

- class **RunService**
- class **xmlparser**

7.3.1 Detailed Description

This file is used to catch the command to run a service through a parser.

The command will be saved into "runservice". "runservice" will be changed into executable.

7.4 Package DWARF.util

Debug.java.

Classes

- class **CorbaInit**

  singleton design pattern.

- class **CorbaInit::OrbRunner**
- class **CorbaNotInitializedException**
- class **CorbaService**

  Implementation of a **DWARF** (p. 59) *Service using the Corba Notification protocol It lets you register Needs and Abilities, if you have described them before.*

- class **DataReceiver**

  Implementation of a CORBA Need: PositionEvent receiving class.

- class **DataSender**

  Implementation of a CORBA Ability: Event sending class.

- class **DataUser**

  Interface PositionEventUser.

- class **Debug**
- class **ObjectExporter**

  The Objects exporter is a **DWARF** (p. 59) *wrapper for the ability to export references to (any) CORBA objects to interested clients.*

- class **ObjectImporter**

  An Object Importers allows an application to gather other references exported by other services, and to use them.

- class **PositionFactory**

  Only a helper class to worry with the Java Date&Time instead o using PositinEvent directly.

- class **StructuredEventFactory**

  Little helper class for getting a prepared, ready to rumble structured event.

- class **UTM**

7.4.1 Detailed Description

Debug.java.

Author: Martin Bauer Description: <describe the Debug class here>

7.5 Package org.globalse.react.applet

Classes

- class **AdminClient**

  The AdminClient Class is the Applet (derived from **ViewerClient** (p. 270)*), which is executed on the admin's computer.*

- class **AdminNullClient**

  The AdminNullClient Class is similar with **AdminClient** (p. 67)*, the only difference is, that a null Layout is used for the chat.*

- class **NullClient**

  The NullClient Class is similar with **ViewerClient** (p. 270)*, the only difference is, that a null Layout is used for the chat.*

- class **QuicktimeClient**

  The QuicktimeClient Class is the Applet, which shows the Quicktime Video Stream in a browser ∗ frame.

- class **ViewerClient**

  The ViewerClient Class is the Applet (derived from **PresentationReceiver** (p. 219)*), which is executed on the viewer's computer.*

7.6 Package org.globalse.react.client

Classes

- class **ClientThread**

  the Client that reads data from the server and reconnects in case of.

- class **PresentationClient**

  a simple exmaple for a Presentation Client that uses the Client and Server classes.

- class **PresentationReceiver**

  the PresentationReceiver, the superclass for all the different clients.

- class **StreamURLManager**

7.7 Package org.globalse.react.io

Interfaces

- interface **CommandListener**

  this is the interface used by client and server to communicate with each other the actual io-work is done unseenly.

Classes

- class **ProtocolHandler**

  this class is meant to do the whole server Client -Protocol It uses DataStreams, because high performance is not necessary here for each command it reads, it calls the corresponding **CommandListener** (p. 76).

- class **ProtocolHandler::Reader**

  this Thread steadily keeps reading commands from the stream.

- class **XMLLogger**

  this class is responsible for the whole server logging all messages will be printed in XML format.

7.8 Package org.globalse.react.server

Classes

- class **PresentationManager**

  this is the main class of the GSE Presentaion Managaer module It is conncted to all remote clients, informs them about changes status and receives the intrrupting questions from them.

- class **ServerThread**

  This class is the interface of the **PresentationManager** (p. 217) *(server) to the client.*

- class **ServerThread::PingThread**

  the thread that steadily pings all the connected Clients to keep the connection.

7.9 Package org.globalse.tramp.services.Airguide

Classes

- class **Airguide**

7.10 Package org.globalse.tramp.services.ExpertSystem

Interfaces

- interface **DOMParserWrapper**

Classes

- class **DOMParser**

  Wraps the Xerces DOM parser and extends NonValidatingDOMParser.

- class **ExpertSystem**

  implements the ExpertSystem.

7.11 Package org.globalse.tramp.services.FLASHview

Classes

- class **FlashClient**

  CSClient
  Client for the FlashServer.

- class **FlashServer**

7.12 Package org.globalse.tramp.services.GarageServer-Application

Classes

- class **GarageServerApplication**

  implements the GarageServer.

- class **GarageServerApplication::UserActionUser**

  creates ActionUser for sending structured Events.

7.13 Package org.globalse.tramp.services.IPAQView

Classes

- class **IPAQView**
- class **MCalibration**
- class **MCalibration::LoadThread**
- class **MInput**
- class **MInputRS**
- class **MPayment**

7.14 Package org.globalse.tramp.services.IPAQViewCR

Classes

- class **CRForm**
- class **IPAQViewCR**

7.15 Package org.globalse.tramp.services.MessageAlerter

Classes

- class **MessageAlerter**

  implements the MessageAlerter Service.

- class **MessageAlerter::MessageAlertUser**

  creates MessageAlertUser for receiving structured Events.

- class **MessageWindow**

  implements the Message Windows for the **MessageAlerter** (p. 168) *Service.*

7.16 Package org.globalse.tramp.services.PersistentDataStorage

Classes

- class **PersistentDataStorage**

  public class PersistentDataStorage extends PersistentDataStoragePOA {.

7.17 Package org.globalse.tramp.services.TaskflowEngine

Classes

- class **TaskflowEngine**

  public class TaskflowEngine extends TaskflowEnginePOA.

7.18 Package org.globalse.tramp.services.Taskflow-Engine.fsaWorkflow

This is the central class of the de.arvika.workflow.fsaWorkflow package.

Interfaces

- interface **CustomListener**
- interface **FsaWorkflowElement**
- interface **LogListener**
- interface **PropertyControlled**
- interface **VariableListener**
- interface **WorkflowEventHandler**

Classes

- class **AndCondition**
- class **BlockCondition**
- class **CommandQueue**
- class **Condition**
- class **CustomEvent**
- class **Equal**
- class **EqualVars**
- class **FireCustomCommand**
- class **FireTransitionCommand**
- class **FsaDocument**
- class **FsaTask**
- class **FsaWorkflow**
- class **FsaWorkflowController**
- class **FsaWorkflowEngine**
- class **FsaWorkflowParser**
- class **FsaWorkflowParser::ElementDetails**

  Helper class that allow the parser to maintain a context Refer to "Professional XML" by Didier Martin et al.

- class **FsaWorkflowParser::Transition**

  HelperClass to collect transition information for later evaluation See the endDocument() method for details.

- class **InEvent**
- class **LogEvent**
- class **LoggingCommand**
- class **MyDocumentListener**

  This interface defines the method a class has to implement so that it may be registered as a listener for DocumentEvents.

- class **MyTaskListener**
- class **MyTaskPossibilities**
- class **OrCondition**

- class **PublishEventCommand**
- class **RequestDocumentCommand**
- class **RequestSubWorkflowCommand**
- class **SetAttributeCommand**
- class **SetVariableCommand**
- class **StartWorkflowCommand**
- class **VariableEvent**
- class **WFEEventMediator**

7.18.1 Detailed Description

This is the central class of the de.arvika.workflow.fsaWorkflow package.

It implements the workflow's DTD as specified in my thesis and provides mechanisms for event handling and variable handling. As we use an event mechanism similar to the Java AWT's event model this class implements a Runnnable's **run**() (p. 325) method that takes Commands (incoming events) from a queue and executes them. The addCommand method simply adds Commands to the end of the queue. Other methods deal with the information handling and variable mechansim of the workflow.

7.19 Package org.globalse.tramp.services.TestingGUI

Classes

- class **DemoService**

  This class implements a trivial (and rather stupid) DemoService for **DWARF** (p. 59) *that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).*

- class **DemoService::PrintPosUser**

  A sample implementation of the interface **DataUser** (p. 94)*, which is used for the callback calls wenn the receiver gets an event.*

- class **DemoStatusImporter**

  This is a Demo Service which queries other **DWARF** (p. 59) *Services for their status.*

- class **GarageUserThread**

  This class implements a trivial (and rather stupid) **DemoService** (p. 96) *for* **DWARF** (p. 59) *that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).*

- class **GarageUserThread::PrintPosUser**

  A sample implementation of the interface **DataUser** (p. 94)*, which is used for the callback calls wenn the receiver gets an event.*

- class **GestureThread**
- class **GestureThread::PrintGestureData**
- class **MarkerThread**
- class **MarkerThread::PrintMarkerThread**
- class **PosThread**
- class **PosThread::PrintPosUser**

  — this prints PositionEvent(of the **DemoService** (p. 96)*-sender), not PoseData — public class* **PrintPosUser** (p. 214) *implements* **DataUser** (p. 94) { *public void* **event***(StructuredEvent se)* (p. 214) { *// Extract our event PositionEvent pose = PositionEventHelper.extract(se.remainder_of_body);.*

- class **TestingGUI**
- class **TestSender**

  This class implements a trivial (and rather stupid) TestSender, it expands the **DemoService** (p. 96) *(version 1.8) for* **DWARF** (p. 59) *that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).*

- class **UserActionSender**
- class **UserActionSender::PrintUserInput**
- class **UserActionThread**
- class **UserActionThread::PrintUserAction**
- class **UserInputSender**
- class **UserInputSender::GestureDataUser**

  creates GestureDataUser for receiving the gesture data as structured Events.

- class **UserInputSender::MarkerInfoUser**

  creates MarkerInfoUser for receiving the marker data as structured Events.

- class **UserInputSender::UserActionUser**

  creates ActionUser for receiving the user action as structured Events.

- class **UserInputThread**
- class **UserInputThread::PrintUserInput**
- class **WearableUserThread**
- class **WearableUserThread::PrintPosUser**

7.19.1 Detailed Description

Author:
TRAMP Testing Team

Version:
1.0

7.20 Package org.globalse.tramp.services.UIController

Interfaces

- interface **UserInterfaceView**

Classes

- class **UIController**
- class **UIController::PrintUserInput**
- class **ViewAdaptor**

7.21 Package org.globalse.tramp.services.UIEngine

Classes

- class **UIEngine**

7.22 Package org.globalse.tramp.services.VRMLView

Interfaces

- interface **VRMLInterface**

Classes

- class **VRMLView**
- class **VRMLViewAdaptor**
- class **VRMLViewAdaptor::VRMLUser**

  the Default location where we expect the service manager to listen.

7.23 Package org.globalse.tramp.services.WearableApplication

Classes

- class **WearableApplication**

  main class of this file(what else?).

- class **WearableApplication::GestureDataUser**

  creates GestureDataUser for receiving the gesture data as structured Events.

- class **WearableApplication::MarkerInfoUser**

  creates MarkerInfoUser for receiving the marker data as structured Events.

- class **WearableApplication::UserActionUser**

  creates ActionUser for receiving the user action as structured Events.

7.24 Package org.globalse.tramp.util

Classes

- class **MathUtil**

  implements common Math procedures.

Chapter 8

TRAMP Namespace Documentation

8.1 DWARF Namespace Reference

dwarfutil.h.

Compounds

- interface **DWARF::Ability**
- interface **DWARF::AbilityConnector**
- interface **DWARF::AbilityDescription**
- interface **DWARF::AbilityIsNotifyConsumer**
- interface **DWARF::AbilityIsNotifySupplier**
- interface **DWARF::ActiveAbilityDescription**
- interface **DWARF::ActiveNeedDescription**
- interface **DWARF::ActiveServiceDescription**
- interface **DWARF::Airguide**
- interface **DWARF::ApplicationMessage**

  interface Applicationmessage is implemented by GarageServerApp and WeareableApp it's used to send messages between both via RPC.

- interface **DWARF::Camera**

  Interface for camera operations.

- interface **DWARF::CameraService**

  This interface creates a **Camera** (p. 71) **DWARF** (p. 59) *service.*

- interface **DWARF::Communication**
- interface **DWARF::ConnectCallback**
- interface **DWARF::Connector**
- interface **DWARF::ConnectorDescription**
- interface **DWARF::ConnectorFactory**
- interface **DWARF::constants**
- interface **DWARF::CorbaObjExporterConnector**

- interface **DWARF::CorbaObjImporterConnector**
- interface **DWARF::DescribeServices**
- interface **DWARF::DwarfStatus**
- interface **DWARF::ExpertSystem**
- struct **DWARF::ExtendedPosition**
- struct **DWARF::GestureData**

  contains information about the type of time of a recognized gesture.

- struct **DWARF::idval**
- struct **DWARF::ImageFormat**

  This struct describes image format properties.

- interface **DWARF::LocatorOffer**
- interface **DWARF::LocatorRequest**
- struct **DWARF::Lost**
- struct **DWARF::LostEvent**
- interface **DWARF::ManualCalibration**

  This interface discribes the marker detection service.

- interface **DWARF::ManualCalibrationService**

  This interface creates a **ManualCalibration** (p. 155) **DWARF** (p. 59) *service.*

- struct **DWARF::Marker**
- struct **DWARF::MarkerEvent**
- struct **DWARF::MessageAlert**
- struct **DWARF::MessageEvent**
- interface **DWARF::MessageSenderService**

  This interface creates a MessageSender **DWARF** (p. 59) *service.*

- interface **DWARF::Need**
- interface **DWARF::NeedDescription**
- interface **DWARF::NeedIsNotifyConsumer**
- interface **DWARF::NeedIsNotifySupplier**
- exception **DWARF::NoMarkerFound**
- exception **DWARF::NoPathFound**
- interface **DWARF::NotifyStructuredPushConsumerConnector**
- interface **DWARF::NotifyStructuredPushSupplierConnector**
- interface **DWARF::PersistentDataStorage**
- interface **DWARF::PersistentDataStorageOps**
- struct **DWARF::PoseData**
- struct **DWARF::Position**
- struct **DWARF::PositionEvent**
- struct **DWARF::Property**
- exception **DWARF::PropertyNotFound**
- interface **DWARF::ProtocolConnector**
- interface **DWARF::RegisterConnectorFactories**
- interface **DWARF::RegisterServices**
- interface **DWARF::RoutePlaner**
- interface **DWARF::RoutePlanerService**

  This interface creates a DetectMarker **DWARF** (p. 59) *service.*

- interface **DWARF::Service**
- interface **DWARF::ServiceAndNeedAndAbility**
- interface **DWARF::ServiceAndNeedAndAbilityIsNotifyConsumer**
- interface **DWARF::ServiceAndNeedAndAbilityIsNotifySupplier**
- interface **DWARF::ServiceAndNeedAndAbilityIsNotifySupplierConsumer**
- interface **DWARF::ServiceDescription**
- interface **DWARF::ServiceLocator**
- interface **DWARF::ServiceManager**
- interface **DWARF::Session**
- interface **DWARF::ShmReaderConnector**
- interface **DWARF::ShmWriterConnector**
- interface **DWARF::StartStop**
- interface **DWARF::StartStopService**
- interface **DWARF::SvcConnCorbaObjExporter**
- interface **DWARF::SvcConnCorbaObjImporter**
- interface **DWARF::SvcConnNotifyStructuredPushConsumer**
- interface **DWARF::SvcConnNotifyStructuredPushSupplier**
- interface **DWARF::SvcConnShmReader**
- interface **DWARF::SvcConnShmWriter**

  these interfaces are for Connectors that exchange shared memory block IDs.

- struct **DWARF::SysMessage**
- interface **DWARF::TaskflowAction**

  interface **TaskflowAction** (p. 246) *is implemented by GarageServerApp and WeareableApp it's used to receive a list of possible TaskflowActions from the* **TaskflowEngine** (p. 247) *via RPC.*

- interface **DWARF::TaskflowEngine**
- interface **DWARF::Thing**
- struct **DWARF::ThingChangedEvent**
- class **DWARF::ThingImpl**

  This class implements a Thing from the WorldModel.

- exception **DWARF::ThingNotFound**
- struct **DWARF::Time**
- interface **DWARF::TrackerData**
- class **DWARF::TrackingFilterSAXHandler**

  XML parser for construction the **PoseDataModelMapping** (p. 207).

- interface **DWARF::TriggerTaskflowTransition**
- interface **DWARF::UIControl**

  This is the interface for the UIController.

- interface **DWARF::UIController**
- interface **DWARF::UIEConfigurationData**

  This is the interface for the **UIEConfigurationData** (p. 265).

- interface **DWARF::UIEngine**
- interface **DWARF::UIView**
- struct **DWARF::UserAction**

UserAction (p. 266) *events are sent to the application depending on the user action.*

- struct **DWARF::UserInput**
- interface **DWARF::View**

  This is the interface for the HTMLView, IPAQView, FlashView.

- interface **DWARF::Wearable**
- interface **DWARF::WorldModel**
- class **DWARF::WorldModelImpl**

  This class the WorldModel.

- class **DWARF::WorldModelSAXHandler**
- interface **DWARF::WorldModelService**
- struct **DWARF::xmlStackElement**

Typedefs

- typedef set< ThingID > **idSet**
- typedef set< **ThingImpl** *> **thingSet**
- typedef set< string > **stringSet**
- typedef string **SystemMessage**
- typedef sequence< octet > **PersistentDataStorage_data**
- typedef double **PositionVector** [3]
- typedef double **OrientationVector** [4]
- typedef double **HomogenousCoords** [16]
- typedef unsigned long **ThingID**
- typedef unsigned long **RelativeToThingID**
- typedef sequence< Property > **PropertySeq**
- typedef string **Path**
- typedef sequence< string > **StringSequence**
- typedef sequence< ThingID > **ThingIDSequence**
- typedef sequence< double > **DoubleSequence**

Enumerations

- enum **GestureType** { **yes**, **no** }

  indicates the type of a recognized gesture.

- enum **MessageType** { **None**, **MsgWarning**, **MsgError**, **MsgInfo**, **MsgQuestion** }
- enum **ThingChangeType** { **ThingAdded**, **ThingDestroyed**, **ThingMoved**, **PropertyAdded**, **PropertyDeleted**, **PropertyChanged** }

8.1.1 Detailed Description

dwarfutil.h.

Author:

Daniel Pustka, Martin Wagner Distributed Wearable Augmented Reality Framework - www.augmentedreality.de

Utility classes to simplify working with the DWARF framework under C++

Id:
dwarfutil.h,v 1.2 2002/01/29 12:15:16 wagnerm Exp

Revision:
1.2

8.1.2 Enumeration Type Documentation

8.1.2.1 enum DWARF::GestureType

indicates the type of a recognized gesture.

Enumeration values:

yes user nodded (meaning yes).

no user shook his head (meaning no).

Chapter 9

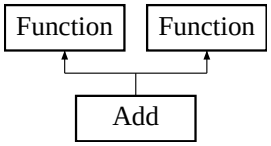
TRAMP Class Documentation

9.1 Add Class Reference

Addition function composition.

`#include <function.h>`

Inheritance diagram for Add::



Public Methods

- **Add** (**Function** *f1, **Function** *f2)
- virtual ~**Add** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** * **getSummand1** () const
- virtual **Function** * **getSummand2** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const
- **Add** (**Function** *f1, **Function** *f2)
- virtual ~**Add** ()

- virtual **Function** ∗ **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** ∗ **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** ∗ **getSummand1** () const
- virtual **Function** ∗ **getSummand2** () const
- virtual **Function** ∗ **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const

9.1.1 Detailed Description

Addition function composition.

Author:
Andreas Krause

See also:
Fct (p. 123)

9.1.2 Member Function Documentation

9.1.2.1 virtual Function∗ Add::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

9.1.2.2 Function ∗ Add::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

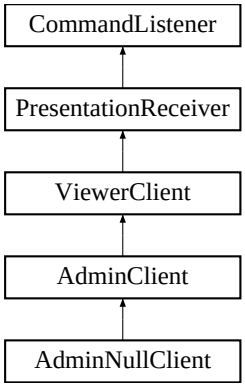
The documentation for this class was generated from the following files:

- **MovementSimulator/function.h**
- **TrackingFilter/function.h**
- MovementSimulator/function.cpp
- TrackingFilter/function.cpp

9.2 AdminClient Class Reference

The AdminClient Class is the Applet (derived from **ViewerClient** (p. 270)), which is executed on the admin's computer.

Inheritance diagram for AdminClient::



Public Methods

- void **init** ()

  init, the same as in **ViewerClient** (p. 270)*, but for changing slides you can also use up and down keys.*

Protected Methods

- void **createPresentationFrame** ()

  Creates a window, in which the UI for the Admin is displayed.

- void **switchSlides** (int direction)

  Loads the next (or previous) slide and sends a command to the server, that it should be displayed to all normal viewers, too.

- void **newQuestion** (String text)

  Displays new question from viewer.

Protected Attributes

- int **slideNr**
- Button **bnSlide**
- Button **bpSlide**
- Button **bpStop**
- Button **bqSave**
- Button **bnQuestion**
- String **aktQuestion**

9.2.1 Detailed Description

The AdminClient Class is the Applet (derived from **ViewerClient** (p. 270)), which is executed on the admin's computer.

It offers some additional functionality to the admin, which is not avilable to the normal user. It allows him to switch to the next slide and to receive questions to the presenter from the viewers.

Author:
Wolfgang Sprung

9.2.2 Member Function Documentation

9.2.2.1 void AdminClient::newQuestion (String *text*) [inline, protected]

Displays new question from viewer.

Parameters:
text the question as a String

Reimplemented from **PresentationReceiver** (p. 220).

9.2.2.2 void AdminClient::switchSlides (int *direction*) [inline, protected]

Loads the next (or previous) slide and sends a command to the server, that it should be displayed to all normal viewers, too.

Parameters:
direction 1 for switching forward, 0 for backwards (as Integers)

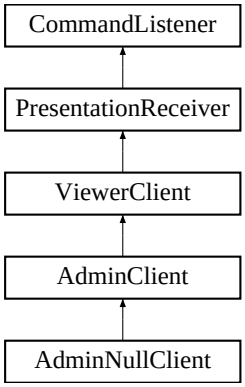
The documentation for this class was generated from the following file:

- AdminClient.java

9.3 AdminNullClient Class Reference

The AdminNullClient Class is similar with **AdminClient** (p. 67), the only difference is, that a null Layout is used for the chat.

Inheritance diagram for AdminNullClient::



Protected Methods

- void **createPresentationFrame** ()

  Creates a window, in which the UI for the Admin is displayed.

9.3.1 Detailed Description

The AdminNullClient Class is similar with **AdminClient** (p. 67), the only difference is, that a null Layout is used for the chat.

The documentation for this class was generated from the following file:

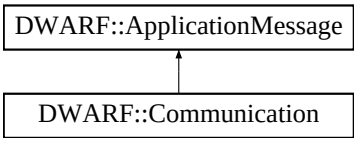
- AdminNullClient.java

9.4 DWARF::ApplicationMessage Interface Reference

interface Applicationmessage is implemented by GarageServerApp and WeareableApp it's used to send messages between both via RPC.

```
import "Application.idl";
```

Inheritance diagram for DWARF::ApplicationMessage::



Public Methods

- void **transferMessage** (in long id, in string mes)

9.4.1 Detailed Description

interface Applicationmessage is implemented by GarageServerApp and WeareableApp it's used to send messages between both via RPC.

The documentation for this interface was generated from the following file:

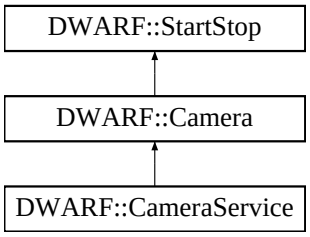
- **Application.idl**

9.5 DWARF::Camera Interface Reference

Interface for camera operations.

`import "Camera.idl";`

Inheritance diagram for DWARF::Camera::



Public Methods

- void **getImageFormat** (out **ImageFormat** format)
  Returns image format properties.
- long **getSHMKey** ()
  retrieve the shared memory key for use with the ringbuffer classes.

9.5.1 Detailed Description

Interface for camera operations.

9.5.2 Member Function Documentation

9.5.2.1 void DWARF::Camera::getImageFormat (out ImageFormat *format*)

Returns image format properties.

Parameters:
format The format description

The documentation for this interface was generated from the following file:

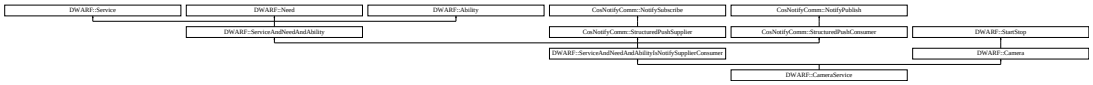
- **Camera.idl**

9.6 DWARF::CameraService Interface Reference

This interface creates a **Camera** (p. 71) **DWARF** (p. 59) service.

`import "Camera.idl";`

Inheritance diagram for DWARF::CameraService::



9.6.1 Detailed Description

This interface creates a **Camera** (p. 71) **DWARF** (p. 59) service.

To call it, please use the **Camera** (p. 71) interface

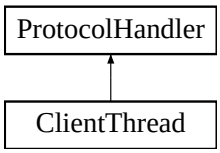
The documentation for this interface was generated from the following file:

- **Camera.idl**

9.7 ClientThread Class Reference

the Client that reads data from the server and reconnects in case of.

Inheritance diagram for ClientThread::



Public Methods

- **ClientThread** (String host, int port, **CommandListener** listener)
  we need host and port and the **CommandListener** (p. 76) *to call.*
- void **kill** ()
  set the PH as killed and close all streams.

Protected Methods

- void **exception** ()
  an IOException has occcured.

9.7.1 Detailed Description

the Client that reads data from the server and reconnects in case of.

error

9.7.2 Member Function Documentation

9.7.2.1 void ClientThread::exception () [inline, protected, virtual]

an IOException has occcured.

The derviving class now has to decide what to do.

Reimplemented from **ProtocolHandler** (p. 223).

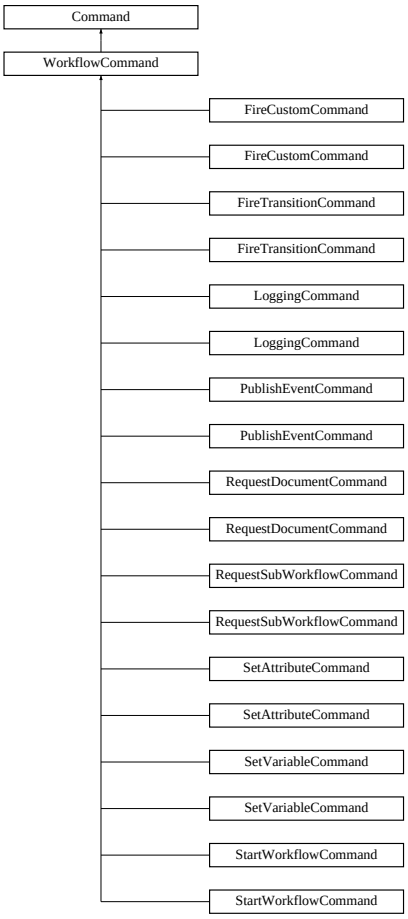
The documentation for this class was generated from the following file:

- ClientThread.java

9.8 Command Interface Reference

This interface is part of a CommandPattern that is used in the WorkflowEngine context to represent actions to be performed on a workflow.

Inheritance diagram for Command::



Public Methods

- void **execute** ()

  Every implementation of the Command interface has to provide this method that executes an operation.

9.8.1 Detailed Description

This interface is part of a CommandPattern that is used in the WorkflowEngine context to represent actions to be performed on a workflow.

Commands are either generated by the subelements of the *Actions* element to describe a reaction to some event or a controlWorkflow call to the WorkflowEngine. To allow a command to be instantiated with only the default constructor a couple of additional interfaces provide methods to

set a command's parameters. Refer to these interface definitions or the Command implementations for details.

See also:
WorkflowEngine::controlWorkflow , de.arvika.workflow.fsaWorkflow.InEvent , de.arvika.workflow.fsaWorkflow.PropertyControlled , de.arvika.workflow.WorkflowCommand

Author:
Stefan Ri"s

Version:

Revision:
1.1

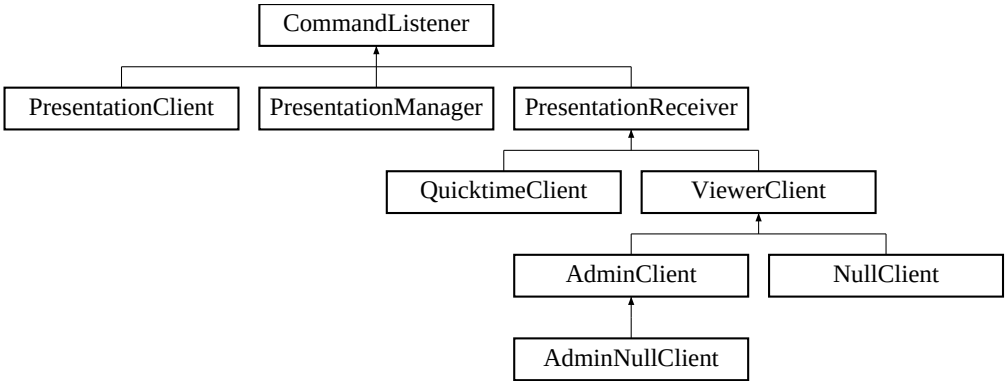
The documentation for this interface was generated from the following file:

- Command.java

9.9 CommandListener Interface Reference

this is the interface used by client and server to communicate with each other the actual io-work is done unseenly.

Inheritance diagram for CommandListener::



Public Methods

- String **commandReceived** (int id, String command)

  server or client receives a command.

Static Public Attributes

- final int **PING** = 0

  this is a simple ping signal it should always be answerd by a commit.

- final int **STREAM_SOURCE** = 1

  transimission of a cource url for a video stream.

- final int **QUESTION** = 2

  a question relevant to the presentation.

- final int **COMMIT** = 3

  a commit signal is just ignored by receipe.

- final int **SLIDE** = 4

  a slide has changed in the presentaion.

- final int **GENERAL_MESSAGE** = 7

  this is the message type used for general chatting.

- final int **PRIVATE_MESSAGE** = 8

  this is a private chat message (for only one receipe).

- final int **ADMIN_NEXTQUESTION** = 5

the admin now takes the next question.

- final int **ADMIN_SAVEQUESTION** = 9

  the admin has decided to place the current question at the end of the presentation.

- final int **ADMIN_CHANGESLIDE** = 6

  the admin changes the slide.

- final int **ADMIN_PRESENTATIONSTOP** = 10

  this admin command signals, that the Presenter it is at the end of his presentation.

- final String **NEXT_SLIDE** = "next"

  String literals for changing slides.

- final String **PREV_SLIDE** = "prev"

  String literals for changing slides.

- final String **MESSAGE_PREFIX** = "message:"

  this String at the beginning of a command signals that it should be sent as general message.

- final String **EMPTY** = "ok"

  the empty response that is not "posted" back.

9.9.1 Detailed Description

this is the interface used by client and server to communicate with each other the actual io-work is done unseenly.

9.9.2 Member Function Documentation

9.9.2.1 String CommandListener::commandReceived (int *id*, String *command*)

server or client receives a command.

Parameters:

id the type of command

command the command as a String

Returns:

the return String that is sent as a response

Reimplemented in **PresentationClient** (p. 215), **PresentationReceiver** (p. 220), and **PresentationManager** (p. 217).

9.9.3 Member Data Documentation

9.9.3.1 final int CommandListener::ADMIN_PRESENTATIONSTOP = 10 [static]

this admin command signals, that the Presenter it is at the end of his presentation.

When send once you cant make it unhappened

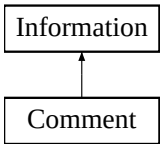
The documentation for this interface was generated from the following file:

- CommandListener.java

9.10 Comment Class Reference

This class stub represents comments that users may add to a Task or Workflow.

Inheritance diagram for Comment::



9.10.1 Detailed Description

This class stub represents comments that users may add to a Task or Workflow.

A final implementation has to work with the persistance layer to store the comment which may be a text, audio or image information. It should also store the user's name and the date and time the comment was created.

Author:
Stefan Ri&szlig, Thomas Reicher;

Version:

Revision:
1.1

The documentation for this class was generated from the following file:

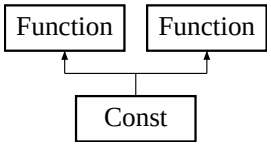
- Comment.java

9.11 Const Class Reference

wraps a constant provided by the constructor.

`#include <function.h>`

Inheritance diagram for Const::



Public Methods

- **Const** (double val)
- virtual ~**Const** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual double **getValue** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const
- **Const** (double val)
- virtual ~**Const** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual double **getValue** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const

9.11.1 Detailed Description

wraps a constant provided by the constructor.

Author:
Andreas Krause

See also:
Fct (p. 123)

9.11.2 Member Function Documentation

9.11.2.1 virtual Function* Const::diff (int *varNo*) const [inline, virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

9.11.2.2 virtual Function* Const::diff (int *varNo*) const [inline, virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

The documentation for this class was generated from the following files:

- **MovementSimulator/function.h**
- **TrackingFilter/function.h**

9.12 CorbaInit Class Reference

singleton design pattern.

Static Public Methods

- void **init** ()
- void **init** (String corbaloc)
- DescribeServices **getDescribeServices** () throws CorbaNotInitializedException

  must be initialized before; throw exception otherwise.

- RegisterServices **getRegisterServices** () throws CorbaNotInitializedException

  must be initialized before; throw exception otherwise.

- ORB **getOrb** () throws CorbaNotInitializedException

  must be initialized before; throw exception otherwise.

9.12.1 Detailed Description

singleton design pattern.

The documentation for this class was generated from the following file:

- CorbaInit.java

9.13 CorbaService Class Reference

Implementation of a **DWARF** (p. 59) Service using the Corba Notification protocol It lets you register Needs and Abilities, if you have described them before.

`#include <corbaservice.h>`

Public Methods

- **CorbaService** (string corbaloc, string name)
- virtual ~**CorbaService** ()
- virtual void **subscription_change** (const EventTypeSeq &added, const EventTypeSeq &removed)
- virtual void **disconnect_structured_push_supplier** ()
- virtual void **connectAbility** (const char *offername, Connector_ptr conn)
- virtual void **disconnectAbility** (const char *offername, Connector_ptr conn)
- virtual void **connectNeed** (const char *reqname, CORBA::Short reqinstance, Connector_ptr conn)
- virtual void **disconnectNeed** (const char *reqname, CORBA::Short reqinstance, Connector_ptr conn)
- virtual void **addAbility** (string name, string type, string attribute, string conn, string connType)
- virtual void **registerAbility** (const char *abilityName, Ability_var ab)
- virtual void **addNeed** (string name, string type, string predicate, string conn, string connType)
- virtual void **registerNeed** (const char *needName, Need_var ab)
- virtual void **activateServiceDescription** ()
- virtual void **registerService** ()
- virtual void **unregisterService** ()

  unregister this service.

- virtual void **startService** ()
- virtual void **stopService** ()
- **CorbaService** (String corbaloc, String s)

  Create new CorbaService.

- ServiceDescription **getServiceDescription** ()

  getServiceDescription for describing new Abilities and Needs
  Exceptions:
  RuntimeException *"Illegal attempt to describe activated service." when* **registerService***()* (p. 84) *has been called before.*

- AbilityDescription **addAbility** (String name, String type, String conn, String connType)
- AbilityDescription **addAbility** (String name, String type, String conn, String connType, String attr)

  addAbility with setAttributes().

- NeedDescription **addNeed** (String name, String type, String conn, String connType)
- NeedDescription **addNeed** (String name, String type, String conn, String connType, String pred)

  addNeed with setPredicate().

- void **activateServiceDescription** ()

  activate the service description.

- void **registerService** ()

  register this service; this activates the service and you can register your Needs and Abilities with the service.

- void **unregisterService** ()

  unregister this service.

- void **registerNeed** (String needName, Need n)

  register the need n with this service.

- void **registerAbility** (String abilityName, Ability ab)

  register the ability ab with this service.

- void **startService** ()

  callback from Service interface.

- void **stopService** ()

  callback from Service interface.

Public Attributes

- string **name**
- ServiceDescription_var **sd**
- ActiveServiceDescription_var **as**
- ORB_var **orb**
- Object_var **obj**
- PortableServer::POA_var **poa**
- DescribeServices_var **describe**
- RegisterServices_var **registrar**
- PortableServer::POAManager_var **pman**
- bool **hasConsumer**
- bool **hasSupplier**
- StructuredProxyPushConsumer_var **notifyConsumer**
- StructuredProxyPushConsumer_var **notifyPosConsumer**
- int **numEvents**

9.13.1 Detailed Description

Implementation of a **DWARF** (p. 59) Service using the Corba Notification protocol It lets you register Needs and Abilities, if you have described them before.

Author:
Oliver Strutynski

9.13.2 Constructor & Destructor Documentation

9.13.2.1 CorbaService::CorbaService (String *corbaloc*, String *s*) [inline]

Create new CorbaService.

Parameters:

corbaloc The location of the ServiceManager;

s The servicename must be unique name!

9.13.3 Member Function Documentation

9.13.3.1 void CorbaService::registerAbility (String *abilityName*, Ability *ab*) [inline]

register the ability ab with this service.

take care only to register described ability.

Parameters:

abilityName The name of the Ability; must be described in the ServiceManager first.

ab The Ability to register

9.13.3.2 void CorbaService::registerNeed (String *needName*, Need *n*) [inline]

register the need n with this service.

take care only to register described needs.

Parameters:

needName The name of the Need; must be described in the ServiceManager first.

n The Need to register

The documentation for this class was generated from the following files:

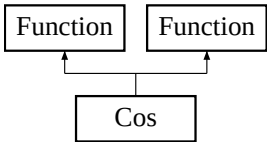
- **corbaservice.h**
- CorbaService.java
- corbaservice.cpp

9.14 Cos Class Reference

Cosine function composition.

`#include <function.h>`

Inheritance diagram for Cos::



Public Methods

- **Cos** (**Function** ∗f)
- virtual ∼**Cos** ()
- virtual **Function** ∗ **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** ∗ **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** ∗ **getOperand** () const
- virtual **Function** ∗ **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const
- **Cos** (**Function** ∗f)
- virtual ∼**Cos** ()
- virtual **Function** ∗ **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** ∗ **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** ∗ **getOperand** () const
- virtual **Function** ∗ **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const

9.14.1 Detailed Description

Cosine function composition.

Author:
Andreas Krause

See also:
Fct (p. 123)

9.14.2 Member Function Documentation

9.14.2.1 virtual Function* Cos::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

9.14.2.2 Function * Cos::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

The documentation for this class was generated from the following files:

- **MovementSimulator/function.h**
- **TrackingFilter/function.h**
- MovementSimulator/function.cpp
- TrackingFilter/function.cpp

9.15 DataReceiver Class Reference

Implementation of a CORBA Need: PositionEvent receiving class.

`#include <datareceiver.h>`

Public Methods

- **DataReceiver** (string name)
- virtual void **setUser** (**DataUser** ∗user)
- void **connectNeed** (const char ∗reqname, CORBA::Short reqinstance, Connector_ptr conn)

  StructuredPushCopnsumer interface.
- void **disconnectNeed** (const char ∗reqname, CORBA::Short reqinstance, Connector_ptr conn)

  StructuredPushConsumer interface.
- virtual void **subscription_change** (const EventTypeSeq &added, const EventTypeSeq &removed)

  StructuredPushSupplier interface ignored so far.
- virtual void **disconnect_structured_push_consumer** ()

  StructuredPushConsumer interface.
- void **offer_change** (const CosNotification::EventTypeSeq &added, const CosNotification::EventTypeSeq &removed)

  StructuredPushConsumer interface.
- void **push_structured_event** (const CosNotification::StructuredEvent &event)

  StructuredPushConsumer interface callback function.
- **DataReceiver** ()

  Create a new empty PositionDataReceiver.
- void **push_structured_event** (StructuredEvent ev)

  callback function from outside.
- void **setUser** (**DataUser** user)

  Specify to which PositionEventUser the PositionEvents should go.
- void **connectNeed** (String reqname, short reqinstance, Connector conn)

  StructuredPushConsumer interface callback fuction for the EventService.
- void **disconnectNeed** (String reqname, short reqinstance, Connector conn)

  StructuredPushConsumer interface callback fuction fr the EventService.
- void **offer_change** (EventType[] added, EventType[] removed)

  StructuredPushSupplier interface ignored so far.
- void **disconnect_structured_push_consumer** ()

  StructuredPushSupplier interface something bad happened, stop pushing or error.

Public Attributes

- bool **hasConsumer**

  guess what...

Protected Attributes

- string **name**
- bool **hasSupplier**
- **DataUser** ∗ **dataUser**

9.15.1 Detailed Description

Implementation of a CORBA Need: PositionEvent receiving class.

Author:
Oliver Strutynski

9.15.2 Member Function Documentation

9.15.2.1 void DataReceiver::connectNeed (String *reqname*, short *reqinstance*, Connector *conn*) [inline]

StructuredPushConsumer interface callback fuction for the EventService.

Parameters:
reqname a String containing the name of the need (note: Need was Requirement before; therfore the name ;-)

reqinstance If you dont want to count how many connections you have now, here is the number.

conn The connector you want to give to me.

9.15.2.2 void DataReceiver::disconnectNeed (String *reqname*, short *reqinstance*, Connector *conn*) [inline]

StructuredPushConsumer interface callback fuction fr the EventService.

Parameters:
reqname a String containing the name of the requirement

conn The connector you want todisconnect.

9.15.2.3 void DataReceiver::setUser (DataUser *user*) [inline]

Specify to which PositionEventUser the PositionEvents should go.

From PositionEventProvider interface.

Parameters:
user The PositionEventUser

9.15.3 Member Data Documentation

9.15.3.1 bool DataReceiver::hasConsumer

guess what...

don't even think about changing this from outside!

The documentation for this class was generated from the following files:

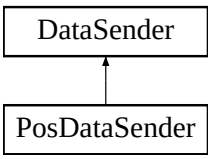
- **datareceiver.h**
- DataReceiver.java
- **datareceiver.cpp**

9.16 DataSender Class Reference

Implementation of a CORBA Ability: Event sending class.

`#include <datasender.h>`

Inheritance diagram for DataSender::



Public Methods

- **DataSender** (string name)
- virtual StructuredProxyPushConsumer_var **getConsumer** ()

  This is simply to get your consumer, when you got a connector.

- virtual void **connectAbility** (const char *offername, Connector_ptr **conn**)

  StructuredPushSupplier interface callback fuction for the EventService.

- virtual void **disconnectAbility** (const char *offername, Connector_ptr **conn**)

  StructuredPushSupplier interface callback fuction fr the EventService.

- virtual void **subscription_change** (const EventTypeSeq &added, const EventTypeSeq &removed)

  StructuredPushSupplier interface ignored so far.

- virtual void **disconnect_structured_push_supplier** ()

  StructuredPushSupplier interface something bad happened, stop pushing or error.

- void **push** (StructuredEvent ev)

  Push a StructuredEvent to your consumer; if the EventService did not provide a consumer so far, all events are simply ignored.

- void **connectAbility** (String offername, Connector **conn**)

  StructuredPushSupplier interface callback fuction for the EventService.

- void **disconnectAbility** (String offername, Connector **conn**)

  StructuredPushSupplier interface callback fuction fr the EventService.

- void **subscription_change** (EventType[] added, EventType[] removed)

  StructuredPushSupplier interface ignored so far.

- void **disconnect_structured_push_supplier** ()

  StructuredPushSupplier interface something bad happened, stop pushing or error.

Public Attributes

- bool **hasConsumer**

  guess what...

Protected Attributes

- StructuredProxyPushConsumer_var **consumer**

  The Consumer of your Events.

- Connector_var **conn**

  The connector to the EventService.

- string **name**

9.16.1 Detailed Description

Implementation of a CORBA Ability: Event sending class.

Author:
Oliver Strutynski

9.16.2 Member Function Documentation

9.16.2.1 void DataSender::connectAbility (String *offername*, Connector *conn*) [inline]

StructuredPushSupplier interface callback fuction for the EventService.

Parameters:

offername a String containing the name of the offer

conn The connector you want to give to me.

9.16.2.2 void DataSender::connectAbility (const char * *offername*, Connector_ptr *conn*) [virtual]

StructuredPushSupplier interface callback fuction for the EventService.

Parameters:

offername a String containing the name of the offer

conn The connector you want to give to me.

9.16.2.3 void DataSender::disconnectAbility (String *offername*, Connector *conn*) [inline]

StructuredPushSupplier interface callback fuction fr the EventService.

Parameters:

offername a String containing the name of the offer

conn The connector you want todisconnect.

9.16.2.4 void DataSender::disconnectAbility (const char * *offername*, Connector_ptr *conn*) [virtual]

StructuredPushSupplier interface callback fuction fr the EventService.

Parameters:

offername a String containing the name of the offer

conn The connector you want todisconnect.

9.16.2.5 void DataSender::push (StructuredEvent *ev*) [inline]

Push a StructuredEvent to your consumer; if the EventService did not provide a consumer so far, all events are simply ignored.

Parameters:

ev The Structured event to be pushed

9.16.3 Member Data Documentation

9.16.3.1 bool DataSender::hasConsumer

guess what...

don't even think about changing this from outside!

The documentation for this class was generated from the following files:

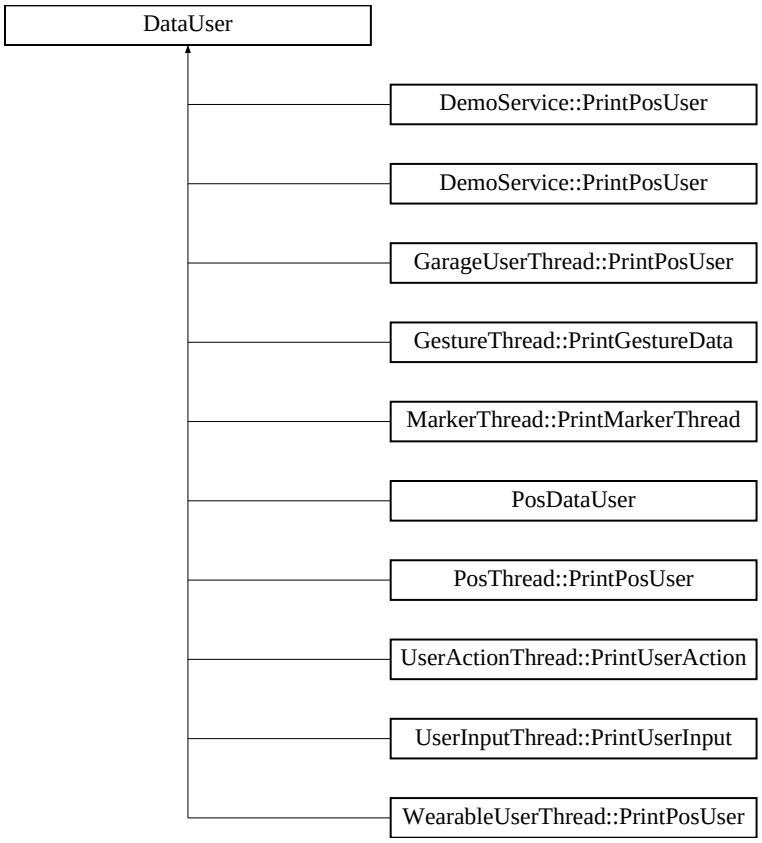
- **datasender.h**
- DataSender.java
- **datasender.cpp**

9.17 DataUser Class Reference

Interface PositionEventUser.

`#include <datauser.h>`

Inheritance diagram for DataUser::



Public Methods

- virtual void **event** (const CosNotification::StructuredEvent &event)=0
- void **event** (StructuredEvent se)

  Callback function an external class calls to push a PositionEvent.

9.17.1 Detailed Description

Interface PositionEventUser.

Author:
Oliver Strutynski

9.17.2 Member Function Documentation

9.17.2.1 void DataUser::event (StructuredEvent *se*)

Callback function an external class calls to push a PositionEvent.

Parameters:
se The PositionEvent

Reimplemented in **DemoService::PrintPosUser** (p. 97), **DemoService::PrintPosUser** (p. 97), **GarageUserThread::PrintPosUser** (p. 139), and **PosThread::PrintPosUser** (p. 214).

The documentation for this class was generated from the following files:

- **datauser.h**
- DataUser.java

9.18 DemoService Class Reference

This class implements a trivial (and rather stupid) DemoService for **DWARF** (p. 59) that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).

Public Methods

- **DemoService** (String corbaloc, boolean doSend, boolean doReceive)
- **DemoService** (String corbaloc)

Static Public Methods

- void **main** (String args[])
- void **main** (String args[])

Static Public Attributes

- final String **DEFLOCATION** = "corbaloc::macbruegge51:39273/ServiceManager"
  the Default location where we expect the service manager to listen.

9.18.1 Detailed Description

This class implements a trivial (and rather stupid) DemoService for **DWARF** (p. 59) that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).

Please use this a a template for your own communication classes.

Author:
The TRAMP Network Team

Version:

Revision:
1.11

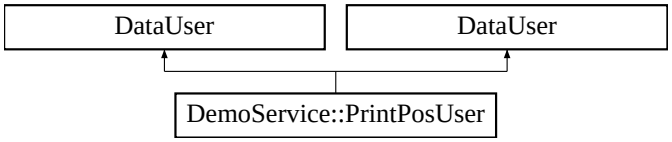
The documentation for this class was generated from the following files:

- dwarf/utiljava/DWARF/DemoService.java
- java/org/globalse/tramp/services/TestingGUI/DemoService.java

9.19 DemoService::PrintPosUser Class Reference

A sample implementation of the interface **DataUser** (p. 94), which is used for the callback calls wenn the receiver gets an event.

Inheritance diagram for DemoService::PrintPosUser::



Public Methods

- void **event** (StructuredEvent se)

  The event is triggered each time an event is received.

- void **event** (StructuredEvent se)

  The event is triggered each time an event is received.

9.19.1 Detailed Description

A sample implementation of the interface **DataUser** (p. 94), which is used for the callback calls wenn the receiver gets an event.

9.19.2 Member Function Documentation

9.19.2.1 void DemoService::PrintPosUser::event (StructuredEvent *se*) [inline]

The event is triggered each time an event is received.

NOTE: The callback implemented in the ORB is multithreaded, so please remember using syncronize() and/or semaphores to handle critical program parts (like modifying object variables) safely!

Parameters:

se the generic Corba StructuredEvent to be used for transmitting the data. Any CorbaObject can be transmitted in the se.remainder_of_body field. What you do with it is up to convention for your specific event ...

Reimplemented from **DataUser** (p. 95).

9.19.2.2 void DemoService::PrintPosUser::event (StructuredEvent *se*) [inline]

The event is triggered each time an event is received.

NOTE: The callback implemented in the ORB is multithreaded, so please remember using syncronize() and/or semaphores to handle critical program parts (like modifying object variables) safely!

Parameters:

se the generic Corba StructuredEvent to be used for transmitting the data. Any CorbaObject can be transmitted in the se.remainder_of_body field. What you do with it is up to convention for your specific event ...

Reimplemented from **DataUser** (p. 95).

The documentation for this class was generated from the following files:

- dwarf/utiljava/DWARF/DemoService.java
- java/org/globalse/tramp/services/TestingGUI/DemoService.java

9.20 DemoStatusExporter Class Reference

This is a Demo Service that can be used as a template for **DWARF** (p. 59) services that want to provide object references to other service as a **DWARF** (p. 59) ability.

Public Methods

- **DemoStatusExporter** (String name, String status)
  Creates new DemoStatusExporter.
- java.lang.String **getStatus** ()
  Status Interface: returns the status of this Service.
- java.lang.String **getName** ()
  Status Interface: returns the name of this service.

Static Public Methods

- void **main** (String args[])
  Dull main method.

Public Attributes

- final String **ID** = "$Id: DemoStatusExporter.java,v 1.6 2002/01/09 13:18:41 richter Exp $"
  Our ID in the CVS:
  Id:
  DemoStatusExporter.java,v 1.6 2002/01/09 13:18:41 richter Exp
  .

Static Public Attributes

- final String **corbaloc** = "corbaloc::localhost:39273/ServiceManager"
  the Default location where we expect the service manager to listen.

9.20.1 Detailed Description

This is a Demo Service that can be used as a template for **DWARF** (p. 59) services that want to provide object references to other service as a **DWARF** (p. 59) ability.

In our case, DemoStatusExporter exports its own reference to enable others to query about its status. (For that reason, it implements the dwarf Status interface)

Note that any Corba Object (subclass of org.omg.CORBA.Object) can be exportert via the object exporter.

Author:
The TRAMP Network team

Version:

Revision:
1.6

9.20.2 Constructor & Destructor Documentation

9.20.2.1 DemoStatusExporter::DemoStatusExporter (String *name*, String *status*) [inline]

Creates new DemoStatusExporter.

Registers a service with the internal name of name+Service, witch has a Name "name" and returns status "status"

Parameters:

name The Name of the Service to be created. Must be unique for all running services.

status The Status to be returned by the service.

9.20.3 Member Function Documentation

9.20.3.1 void DemoStatusExporter::main (String *args*[]) [inline, static]

Dull main method.

Just checks the command line arguments and instancitates the class

Parameters:

args the command line arguments

The documentation for this class was generated from the following file:

- DemoStatusExporter.java

9.21 DemoStatusImporter Class Reference

This is a Demo Service which queries other **DWARF** (p. 59) Services for their status.

Public Methods

- **DemoStatusImporter** ()
  Creates new DemoStatusImporter.
- void **showStatus** ()
  showStatus.
- void **run** ()
  does a loop and shows the status, over and over again.
- **DemoStatusImporter** ()
  Creates new DemoStatusImporter.
- void **showStatus** ()
  showStatus.
- void **run** ()
  does a loop and shows the status, over and over again.

Static Public Methods

- void **main** (String args[])
  Main method.
- void **main** (String args[])
  Main method.

Public Attributes

- final String **ID** = "$Id: DemoStatusImporter.java,v 1.6 2002/01/09 13:18:41 richter Exp $"
  Our ID in the CVS:
  Id:
  DemoStatusImporter.java,v 1.2 2002/01/29 12:23:56 winterm Exp
  .

Static Public Attributes

- final String **corbaloc** = "corbaloc::localhost:39273/ServiceManager"
  the Default location where we expect the service manager to listen.

9.21.1 Detailed Description

This is a Demo Service which queries other **DWARF** (p. 59) Services for their status.

Can be used as a template for any service that imports Object References.

NOTE: The **ObjectImporter** (p. 201) interface is not yet considered stable. It will probably change some time after christmas.

Author:
://tramp.globalse.org/teams.html#network The TRAMP Network team

Version:

Revision:
1.6

9.21.2 Member Function Documentation

9.21.2.1 void DemoStatusImporter::main (String *args*[]) [inline, static]

Main method.

Nothing to it.

Parameters:
args the command line arguments

9.21.2.2 void DemoStatusImporter::main (String *args*[]) [inline, static]

Main method.

Nothing to it.

Parameters:
args the command line arguments

9.21.2.3 void DemoStatusImporter::run () [inline]

does a loop and shows the status, over and over again.

What a dull job...

9.21.2.4 void DemoStatusImporter::run () [inline]

does a loop and shows the status, over and over again.

What a dull job...

9.21.2.5 void DemoStatusImporter::showStatus () [inline]

showStatus.

Shows what we know about other services

9.21.2.6 void DemoStatusImporter::showStatus () [inline]

showStatus.

Shows what we know about other services

The documentation for this class was generated from the following files:

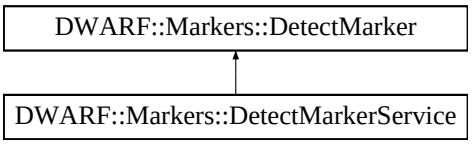
- dwarf/utiljava/DWARF/DemoStatusImporter.java
- java/org/globalse/tramp/services/TestingGUI/DemoStatusImporter.java

9.22 DWARF::Markers::DetectMarker Interface Reference

This interface discribes the marker detection service.

`import "DetectMarkers.idl";`

Inheritance diagram for DWARF::Markers::DetectMarker::



Public Methods

- long **startDetection** (in string uri)
  Starts the detection of a marker.
- void **stopDetection** (in long id)
  Stops the detection of a marker.

9.22.1 Detailed Description

This interface discribes the marker detection service.

9.22.2 Member Function Documentation

9.22.2.1 long DWARF::Markers::DetectMarker::startDetection (in string *uri*)

Starts the detection of a marker.

Parameters:
uri The uri of the marker to detect

Returns:
The ID of the marker that will be used to refer to it in subsequent function calls.

9.22.2.2 void DWARF::Markers::DetectMarker::stopDetection (in long *id*)

Stops the detection of a marker.

Parameters:
id The id of the marker whose detection to be stopped

The documentation for this interface was generated from the following file:

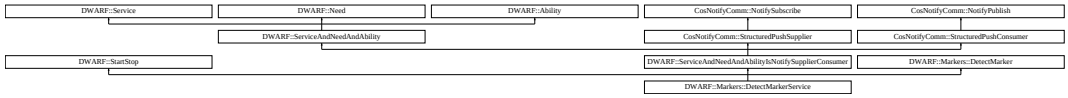
- **DetectMarkers.idl**

9.23 DWARF::Markers::DetectMarkerService Interface Reference

This interface creates a **DetectMarker** (p. 104) **DWARF** (p. 59) service.

```
import "DetectMarkers.idl";
```

Inheritance diagram for DWARF::Markers::DetectMarkerService::



9.23.1 Detailed Description

This interface creates a **DetectMarker** (p. 104) **DWARF** (p. 59) service.

To call it, please use the **DetectMarker** (p. 104) interface

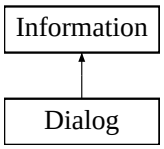
The documentation for this interface was generated from the following file:

- **DetectMarkers.idl**

9.24 Dialog Class Reference

Some workflows require that complex dialogs like checklists are presented to the user - these should be represented by this class stub.

Inheritance diagram for Dialog::



9.24.1 Detailed Description

Some workflows require that complex dialogs like checklists are presented to the user - these should be represented by this class stub.

As these dialogs may also include information that is relevant to the execution of the workflow this special class was introduced. It should particulary manage the storage of the dialogs output (e.g. the users input) but should also relay necessary information to the workflow. An implementation might use a special **Document** (p. 109) tag in the workflow's description to describe the actions to be performed after the execution of the dialog.

Author:
Stefan Ri"s

Version:

Revision:
1.1

The documentation for this class was generated from the following file:

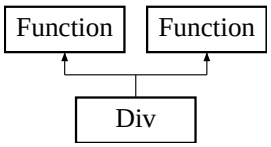
- Dialog.java

9.25 Div Class Reference

Division function composition.

`#include <function.h>`

Inheritance diagram for Div::



Public Methods

- **Div** (**Function** *f1, **Function** *f2)
- virtual ~**Div** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** * **getEnumerator** () const
- virtual **Function** * **getDenominator** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const
- **Div** (**Function** *f1, **Function** *f2)
- virtual ~**Div** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** * **getEnumerator** () const
- virtual **Function** * **getDenominator** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const

9.25.1 Detailed Description

Division function composition.

Author:
Andreas Krause

See also:
Fct (p. 123)

9.25.2 Member Function Documentation

9.25.2.1 virtual Function* Div::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

9.25.2.2 Function * Div::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

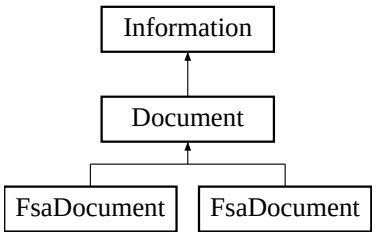
The documentation for this class was generated from the following files:

- **MovementSimulator/function.h**
- **TrackingFilter/function.h**
- MovementSimulator/function.cpp
- TrackingFilter/function.cpp

9.26 Document Class Reference

This class should include the information necessary to describe a document.

Inheritance diagram for Document::



Public Methods

- **Document** ()

9.26.1 Detailed Description

This class should include the information necessary to describe a document.

As a subclass of **Information** (p. 145) it already has an associated URI. Additional information could be time of creation, author, keywords or search index information.

See also:
de.arvika.workflow.fsaWorkflow.FsaDocument

Author:
Stefan Ri"s

Version:

Revision:
1.1

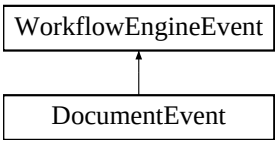
The documentation for this class was generated from the following file:

- Document.java

9.27 DocumentEvent Class Reference

This class is used to provide a *DocumentEventListener* with information about avilable or requested documents.

Inheritance diagram for DocumentEvent::



Public Methods

- **DocumentEvent** (**Information** src, int id)

  Use this constructor to create an evnet that directly refers to a specific document.

- **DocumentEvent** (**InformationList** src, int id)

  As described in the introduction of this class, DocumentEvents may also be used to notify a listener about a list of available Documents.

- String **toString** ()

  Overrides Object toString to produce nicer output in debug statements.

- **Information getInformation** ()

  Retrieves the provided **Information** (p. 145) *using the EventObjects getSource() method.*

- **InformationList getInformationList** ()

  Retrieves the provided **InformationList** (p. 147) *using the EventObjects getSource() method.*

Static Public Attributes

- final int **SHOW_REQUESTED** = 1

  Identifies a request to show a certain document.

- final int **NEW_TASK_DOCUMENTS** = 2

  An event whose ID equals this constant carries a list of documents that belong to a new task.

- final int **WORKFLOW_DOCUMENTS** = 4

  The description language allows for documents that belong to a complete workflow.

- final int **SUBWORKFLOW_STARTREQUEST** = 8

  As a workflow may refer to subworkflow in a normal **Document** (p. 109) *tag and request the start of that workflow with the GoSub action, we consequently introduced a DocumentEvent type that covers such requests.*

9.27.1 Detailed Description

This class is used to provide a *DocumentEventListener* with information about avilable or requested documents.

For example whenever a Workflows gets into a new Task it should send out a DocumentEvent containing a list of all available documents within the task. On the other hand this event tells a listener to show a certain document this is normally caused by a call to a WorkflowEngines requestDocument method. In this case the subsequent calls to the WorkflowController and Workflow should produce a DocumentEvent containing a reference to the **Information** (p. 145) to be displayed. Please note the this handles not only Documents but all **Information** (p. 145) types. The name *DocumentEvent* is just a little bit more descriptive than *InformationEvent* would be.

Implementation citique: It would have made more sense to split this class into two classes. One that handles single document events like this and another that handles InformatinListEvents. This would also be more consistent with the AWT's event concept which was the model for the Workflow event mechanism.

See also:
DocumentListener (p. 113) , de.arvika.workflow.fsaWorkflow.RequestDocumentCommand , de.arvika.workflow.WorkflowEnginerequestDocument

Author:
Stefan Ri"s

Version:

Revision:
1.1

9.27.2 Constructor & Destructor Documentation

9.27.2.1 DocumentEvent::DocumentEvent (Information *src*, int *id*) [inline]

Use this constructor to create an evnet that directly refers to a specific document.

Should be used with types SHOW_REQUESTED and SUBWORKFLOW_STARTREQUEST.

Parameters:
src An **Information** (p. 145) that includes a reference to the document to be displayed or to the subworkflow that should be started.
id Pass one of this class' constants to this constructor to further specify the event type.

9.27.2.2 DocumentEvent::DocumentEvent (InformationList *src*, int *id*) [inline]

As described in the introduction of this class, DocumentEvents may also be used to notify a listener about a list of available Documents.

This is constructor should therfore be used for the types NEW_TASK_DOCUMENTS and WORKFLOW_DOCUMENTS.

Parameters:
src An **InformationList** (p. 147) that should contain all the Documents that are available in a workflow or a certain task.
id Pass one of this class' constants to this constructor to further specify the event type.

9.27.3 Member Function Documentation

9.27.3.1 Information DocumentEvent::getInformation () [inline]

Retrieves the provided **Information** (p. 145) using the EventObjects getSource() method.

See also:
java.util.EventObjectgetSource

Returns:
The **Information** (p. 145) that is to be displayed or used as a subworkflow.

9.27.3.2 InformationList DocumentEvent::getInformationList () [inline]

Retrieves the provided **InformationList** (p. 147) using the EventObjects getSource() method.

See also:
java.util.EventObjectgetSource

Returns:
The **InformationList** (p. 147) that references the available documents.

9.27.4 Member Data Documentation

9.27.4.1 final int DocumentEvent::WORKFLOW_DOCUMENTS = 4 [static]

The description language allows for documents that belong to a complete workflow.

These events deliver that list of documents to the listener.

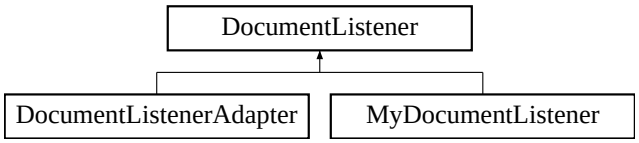
The documentation for this class was generated from the following file:

- DocumentEvent.java

9.28 DocumentListener Interface Reference

This interface defines the method a class has to implement so that it may be registered as a listener for DocumentEvents.

Inheritance diagram for DocumentListener::



Public Methods

- abstract void **showRequested** (**DocumentEvent** e)

  This method is called when a publisher requests a document to be shown.

- abstract void **newDocuments** (**DocumentEvent** e)

  Whenever a workflow changes its current state or a new workflow is started it publishes a list of avilable methods.

- abstract void **startSubWorkflow** (**DocumentEvent** e)

  The event publisher (e.g.

9.28.1 Detailed Description

This interface defines the method a class has to implement so that it may be registered as a listener for DocumentEvents.

Smilar to the **DocumentEvent** (p. 110) this class would also have been best to splitted into a DocumentListner and a DocumentListListener. In analogy to the AWT event model we provided an adapter with empty implementations of this interface's methods.

See also:
DocumentEvent (p. 110) , **DocumentListenerAdapter** (p. 115)

Author:
Stefan Ri"s

Version:

Revision:
1.1

9.28.2 Member Function Documentation

9.28.2.1 abstract void DocumentListener::newDocuments (DocumentEvent *e*) [pure virtual]

Whenever a workflow changes its current state or a new workflow is started it publishes a list of avilable methods.

Which eventually leads to a call of this method. **DocumentEvent** (p. 110) types: NEW_TASK_-DOCUMENTS, WORKFLOW_DOCUMENTS

Reimplemented in **DocumentListenerAdapter** (p. 115), and **MyDocumentListener** (p. 177).

9.28.2.2 abstract void DocumentListener::showRequested (DocumentEvent *e*) [pure virtual]

This method is called when a publisher requests a document to be shown.

The listener then should take care of the document's display.

Reimplemented in **DocumentListenerAdapter** (p. 116), and **MyDocumentListener** (p. 177).

9.28.2.3 abstract void DocumentListener::startSubWorkflow (DocumentEvent *e*) [pure virtual]

The event publisher (e.g.

the WorkflowEngine) calls this method if a workflow issues a GoSub request. In this case the listner has not to handle the start of this subworkflow. This is normally done by the WorkflowController.

Reimplemented in **DocumentListenerAdapter** (p. 116), and **MyDocumentListener** (p. 177).

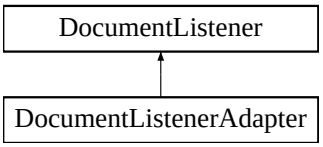
The documentation for this interface was generated from the following file:

- DocumentListener.java

9.29 DocumentListenerAdapter Class Reference

This class provides empty bodies for all the methods of the **DocumentListener** (p. 113) interface.

Inheritance diagram for DocumentListenerAdapter::



Public Methods

- void **showRequested** (**DocumentEvent** e)

  This method is called when a publisher requests a document to be shown.

- void **newDocuments** (**DocumentEvent** e)

  Whenever a workflow changes its current state or a new workflow is started it publishes a list of avilable methods.

- void **startSubWorkflow** (**DocumentEvent** e)

  The event publisher (e.g.

9.29.1 Detailed Description

This class provides empty bodies for all the methods of the **DocumentListener** (p. 113) interface.

Most of the time it is more convinient to override this class' methods than to implment the interface yourself. Refer to the **DocumentListener** (p. 113) to get a detailed description of the methods.

See also:
DocumentListener (p. 113)

Author:
Stefan Ri"s

Version:

Revision:
1.1

9.29.2 Member Function Documentation

9.29.2.1 void DocumentListenerAdapter::newDocuments (DocumentEvent *e*) [inline, virtual]

Whenever a workflow changes its current state or a new workflow is started it publishes a list of avilable methods.

Which eventually leads to a call of this method. **DocumentEvent** (p. 110) types: NEW_TASK_DOCUMENTS, WORKFLOW_DOCUMENTS

Reimplemented from **DocumentListener** (p. 114).

9.29.2.2 void DocumentListenerAdapter::showRequested (DocumentEvent *e*) [inline, virtual]

This method is called when a publisher requests a document to be shown.

The listener then should take care of the document's display.

Reimplemented from **DocumentListener** (p. 114).

9.29.2.3 void DocumentListenerAdapter::startSubWorkflow (DocumentEvent *e*) [inline, virtual]

The event publisher (e.g.

the WorkflowEngine) calls this method if a workflow issues a GoSub request. In this case the listner has not to handle the start of this subworkflow. This is normally done by the WorkflowController.

Reimplemented from **DocumentListener** (p. 114).

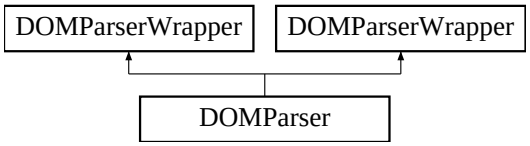
The documentation for this class was generated from the following file:

- DocumentListenerAdapter.java

9.30 DOMParser Class Reference

Wraps the Xerces DOM parser and extends NonValidatingDOMParser.

Inheritance diagram for DOMParser::



Public Methods

- **DOMParser** ()

  Default constructor.

- **Document parse** (String uri) throws Exception

  Parses the specified URI and returns the document.

- void **setFeature** (String featureId, boolean state) throws SAXNotRecognizedException
- **DOMParser** ()

  Default constructor.

- **Document parse** (String uri) throws Exception

  Parses the specified URI and returns the document.

- void **setFeature** (String featureId, boolean state) throws SAXNotRecognizedException

9.30.1 Detailed Description

Wraps the Xerces DOM parser and extends NonValidatingDOMParser.

Version:
id

The documentation for this class was generated from the following files:

- dwarf/utiljava/DWARF/DOMParser.java
- java/org/globalse/tramp/services/ExpertSystem/DOMParser.java

9.31 ExpertSystem Class Reference

implements the ExpertSystem.

Public Methods

- **ExpertSystem** (String corbaloc)

  Constructor-method of ExpertSystem, registers needs and abilities.

- String **getName** ()

  @section getName returns the name of this subsystem.

- String **getStatus** ()

  @section getStatus returns the status of this subsystem.

- int **FindTaskflow** (String searchstring, org.omg.CORBA.StringHolder result)

  @section FindTaskflow finds the right taskflow according to the given string.

- void **run** ()

  thread holding the application running.

Static Public Methods

- void **main** (String args[])

  main-method of ExpertSystem.

Public Attributes

- final String **ID** = "$Id: ExpertSystem.java,v 1.13 2002/02/06 12:07:14 hu Exp $"

  Our ID in the CVS:
  Id:
  ExpertSystem.java,v 1.13 2002/02/06 12:07:14 hu Exp
  .

Static Public Attributes

- final String **DEFLOCATION** = "corbaloc::localhost:39273/ServiceManager"

  the Default location where we expect the service manager to listen.

9.31.1 Detailed Description

implements the ExpertSystem.

9.31.2 Constructor & Destructor Documentation

9.31.2.1 ExpertSystem::ExpertSystem (String *corbaloc*) [inline]

Constructor-method of ExpertSystem, registers needs and abilities.

Parameters:
corbaloc the url where we find the service manager

Author:
otto

Date:
2001/01/29

9.31.3 Member Function Documentation

9.31.3.1 int ExpertSystem::FindTaskflow (String *searchstring*, org.omg.CORBA.StringHolder *result*) [inline]

@section FindTaskflow finds the right taskflow according to the given string.

Parameters:
searchstring the string to be searched

result consisting of the descriptor of the found workflow

Returns:
ErrorCode the error code, 0 when no error occurs

Author:
otto

Date:
22.1.02

9.31.3.2 String ExpertSystem::getName () [inline]

@section getName returns the name of this subsystem.

Returns:
name of this subsystem is returned.

Author:
otto

Date:
22.1.02

9.31.3.3 String ExpertSystem::getStatus () [inline]

@section getStatus returns the status of this subsystem.

Returns:
status of this subsystem is returned.

Author:
otto

Date:
22.1.02

9.31.3.4 void ExpertSystem::main (String *args*[]) [inline, static]

main-method of ExpertSystem.

Parameters:
args

Author:
otto

Date:
22.1.02

9.31.3.5 void ExpertSystem::run () [inline]

thread holding the application running.

Author:
otto

Date:
22.1.02

The documentation for this class was generated from the following file:

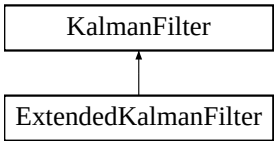
- **ExpertSystem.java**

9.32 ExtendedKalmanFilter Class Reference

A nonlinear Kalman Filter implementation.

`#include <kalmanfilter.h>`

Inheritance diagram for ExtendedKalmanFilter::



Public Methods

- **ExtendedKalmanFilter** ()
- ~**ExtendedKalmanFilter** ()
- void **setF** (const FctMatrix &pnewVal)
- const FctMatrix & **getF** ()
- const Matrix & **getP0** ()

  Read property of Matrix P0.

- Matrix **update** (Matrix z, Matrix u)

  does measurement and prediction update according to measurement z.

- const Matrix & **getX** ()

  Read property of Matrix X.

- void **setH** (const FctMatrix &pnewVal)

  Write property of Matrix H.

- const FctMatrix & **getH** ()

  Read property of Matrix H.

- void **setQ** (const Matrix &pnewVal)

  Write property of Matrix Q.

- const Matrix & **getQ** ()

  Read property of Matrix Q.

- void **setR** (const Matrix &pnewVal)

  Write property of Matrix R.

- const Matrix & **getR** ()

  Read property of Matrix R.

- void **init** (Matrix pX0, Matrix pP0, FctMatrix pF, FctMatrix pH, Matrix pQ, Matrix pR, int u)

  initializes the Kalman filter.

9.32.1 Detailed Description

A nonlinear Kalman Filter implementation.

Author:
Andreas Krause implements the nonlinear version of the **KalmanFilter** (p. 149), used by the TrackingFilter

9.32.2 Member Function Documentation

9.32.2.1 void ExtendedKalmanFilter::init (Matrix *pX0*, Matrix *pP0*, FctMatrix *pF*, FctMatrix *pH*, Matrix *pQ*, Matrix *pR*, int *u*)

initializes the Kalman filter.

Parameters:
X0 initial measurement prediction
P0 initial predicted covariance
F nonlinear model description
H nonlinear relation of measurement to state vector
Q initial process noice
R initial measurement error covariance
u controlVars

9.32.2.2 Matrix ExtendedKalmanFilter::update (Matrix *z*, Matrix *u*) [virtual]

does measurement and prediction update according to measurement z.

Parameters:
z current measurement
u current process control

Returns:
updated estimate; use **getX**() (p. 121) to get new prediction

Reimplemented from **KalmanFilter** (p. 149).

The documentation for this class was generated from the following files:

- **kalmanfilter.h**
- kalmanfilter.cpp

9.33 Fct Class Reference

wraps a symbolic function.

`#include <function.h>`

Public Methods

- **Fct** ()
- **Fct** (double val)
- **Fct** (int val)
- **Fct** (**Function** ∗fct)
- ∼**Fct** ()
- Fct **diff** (int varNo)

  differentiates the function with respect to variable varNo.

- Fct **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- double **operator()** (Matrix **values**) const
- void **print** (ostream &ostrm) const

  used to print the function.

- **Function** ∗ **getFct** () const
- Fct **operator+=** (const Fct &f)
- Fct **operator** ∗= (const Fct &f)
- Fct **operator/=** (const Fct &f)
- Fct **operator-=** (const Fct &f)
- Fct **operator-** ()
- void **noDel** ()
- **Fct** ()
- **Fct** (double val)
- **Fct** (int val)
- **Fct** (**Function** ∗fct)
- ∼**Fct** ()
- Fct **diff** (int varNo)

  differentiates the function with respect to variable varNo.

- Fct **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- double **operator()** (Matrix **values**) const
- void **print** (ostream &ostrm) const

  used to print the function.

- **Function** ∗ **getFct** () const
- Fct **operator+=** (const Fct &f)
- Fct **operator** ∗= (const Fct &f)

- Fct **operator/=** (const Fct &f)
- Fct **operator-=** (const Fct &f)
- Fct **operator-** ()
- void **noDel** ()

Friends

- class **Function**

9.33.1 Detailed Description

wraps a symbolic function.

Author:
Andreas Krause This is the class instantiated by the filter Example Fct f = sin(var(1))+3.14+var(0)∗var(2); Fct g = f.diff(1).**simplify**() (p. 123);

9.33.2 Member Function Documentation

9.33.2.1 Fct Fct::diff (int *varNo*) [inline]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

9.33.2.2 Fct Fct::diff (int *varNo*) [inline]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

The documentation for this class was generated from the following files:

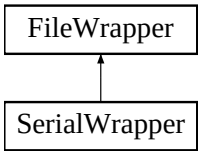
- **MovementSimulator/function.h**
- **TrackingFilter/function.h**

9.34 FileWrapper Class Reference

A class that wrapps access to a file. The file can be given by name and path and is remembered by the FileWrapper for further use. Arbitrary open and close operations can be done by the FileWrapper on the wrapped file. The FileWrapper can also be used as interface for wrapping more complex data sources. (e.g. serail ports, raw devices, sockets).

`#include <FileWrapper.h>`

Inheritance diagram for FileWrapper::



Public Methods

- **FileWrapper** (char ∗**name**, char ∗**mode**)

  Creates a new FileWrapper for the given file.

- **∼FileWrapper** ()

  Destroys this FileWrapper.

- virtual void **setName** (char ∗**name**)

  Sets the file to be wrapped by this FileWrapper.

- virtual void **setMode** (char ∗**mode**)

  Sets the mode to use for accessing the wrapped file.

- virtual FILE ∗ **open** ()

  Tries to opens the wrapped file.

- virtual FILE ∗ **getFile** ()

  If the wrapped file is open, its FILE handle is returned.

- virtual bool **isOpen** ()

  Checks if the wrapped file is open.

- virtual void **close** ()

  Tries to close the wrapped file.

Protected Attributes

- char ∗ **name**

  The name of the wrapped file.

- char ∗ **mode**

The access mode of the wrapped file.

- FILE * **file**

  The FILE handle of the wrapped file.

9.34.1 Detailed Description

A class that wrapps access to a file. The file can be given by name and path and is remembered by the FileWrapper for further use. Arbitrary open and close operations can be done by the FileWrapper on the wrapped file. The FileWrapper can also be used as interface for wrapping more complex data sources. (e.g. serail ports, raw devices, sockets).

9.34.2 Constructor & Destructor Documentation

9.34.2.1 FileWrapper::FileWrapper (char * *name*, char * *mode*)

Creates a new FileWrapper for the given file.

Parameters:

name The name (including path) of the file to wrap. It is not checked, if this file exists.

mode The access mode to use when opening the file.

9.34.2.2 FileWrapper::~FileWrapper ()

Destroys this FileWrapper.

If the wrapped file, is still open, It is closed before.

9.34.3 Member Function Documentation

9.34.3.1 void FileWrapper::close () [virtual]

Tries to close the wrapped file.

If the file is not open, nothing happens.

9.34.3.2 FILE * FileWrapper::getFile () [virtual]

If the wrapped file is open, its FILE handle is returned.

Returns:

The FILE handle of the wrapped file, or NULL, if it is not open.

9.34.3.3 bool FileWrapper::isOpen () [virtual]

Checks if the wrapped file is open.

Returns:

true, if the wrapped file is open, otherwise false.

9.34.3.4 FILE * FileWrapper::open () [virtual]

Tries to opens the wrapped file.

This is only possible, if it exists, and can be accessed.

Returns:
The FILE handle of the wrapped file, or NULL, if it could not be opened.

Reimplemented in **SerialWrapper** (p. 235).

9.34.3.5 void FileWrapper::setMode (char * *mode*) [virtual]

Sets the mode to use for accessing the wrapped file.

If the file was already open, it is closed before.

Parameters:
mode The file access mode to use. It can be one of r, r+, w, w+, a or a+.

9.34.3.6 void FileWrapper::setName (char * *name*) [virtual]

Sets the file to be wrapped by this FileWrapper.

If another file was wrapped before, it is closed before.

Parameters:
name The name (including path) of the file to wrap. It is not checked, if this file exists.

The documentation for this class was generated from the following files:

- **FileWrapper.h**
- **FileWrapper.cpp**

9.35 FlashClient Class Reference

CSClient

Client for the FlashServer.

Public Methods

- **FlashClient** (FlashServer app, Socket socket)

  ——————————————————————————- *Constructor for the FlashClient.*

- void **run** ()

  ——————————————————————————- *Thread run method.*

- String **getIP** ()

  ——————————————————————————- *Gets the ip of this client.*

- void **send** (String message)

  ——————————————————————————- *Sends a message to this client.*

Protected Attributes

- BufferedReader **in**
- PrintWriter **out**

9.35.1 Detailed Description

CSClient

Client for the FlashServer.

9.35.2 Constructor & Destructor Documentation

9.35.2.1 FlashClient::FlashClient (FlashServer *app*, Socket *socket*) [inline]

——————————————————————————- Constructor for the FlashClient.

Initializes the FlashClient properties.

Parameters:

server The server to which this client is connected.

socket The socket through which this client has connected.

app The FlashServerApplet in which the server is started. ——————————————————————————

9.35.3 Member Function Documentation

9.35.3.1 String FlashClient::getIP () [inline]

- Gets the ip of this client.

Returns:
ip this client's ip

9.35.3.2 void FlashClient::run () [inline]

- Thread run method.

Monitors incoming messages.

9.35.3.3 void FlashClient::send (String *message*) [inline]

- Sends a message to this client.

Called by the server's broadcast method.

Parameters:
message The message to send.

The documentation for this class was generated from the following file:

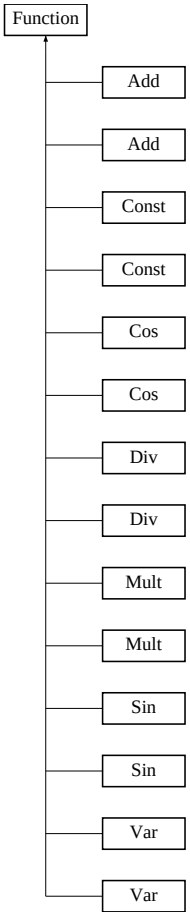
- FlashClient.java

9.36 Function Class Reference

Generic implementation of a function class.

`#include <function.h>`

Inheritance diagram for Function::



Public Methods

- **Function** ()
- virtual ~**Function** ()
- virtual Function * **diff** (int varNo) const=0

  differentiates the function with respect to variable varNo.

- virtual Function * **simplify** ()=0

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual int **getType** () const=0
- virtual double **operator()** (Matrix **values**) const=0
- virtual void **print** (ostream &ostrm) const=0

used to print the function.

- virtual Function * **clone** () const=0
- **Function** ()
- virtual ~**Function** ()
- virtual Function * **diff** (int varNo) const=0

  differentiates the function with respect to variable varNo.

- virtual Function * **simplify** ()=0

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual int **getType** () const=0
- virtual double **operator()** (Matrix **values**) const=0
- virtual void **print** (ostream &ostrm) const=0

  used to print the function.

- virtual Function * **clone** () const=0

9.36.1 Detailed Description

Generic implementation of a function class.

Author:
Andreas Krause

See also:
Fct (p. 123) this class is superclass for specific functions, it is wrapped by the **Fct** (p. 123) class

9.36.2 Member Function Documentation

9.36.2.1 virtual Function* Function::diff (int *varNo*) const [pure virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented in **Add** (p. 66), **Mult** (p. 174), **Div** (p. 108), **Cos** (p. 87), **Sin** (p. 242), **Var** (p. 268), **Const** (p. 81), **Add** (p. 66), **Mult** (p. 174), **Div** (p. 108), **Cos** (p. 87), **Sin** (p. 242), **Var** (p. 268), and **Const** (p. 81).

9.36.2.2 virtual Function* Function::diff (int *varNo*) const [pure virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented in **Add** (p. 66), **Mult** (p. 174), **Div** (p. 108), **Cos** (p. 87), **Sin** (p. 242), **Var** (p. 268), **Const** (p. 81), **Add** (p. 66), **Mult** (p. 174), **Div** (p. 108), **Cos** (p. 87), **Sin** (p. 242), **Var** (p. 268), and **Const** (p. 81).

The documentation for this class was generated from the following files:

- **MovementSimulator/function.h**
- **TrackingFilter/function.h**
- MovementSimulator/function.cpp
- TrackingFilter/function.cpp

9.37 GarageServerApplication Class Reference

implements the GarageServer.

Public Methods

- **GarageServerApplication** (String corbaloc)
  Constructor-method of GarageServerApplication.
- int **getTaskflowAction** (idval data)
  Interface for Taskflow engine to transmit Taskflow options.
- void **transferMessage** (int id, String mes)
  message Interface for intra Application communication.
- String **getName** ()
  returns the status of this service.
- String **getStatus** ()
  returns the status of this service.
- boolean **checkStatus** ()
  checks if all services we need are available.
- void **sendAppMessage** (int id, String mes)
  sends our messages messages.
- void **sendUIEConfigurationData** (String path)
  sends adress of a xml-file where configuration-data is listed.
- int **triggerTaskflowTransition** (idval Taskflowoption)
  sends a list with instructions to Taskflow Engine.
- int **getTaskflow** (String search, org.omg.CORBA.StringHolder result)
  gets results of a question to the **ExpertSystem** (p. 118).

Static Public Methods

- void **main** (String args[])
  main-method of GarageServerApplication.

Static Public Attributes

- final String **DEFLOCATION** = "corbaloc::localhost:39273/ServiceManager"
  the Default location where we expect the service manager to listen.

9.37.1 Detailed Description

implements the GarageServer.

9.37.2 Constructor & Destructor Documentation

9.37.2.1 GarageServerApplication::GarageServerApplication (String *corbaloc*) [inline]

Constructor-method of GarageServerApplication.

Parameters:
corbaloc the url where we find the service manager

Author:
meierb

Date:
22.1.02

9.37.3 Member Function Documentation

9.37.3.1 boolean GarageServerApplication::checkStatus () [inline]

checks if all services we need are available.

Precondition:

Postcondition:
all needed services are available

Author:
meierb

Date:
25.1.02

9.37.3.2 String GarageServerApplication::getName () [inline]

returns the status of this service.

Precondition:

Postcondition:

Author:
meierb

Date:
25.1.02

9.37.3.3 String GarageServerApplication::getStatus () [inline]

returns the status of this service.

Precondition:

Postcondition:

Author:
meierb

Date:
25.1.02

9.37.3.4 int GarageServerApplication::getTaskflow (String *search*, org.omg.CORBA.StringHolder *result*) [inline]

gets results of a question to the **ExpertSystem** (p. 118).

Parameters:
search the search string
result the CUIML file to the UI

Precondition:

Postcondition:

Author:
meierb

Date:
22.1.02

9.37.3.5 int GarageServerApplication::getTaskflowAction (idval *data*) [inline]

Interface for Taskflow engine to transmit Taskflow options.

Parameters:
data

Precondition:
a Taskflow was triggered

Postcondition:
The path for a ∗.TUIML file has been send to UIController

Author:
hilliges

Date:
23.1.02

9.37.3.6 void GarageServerApplication::main (String *args*[]) [inline, static]

main-method of GarageServerApplication.

Parameters:
args

Author:
meierb

Date:
22.1.02

9.37.3.7 void GarageServerApplication::sendAppMessage (int *id*, String *mes*) [inline]

sends our messages messages.

Parameters:
id the message id

mes the message string

Precondition:

Postcondition:

Author:
meierb

Date:
22.1.02

9.37.3.8 void GarageServerApplication::sendUIEConfigurationData (String *path*) [inline]

sends adress of a xml-file where configuration-data is listed.

Parameters:
path the relative path of the xml file

Precondition:

Postcondition:

Author:
meierb

Date:
22.1.02

9.37.3.9 void GarageServerApplication::transferMessage (int *id*, String *mes*) [inline]

message Interface for intra Application communication.

Parameters:

id the message id

mes the message string

Precondition:

Postcondition:

either the state of this Service is changed or the Taskflow will be changed use this option with care because their is no kind of security and this might crash the whole system.

Author:

hilliges

Date:

23.1.02

9.37.3.10 int GarageServerApplication::triggerTaskflowTransition (idval *Taskflowoption*) [inline]

sends a list with instructions to Taskflow Engine.

list of instructions contains an ID and value in each element

Parameters:

Taskflowoption the option of the tasklflow

Precondition:

Postcondition:

Author:

meierb

Date:

22.1.02

The documentation for this class was generated from the following file:

- **GarageServerApplication.java**

9.38 GarageUserThread Class Reference

This class implements a trivial (and rather stupid) **DemoService** (p. 96) for **DWARF** (p. 59) that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).

Public Methods

- **GarageUserThread** (String corbaloc)

Static Public Methods

- void **main** (String args[])

Static Public Attributes

- final String **DEFLOCATION** = "corbaloc::localhost:39273/ServiceManager"
  the Default location where we expect the service manager to listen.

9.38.1 Detailed Description

This class implements a trivial (and rather stupid) **DemoService** (p. 96) for **DWARF** (p. 59) that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).

Please use this a a template for your own communication classes.

Author:
The TRAMP Network Team

Version:

Revision:
1.1

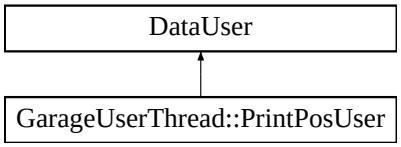
The documentation for this class was generated from the following file:

- GarageUserThread.java

9.39 GarageUserThread::PrintPosUser Class Reference

A sample implementation of the interface **DataUser** (p. 94), which is used for the callback calls wenn the receiver gets an event.

Inheritance diagram for GarageUserThread::PrintPosUser::



Public Methods

- void **event** (StructuredEvent se)

  The event is triggered each time an event is received.

9.39.1 Detailed Description

A sample implementation of the interface **DataUser** (p. 94), which is used for the callback calls wenn the receiver gets an event.

9.39.2 Member Function Documentation

9.39.2.1 void GarageUserThread::PrintPosUser::event (StructuredEvent *se*) [inline]

The event is triggered each time an event is received.

NOTE: The callback implemented in the ORB is multithreaded, so please remember using syncronize() and/or semaphores to handle critical program parts (like modifying object variables) safely!

Parameters:

se the generic Corba StructuredEvent to be used for transmitting the data. Any CorbaObject can be transmitted in the se.remainder_of_body field. What you do with it is up to convention for your specific event ...

Reimplemented from **DataUser** (p. 95).

The documentation for this class was generated from the following file:

- GarageUserThread.java

9.40 DWARF::GestureData Struct Reference

contains information about the type of time of a recognized gesture.

`import "GestureData.idl";`

Public Attributes

- **GestureType type**

  type of the gesture (yes or no).

- Time **t**

  time when the gesture was recognized.

9.40.1 Detailed Description

contains information about the type of time of a recognized gesture.

The documentation for this struct was generated from the following file:

- **GestureData.idl**

9.41 gestureEvent Struct Reference

struct containing data from the inertial tracker (converted due to easier use for the gesture recognition).

`#include <gesturerecognition.h>`

Public Attributes

- float **yaw**

  rotation around body axis (in degrees).

- float **pitch**

  rotation around side axis (in degrees).

- double **time**

  timestamp (in seconds).

9.41.1 Detailed Description

struct containing data from the inertial tracker (converted due to easier use for the gesture recognition).

The documentation for this struct was generated from the following file:

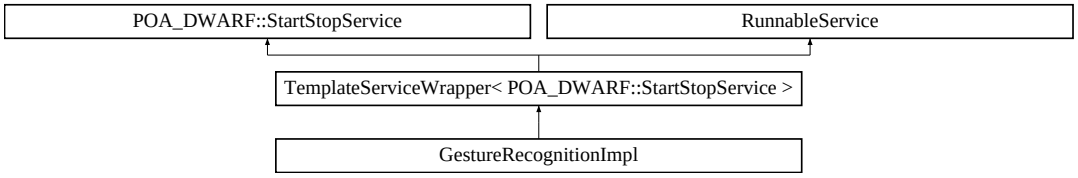
- **gesturerecognition.h**

9.42 GestureRecognitionImpl Class Reference

Main class for the gesture recognition service.

`#include <gesturerecognition.h>`

Inheritance diagram for GestureRecognitionImpl::



Public Methods

- **GestureRecognitionImpl** ()
  This constructor initialises the attributes of the class.
- void **describe** ()
  Describes the service, including its needs and abilites.
- void **push_structured_event** (const CosNotification::StructuredEvent &event)
  This method receives the events from the tracker.
- char * **getName** ()
  Returns the name of the service (implements the DWARFStatus interface).
- char * **getStatus** ()
  Returns the status of the service (implements the DWARFStatus interface).
- void **start** ()
  Starts the gesture recognition.
- void **stop** ()
  Stops the gesture recognition.

9.42.1 Detailed Description

Main class for the gesture recognition service.

9.42.2 Constructor & Destructor Documentation

9.42.2.1 GestureRecognitionImpl::GestureRecognitionImpl ()

This constructor initialises the attributes of the class.

If the parameters for the recognition have to be changed, this has to be done in the sourcecode of the constructor (there is no pulic method for that because it shouldn't be necessary to change them)

9.42.3 Member Function Documentation

9.42.3.1 char * GestureRecognitionImpl::getName ()

Returns the name of the service (implements the DWARFStatus interface).

Returns:
string containing the service name

9.42.3.2 char * GestureRecognitionImpl::getStatus ()

Returns the status of the service (implements the DWARFStatus interface).

Returns:
string containing the service status

9.42.3.3 void GestureRecognitionImpl::push_structured_event (const CosNotification::StructuredEvent & *event*) [virtual]

This method receives the events from the tracker.

Parameters:
event the incoming event

Reimplemented from **TemplateServiceWrapper** (p. 251).

9.42.3.4 void GestureRecognitionImpl::start ()

Starts the gesture recognition.

The recognition will stay active until you terminate it via the **stop**() (p. 142) method

The documentation for this class was generated from the following files:

- **gesturerecognition.h**
- **gesturerecognition.cpp**

9.43 DWARF::ImageFormat Struct Reference

This struct describes image format properties.

`import "Camera.idl";`

Public Attributes

- long **width**
  the width of the image in pixels.
- long **height**
  the height of the image in pixels.
- long **bytesPerRow**
  line width in bytes, including padding.
- short **bitsPerPixel**
  bits per pixel.

9.43.1 Detailed Description

This struct describes image format properties.

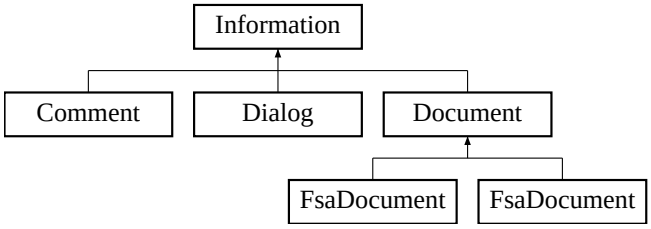
The documentation for this struct was generated from the following file:

- **Camera.idl**

9.44 Information Class Reference

The WorkflowEngine's main purpose is to deliver information to the client and thus we encapsulate this need in this information.

Inheritance diagram for Information::



Public Methods

- **Information** ()

  If an id is not known from the beginning this constructor instantiates it using a default id.

- **Information** (String id)

  This is the normal way to instantiate an Information object.

- void **setId** (String id)
- String **getId** ()
- String **getLabel** ()

  A GUI may get a label to display a link to the information.

- String **toString** ()

  Override Object's toStrig to produce nicer debug output.

Public Attributes

- String **URI**

  URI where a client can get the information.

9.44.1 Detailed Description

The WorkflowEngine's main purpose is to deliver information to the client and thus we encapsulate this need in this information.

Normally a Workflow refers to subclasses of Information like **Document** (p. 109) or **Dialog** (p. 106). The most important attriute in this class is the URI which should point to a file or other resource (like a database) where the information is located. Information instances are normally delivered to the application by DocumentEvents.

See also:
DocumentEvent (p. 110) , **DocumentListener** (p. 113)

Author:
Stefan Ri"s

Version:

Revision:
1.1

9.44.2 Constructor & Destructor Documentation

9.44.2.1 Information::Information () [inline]

If an id is not known from the beginning this constructor instantiates it using a default id.

It can be set later using the setId() method

The documentation for this class was generated from the following file:

- Information.java

9.45 InformationList Class Reference

Utility class for easier access to an InformationList.

Public Methods

- synchronized **Information getInformation** (Object key)

  This method is used to retrieve an information from the list.

- synchronized **Information putInformation** (Object key, **Information** value) throws NullPointerException

  This method puts information into the Hashtable.

- synchronized Enumeration **keys** ()

  Use this method to get a sorted list of keys.

9.45.1 Detailed Description

Utility class for easier access to an InformationList.

Essentially this is a wrapper for java.util.Hashtable. An InformationList is used to keep track of the documents belonging to a task or workflow but are also delivered to a **DocumentListener** (p. 113) as the source of a **DocumentEvent** (p. 110).

Implementation critique: It would have been better to use a private instance variable to refers to a Hashtable that to subclass it. On the other hand we did not need to provide methods like size() because we simply inherit them from Hashtable.

See also:
DocumentListener (p. 113) , **DocumentEvent** (p. 110)

Author:
Stefan Ri"s

Version:

Revision:
1.1

9.45.2 Member Function Documentation

9.45.2.1 synchronized Information InformationList::getInformation (Object *key*) [inline]

This method is used to retrieve an information from the list.

Parameters:
As a key any object may be used, but Strings are normally preferred.

Returns:
The information associated with the key.

The documentation for this class was generated from the following file:

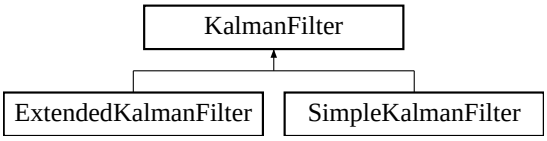
- InformationList.java

9.46 KalmanFilter Class Reference

a superclass for the EKF and SKF filters.

`#include <kalmanfilter.h>`

Inheritance diagram for KalmanFilter::



Public Methods

- virtual Matrix **update** (Matrix z, Matrix u)=0
  does measurement and prediction update according to measurement z.
- virtual const Matrix & **getX** ()=0

9.46.1 Detailed Description

a superclass for the EKF and SKF filters.

9.46.2 Member Function Documentation

9.46.2.1 virtual Matrix KalmanFilter::update (Matrix *z*, Matrix *u*) `[pure virtual]`

does measurement and prediction update according to measurement z.

Parameters:

z current measurement

u current process control

Returns:

updated estimate; use getX() to get new prediction

Reimplemented in **SimpleKalmanFilter** (p. 239), and **ExtendedKalmanFilter** (p. 122).

The documentation for this class was generated from the following file:

- **kalmanfilter.h**

9.47 LabeledIdTable Class Reference

This helper class associates ids with labels that may be used to represent the entity behind the id.

Public Methods

- Enumeration **keys** ()

  This methods returns an Enumeration of all ids in the table.

- void **addLabel** (String id, **LocalizedLabels** l)

  If there is a localiced label available you may add it to the table using this method.

- void **addLabel** (String id, String l)

  If the object associated with the id has only one label it may be put into the tabel using this method.

- String **getLabel** (String key, Locale l)

  This method is used to retrieve a Label using a Locale object and a key.

- String **getLabel** (String key)

  Use thjis method to retrieve a label if you do not know the current locale.

9.47.1 Detailed Description

This helper class associates ids with labels that may be used to represent the entity behind the id.

The label may be localized and the public methods of this class provide differnet ways to add and retrieve labels for a certain id. The method getCustomEvents() of the class FsaTask produces a LabeledIdTable in order to hide the Event objects behind the ids.

See also:
de.arvika.workflow.fsaWorkflow.FsaTask , **LocalizedLabels** (p. 152)

Author:
Stefan Ri"s

Version:

Revision:
1.1

9.47.2 Member Function Documentation

9.47.2.1 void LabeledIdTable::addLabel (String *id*, String *l*) [inline]

If the object associated with the id has only one label it may be put into the tabel using this method.

Parameters:
id identifies the label and the object the label is associated with
l is a String that describes the object.

9.47.2.2 void LabeledIdTable::addLabel (String *id*, LocalizedLabels *l*) [inline]

If there is a localiced label available you may add it to the table using this method.

Parameters:
id identifies the label and the object the label is associated with
l is a list of labels (**LocalizedLabels** (p. 152)) that is associated with different locales

9.47.2.3 String LabeledIdTable::getLabel (String *key*) [inline]

Use thjis method to retrieve a label if you do not know the current locale.

Parameters:
key This String identifies the label

Returns:
A label (String) that is associated with the key

9.47.2.4 String LabeledIdTable::getLabel (String *key*, Locale *l*) [inline]

This method is used to retrieve a Label using a Locale object and a key.

Parameters:
key This String identifies the label
l look for a label suitable for this locale

Returns:
A label (String) that is associated with the key and that belongs to the specified locale.

9.47.2.5 Enumeration LabeledIdTable::keys () [inline]

This methods returns an Enumeration of all ids in the table.

Returns:
Enumeration containing keys.

The documentation for this class was generated from the following file:

- LabeledIdTable.java

9.48 LocalizedLabels Class Reference

This class enables us to have a single object for each element in the workflow.

Public Methods

- void **addLabel** (String label, Locale loc)

  Add (p. 65) *a label and associate it with an existing locale.*

- void **addLabel** (String label, String countryCode)

  Add (p. 65) *a label using only the countrycode.*

- void **addLabel** (String label, String countryCode, String language)

  Add (p. 65) *a label using the countrycode and the language identifier in the case that a country code is not sufficient to describe the language.*

- void **addLabel** (String label)

  Add (p. 65) *a label without specifiying a locale.*

- String **getLabel** (Locale loc)

  Use this method to retrieve a label in a certain language by specifying its Locale.

- String **getLabel** ()

  Get the label associated with the default Locale.

9.48.1 Detailed Description

This class enables us to have a single object for each element in the workflow.

This class encapsulates the behaviour that is necessary to localize the labels that all describe the same object.

Author:
Stefan Ri"s

Version:

Revision:
1.1

9.48.2 Member Function Documentation

9.48.2.1 void LocalizedLabels::addLabel (String *label*) [inline]

Add (p. 65) a label without specifiying a locale.

Parameters:
label The label that will be added simply using the default locale

9.48.2.2 void LocalizedLabels::addLabel (String *label*, String *countryCode*, String *language*) [inline]

Add (p. 65) a label using the countrycode and the language identifier in the case that a country code is not sufficient to describe the language.

Parameters:

label The label in the language specified by the countrycode

countryCode A country identifier as defined in the JAVA Locale specification

language A language identifier as defined in the JAVA Locale specification

9.48.2.3 void LocalizedLabels::addLabel (String *label*, String *countryCode*) [inline]

Add (p. 65) a label using only the countrycode.

Parameters:

label The label in the language specified by the countrycode

countryCode A country identifier as defined in the JAVA Locale specification (examples: de, en)

9.48.2.4 void LocalizedLabels::addLabel (String *label*, Locale *loc*) [inline]

Add (p. 65) a label and associate it with an existing locale.

Parameters:

label The label in the language specified by the locale

loc The Locale object that describes the language of the label

9.48.2.5 String LocalizedLabels::getLabel () [inline]

Get the label associated with the default Locale.

Returns:

A Label.

9.48.2.6 String LocalizedLabels::getLabel (Locale *loc*) [inline]

Use this method to retrieve a label in a certain language by specifying its Locale.

Parameters:

loc A locale object that identifies the language of the required label.

Returns:

To make LocalizedLabels more robust this method never returns a null String. Instead it will return "Unknown Label" if it can find neither the label belonging to the specified locale nor one associated with the default locale.

The documentation for this class was generated from the following file:

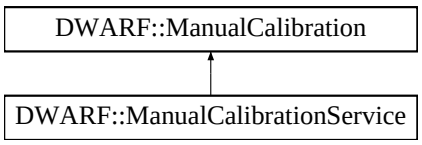
- LocalizedLabels.java

9.49 DWARF::ManualCalibration Interface Reference

This interface discribes the marker detection service.

`import "ManualCalibration.idl";`

Inheritance diagram for DWARF::ManualCalibration::



Public Methods

- void **calibrate** ()

  Calibrates The user has to look straight ahead when this method is called.

9.49.1 Detailed Description

This interface discribes the marker detection service.

The documentation for this interface was generated from the following file:

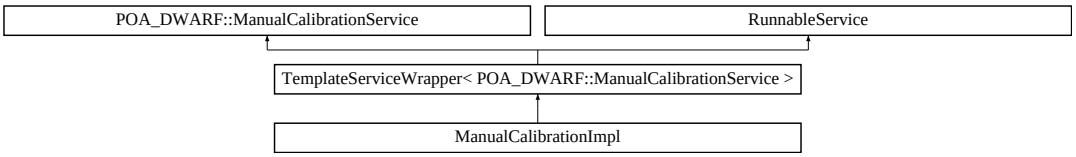
- **ManualCalibration.idl**

9.50 ManualCalibrationImpl Class Reference

the implementation of the ManualCalibration service.

`#include <ManualCalibration.h>`

Inheritance diagram for ManualCalibrationImpl::



Public Methods

- virtual void **describe** ()

  describes the service at the service manager.

- **ManualCalibrationImpl** ()

  This constructor is always needed as it is called by the ServiceRunnerWrapper.

- bool **mainLoop** ()

  the main service actions.

- char * **getName** ()
- char * **getStatus** ()
- void **calibrate** ()

  Calibrates The user has to look straight ahead when this method is called Gets the PoseData from the nmeatrackingmodule and publishes it to all recognized trackers.

9.50.1 Detailed Description

the implementation of the ManualCalibration service.

Author:
Andreas Krause

9.50.2 Member Function Documentation

9.50.2.1 bool ManualCalibrationImpl::mainLoop () `[virtual]`

the main service actions.

Returns:
true if service should continue to run Does nothing unless there was an unfulfilled calibration request, upon which it will try again after 10 seconds

Reimplemented from **RunnableService** (p. 233).

The documentation for this class was generated from the following files:

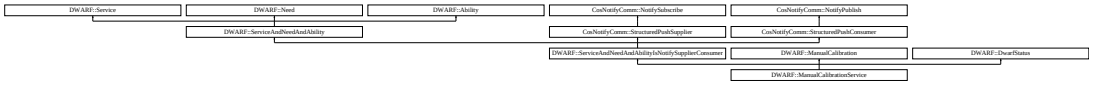
- **ManualCalibration.h**
- ManualCalibration.cpp

9.51 DWARF::ManualCalibrationService Interface Reference

This interface creates a **ManualCalibration** (p. 155) **DWARF** (p. 59) service.

```
import "ManualCalibration.idl";
```

Inheritance diagram for DWARF::ManualCalibrationService::



9.51.1 Detailed Description

This interface creates a **ManualCalibration** (p. 155) **DWARF** (p. 59) service.

To call it, please use the **ManualCalibration** (p. 155) interface

The documentation for this interface was generated from the following file:

- **ManualCalibration.idl**

9.52 DWARF::Markers::MarkerInfo Struct Reference

This struct is filled out when an optical marker is detected and sent in the body of a structured event.

`import "DetectMarkers.idl";`

Public Attributes

- long **area**
- long **id**
- long **dir**
- double **cf**
- double **pos** [2]
- double **line** [4][3]
- double **vertex** [4][2]
- Time **timeStamp**

  The time the marker was detected.

9.52.1 Detailed Description

This struct is filled out when an optical marker is detected and sent in the body of a structured event.

Only the "id" attribute is meaningful to an application that is only interested in the visibility of a marker. The rest is specific to the AR-Toolkit and only used internally.

The documentation for this struct was generated from the following file:

- **DetectMarkers.idl**

9.53 MathUtil Class Reference

implements common Math procedures.

Static Public Methods

- final double[] **quaternionToYawPitchRoll** (double[] quat, double[] ypr)
  REMARK: Entries in result are [yaw(-pi..pi), pitch(-pi/2..pi/2), roll(-pi..pi)].
- final double[] **yawPitchRollToQuaternion** (double[] ypr, double[] qResult)
  computes a quaternion from euler angles representing a rotation.
- final double[] **normalizeQuaternion** (double[] q)
  normalizes a quaternion to a unit quaternion.
- final double[] **multiplyQuaternion** (double[] q1, double[] q2, double[] qResult)
  multiply two quaternions.
- final double[] **rotateVector** (double[] q, double[] v, double[] vResult)
  rotates a vector by a quaternion.
- final double[] **invertQuaternion** (double[] q, double[] qResult)
  inverts a quaternion.
- double **getAngleToObservedObject** (double observerX, double observerY, double targetX, double targetY, double[] currentOrientation)
  calculates the angle between the observer's current line of sight and an object.
- void **main** (String[] args)

9.53.1 Detailed Description

implements common Math procedures.

This class is a BFH and contains only methods that are useful for the TRAMP1 demo, please redesign it ASAP.

9.53.2 Member Function Documentation

9.53.2.1 double MathUtil::getAngleToObservedObject (double *observerX*, double *observerY*, double *targetX*, double *targetY*, double *currentOrientation*[]) [inline, static]

calculates the angle between the observer's current line of sight and an object.

Author:
Martin Wagner (wagnerm@in.tum.de)

This method is designed specifically for the TRAMP1 demo and is primarily used to allow the VRMLViewto show an arrow that targets to an arbitrary object. The coordinate system is as follows: the x coordinate is headed eastbound, the y coordinate is headed northbound. If the current orientation is zero, the observer's line of sight is in the direction of the y axis.

Parameters:

observerX the observer's x coordinate

observerY the observer's y coordinate

targetX the target's x coordinate

targetY the target's y coordinate

currentOrientation double[4] quaternion with the current observer orientation

Returns:

the angle between the current line of sight and the line towards the target

9.53.2.2 final double [] MathUtil::invertQuaternion (double *q*[], double *qResult*[]) [inline, static]

inverts a quaternion.

Parameters:

q double[4] quaternion to be inverted

qResult double[4] resulting quaternion

Returns:

pointer to the result stored in qResult as well

9.53.2.3 final double [] MathUtil::multiplyQuaternion (double *q1*[], double *q2*[], double *qResult*[]) [inline, static]

multiply two quaternions.

Parameters:

q1 double[4] quaternion 1

q2 double[4] quaternion 2

qResult double[4] resulting quaternion

Returns:

pointer to the result stored in qResult as well

9.53.2.4 final double [] MathUtil::normalizeQuaternion (double *q*[]) [inline, static]

normalizes a quaternion to a unit quaternion.

Parameters:

q double[4] quaternion to be normalized

Returns:

pointer to the result stored in q as well

9.53.2.5 final double [] MathUtil::quaternionToYawPitchRoll (double *quat*[], double *ypr*[]) [inline, static]

REMARK: Entries in result are [yaw(-pi..pi), pitch(-pi/2..pi/2), roll(-pi..pi)].

Parameters:

qIn double[4] storing the quaternion

result double[3] result: yaw, pitch, roll

Returns:

pointer to result array

9.53.2.6 final double [] MathUtil::rotateVector (double *q*[], double *v*[], double *vResult*[]) [inline, static]

rotates a vector by a quaternion.

Parameters:

q double[4] quaternion that gives the rotation

v double[3] vector to be rotated

vResult double[4] resulting vecor

Returns:

pointer to the result stored in vResult as well

9.53.2.7 final double [] MathUtil::yawPitchRollToQuaternion (double *ypr*[], double *qResult*[]) [inline, static]

computes a quaternion from euler angles representing a rotation.

Parameters:

spherical double[3] with yaw, pitch, roll (in this sequence)

qResult double[4] where the result is stored

Returns:

pointer to result array

The documentation for this class was generated from the following file:

- **MathUtil.java**

9.54 MathUtils Class Reference

This class implements some utility classes for matrix and quaternion calculations.

`#include <MathUtils.h>`

Static Public Methods

- double * **axisAngleToQuaternion** (double *axisa, double *qResult)
  converts an axis angle representation into a quaternion.
- double * **eulerToQuaternion** (double roll, double pitch, double yaw, double *qResult)
  computes a quaternion from euler angles representing a rotation.
- double * **yawPitchRollToQuaternion** (const double *ypr, double *result)
  computes a quaternion from ypr/zxy euler angles representing a rotation.
- double * **quaternionToYawPitchRoll** (double *quat, double *ypr)
  computes ypr/zxy euler angles for a quaterion representing a rotation.
- double * **quaternionToEuler** (const double *q, double *result)
  Converts a quaternion to euler angles.
- double * **invertQuaternion** (double *q, double *qResult)
  inverts a quaternion.
- double * **matrixToQuaternion** (double matrix[3][3], double *qResult)
  converts a rotational matrix to a quaternion.
- double * **multiplyQuaternion** (double *q1, double *q2, double *qResult)
  multiplies two quaternions and stores result in a third.
- double * **normalizeQuaternion** (double *q)
  normalizes quaternion to unit length.
- double * **rotateVector** (double *q, double *v, double *vResult)
  rotates a vector using a given unit quaternion.

9.54.1 Detailed Description

This class implements some utility classes for matrix and quaternion calculations.

These are static methods, I just used a class to have a common place for such things, like in Java. These do not handle any memory issues, all arrays etc. have to be created and managed by the calling code.

Todo:
think about using double in implementation to make it more accurate.

Author:
Gerhard Reitmayr

9.54.2 Member Function Documentation

9.54.2.1 double * MathUtils::axisAngleToQuaternion (double * *axisa*, double * *qResult*) [static]

converts an axis angle representation into a quaternion.

Parameters:

axisa double[4] containing axis and angle in radiants

qResult double[4] where the result is stored

Returns:

pointer to result array

9.54.2.2 double * MathUtils::eulerToQuaternion (double *roll*, double *pitch*, double *yaw*, double * *qResult*) [static]

computes a quaternion from euler angles representing a rotation.

Parameters:

roll rotation around looking axis

pitch rotation around up axis

yaw rotation around side axis

qResult double[4] where the result is stored

Returns:

pointer to result array

9.54.2.3 double * MathUtils::invertQuaternion (double * *q*, double * *qResult*) [static]

inverts a quaternion.

Parameters:

q double[4] storing the quaternion

qResult double[4] where the result is stored

Returns:

pointer to result array

9.54.2.4 double * MathUtils::matrixToQuaternion (double *matrix*[3][3], double * *qResult*) [static]

converts a rotational matrix to a quaternion.

Parameters:

matrix double[3][3] storing the rotational matrix

qResult double[4] where the result is stored

Returns:

pointer to result array

9.54.2.5 double * MathUtils::multiplyQuaternion (double * *q1*, double * *q2*, double * *qResult*) [static]

multiplies two quaternions and stores result in a third.

Parameters:

q1 double[4] storing first quaternion

q2 double[4] storing second quaternion

qResult double[4] where the result is stored

Returns:

pointer to result array

9.54.2.6 double * MathUtils::normalizeQuaternion (double * *q*) [static]

normalizes quaternion to unit length.

Here the computation is done in place and the parameter is changed !

Parameters:

q double[4] storing quaternion

Returns:

pointer to result array

9.54.2.7 double * MathUtils::quaternionToEuler (const double * *q*, double * *result*) [static]

Converts a quaternion to euler angles.

Parameters:

q double[4] storing the quaternion

result double[3] result: roll, pitch, yaw

Returns:

pointer to result array

9.54.2.8 double * MathUtils::quaternionToYawPitchRoll (double * *quat*, double * *ypr*) [static]

computes ypr/zxy euler angles for a quaterion representing a rotation.

Parameters:

quat double[4] input quaternion

ypr double[3] yaw, pitch and roll angles

Returns:

pointer to ypr array

9.54.2.9 double * MathUtils::rotateVector (double * *q*, double * *v*, double * *vResult*) [static]

rotates a vector using a given unit quaternion.

It does not normalize the quaternion or check for unit length !

Parameters:

q double[4] storing quaternion

v double[3] storing vector

vResult double[3] where the result is stored

Returns:

pointer to result array

9.54.2.10 double * MathUtils::yawPitchRollToQuaternion (const double * *ypr*, double * *qResult*) [static]

computes a quaternion from ypr/zxy euler angles representing a rotation.

Parameters:

ypr double[3] yaw, pitch and roll angles

result double[4] where the result is stored

Returns:

pointer to result array

The documentation for this class was generated from the following files:

- **MathUtils.h**
- **MathUtils.cpp**

9.55 MeasureMap Struct Reference

defines the Mapping for each member of the PoseData struct defines the entries of the Model-Mapping.

`#include <posedatamodelmapping.h>`

Public Methods

- **MeasureMap** ()

Public Attributes

- bool **isMapped**

  default constructor.

- **KalmanFilter** ∗ **filter**

  true if PoseData entry is being mapped.

- vector< int > **trackerIndex**

  reference to the involved **KalmanFilter** (p. 149) *subclass.*

9.55.1 Detailed Description

defines the Mapping for each member of the PoseData struct defines the entries of the Model-Mapping.

The documentation for this struct was generated from the following file:

- **posedatamodelmapping.h**

9.56 MessageAlerter Class Reference

implements the MessageAlerter Service.

Public Methods

- **MessageAlerter** (String corbaloc)

  Constructor-method of **GarageServerApplication** (p. 133).

- String **getName** ()

  implements a function that returns the name of this service.

- String **getStatus** ()

  implements a function that returns the status of this service.

Static Public Methods

- void **main** (String args[])

Static Public Attributes

- final String **DEFLOCATION** = "corbaloc::localhost:39273/ServiceManager"

  the Default location where we expect the service manager to listen.

9.56.1 Detailed Description

implements the MessageAlerter Service.

9.56.2 Constructor & Destructor Documentation

9.56.2.1 MessageAlerter::MessageAlerter (String *corbaloc*) [inline]

Constructor-method of **GarageServerApplication** (p. 133).

Parameters:
corbaloc the url where we find the service manager

Author:
meierb

Date:
22.1.02

9.56.3 Member Function Documentation

9.56.3.1 String MessageAlerter::getName () [inline]

implements a function that returns the name of this service.

Precondition:

Postcondition:

Author:
Martin Wagner wagnerm@in.tum.de

Date:
29.1.02

Returns:
the service's name

9.56.3.2 String MessageAlerter::getStatus () [inline]

implements a function that returns the status of this service.

Precondition:

Postcondition:

Author:
Martin Wagner wagnerm@in.tum.de

Date:
29.1.02

Returns:
the service's status

9.56.3.3 void MessageAlerter::main (String *args*[]) [inline, static]

Parameters:
args the command line arguments

The documentation for this class was generated from the following file:

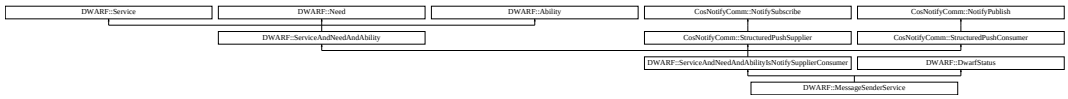
- **MessageAlerter.java**

9.57 DWARF::MessageSenderService Interface Reference

This interface creates a MessageSender **DWARF** (p. 59) service.

`import "MessageSender.idl";`

Inheritance diagram for DWARF::MessageSenderService::



9.57.1 Detailed Description

This interface creates a MessageSender **DWARF** (p. 59) service.

The documentation for this interface was generated from the following file:

- **MessageSender.idl**

9.58 MessageWindow Class Reference

implements the Message Windows for the **MessageAlerter** (p. 168) Service.

Public Methods

- **MessageWindow** (String message, MessageType msgType, long showTime)
- void **run** ()

  implements the main method of the class. it creates messageWindows and deletes them after some time.

Static Public Methods

- void **showMessage** (MessageAlert ma)

9.58.1 Detailed Description

implements the Message Windows for the **MessageAlerter** (p. 168) Service.

9.58.2 Constructor & Destructor Documentation

9.58.2.1 MessageWindow::MessageWindow (String *message*, MessageType *msgType*, long *showTime*) [inline]

Parameters:

message The Message to be shown

type Indeicates the icon next to the message @ showTime the time in ms the message will be shown

9.58.3 Member Function Documentation

9.58.3.1 void MessageWindow::run () [inline]

implements the main method of the class. it creates messageWindows and deletes them after some time.

Precondition:

Postcondition:

Author:

Martin Wagner wagnerm@in.tum.de

Date:

29.1.02

The documentation for this class was generated from the following file:

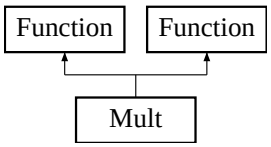
- **MessageWindow.java**

9.59 Mult Class Reference

Multiplication function composition.

`#include <function.h>`

Inheritance diagram for Mult::



Public Methods

- **Mult** (**Function** *f1, **Function** *f2)
- virtual ~**Mult** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** * **getFactor1** () const
- virtual **Function** * **getFactor2** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const
- **Mult** (**Function** *f1, **Function** *f2)
- virtual ~**Mult** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** * **getFactor1** () const
- virtual **Function** * **getFactor2** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const

9.59.1 Detailed Description

Multiplication function composition.

Author:
Andreas Krause

See also:
Fct (p. 123)

9.59.2 Member Function Documentation

9.59.2.1 virtual Function* Mult::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

9.59.2.2 Function * Mult::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

The documentation for this class was generated from the following files:

- **MovementSimulator/function.h**
- **TrackingFilter/function.h**
- MovementSimulator/function.cpp
- TrackingFilter/function.cpp

9.60 MultipleDisplayDocument Interface Reference

A **Document** (p. 109) that implements this interface allows a multiple display user interface to decide if the document is suited for the primary (usually head mounted) display.

Public Methods

- boolean **isPrimary** ()

9.60.1 Detailed Description

A **Document** (p. 109) that implements this interface allows a multiple display user interface to decide if the document is suited for the primary (usually head mounted) display.

Author:
Stefan Ri"s

Version:

Revision:
1.1

Deprecated:

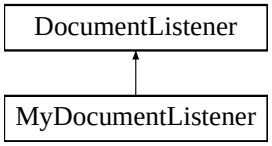
The documentation for this interface was generated from the following file:

- MultipleDisplayDocument.java

9.61 MyDocumentListener Class Reference

This interface defines the method a class has to implement so that it may be registered as a listener for DocumentEvents.

Inheritance diagram for MyDocumentListener::



Public Methods

- **MyDocumentListener** (MyTaskPossibilities mtp)
- void **showRequested** (**DocumentEvent** e)

  This method is called when a publisher requests a document to be shown.

- void **newDocuments** (**DocumentEvent** e)

  Whenever a workflow changes its current state or a new workflow is started it publishes a list of avilable methods.

- void **startSubWorkflow** (**DocumentEvent** e)

  The event publisher (e.g.

9.61.1 Detailed Description

This interface defines the method a class has to implement so that it may be registered as a listener for DocumentEvents.

Smilar to the **DocumentEvent** (p. 110) this class would also have been best to splitted into a DocumentListner and a DocumentListListener. In analogy to the AWT event model we provided an adapter with empty implementations of this interface's methods.

See also:
DocumentEvent (p. 110) , **DocumentListenerAdapter** (p. 115)

Author:
Stefan Ri"s

Version:

Revision:
1.3

9.61.2 Member Function Documentation

9.61.2.1 void MyDocumentListener::newDocuments (DocumentEvent *e*) [inline, virtual]

Whenever a workflow changes its current state or a new workflow is started it publishes a list of avilable methods.

Which eventually leads to a call of this method. **DocumentEvent** (p. 110) types: NEW_TASK_DOCUMENTS, WORKFLOW_DOCUMENTS

Reimplemented from **DocumentListener** (p. 114).

9.61.2.2 void MyDocumentListener::showRequested (DocumentEvent *e*) [inline, virtual]

This method is called when a publisher requests a document to be shown.

The listener then should take care of the document's display.

Reimplemented from **DocumentListener** (p. 114).

9.61.2.3 void MyDocumentListener::startSubWorkflow (DocumentEvent *e*) [inline, virtual]

The event publisher (e.g.

the WorkflowEngine) calls this method if a workflow issues a GoSub request. In this case the listner has not to handle the start of this subworkflow. This is normally done by the WorkflowController.

Reimplemented from **DocumentListener** (p. 114).

The documentation for this class was generated from the following file:

- MyDocumentListener.java

9.62 NMEAConfiguration Class Reference

An NMEAConfiguration stores all data needed to configure a **NMEATrackingModule** (p. 194). Since the main purpuse of this class is to wrapp configuration data, those data are simply stored in public attributes. They can be read and changed directly. For details see the discription of the specific attributes. NMEAConfiguration also provides the method **readConfigFile**() (p. 180), which alows to read the whole configuration or parts of it from a XML based configuration file. Therefore it implements the SAX2 interfaces ContentHandler and ErrorHandler.

`#include <NMEAConfiguration.h>`

Public Methods

- **NMEAConfiguration** ()

  Creates a new NMEAConfiguration.

- ∼**NMEAConfiguration** ()

  Default destructor.

- bool **readConfigFile** (char ∗configFile)

  Reads the NMEAConfiguration from a XML file.

- void **startElement** (const XMLCh ∗const uri, const XMLCh ∗const localname, const XMLCh ∗const qname, const Attributes &attrs)

  Handles the elements of the NMEA config file.

- void **endElement** (const XMLCh ∗const uri, const XMLCh ∗const localname, const XMLCh ∗const qname)

  Handles the elements of the NMEA config file.

- void **warning** (const SAXParseException &e)

  Handles warnings in the NMEA config file.

- void **error** (const SAXParseException &e)

  Handles errors in the NMEA config file.

- void **fatalError** (const SAXParseException &e)

  Handles fatal errors in the NMEA config file.

Public Attributes

- **FileWrapper** ∗ **inputSource**

  The input source for the NMEA data.

- UTM ∗ **targetCoordinateSystem**

  Handles the transformation of NMEA data to the desired coordinate system of the PoseData.

- double **positionAccuracy**

  Overrides the position accuracy of the GPS data.

- double **orientationAccuracy**

  Overrides the orientation accuracy of the GPS data.

9.62.1 Detailed Description

An NMEAConfiguration stores all data needed to configure a **NMEATrackingModule** (p. 194). Since the main purpuse of this class is to wrapp configuration data, those data are simply stored in public attributes. They can be read and changed directly. For details see the discription of the specific attributes. NMEAConfiguration also provides the method **readConfigFile**() (p. 180), which alows to read the whole configuration or parts of it from a XML based configuration file. Therefore it implements the SAX2 interfaces ContentHandler and ErrorHandler.

9.62.2 Constructor & Destructor Documentation

9.62.2.1 NMEAConfiguration::NMEAConfiguration ()

Creates a new NMEAConfiguration.

All parameters are set to default values.

9.62.2.2 NMEAConfiguration::~NMEAConfiguration ()

Default destructor.

Deletes this NMEAConfiguration and the inputSource and targetCoordinateSystem contained.

9.62.3 Member Function Documentation

9.62.3.1 void NMEAConfiguration::endElement (const XMLCh *const *uri*, const XMLCh *const *localname*, const XMLCh *const *qname*)

Handles the elements of the NMEA config file.

Needed to implement SAX2.

9.62.3.2 void NMEAConfiguration::error (const SAXParseException & *e*)

Handles errors in the NMEA config file.

Needed to implement SAX2.

9.62.3.3 void NMEAConfiguration::fatalError (const SAXParseException & *e*)

Handles fatal errors in the NMEA config file.

Needed to implement SAX2.

9.62.3.4 bool NMEAConfiguration::readConfigFile (char * *configFile*)

Reads the NMEAConfiguration from a XML file.

Any part of the configuration, appearing in the file will be set accordingly. The rest of the configuration will be left unchanged. This file should be valid against XMLConfiguration.dtd, which also contains detailed information on the use of the config file.

Parameters:

configFile A file containing data compliant to NMEAConfiguration.dtd.

9.62.3.5 void NMEAConfiguration::startElement (const XMLCh *const *uri*, const XMLCh *const *localname*, const XMLCh *const *qname*, const Attributes & *attrs*)

Handles the elements of the NMEA config file.

Needed to implement SAX2.

9.62.3.6 void NMEAConfiguration::warning (const SAXParseException & *e*)

Handles warnings in the NMEA config file.

Needed to implement SAX2.

9.62.4 Member Data Documentation

9.62.4.1 FileWrapper* NMEAConfiguration::inputSource

The input source for the NMEA data.

The class **FileWrapper** (p. 125) can be used for plain files as input. By deriving from **FileWrapper** (p. 125) more complex input sources can be specified. (e.g. serial ports, raw devices) There is now default value for the input source. It must always be set explicitly, to enable the **NMEATrackingModule** (p. 194) to work.

9.62.4.2 double NMEAConfiguration::orientationAccuracy

Overrides the orientation accuracy of the GPS data.

Positive values are applied as factor to the accuracy calculated from the NMEA data. The absolute value of a nagative number is used for total override. Default value is +1 which does not do anything.

Todo:

This is not used yet. Must be included in the specific implementations of MNEASentence.

9.62.4.3 double NMEAConfiguration::positionAccuracy

Overrides the position accuracy of the GPS data.

Positive values are applied as factor to the accuracy calculated from the NMEA data. The absolute value of a nagative number is used for total override. Default value is +1 which does not do anything.

Todo:
This is not used yet. Must be included in the specific implementations of MNEASentence.

9.62.4.4 UTM* NMEAConfiguration::targetCoordinateSystem

Handles the transformation of NMEA data to the desired coordinate system of the PoseData.

A UTMCoordinateSystem with automatic zone choise is the default value.

Todo:
Later a common baseclass for UTM and other coordinates should be included.

The documentation for this class was generated from the following files:

- **NMEAConfiguration.h**
- **NMEAConfiguration.cpp**

9.63 NMEAInterpreter Class Reference

This class is parser for NMEA data. It has setter methods to specify a input source, and a PoseData which shall be updated according to the NMEA data. Then the read method can be used to parse the next NMEA sentence, and update the PoseData. The implementation of this class is generated by lex. This tool takes care of all the token recognition, and makes the implementation easily adjustable to different NMEA sentences.

`#include <NMEAInterpreter.h>`

Public Methods

- **NMEAInterpreter** (FILE *input, PoseData *output, omni_mutex *outputSemaphore)

  Creates a new NMEAInterpreter, which will read input from a file stream, and write the interpreted output to a PoseData.

- **~NMEAInterpreter** ()

  Deletes this NMEAInterpreter.

- void **read** ()

  Parses the input file for the next available NMEA sentence.

- void **setInput** (FILE *input)

  Changes the file stream to read from.

- void **setOutput** (PoseData *output)

  Changes the PoseData to write the interpreted output to.

- void **setOutputSemaphore** (omni_mutex *outputSemaphore)

  Changes the semaphore used to protect the output PoseData.

- char * **getStatus** ()

  Gets statistical information about all the NMEA sentences received.

9.63.1 Detailed Description

This class is parser for NMEA data. It has setter methods to specify a input source, and a PoseData which shall be updated according to the NMEA data. Then the read method can be used to parse the next NMEA sentence, and update the PoseData. The implementation of this class is generated by lex. This tool takes care of all the token recognition, and makes the implementation easily adjustable to different NMEA sentences.

Todo:

No information about past or future NMEA Sentences is evaluated. Such a feature should be added, since there are some NMEA sentences consisting of more than one parts. Currently all NMEA sentences are handled singularily.

9.63.2 Constructor & Destructor Documentation

9.63.2.1 NMEAInterpreter::NMEAInterpreter (FILE * *input*, PoseData * *output*, omni_mutex * *outputSemaphore*)

Creates a new NMEAInterpreter, which will read input from a file stream, and write the interpreted output to a PoseData.

That PoseData is protected by a semaphor, to avoid invalid PoseDate to be read, during it is updated.

Parameters:

input A filestream containing the sentences to read.

output A PoseData object to write the interpreted NMEA data to.

outputSemaphore A semaphore to use for locking the PoseData.

9.63.3 Member Function Documentation

9.63.3.1 char* NMEAInterpreter::getStatus ()

Gets statistical information about all the NMEA sentences received.

Returns:

Status information about this NMEAInterpreter.

Todo:

Currently only a predefined status string is returned. Implement some usefull output here.

9.63.3.2 void NMEAInterpreter::read ()

Parses the input file for the next available NMEA sentence.

If a supported NMEA sentence is found, it is handled using a matching subclass of **NMEASentence** (p. 185). If the **NMEASentence** (p. 185) can be handled correctly, the output PoseData is updated according to its contents. Unknown or invalid NMEA sentences are ignored, and the output PoseData is left unchanged. This method is blocking. It waits for input, even if the inputFile does provide any currently. It always reads a whole line, before returning.

Todo:

Support for more NMEA sentences can easily be added later.

9.63.3.3 void NMEAInterpreter::setInput (FILE * *input*)

Changes the file stream to read from.

Parameters:

input The new file stream containing the sentences to read.

9.63.3.4 void NMEAInterpreter::setOutput (PoseData * *output*)

Changes the PoseData to write the interpreted output to.

Parameters:
output The new poseData for updating with the output interpreted.

9.63.3.5 void NMEAInterpreter::setOutputSemaphore (omni_mutex * *outputSemaphore*)

Changes the semaphore used to protect the output PoseData.

Parameters:
outputSemaphore The new semaphore for locking the PoseData.

The documentation for this class was generated from the following file:

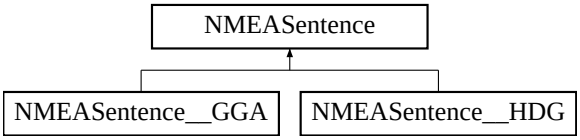
- **NMEAInterpreter.h**

9.64 NMEASentence Class Reference

Abstract class to represent the data of a NMEA sentence. Derive from this class to handle different types of NMEA sentences. This can be usefull as a helper class while parsing NMEA.

`#include <NMEASentence.h>`

Inheritance diagram for NMEASentence::



Public Methods

- **NMEASentence** (char *header)
  Initializes this NMEASentence for the given NMEA sentence header.
- void **nextParam** (char *delimiter)
  Increases the number of the current parameter.
- int **getParamNr** ()
  Gets the number of the current parameter.
- virtual bool **setParam** (char *paramVal)=0
  Sets the value of the current parameter.
- void **setChecksum** (char *checksum)
  Adds the checksum to the raw data of this NMEASentence, and resets the current parameter to -1.
- int **validateChecksum** ()
  Validates the current status of the checksum, based on the rawData.
- virtual void **updatePose** (PoseData *pose, omni_mutex *poseSemaphore)=0
  Applies the data of this NMEASentence to a PoseData.
- char * **getRawData** ()
  Gets the raw data represented by this NMEA sentence.
- virtual ostream & **print** (ostream &stream)=0
  An implementing method should prints the data represented by the specific NMEASentence to the given stream.

Protected Methods

- void **addRawData** (char *text)
  Adds the given string to the raw data of this NMEA sentence.

9.64.1 Detailed Description

Abstract class to represent the data of a NMEA sentence. Derive from this class to handle different types of NMEA sentences. This can be usefull as a helper class while parsing NMEA.

9.64.2 Constructor & Destructor Documentation

9.64.2.1 NMEASentence::NMEASentence (char * *header*)

Initializes this NMEASentence for the given NMEA sentence header.

A NMEA sentence header consists of "$" TalkerID SentenceID. However, the header data are not evalueated, but only used to remember the raw data of the NMEA sentence. A derived class should always call this constructor, and then initialize its parameters to default values. This is needed, since many parameters may be ommitted in a NMEA sentence.

Parameters:

header The raw data of the NMEA header. Usually the first 6 characters, including the leading "$".

9.64.3 Member Function Documentation

9.64.3.1 void NMEASentence::addRawData (char * *text*) [protected]

Adds the given string to the raw data of this NMEA sentence.

Parameters:

text The text fragment to add to the raw data.

9.64.3.2 int NMEASentence::getParamNr ()

Gets the number of the current parameter.

The method **setParam**() (p. 187) needs it, to find out which parameter is to be set.

Returns:

The number of the current parameter.

9.64.3.3 char * NMEASentence::getRawData ()

Gets the raw data represented by this NMEA sentence.

For any subclass implemented correctly, this should match exactly the data received from the input device.

Returns:

The raw data of this NMEASentence.

9.64.3.4 void NMEASentence::nextParam (char * *delimiter*)

Increases the number of the current parameter.

This will influence further calls of **setParam**() (p. 187).

Parameters:

delimiter The string which was used in the NMEASentence to delimit the next parameter from the last one (usuallly ","). This is necessary to remember the raw data of the NMEA sentence correctly.

9.64.3.5 virtual ostream& NMEASentence::print (ostream & *stream*) [pure virtual]

An implementing method should prints the data represented by the specific NMEASentence to the given stream.

Parameters:

stream The output stream to print to.

Returns:

The same stream given as parameter.

Reimplemented in **NMEASentence_GGA** (p. 190), and **NMEASentence_HDG** (p. 193).

9.64.3.6 void NMEASentence::setChecksum (char * *checksum*)

Adds the checksum to the raw data of this NMEASentence, and resets the current parameter to -1.

This avoids more parameters to be added, since the checksum is is always the last part of an NMEASentence.

Parameters:

checksum A string representing the checksum. Must be an "*" followed by a two digit hex number (e.g. "*B7"), or an empty string, if no checksum is given.

9.64.3.7 virtual bool NMEASentence::setParam (char * *paramVal*) [pure virtual]

Sets the value of the current parameter.

An implementing method should set its specific parameter, which corresponds to the current parameter number. The current parameter number is returned by **getParamNr**() (p. 186). Additionally it should use addRawData(paramVal), to support correct checksum validation.

Parameters:

paramVal A string representing the parameter to add.

Returns:

false, if a parameter with the number returned by getParam() is invalid, or another error occurred, otherwise true.

Reimplemented in **NMEASentence_GGA** (p. 190), and **NMEASentence_HDG** (p. 193).

9.64.3.8 virtual void NMEASentence::updatePose (PoseData * *pose*, omni_mutex * *poseSemaphore*) [pure virtual]

Applies the data of this NMEASentence to a PoseData.

An implementing method should update the given PoseData according to the values contained in the concrete NMEASentence. During the update, access to the Posedata should be locked, using the given semaphore.

Parameters:

pose The PoseData to update.

poseSemaphore The semaphore to use for locking the PoseData.

Reimplemented in **NMEASentence_GGA** (p. 190), and **NMEASentence_HDG** (p. 193).

9.64.3.9 int NMEASentence::validateChecksum ()

Validates the current status of the checksum, based on the rawData.

This only works, if all derived methods are adding the raw data correctly, using **addRawData**() (p. 186).

Returns:

0 if invalid, -1 if not validated, 1 if valid

Todo:

Implement this method! Currently it always returns -1.

The documentation for this class was generated from the following files:

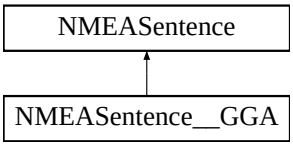
- **NMEASentence.h**
- **NMEASentence.cpp**

9.65 NMEASentence__GGA Class Reference

Represents a **NMEASentence** (p. 185) with the sentence ID GGA (Global Positioning Szstem Fix Data).

`#include <NMEASentence__GGA.h>`

Inheritance diagram for NMEASentence__GGA::



Public Methods

- **NMEASentence__GGA** (char *header)

  Creates a new NMEASentence__GGA for the given NMEA sentence header.

- virtual bool **setParam** (char *paramVal)

  Handles the current parameter according to the specification of an **NMEASentence** (p. 185) *of type GGA.*

- virtual void **updatePose** (PoseData *pose, omni_mutex *poseSemaphore)

  Updates the given PoseData with the GPS data received.

- virtual ostream & **print** (ostream &stream)

  Prints the data represented by this NMEASentence__GGA to the given stream.

9.65.1 Detailed Description

Represents a **NMEASentence** (p. 185) with the sentence ID GGA (Global Positioning Szstem Fix Data).

9.65.2 Constructor & Destructor Documentation

9.65.2.1 NMEASentence__GGA::NMEASentence__GGA (char * *header*)

Creates a new NMEASentence__GGA for the given NMEA sentence header.

Creating a NMEASentence__GGA only makes sense, if SentenceID is "GGA". However, the header data are not evalueated, but only used to remember the raw data of the message. All parameters of the new NMEASentence__GGA are set to default values. This is necessary, since an **NMEASentence** (p. 185) does not have to contain all parameters explicitly.

Parameters:

header The raw data of the NMEA header. Should match "$__GGA".

9.65.3 Member Function Documentation

9.65.3.1 ostream & NMEASentence__GGA::print (ostream & *stream*) [virtual]

Prints the data represented by this NMEASentence__GGA to the given stream.

The output will be a list of all parameters and their values.

Parameters:
stream The output stream to print to.

Returns:
The same stream given as parameter.

Reimplemented from **NMEASentence** (p. 187).

9.65.3.2 bool NMEASentence__GGA::setParam (char * *paramVal*) [virtual]

Handles the current parameter according to the specification of an **NMEASentence** (p. 185) of type GGA.

The given parameter is only set successfully, if it fits the syntax of the current parameter.

Parameters:
paramVal According to the value of **getParamNr**() (p. 186), the given paramVal is interpreted as follows:

If **getParamNr**() (p. 186) returns other values, the operation fails, and no parameter is set.

Returns:
true, if parameter was set successfully, otherwise false.

Todo:
Not all parameters are evaluated yet.

Reimplemented from **NMEASentence** (p. 187).

9.65.3.3 void NMEASentence__GGA::updatePose (PoseData * *pose*, omni_mutex * *poseSemaphore*) [virtual]

Updates the given PoseData with the GPS data received.

The precision of the PoseData is set according to the precision of the GPS data received. Orientation data is leaved unchanged, since GPS does not provide any.

Parameters:
pose The PoseData to update.

poseSemaphore The semaphore to use for locking the PoseData.

Todo:
The accuracy is not calculated from the GPS-data yet, but set to a fixed value.
Before the PoseData is updated the NMEA checksum might be checked. However, this is not that necessary on modern hardware.

Reimplemented from **NMEASentence** (p. 188).

The documentation for this class was generated from the following files:

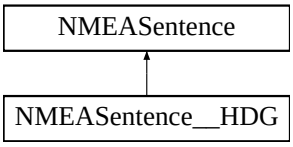
- **NMEASentence_GGA.h**
- **NMEASentence_GGA.cpp**

9.66 NMEASentence__HDG Class Reference

Represents a **NMEASentence** (p. 185) with the sentence ID HDG (Heading - Deviation & Variation).

`#include <NMEASentence__HDG.h>`

Inheritance diagram for NMEASentence__HDG::



Public Methods

- **NMEASentence__HDG** (char *header)

  Creates a new NMEASentence__HDG for the given NMEA sentence header.

- virtual bool **setParam** (char *paramVal)

  Handles the current parameter according to the specification of an **NMEASentence** (p. 185) *of type HDG.*

- virtual void **updatePose** (PoseData *pose, omni_mutex *poseSemaphore)

  Updates the given PoseData with the compass data received.

- virtual ostream & **print** (ostream &stream)

  Prints the data represented by this NMEASentence__HDG to the given stream.

9.66.1 Detailed Description

Represents a **NMEASentence** (p. 185) with the sentence ID HDG (Heading - Deviation & Variation).

9.66.2 Constructor & Destructor Documentation

9.66.2.1 NMEASentence__HDG::NMEASentence__HDG (char * *header*)

Creates a new NMEASentence__HDG for the given NMEA sentence header.

Creating a NMEASentence__HDG only makes sense, if SentenceID is "HDG". However, the header data are not evalueated, but only used to remember the raw data of the message. All parameters of the new NMEASentence__HDG are set to default values. This is necessary, since an **NMEASentence** (p. 185) does not have to contain all parameters explicitly.

Parameters:

header The raw data of the NMEA header. Should match "$__HDG".

9.66.3 Member Function Documentation

9.66.3.1 ostream & NMEASentence_HDG::print (ostream & *stream*) [virtual]

Prints the data represented by this NMEASentence_HDG to the given stream.

The output will be a list of all parameters and their values.

Parameters:
stream The output stream to print to.

Returns:
The same stream given as parameter.

Reimplemented from **NMEASentence** (p. 187).

9.66.3.2 bool NMEASentence_HDG::setParam (char * *paramVal*) [virtual]

Handles the current parameter according to the specification of an **NMEASentence** (p. 185) of type HDG.

The given parameter is only set successfully, if it fits the syntax of the current parameter.

Parameters:
paramVal According to the value of **getParamNr**() (p. 186), the given paramVal is interpreted as follows:

If **getParamNr**() (p. 186) returns other values, the operation fails, and no parameter is set.

Returns:
true, if parameter was set successfully, otherwise false.

Reimplemented from **NMEASentence** (p. 187).

9.66.3.3 void NMEASentence_HDG::updatePose (PoseData * *pose*, omni_mutex * *poseSemaphore*) [virtual]

Updates the given PoseData with the compass data received.

The precision of the orientation is set according to the precision of the compass data received. Position data is leaved unchanged, since a compass does not provide any.

Parameters:
poseSemaphore The semaphore to use for locking the PoseData.

Todo:
The accuracy is not calculated from the compas-data yet, but set to a fixed value.
Before the PoseData is updated the NMEA checksum might be checked. However, this is not that necessary on modern hardware.

Reimplemented from **NMEASentence** (p. 188).

The documentation for this class was generated from the following files:

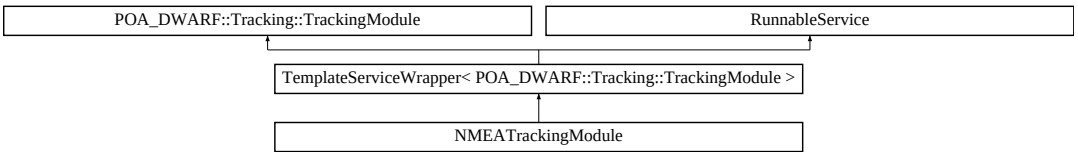
- **NMEASentence_HDG.h**
- **NMEASentence_HDG.cpp**

9.67 NMEATrackingModule Class Reference

This is the main class implementing the NMEA Tracking service. NMEATracking module is implemented as a **DWARF** (p. 59) service, and describes its needs & abilities according to the **DWARF** (p. 59) standard: a: PoseSource a: poseData a: status Since NMEATrackingModule implements the TrackingModule interface, its main purpuse is to calculate PoseData and deliver them to other **DWARF** (p. 59) services. Therefore it uses several helper classes to set up its configuration, parse an input source, interpret NMEA messages and transform them to PoseDate in a target coordinate system. All this is part of the method **mainLoop**() (p. 194) which has to be called frequently from a ServiceRunner. Additionally the method **getPosition**() (p. 196) can be invoke by remote **DWARF** (p. 59) services, to get the PoseData currently tracked.

`#include <nmeatrackingmodule.h>`

Inheritance diagram for NMEATrackingModule::



Public Methods

- **NMEATrackingModule** (char *fileName)

  Creates a new NMEATracking module, and sets up the nmeaInput and the nmeaInterpreter needed.

- **~NMEATrackingModule** ()

  Frees all used resources, especially the nmeaInput and the nmeaInterpreter used.

- bool **mainLoop** ()

  The method **mainLoop***()* (p. 194) *is abstract in the template and must be concretized as it carries out the main service actions (as long as it is not pushed - for push reaction, we have to redefine the virtual push method.*

- PoseData **getPosition** (const Time &time)

  Gets the PoseData calculated by this NMEATrackingModule.

- PoseSourceCapabilities **getCapabilities** ()

  Gets the capabilities of this NMEATracking module.

- void **calibrate** (const PoseData &currentposition)

  Needed to implement the PoseSource interface.

- virtual void **describe** ()

  Describes this NMEATrackingModule as a **DWARF** (p. 59) *service: This NMAETracking-Module calls itself "nmeatrackingmodule", and has the abilities "PoseSource", "PoseData" and "status".*

- char * **getName** ()

  Implements the status interface of Dwarf.

- char * **getStatus** ()

  Implements the status interface of **DWARF** (p. 59).

9.67.1 Detailed Description

This is the main class implementing the NMEA Tracking service. NMEATracking module is implemented as a **DWARF** (p. 59) service, and describes its needs & abilities according to the **DWARF** (p. 59) standard: a: PoseSource a: poseData a: status Since NMEATrackingModule implements the TrackingModule interface, its main purpuse is to calculate PoseData and deliver them to other **DWARF** (p. 59) services. Therefore it uses several helper classes to set up its configuration, parse an input source, interpret NMEA messages and transform them to PoseDate in a target coordinate system. All this is part of the method **mainLoop**() (p. 194) which has to be called frequently from a ServiceRunner. Additionally the method **getPosition**() (p. 196) can be invoke by remote **DWARF** (p. 59) services, to get the PoseData currently tracked.

Todo:

Currently the PoseDate ability is turned of, so an NMEATrackingModule does not push PoseData by itself, but only delivers them on request.

9.67.2 Member Function Documentation

9.67.2.1 void NMEATrackingModule::calibrate (const PoseData & *currentposition*)

Needed to implement the PoseSource interface.

Does nothing, since the NMEATracker cannot be calibrated.

9.67.2.2 void NMEATrackingModule::describe () [virtual]

Describes this NMEATrackingModule as a **DWARF** (p. 59) service: This NMAETrackingModule calls itself "nmeatrackingmodule", and has the abilities "PoseSource", "PoseData" and "status".

Todo:

Currently the PoseDate ability is turned of, so an NMEATrackingModule does not push PoseData by itself, but only delivers them on request.

Reimplemented from **RunnableService** (p. 230).

9.67.2.3 PoseSourceCapabilities NMEATrackingModule::getCapabilities ()

Gets the capabilities of this NMEATracking module.

This discribes its update frequency, its type (absolute or relative) and its precision in position and orientation.

Returns:

The PoseSourceCapabilities of this NMEATrackingModule.

9.67.2.4 char * NMEATrackingModule::getName ()

Implements the status interface of Dwarf.

Returns:
is always "nmeatrackingmodule".

9.67.2.5 PoseData NMEATrackingModule::getPosition (const Time & *t*)

Gets the PoseData calculated by this NMEATrackingModule.

Parameters:
time The time for which the position in desired.

Returns:
The PoseData at the desired time.

Todo:
The given time is not evaluated yet. Instead the latest PoseData is always returned.

9.67.2.6 char * NMEATrackingModule::getStatus ()

Implements the status interface of **DWARF** (p. 59).

Returns:
A brief discription of this NMEATrackingModule's status.

Todo:
Make the output more individual according to the current status. Currently just a fixed message is used.

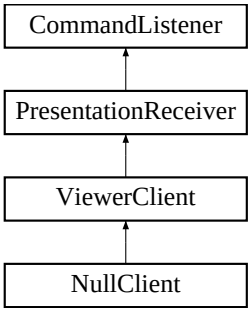
The documentation for this class was generated from the following files:

- **nmeatrackingmodule.h**
- **nmeatrackingmodule.cpp**

9.68 NullClient Class Reference

The NullClient Class is similar with **ViewerClient** (p. 270), the only difference is, that a null Layout is used for the chat.

Inheritance diagram for NullClient::



Public Methods

- void **init** ()

  This method initializes the applet and creates the User Interface.

9.68.1 Detailed Description

The NullClient Class is similar with **ViewerClient** (p. 270), the only difference is, that a null Layout is used for the chat.

The documentation for this class was generated from the following file:

- NullClient.java

9.69 ObjectExporter Class Reference

The Objects exporter is a **DWARF** (p. 59) wrapper for the ability to export references to (any) CORBA objects to interested clients.

`#include <objectexporter.h>`

Public Methods

- **ObjectExporter** (Object exportObj)

  Creates new ObjectExporter.

- void **connectAbility** (const char *offername, Connector_ptr conn)

  StructuredPushSupplier interface callback fuction for the EventService.

- void **disconnectAbility** (const char *offername, Connector_ptr conn)

  StructuredPushSupplier interface callback fuction fr the EventService.

- **ObjectExporter** (org.omg.CORBA.Object exportObj)

  Creates a ObjectExporter.

- void **connectAbility** (String offername, Connector conn)

  DWARF.Ability interface callback fuction for the Service Manager.

- void **disconnectAbility** (String offername, Connector conn)

  DWARF.Ability interface callback fuction for the Service Manager.

Public Attributes

- Object **exportObj**
- final String **ID** = "$Id: ObjectExporter.java,v 1.4 2001/12/19 11:16:53 wundsam Exp $"

  Our ID in the CVS:
  Id:
  ObjectExporter.java,v 1.4 2001/12/19 11:16:53 wundsam Exp

  .

9.69.1 Detailed Description

The Objects exporter is a **DWARF** (p. 59) wrapper for the ability to export references to (any) CORBA objects to interested clients.

Author:
Oliver Strutynski

9.69.2 Constructor & Destructor Documentation

9.69.2.1 ObjectExporter::ObjectExporter (org.omg.CORBA.Object *exportObj*) [inline]

Creates a ObjectExporter.

Parameters:

exportObj the Object to be exported

9.69.3 Member Function Documentation

9.69.3.1 void ObjectExporter::connectAbility (String *offername*, Connector *conn*) [inline]

DWARF.Ability interface callback fuction for the Service Manager.

Parameters:

offername a String containing the name of the offer

conn The connector you want to give to me.

9.69.3.2 void ObjectExporter::connectAbility (const char * *offername*, Connector_ptr *conn*)

StructuredPushSupplier interface callback fuction for the EventService.

Parameters:

offername a String containing the name of the offer

conn The connector you want to give to me.

9.69.3.3 void ObjectExporter::disconnectAbility (String *offername*, Connector *conn*) [inline]

DWARF.Ability interface callback fuction for the Service Manager.

Parameters:

offername a String containing the name of the offer

conn The connector you want todisconnect.

9.69.3.4 void ObjectExporter::disconnectAbility (const char * *offername*, Connector_ptr *conn*)

StructuredPushSupplier interface callback fuction fr the EventService.

Parameters:

offername a String containing the name of the offer

conn The connector you want todisconnect.

The documentation for this class was generated from the following files:

- **objectexporter.h**
- ObjectExporter.java
- objectexporter.cpp

9.70 ObjectImporter Class Reference

An Object Importers allows an application to gather other references exported by other services, and to use them.

`#include <objectimporter.h>`

Public Methods

- **ObjectImporter** ()

  Creates new ObjectImporter.

- void **connectNeed** (const char *accepternam*e, CORBA::Short reqInstance, Connector_ptr conn)

  StructuredPushSupplier interface callback fuction for the EventService.

- void **disconnectNeed** (const char *accepternam*e, CORBA::Short reqInstance, Connector_ptr conn)

  StructuredPushSupplier interface callback fuction for the EventService.

- int **size** ()

  methods for accessing the CORBA object vector.

- Object_var **get** (int)

  returns the vector size.

- **ObjectImporter** ()

  Creates new ObjectImporter.

- int **size** ()

  returns the number of Objects we know about.

- org.omg.CORBA.Object **get** (int i)

  returns the ith Object from the Vector of imported Objects NOTE: This is not a very bright way to do this (list may change while being accessed).

- void **connectNeed** (String accepternam*e*, short reqInstance, Connector conn)

  DWARF.Need interface callback fuction for the Service Manager.

- void **disconnectNeed** (String accepternam*e*, short reqInstance, Connector conn)

  DWARF.Need interface callback fuction for the Service Manager.

Public Attributes

- final String **ID** = "$Id: ObjectImporter.java,v 1.5 2001/12/19 11:16:53 wundsam Exp $"

  Our ID in the CVS:
  Id:
  ObjectImporter.java,v 1.5 2001/12/19 11:16:53 wundsam Exp
  .

9.70.1 Detailed Description

An Object Importers allows an application to gather other references exported by other services, and to use them.

Author:
Jan-Gregor Fischer

9.70.2 Constructor & Destructor Documentation

9.70.2.1 ObjectImporter::ObjectImporter ()

Creates new ObjectImporter.

Author:
fischerj @revision

Revision:
1.5

9.70.3 Member Function Documentation

9.70.3.1 void ObjectImporter::connectNeed (String *accepternam*e, short *reqInstance*, Connector *conn*) [inline]

DWARF.Need interface callback fuction for the Service Manager.

adds the supplied Object to the Vector of imported Objects.

Parameters:
***accepternam*e** a String containing the name of the accepter

conn The connector I want to give to you.

9.70.3.2 void ObjectImporter::connectNeed (const char * *accepternam*e, CORBA::Short *reqInstance*, Connector_ptr *conn*)

StructuredPushSupplier interface callback fuction for the EventService.

Parameters:
***accepternam*e** a String containing the name of the accepter

conn The connector I want to give to you.

9.70.3.3 void ObjectImporter::disconnectNeed (String *accepternam*e, short *reqInstance*, Connector *conn*) [inline]

DWARF.Need interface callback fuction for the Service Manager.

called when an object becomes unavaillable. Removes that object from the vector of imported objects.

Parameters:

accepternname a String containing the name of the accepter

conn The connector I want to disconnect.

9.70.3.4 void ObjectImporter::disconnectNeed (const char * *accepternname*, CORBA::Short *reqInstance*, Connector_ptr *conn*)

StructuredPushSupplier interface callback fuction for the EventService.

Parameters:

accepternname a String containing the name of the accepter

conn The connector I want to disconnect.

9.70.3.5 org.omg.CORBA.Object ObjectImporter::get (int *i*) [inline]

returns the ith Object from the Vector of imported Objects NOTE: This is not a very bright way to do this (list may change while being accessed).

So will will have to change this after christmas.

Parameters:

i the index of the Object to be returned

Returns:

the imported Objects

9.70.3.6 int ObjectImporter::size () [inline]

returns the number of Objects we know about.

Returns:

s.a.

The documentation for this class was generated from the following files:

- **objectimporter.h**
- ObjectImporter.java
- objectimporter.cpp

9.71 PersistentDataStorage Class Reference

public class PersistentDataStorage extends PersistentDataStoragePOA {.

9.71.1 Detailed Description

public class PersistentDataStorage extends PersistentDataStoragePOA {.

/** GLOBAL variables and constants

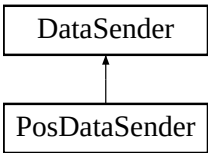
The documentation for this class was generated from the following file:

- **PersistentDataStorage.java**

9.72 PosDataSender Class Reference

`#include <posdatasender.h>`

Inheritance diagram for PosDataSender::



Public Methods

- **PosDataSender** (string name)
- virtual void **push** (DWARF::PositionEvent ev)

  Push an PositionEvent to your consumer; if the EventService did not provide a consumer so far, all events are simply ignored.

9.72.1 Detailed Description

Author:
Oliver Strutynski

9.72.2 Member Function Documentation

9.72.2.1 void PosDataSender::push (DWARF::PositionEvent *ev*) [virtual]

Push an PositionEvent to your consumer; if the EventService did not provide a consumer so far, all events are simply ignored.

Parameters:
ev The PositionEvent

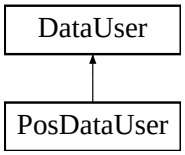
The documentation for this class was generated from the following files:

- **posdatasender.h**
- **posdatasender.cpp**

9.73 PosDataUser Class Reference

`#include <posdatauser.h>`

Inheritance diagram for PosDataUser::



Public Methods

- **PosDataUser** ()
- virtual void **event** (const CosNotification::StructuredEvent &event)

  PosDataSender::event method receives a structured event and from the channel.

9.73.1 Detailed Description

Author:
Oliver Strutynski

The documentation for this class was generated from the following files:

- **posdatauser.h**
- posdatauser.cpp

9.74 PoseDataModelMapping Class Reference

wrapper to make the PoseData struct 'intelligent' it combines each member of PoseData with a **KalmanFilter** (p. 149) if Trackers providing the variable are available.

`#include <posedatamodelmapping.h>`

Public Methods

- **PoseDataModelMapping** ()
  posedatamodelmapping.cpp.
- PoseData **getPoseData** (Matrix &trackerStates)
  updates the mapped KalmanFilters and returns the new state estimate.
- void **clear** ()
  resets (clears) the mapping and destroys the KalmanFilters.
- void **setNewModel** (int pos, **MeasureMap** &map)
  sets a new entry to the mVars array.
- bool **getIsMapped** (int pos)
  returns the "mappedness" of the PoseData entries.

9.74.1 Detailed Description

wrapper to make the PoseData struct 'intelligent' it combines each member of PoseData with a **KalmanFilter** (p. 149) if Trackers providing the variable are available.

Author:
Andreas Krause

9.74.2 Constructor & Destructor Documentation

9.74.2.1 PoseDataModelMapping::PoseDataModelMapping ()

posedatamodelmapping.cpp.

Author:
Andreas Krause Traveling Repair And Maintenance Platform - tramp.globalse.org

A SAX based XML parser for parsing and constructing the filter<->PoseData mapping

Id:
posedatamodelmapping.cpp,v 1.16 2002/02/05 19:41:09 wagnerm Exp

Revision:
1.16

Implementation of class: PoseDataModelMapping

9.74.3 Member Function Documentation

9.74.3.1 bool PoseDataModelMapping::getIsMapped (int *pos*) [inline]

returns the "mappedness" of the PoseData entries.

Parameters:
pos the mapped variable -> enum values

Returns:
true if specified variable is mapped

9.74.3.2 PoseData PoseDataModelMapping::getPoseData (Matrix & *trackerStates*)

updates the mapped KalmanFilters and returns the new state estimate.

Parameters:
trackerStates the matrix containing the tracker input

Returns:
the new PoseData estimate

9.74.3.3 void PoseDataModelMapping::setNewModel (int *pos*, MeasureMap & *map*)

sets a new entry to the mVars array.

Parameters:
pos the mapped variable -> enum values
map the **MeasureMap** (p. 167) to be copied

The documentation for this class was generated from the following files:

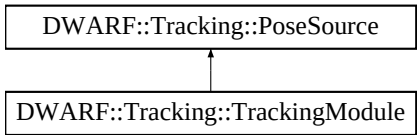
- **posedatamodelmapping.h**
- posedatamodelmapping.cpp

9.75 DWARF::Tracking::PoseSource Interface Reference

This interface is implemented by all tracking modules and implements a pull-style tracker interface.

`import "Tracker.idl";`

Inheritance diagram for DWARF::Tracking::PoseSource::



Public Methods

- **PoseSourceCapabilities getCapabilities** ()
- PoseData **getPosition** (in Time t)

  read the current position.

- void **calibrate** (in PoseData currentposition)

  set the current position (for relative trackers).

9.75.1 Detailed Description

This interface is implemented by all tracking modules and implements a pull-style tracker interface.

9.75.2 Member Function Documentation

9.75.2.1 PoseData DWARF::Tracking::PoseSource::getPosition (in Time *t*)

read the current position.

Parameters:

time The time for which the position in desired. Only used for prediction filters, so check the time member of the returned position!

The documentation for this interface was generated from the following file:

- **Tracker.idl**

9.76 DWARF::Tracking::PoseSourceCapabilities Struct Reference

Description of the tracker capabilities.

`import "Tracker.idl";`

Public Attributes

- unsigned long **minUpdateInterval**
  the best-case update interval in ms.
- TrackerType **positionType**
  The type of tracker.
- TrackerType **orientationType**
- double **positionPrecision** [3]
  standard deviation per axis in meters, negative if not available.
- double **orientationPrecision** [3]
  standard deviation per axis in radians, negative if not available.

9.76.1 Detailed Description

Description of the tracker capabilities.

The documentation for this struct was generated from the following file:

- **Tracker.idl**

9.77 PositionFactory Class Reference

Only a helper class to worry with the Java Date&Time instead o using PositinEvent directly.

Static Public Methods

- PositionEvent **getPositionEvent** (int thingId, int parentId, short hopcount, boolean hasRotation, boolean hasTranslation, boolean hasPose, double[] translation, double[] rotation, double[] pose, double accuracy, int lagUsec)

  Create a PositionEvent from the specified parameters.

- PositionEvent **getNullPositionEvent** ()

  Create a PositionEvent containing no rotation and no translation of the thing with ID 0, in the coordinate system 0.

- StructuredEvent **getStructuredEvent** (PositionEvent pe)

  Pack a PositionEvent into a CORBA structured event.

- StructuredEvent **getStructuredEvent** (int thingId, int parentId, short hopcount, boolean hasRotation, boolean hasTranslation, boolean hasPose, double[] translation, double[] rotation, double[] pose, double accuracy, int lagUsec)

  Create a StructuredEvent from the specified parameters by creating a PositionEvent first and then packing it into a StructuredEvent.

9.77.1 Detailed Description

Only a helper class to worry with the Java Date&Time instead o using PositinEvent directly.

Version:
1.0

Author:
Martin Bauer

9.77.2 Member Function Documentation

9.77.2.1 PositionEvent PositionFactory::getNullPositionEvent () `[inline, static]`

Create a PositionEvent containing no rotation and no translation of the thing with ID 0, in the coordinate system 0.

Returns:
The generated default PositionEvent

9.77.2.2 PositionEvent PositionFactory::getPositionEvent (int *thingId*, int *parentId*, short *hopcount*, boolean *hasRotation*, boolean *hasTranslation*, boolean *hasPose*, double *translation*[], double *rotation*[], double *pose*[], double *accuracy*, int *lagUsec*) [inline, static]

Create a PositionEvent from the specified parameters.

Parameters:

thingId The ID of the thing.

parentId The parent ID of the things coordinate system.

hopcount The hopcount, how many preprocesing steps the Event went through already.

hasRotation If the PositionEvent contains a valid rotation. This does **not** mean that the rotation vector can be invalid; use something like {1,0,0,0} if you dont know the rotation!

hasTranslation If the PositionEvent contains a valid Translation. This does **not** mean that the translation vector can be invalid; use {0,0,0} if you dont know the translation!

hasPose If the PositionEvent contaiins a valid pose Matrix.

translation The translation vector. This is the vector {x,y,z}

rotation The rotation vector; This is the vector {x,y,z,alpha} in rodriguez coordinates.

pose The 4x4 pose matrix of homogeneous coordinates.

accuracy The accuracy of the Position estimation.

lagUsec The internal lag of the event.

Returns:

The generated PositionEvent

9.77.2.3 StructuredEvent PositionFactory::getStructuredEvent (int *thingId*, int *parentId*, short *hopcount*, boolean *hasRotation*, boolean *hasTranslation*, boolean *hasPose*, double *translation*[], double *rotation*[], double *pose*[], double *accuracy*, int *lagUsec*) [inline, static]

Create a StructuredEvent from the specified parameters by creating a PositionEvent first and then packing it into a StructuredEvent.

Parameters:

thingId The ID of the thing.

parentId The parent ID of the things coordinate system.

hopcount The hopcount, how many preprocesing steps the Event went through already.

hasRotation If the PositionEvent contains a valid rotation. This does **not** mean that the rotation vector can be invalid; use something like {1,0,0,0} if you dont know the rotation!

hasTranslation If the PositionEvent contains a valid Translation. This does **not** mean that the translation vector can be invalid; use {0,0,0} if you dont know the translation!

hasPose If the PositionEvent contaiins a valid pose Matrix.

translation The translation vector. This is the vector {x,y,z}

rotation The rotation vector; This is the vector {x,y,z,alpha} in rodriguez coordinates.

pose The 4x4 pose matrix of homogeneous coordinates.

accuracy The accuracy of the Position estimation.

lagUsec The internal lag of the event.

Returns:

The generated StructuredEvent

9.77.2.4 StructuredEvent PositionFactory::getStructuredEvent (PositionEvent *pe*) [inline, static]

Pack a PositionEvent into a CORBA structured event.

Parameters:
pe The PositionEvent

Returns:
The StructuredEvent

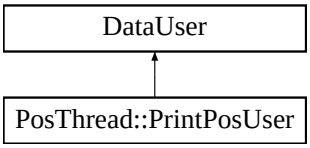
The documentation for this class was generated from the following file:

- PositionFactory.java

9.78 PosThread::PrintPosUser Class Reference

— this prints PositionEvent(of the **DemoService** (p. 96)-sender), not PoseData — public class **PrintPosUser** (p. 214) implements **DataUser** (p. 94) { public void **event**(StructuredEvent se) (p. 214) { // Extract our event PositionEvent pose = PositionEventHelper.extract(se.remainder_of_body);.

Inheritance diagram for PosThread::PrintPosUser::



Public Methods

- void **event** (StructuredEvent se)

  Callback function an external class calls to push a PositionEvent.

9.78.1 Detailed Description

— this prints PositionEvent(of the **DemoService** (p. 96)-sender), not PoseData — public class **PrintPosUser** (p. 214) implements **DataUser** (p. 94) { public void **event**(StructuredEvent se) (p. 214) { // Extract our event PositionEvent pose = PositionEventHelper.extract(se.remainder_of_body);.

// Just print some info System.out.println("Position-Event occured: Thing " + pose.thing + ", Pos: " + pose.pos); area.setText(area.getText() + "Position-Event occured: Thing " + pose.thing + ", Pos: " + pose.pos + "

"); } };

9.78.2 Member Function Documentation

9.78.2.1 void PosThread::PrintPosUser::event (StructuredEvent *se*) [inline]

Callback function an external class calls to push a PositionEvent.

Parameters:

se The PositionEvent

Reimplemented from **DataUser** (p. 95).

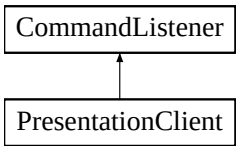
The documentation for this class was generated from the following file:

- PosThread.java

9.79 PresentationClient Class Reference

a simple exmaple for a Presentation Client that uses the Client and Server classes.

Inheritance diagram for PresentationClient::



Public Methods

- **PresentationClient** (String host, int port)
- String **commandReceived** (int id, String command)

  server or client receives a command.

- void **newSlide** (URL url)

  the servers says, that a new slide is now being displayed.

9.79.1 Detailed Description

a simple exmaple for a Presentation Client that uses the Client and Server classes.

9.79.2 Member Function Documentation

9.79.2.1 String PresentationClient::commandReceived (int *id*, String *command*) [inline]

server or client receives a command.

Parameters:

id the type of command

command the command as a String

Returns:

the return String that is sent as a response

Reimplemented from **CommandListener** (p. 77).

9.79.2.2 void PresentationClient::newSlide (URL *url*) [inline]

the servers says, that a new slide is now being displayed.

Parameters:

url where the slide can be retrieved

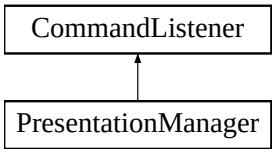
The documentation for this class was generated from the following file:

- PresentationClient.java

9.80 PresentationManager Class Reference

this is the main class of the GSE Presentaion Managaer module It is conncted to all remote clients, informs them about changes status and receives the intrrupting questions from them.

Inheritance diagram for PresentationManager::



Public Methods

- **PresentationManager** (Properties props) throws Exception
  parse the Properties and launch the Server.
- synchronized String **commandReceived** (int id, String command)
  server or client receives a command.

Static Public Methods

- void **main** (String[] args) throws Exception
  reads the iniFile intro a Properties.

9.80.1 Detailed Description

this is the main class of the GSE Presentaion Managaer module It is conncted to all remote clients, informs them about changes status and receives the intrrupting questions from them.

9.80.2 Member Function Documentation

9.80.2.1 synchronized String PresentationManager::commandReceived (int *id*, String *command*) [inline]

server or client receives a command.

Parameters:

id the type of command

command the command as a String

Returns:

the return String that is sent as a response

Reimplemented from **CommandListener** (p. 77).

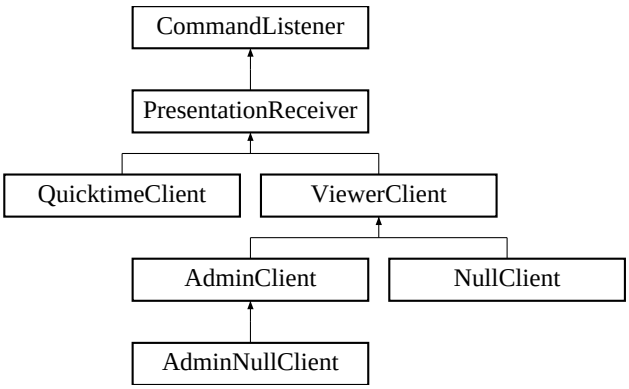
The documentation for this class was generated from the following file:

- PresentationManager.java

9.81 PresentationReceiver Class Reference

the PresentationReceiver, the superclass for all the different clients.

Inheritance diagram for PresentationReceiver::



Public Methods

- String **commandReceived** (int id, String command)

  server or client receives a command.

Protected Methods

- void **connect** (String host, int port)

  connect to the server that is running on this host and port.

- void **disconnect** ()

  kill the **ProtocolHandler** (p. 222) *-> disconnect from Server.*

- void **newSlide** (URL url)

  the servers says, that a new slide is now being displayed the applet shows the URL in the frame with the name "slides".

- void **updateStreams** (StreamURLManager m)

  to implement by derived classes, there are new stream urls available.

- synchronized void **askQuestion** (String text)

  the user asks a question.

- void **newQuestion** (String text)

  any client has asked a question.

- void **executeCommand** (int commandId, String command)

  this action will be delegated to the ProtocHandler.

- void **newMessage** (String text)

this is a general message to be maybe printed to a screen, if wanted.

- void **privateMessage** (String person, String text)

  this is a private message for anybody with this name it should be printed to screen, but only for this person.

9.81.1 Detailed Description

the PresentationReceiver, the superclass for all the different clients.

9.81.2 Member Function Documentation

9.81.2.1 synchronized void PresentationReceiver::askQuestion (String *text*) [inline, protected]

the user asks a question.

It will be sent to the server

9.81.2.2 String PresentationReceiver::commandReceived (int *id*, String *command*) [inline]

server or client receives a command.

Parameters:

id the type of command

command the command as a String

Returns:

the return String that is sent as a response

Reimplemented from **CommandListener** (p. 77).

9.81.2.3 void PresentationReceiver::newQuestion (String *text*) [inline, protected]

any client has asked a question.

This question will now be answered to be implemented by the **AdminClient** (p. 67)

Reimplemented in **AdminClient** (p. 68).

9.81.2.4 void PresentationReceiver::newSlide (URL *url*) [inline, protected]

the servers says, that a new slide is now being displayed the applet shows the URL in the frame with the name "slides".

Parameters:

url where the slide can be retrieved

Reimplemented in **ViewerClient** (p. 272).

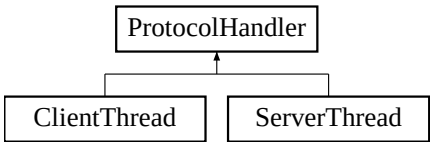
The documentation for this class was generated from the following file:

- PresentationReceiver.java

9.82 ProtocolHandler Class Reference

this class is meant to do the whole server Client -Protocol It uses DataStreams, because high peformance is not necessary here for each command it reads, it calls the corresponding **CommandListener** (p. 76).

Inheritance diagram for ProtocolHandler::



Public Methods

- **ProtocolHandler** (**CommandListener** listener)
- void **read** () throws IOException

  steadily read commands.

- synchronized void **write** (int type, String command)

  write out any command.

- void **kill** ()

  calling this method means for the Reader Thread to return this class cannot be used again.

Static Public Methods

- void **close** (Socket s, InputStream **in**, OutputStream **out**)

  close the given streams a hundred percent.

Protected Methods

- abstract void **exception** ()

  an IOException has occcured.

- void **start** ()

  start the reader.

Protected Attributes

- DataInputStream **in**

  the Stream to read the commands from.

- DataOutputStream **out**

  the Stream to write out own commands.

9.82.1 Detailed Description

this class is meant to do the whole server Client -Protocol It uses DataStreams, because high peformance is not necessary here for each command it reads, it calls the corresponding **Command-Listener** (p. 76).

9.82.2 Member Function Documentation

9.82.2.1 abstract void ProtocolHandler::exception () [protected, pure virtual]

an IOException has occcured.

The derviving class now has to decide what to do.

Reimplemented in **ClientThread** (p. 73), and **ServerThread** (p. 237).

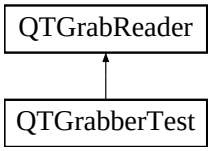
The documentation for this class was generated from the following file:

- ProtocolHandler.java

9.83 QTGrabberTest Class Reference

grabtest.cpp.

Inheritance diagram for QTGrabberTest::



Public Methods

- **QTGrabberTest** ()
- void **ProcessFrame** (void *pImage)

9.83.1 Detailed Description

grabtest.cpp.

Author:
Daniel Pustka <daniel@pustka.de> @id

Id:
grabtest.cpp,v 1.4 2002/01/21 18:26:41 pustka Exp

@revision

Revision:
1.4

The documentation for this class was generated from the following file:

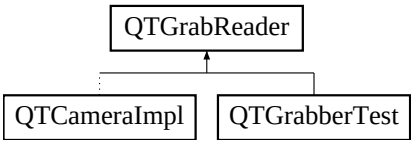
- grabtest.cpp

9.84 QTGrabReader Class Reference

defines a callback interface.

`#include <qtgrabber.h>`

Inheritance diagram for QTGrabReader::



Public Methods

- virtual void **ProcessFrame** (void *pImage)=0

9.84.1 Detailed Description

defines a callback interface.

Author:
Daniel Pustka <daniel@pustka.de> @id

Id:
qtgrabber.h,v 1.5 2002/01/24 13:35:20 pustka Exp

@revision

Revision:
1.5

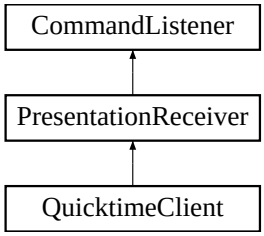
The documentation for this class was generated from the following file:

- **qtgrabber.h**

9.85 QuicktimeClient Class Reference

The QuicktimeClient Class is the Applet, which shows the Quicktime Video Stream in a browser ∗ frame.

Inheritance diagram for QuicktimeClient::



Public Methods

- void **init** ()
- void **start** ()
- void **stop** ()

  Stops the Applet.

- void **destroy** ()

  Destroys the Applet and closes Quicktime.

Protected Methods

- void **updateStreams** (StreamURLManager m)

  Inherited from Presentation Receiver.

- void **openStream** (String StreamURL)

  Opens up a stream and displays it in the Quicktime Player.

9.85.1 Detailed Description

The QuicktimeClient Class is the Applet, which shows the Quicktime Video Stream in a browser ∗ frame.

Author:
Wolfgang Sprung

9.85.2 Member Function Documentation

9.85.2.1 void QuicktimeClient::openStream (String *StreamURL*) [inline, protected]

Opens up a stream and displays it in the Quicktime Player.

Parameters:
StreamURL URL of the stream to open (as a string)

9.85.2.2 void QuicktimeClient::updateStreams (StreamURLManager *m*) [inline, protected]

Inherited from Presentation Receiver.

Gets new stream sources from the StreamManager and opens them.

Parameters:
m the StreamURLManager, in which the URLs of the available streams are saved.

Reimplemented from **PresentationReceiver** (p. 219).

The documentation for this class was generated from the following file:

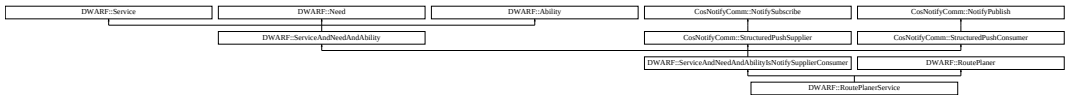
- QuicktimeClient.java

9.86 DWARF::RoutePlanerService Interface Reference

This interface creates a DetectMarker **DWARF** (p. 59) service.

`import "RoutePlaner.idl";`

Inheritance diagram for DWARF::RoutePlanerService::



9.86.1 Detailed Description

This interface creates a DetectMarker **DWARF** (p. 59) service.

To call it, please use the DetectMarker interface

The documentation for this interface was generated from the following file:

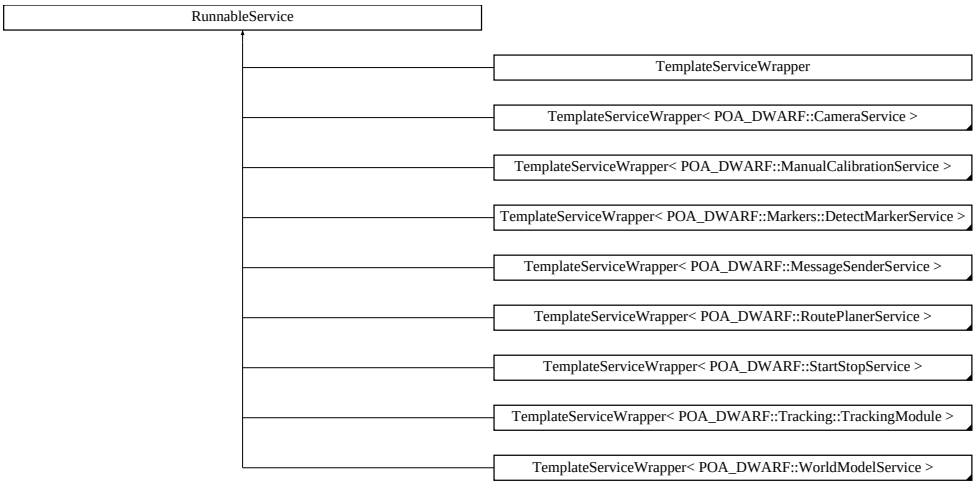
- **RoutePlaner.idl**

9.87 RunnableService Class Reference

RunnableService is an abstract interface, which services have to inherit from, if they ought to be run by using a MultipleServiceRunner.

`#include <templateservicewrapper.h>`

Inheritance diagram for RunnableService::



Public Methods

- **RunnableService** ()

  the default constructor.

- virtual ~**RunnableService** ()
- void **handle_startService** ()
- void **handle_stopService** ()
- void **handle_connectAbility** (const char *offername, Connector_ptr conn)
- void **handle_disconnectAbility** (const char *offername, Connector_ptr conn)
- void **handle_connectNeed** (const char *reqname, CORBA::Short reqinstance, Connector_-ptr conn)
- void **handle_disconnectNeed** (const char *reqname, CORBA::Short reqinstance, Connector_ptr conn)
- void **handle_subscription_change** (const EventTypeSeq &added, const EventTypeSeq &removed)
- void **handle_disconnect_structured_push_supplier** ()
- void **pushStructuredEvent** (StructuredEvent &sevnt)

  Method for pushing structured events.

- bool **getHasConsumer** ()

  used to determine whether there is a push consumer registered.

- ObjectVarVector * **getImportedObjectsByNeedName** (const char *needName)

  used to access imported CORBA objects returns a vector of all imported services providing the specified need.

- void **setRegistrar** (RegisterServices_ptr r)
- virtual bool **mainLoop** ()

  must be concretized by the inheriting class!

- virtual void **run** ()

  run method constantly calls the **mainLoop***()* (p. 233).

- virtual void **describe** ()=0

  has to be implemented and is called by MultipleServiceRunner to give the service a chance to describe its needs and abilities.

- void **setDescribe** (DescribeServices_ptr desc)

  set the describer member variable.

- void **setParameters** (vector< string *> *pVect)

  set parameter vector.

- vector< string *> * **getParameters** ()

  get parameter vector.

- virtual void **onImportChange** (const char *needName)=0

  abstract method concretized by wrapper template.

- const char * **getServiceName** ()

Static Public Methods

- void * **threadStartRoutine** (void *data)

  static routine needed for thread initialization - just calls the virtual run method.

Protected Methods

- bool **describeService** (char ***srvcName**, bool startOnDemand, bool stopOnNoUse)

  call before adding abilities and needs.

- bool **addAbility** (char *abilityName, char *abilityType, char *attributes, int connector)

  used to add an ability to a service description.

- bool **addNeed** (char *needName, char *needType, char *predicate, int connector, int minInstances=0, int maxInstances=10000)

  used to add a need to a service description.

- virtual StructuredPushSupplier_ptr **getCORBA_thisStructuredPushSupplier** ()=0
- virtual StructuredPushConsumer_ptr **getCORBA_thisStructuredPushConsumer** ()=0
- virtual ServiceAndNeedAndAbility_ptr **getCORBA_thisServiceAndNeedAndAbility** ()=0

Protected Attributes

- RegisterServices_var **registrar**
  variable for registering services.
- bool **hasSupplier**
  indicated whether there is a push supplier available.
- bool **hasConsumer**
  indicates whether there is a consumer available or not.
- StructuredProxyPushConsumer_var **notifyConsumer**
  the Consumer.
- Connector_ptr **notifyConsumerConnector**
- map< string, ObjectVarVector *> **needRefVctMap**
  stores imported CORBA interfaces in vectors, hashing is done by the need name.
- int **numEvents**
  just for counting.
- ServiceDescription_var **sd**
- NeedDescription_var **nd**
  the service describer.
- AbilityDescription_var **ad**
  only locally used.
- ConnectorDescription_var **cd**
  only locally used.
- vector< string *> * **paramVector**
  only locally used.
- DescribeServices_var **describevar**
- char * **srvcName**
  stores reference to the describer.

9.87.1 Detailed Description

RunnableService is an abstract interface, which services have to inherit from, if they ought to be run by using a MultipleServiceRunner.

9.87.2 Member Function Documentation

9.87.2.1 bool RunnableService::addAbility (char * *abilityName*, char * *abilityType*, char * *attributes*, int *connector*) [protected]

used to add an ability to a service description.

Parameters:

abilityName locally used name to distinguish abilities

abilityType globally used ability type (e.g. DwarfStatus, ...)

attributes further details about the ability's quality, used by SLP (use SLP syntax)

connector determines the used connector type use the constants specified in the enum above

9.87.2.2 bool RunnableService::addNeed (char * *needName*, char * *needType*, char * *predicate*, int *connector*, int *minInstances* = 0, int *maxInstances* = 10000) [protected]

used to add a need to a service description.

Parameters:

needName locally used name to distinguish needs

needType globally used need type (e.g. DwarfStatus, ...) - must match corresponding abil type

predicate further details about the needs's quality, used by SLP (use SLP syntax)

connector determines the used connector type use the constants specified in the enum above

minInstances minimum no. of available ability instances to allow connection

minInstances maximum no. of available ability instances to be presented

9.87.2.3 bool RunnableService::describeService (char * *srvcName*, bool *startOnDemand*, bool *stopOnNoUse*) [protected]

call before adding abilities and needs.

Parameters:

srvcName the service name (unique per service manager)

startOnDemand not currently implemented by the Service Manager

stopOnNoUse not currently implemented by the Service Manager

9.87.2.4 bool RunnableService::getHasConsumer () [inline]

used to determine whether there is a push consumer registered.

Returns:

true if there is a consumer registered

9.87.2.5 ObjectVarVector * RunnableService::getImportedObjectsByNeedName (const char * *name*)

used to access imported CORBA objects returns a vector of all imported services providing the specified need.

Parameters:

name the name of the imported need

Returns:

a pointer to the object vector stored in the hastable

9.87.2.6 virtual bool RunnableService::mainLoop () [inline, virtual]

must be concretized by the inheriting class!

Returns:
false if service should stop

Reimplemented in **ManualCalibrationImpl** (p. 156), **NMEATrackingModule** (p. 194), **TrackingFilterImpl** (p. 260), **DWARF::WorldModelImpl** (p. 283), and **DWARF::WorldModelImpl** (p. 283).

9.87.2.7 void RunnableService::pushStructuredEvent (StructuredEvent & *sevnt*)

Method for pushing structured events.

Parameters:
sevnt the structured event to be pushed

The documentation for this class was generated from the following files:

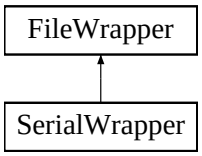
- **templateservicewrapper.h**
- templateservicewrapper.cpp

9.88 SerialWrapper Class Reference

Implements a **FileWrapper** (p. 125) for SerialPorts. Many additional properties can be used to configure the SerialPort properly.

`#include <SerialWrapper.h>`

Inheritance diagram for SerialWrapper::



Public Methods

- **SerialWrapper** (char ∗**name**, char ∗**mode**, int **baudRate**, int **dataBits**, int **stopBits**, int **parity**)
  Creates a new SerialWrapper for the given port, and sets the given parameters for it.
- virtual FILE ∗ **open** ()
  Tries to open the wrapped serial port.
- virtual void **setBaudRate** (int **baudRate**)
  Sets the baud rate.
- virtual void **setDataBits** (int **dataBits**)
  Sets the data bits.
- virtual void **setStopBits** (int **stopBits**)
  Sets the stop bits.
- virtual void **setParity** (int **parity**)
  Sets the parity.
- ∼**SerialWrapper** ()

Protected Attributes

- int **baudRate**
  The Baud Rate of the serial port.
- int **dataBits**
  The number of data bits to use.
- int **stopBits**
  The number of stop bits to use.
- int **parity**
  The parity settings of the serial port.

9.88.1 Detailed Description

Implements a **FileWrapper** (p. 125) for SerialPorts. Many additional properties can be used to configure the SerialPort properly.

9.88.2 Constructor & Destructor Documentation

9.88.2.1 SerialWrapper::SerialWrapper (char * *name*, char * *mode*, int *baudRate*, int *dataBits*, int *stopBits*, int *parity*)

Creates a new SerialWrapper for the given port, and sets the given parameters for it.

Parameters:

name The name of the serial port's device file. This is usually a file in the /dev directory (e.g. on Linux /dev/ttyS0, /dev/ttyS1, /dev/ttyS2, ...). It is not checked, if the file exists and describes a serial port.

mode The access mode to use when opening the serial port.

baudRate The baud rate to use for the serail port.

dataBits The number of data bits to use for serial communication.

stopBits The number of stop bits to use for serial communication.

parity The parity settings for the serial port.

9.88.3 Member Function Documentation

9.88.3.1 FILE * SerialWrapper::open () [virtual]

Tries to open the wrapped serial port.

This is only possible if the given device file is accessible and describes a serial port.

Returns:

The FILE handle of the wrapped serial port, or NULL, if it could not be opened.

Reimplemented from **FileWrapper** (p. 127).

9.88.3.2 void SerialWrapper::setBaudRate (int *baudRate*) [virtual]

Sets the baud rate.

Parameters:

baudRate The baud rate to use for the serial port.

9.88.3.3 void SerialWrapper::setDataBits (int *dataBits*) [virtual]

Sets the data bits.

Parameters:

dataBits The the number of data bits to use for serial communication.

9.88.3.4 void SerialWrapper::setParity (int *parity*) [virtual]

Sets the parity.

Parameters:
parity The parity settings for the serial port.

9.88.3.5 void SerialWrapper::setStopBits (int *stopBits*) [virtual]

Sets the stop bits.

Parameters:
stopBits The the number of stop bits to use for serial communication.

9.88.4 Member Data Documentation

9.88.4.1 int SerialWrapper::baudRate [protected]

The Baud Rate of the serial port.

It's speed.

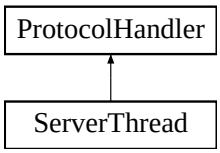
The documentation for this class was generated from the following files:

- **SerialWrapper.h**
- **SerialWrapper.cpp**

9.89 ServerThread Class Reference

This class is the interface of the **PresentationManager** (p. 217) (server) to the client.

Inheritance diagram for ServerThread::



Public Methods

- **ServerThread** (Socket socket, **CommandListener** listener) throws IOException
  the ServerThread puts itself into the list of Threads.

Static Public Methods

- void **writeToAll** (int id, String command)
  spread common comamnds to all Server Threads.

Protected Methods

- void **exception** ()
  an IOException has occcured.

9.89.1 Detailed Description

This class is the interface of the **PresentationManager** (p. 217) (server) to the client.

It handles all the Client requests, writes out server commands and terminates when an exception has occured

9.89.2 Member Function Documentation

9.89.2.1 void ServerThread::exception () [inline, protected, virtual]

an IOException has occcured.

The derviving class now has to decide what to do.

Reimplemented from **ProtocolHandler** (p. 223).

The documentation for this class was generated from the following file:

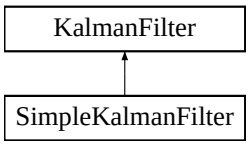
- ServerThread.java

9.90 SimpleKalmanFilter Class Reference

A simple Kalman Filter implementation.

`#include <kalmanfilter.h>`

Inheritance diagram for SimpleKalmanFilter::



Public Methods

- **SimpleKalmanFilter** ()
- ~**SimpleKalmanFilter** ()
- void **setA** (const Matrix &pnewVal)

  Write property of Matrix mA.
- const Matrix & **getA** ()

  Read property of Matrix mA.
- void **setB** (const Matrix &pnewVal)

  Write property of Matrix B.
- const Matrix & **getB** ()

  Read property of Matrix B.
- const Matrix & **getP0** ()

  Read property of Matrix P0.
- Matrix **update** (Matrix z, Matrix u)

  does measurement and prediction update according to measurement z.
- const Matrix & **getX** ()

  Read property of Matrix X.
- void **setH** (const Matrix &pnewVal)

  Write property of Matrix H.
- const Matrix & **getH** ()

  Read property of Matrix H.
- void **setQ** (const Matrix &pnewVal)

  Write property of Matrix Q.
- const Matrix & **getQ** ()

  Read property of Matrix Q.

- void **setR** (const Matrix &pnewVal)
  Write property of Matrix R.
- const Matrix & **getR** ()
  Read property of Matrix R.
- void **init** (Matrix pX0, Matrix pP0, Matrix pA, Matrix pB, Matrix pH, Matrix pQ, Matrix pR)
  initializes the Kalman filter.

9.90.1 Detailed Description

A simple Kalman Filter implementation.

Author:
Andreas Krause implements the linear version of the **KalmanFilter** (p. 149)

9.90.2 Member Function Documentation

9.90.2.1 void SimpleKalmanFilter::init (Matrix *pX0*, Matrix *pP0*, Matrix *pA*, Matrix *pB*, Matrix *pH*, Matrix *pQ*, Matrix *pR*)

initializes the Kalman filter.

Parameters:
X0 initial measurement prediction
P0 initial predicted covariance
A linear model description
B control influence
H relation of measurement to state vector
Q initial process noice
R initial measurement error covariance

9.90.2.2 Matrix SimpleKalmanFilter::update (Matrix *z*, Matrix *u*) [virtual]

does measurement and prediction update according to measurement z.

Parameters:
z current measurement
u current process control

Returns:
updated estimate; use **getX**() (p. 238) to get new prediction

Reimplemented from **KalmanFilter** (p. 149).

The documentation for this class was generated from the following files:

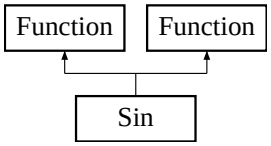
- **kalmanfilter.h**
- kalmanfilter.cpp

9.91 Sin Class Reference

Sine function composition.

`#include <function.h>`

Inheritance diagram for Sin::



Public Methods

- **Sin** (**Function** ∗f)
- virtual ∼**Sin** ()
- virtual **Function** ∗ **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** ∗ **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** ∗ **getOperand** () const
- virtual **Function** ∗ **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const
- **Sin** (**Function** ∗f)
- virtual ∼**Sin** ()
- virtual **Function** ∗ **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** ∗ **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual **Function** ∗ **getOperand** () const
- virtual **Function** ∗ **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const

9.91.1 Detailed Description

Sine function composition.

Author:
Andreas Krause

See also:
Fct (p. 123)

9.91.2 Member Function Documentation

9.91.2.1 virtual Function* Sin::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

9.91.2.2 Function * Sin::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

The documentation for this class was generated from the following files:

- **MovementSimulator/function.h**
- **TrackingFilter/function.h**
- MovementSimulator/function.cpp
- TrackingFilter/function.cpp

9.92 StructuredEventFactory Class Reference

Little helper class for getting a prepared, ready to rumble structured event.

Static Public Methods

- StructuredEvent **getStructuredEvent** (String type)

  returns a prepared StructuredEvent, ready to have its remainder_of_body filled with the structure you desire (No sexism intended here ;-) Use this as a template:.

9.92.1 Detailed Description

Little helper class for getting a prepared, ready to rumble structured event.

9.92.2 Member Function Documentation

9.92.2.1 StructuredEvent StructuredEventFactory::getStructuredEvent (String *type*) [inline, static]

returns a prepared StructuredEvent, ready to have its remainder_of_body filled with the structure you desire (No sexism intended here ;-) Use this as a template:.

StructuredEvent se= StructuredEventFactory.getStructuredEvent("PositionData"); PositionEventHelper.insert(se.remainder_of_body, pe); mysender.push(se);

Returns:
The StructuredEvent

The documentation for this class was generated from the following file:

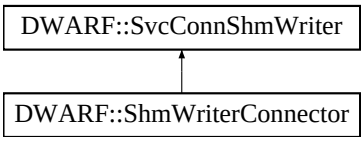
- StructuredEventFactory.java

9.93 DWARF::SvcConnShmWriter Interface Reference

these interfaces are for Connectors that exchange shared memory block IDs.

`import "ServiceCommShm.idl";`

Inheritance diagram for DWARF::SvcConnShmWriter::



Public Methods

- long **initShmRingBuffer** (in long numbuffer, in long bufsize)

9.93.1 Detailed Description

these interfaces are for Connectors that exchange shared memory block IDs.

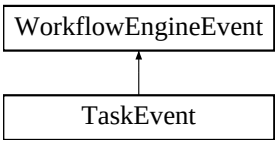
The documentation for this interface was generated from the following file:

- **ServiceCommShm.idl**

9.94 TaskEvent Class Reference

This Event delivers a task to the TaskListener.

Inheritance diagram for TaskEvent::



Public Methods

- **TaskEvent** (Task src, int id)

Static Public Attributes

- final int **TASK_CHANGED** = 0

  TaskEvents are only sent if the current Task changed.

9.94.1 Detailed Description

This Event delivers a task to the TaskListener.

See also:
TaskListener

Author:
Stefan Ri"s

Version:

Revision:
1.1

The documentation for this class was generated from the following file:

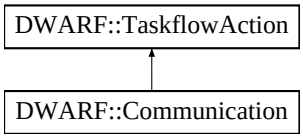
- TaskEvent.java

9.95 DWARF::TaskflowAction Interface Reference

interface **TaskflowAction** (p. 246) is implemented by GarageServerApp and WeareableApp it's used to receive a list of possible TaskflowActions from the **TaskflowEngine** (p. 247) via RPC.

`import "Application.idl";`

Inheritance diagram for DWARF::TaskflowAction::



Public Methods

- long **getTaskflowAction** (in idval data)

9.95.1 Detailed Description

interface **TaskflowAction** (p. 246) is implemented by GarageServerApp and WeareableApp it's used to receive a list of possible TaskflowActions from the **TaskflowEngine** (p. 247) via RPC.

(See TF_* constants for meaning of values in Idvallist.idl) the returned long (which is an int in java) is used to check if data was received.

The documentation for this interface was generated from the following file:

- **Application.idl**

9.96 TaskflowEngine Class Reference

public class TaskflowEngine extends TaskflowEnginePOA.

Public Methods

- **TaskflowEngine** (String corbaloc, String wearable)

  add our abilities, add our needs, register our service, initialize the ObjectExporters, register our abilities, initialize the ObjectImporters, register our needs.

- String **getName** ()

  If the tfe is running on the wearable this returns TaskflowEngineWearable. Running on the Garage server this returns TaskflowEngine.

- String **getStatus** ()

  If the taskflowAction and the persistent data storage are running:return OK else WAITING.

- int **ChangeTaskflow** (idval data)

  either initializes a new workflow or changes the state of a current workflow.

- void **p** (String s)
- void **run** ()

Static Public Methods

- void **main** (String args[])

  And sleep for some time.

Public Attributes

- final String **ID** = "$Id: TaskflowEngine.java,v 1.39 2002/02/04 13:16:32 groher Exp $"

  Our ID in the CVS:
  Id:
  TaskflowEngine.java,v 1.39 2002/02/04 13:16:32 groher Exp
  .

Static Public Attributes

- final String **DEFLOCATION** = "corbaloc::localhost:39273/ServiceManager"

  the Default location where we expect the service manager to listen.

9.96.1 Detailed Description

public class TaskflowEngine extends TaskflowEnginePOA.

9.96.2 Constructor & Destructor Documentation

9.96.2.1 TaskflowEngine::TaskflowEngine (String *corbaloc*, String *wearable*) [inline]

add our abilities, add our needs, register our service, initialize the ObjectExporters, register our abilities, initialize the ObjectImporters, register our needs.

Parameters:

corbaloc, the location where we find the service manager

wearable, is "Wearable" if the TaskflowEngine is run on the Wearable computer, otherwise it is "".

Author:

ottmi

Date:

22.1.02

9.96.3 Member Function Documentation

9.96.3.1 int TaskflowEngine::ChangeTaskflow (idval *data*) [inline]

either initializes a new workflow or changes the state of a current workflow.

Parameters:

data - an object with two attributes, val and id. id can be TF_DOCUMENT which causes the method to open a new taskflow and start it on the current workflowEngine, or TF_CHANGETRANSITION which makes a running Taskflow change the state. The value field contains the identifier to a workflow xml file or a custom event (right now only NEXT and PREV is possible) to address the proper transition to be fired

Author:

ottmi

Date:

23.1.02

9.96.3.2 String TaskflowEngine::getName () [inline]

If the tfe is running on the wearable this returns TaskflowEngineWearable. Running on the Garage server this returns TaskflowEngine.

Parameters:

author ottmi

Date:

23.1.02

9.96.3.3 String TaskflowEngine::getStatus () [inline]

If the taskflowAction and the persistent data storage are running:return OK else WAITING.

Parameters:
author ottmi

Date:
23.1.02

9.96.4 Member Data Documentation

9.96.4.1 final String TaskflowEngine::DEFLOCATION = "corbaloc::localhost:39273/ServiceManager" [static]

the Default location where we expect the service manager to listen.

Parameters:
author ottmi

Date:
23.1.02

9.96.4.2 final String TaskflowEngine::ID = "$Id: TaskflowEngine.java,v 1.39 2002/02/04 13:16:32 groher Exp $"

Our ID in the CVS:

Id:
TaskflowEngine.java,v 1.39 2002/02/04 13:16:32 groher Exp

.

Parameters:
author ottmi

Date:
23.1.02

The documentation for this class was generated from the following file:

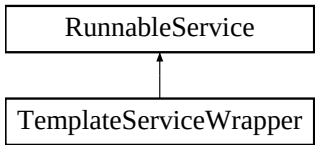
- **TaskflowEngine.java**

9.97 TemplateServiceWrapper Class Template Reference

class TemplateServiceWrapper.

`#include <templateservicewrapper.h>`

Inheritance diagram for TemplateServiceWrapper::



Protected Methods

- **TemplateServiceWrapper** ()

  the default constructor.

- virtual ∼**TemplateServiceWrapper** ()
- void **startService** ()

  Service interface - used by the ServiceManager - DO NOT call these directly!

- void **stopService** ()

  Service interface - used by the ServiceManager - DO NOT call these directly!

- void **connectAbility** (const char ∗offername, Connector_ptr conn)
- void **disconnectAbility** (const char ∗offername, Connector_ptr conn)
- void **connectNeed** (const char ∗reqname, CORBA::Short reqinstance, Connector_ptr conn)

  Need interface - used by the ServiceManager - DO NOT call these directly!

- void **disconnectNeed** (const char ∗reqname, CORBA::Short reqinstance, Connector_ptr conn)

  Need interface - used by the ServiceManager - DO NOT call these directly!

- void **subscription_change** (const EventTypeSeq &added, const EventTypeSeq &removed)

  StructuredPushSupplier interface - used by the ServiceManager - DO NOT call these directly!

- void **disconnect_structured_push_supplier** ()

  StructuredPushSupplier interface - used by the ServiceManager - DO NOT call these directly!

- virtual void **push_structured_event** (const CosNotification::StructuredEvent &event)

  this virtual method is called if this service recieves a pushed event Must be concretized if subclass wants to receive pushed events.

- void **disconnect_structured_push_consumer** ()

  StructuredPushConsumer interface - used by the ServiceManager - DO NOT call these directly!

- void **offer_change** (const CosNotification::EventTypeSeq &added, const CosNotification::EventTypeSeq &removed)

  StructuredPushConsumer interface - used by the ServiceManager - DO NOT call these directly!

- virtual void **onImportChange** (const char *needName)

  this method is called as soon as the vector map of imported objects changes.

- StructuredPushSupplier_ptr **getCORBA_thisStructuredPushSupplier** ()
- StructuredPushConsumer_ptr **getCORBA_thisStructuredPushConsumer** ()
- ServiceAndNeedAndAbility_ptr **getCORBA_thisServiceAndNeedAndAbility** ()

9.97.1 Detailed Description

template<class INTERFACETOBEINHERITED> class TemplateServiceWrapper< INTERFACETOBEINHERITED >

class TemplateServiceWrapper.

Author:
Andreas Krause

9.97.2 Member Function Documentation

9.97.2.1 template<class INTERFACETOBEINHERITED> virtual void TemplateServiceWrapper< INTERFACETOBEINHERITED >::push_structured_event (const CosNotification::StructuredEvent & *event*) [inline, protected, virtual]

this virtual method is called if this service recieves a pushed event Must be concretized if subclass wants to receive pushed events.

Parameters:
event reference to the received structured event

Reimplemented in **GestureRecognitionImpl** (p. 143), **DWARF::WorldModelImpl** (p. 283), and **DWARF::WorldModelImpl** (p. 283).

The documentation for this class was generated from the following file:

- **templateservicewrapper.h**

9.98 TestSender Class Reference

This class implements a trivial (and rather stupid) TestSender, it expands the **DemoService** (p. 96) (version 1.8) for **DWARF** (p. 59) that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).

Public Methods

- **TestSender** (String corbaloc)

Static Public Methods

- void **main** (String args[])

Static Public Attributes

- final String **DEFLOCATION** = "corbaloc::localhost:39273/ServiceManager"
  the Default location where we expect the service manager to listen.

9.98.1 Detailed Description

This class implements a trivial (and rather stupid) TestSender, it expands the **DemoService** (p. 96) (version 1.8) for **DWARF** (p. 59) that can either send out or receive PositionEvents via The Corba Notification Service (or even both :-)).

*

Author:
The TRAMP Testing Team

Version:

Revision:
1.1

The documentation for this class was generated from the following file:

- TestSender.java

9.99 TestShmRead Class Reference

testshmread.h.

`#include <testshmread.h>`

9.99.1 Detailed Description

testshmread.h.

Created on January 26, 2001 early in the morning ;)

Author:
Lothar Richter and Jan-Gregor Fischer

Revision:
1.2

Id:
testshmread.h,v 1.2 2002/01/26 19:11:24 fischerj Exp

The documentation for this class was generated from the following file:

- **testshmread.h**

9.100 TestShmWrite Class Reference

testshmwrite.h.

`#include <testshmwrite.h>`

9.100.1 Detailed Description

testshmwrite.h.

Created on January 26, 2001 early in the morning ;)

Author:
Lothar Richter and Jan-Gregor Fischer

Revision:
1.2

Id:
testshmwrite.h,v 1.2 2002/01/26 19:11:24 fischerj Exp

The documentation for this class was generated from the following file:

- **testshmwrite.h**

9.101 TestStatus Class Reference

testobjimport.h.

`#include <testobjimport.h>`

9.101.1 Detailed Description

testobjimport.h.

Created on December 17, 2001

Author:
fischerj @revision

Revision:
1.2

The documentation for this class was generated from the following file:

- **testobjexport.h**

9.102 DWARF::ThingImpl Class Reference

This class implements a Thing from the WorldModel.

`#include <WorldModel.hpp>`

Public Methods

- **ThingImpl** (ThingImpl *_parent, const char *_name, **WorldModelImpl** *_theWorldModel, Position *_pos=NULL)
- ~**ThingImpl** ()
- ThingImpl * **addChild** (const char *_name, Position *_pos)
- void **SetMarkerType** (char *ttype)
- void **SetThingType** (char *ttype)
- void **SetMarkerFileName** (char *tfile)
- void **SetMarkerSize** (char *tsize)
- void **SetMarkerPosition** (DoubleSequence *seq)
- void **SetMarkerOrientation** (DoubleSequence *seq)
- int **getNumberOfChildren** ()
- ThingID **getID** ()
- char * **getName** ()
- char * **getPath** ()
- char * **getThingType** ()
- char * **getMarkerType** ()
- char * **getMarkerFileName** ()
- char * **getMarkerSize** ()
- DoubleSequence * **getMarkerPosition** ()
- DoubleSequence * **getMarkerOrientation** ()
- Thing_ptr **getParent** ()
- ThingID **getParentID** ()
- void **changeParentToID** (ThingID newParent)
- StringSequence * **getChildrenNames** ()
- StringSequence * **getChildrenPaths** ()
- ThingIDSequence * **getChildrenIDs** ()
- void **addChild** (const char *name)
- Position **getAbsolutePosition** ()
- Position **getPositionToParent** ()
- Position **getPositionToID** (ThingID toID)
- Position **getPositionToPath** (const char *toPath)
- void **setPosition** (const Position &pos)
- DWARF::PropertySeq * **getProperties** ()
- CORBA::Boolean **hasProperty** (const char *prop)
- CORBA::Any * **getProperty** (const char *prop)
- void **setProperty** (const char *prop, const CORBA::Any &value)
- void **deleteProperty** (const char *prop)
- void **destroy** ()
- **ThingImpl** (ThingImpl *_parent, const char *_name, **WorldModelImpl** *_theWorldModel, Position *_pos=NULL)
- ~**ThingImpl** ()
- ThingImpl * **addChild** (const char *_name, Position *_pos)

- void **SetMarkerType** (char *ttype)
- void **SetThingType** (char *ttype)
- void **SetMarkerFileName** (char *tfile)
- void **SetMarkerSize** (char *tsize)
- void **SetMarkerPosition** (DoubleSequence *seq)
- void **SetMarkerOrientation** (DoubleSequence *seq)
- int **getNumberOfChildren** ()
- ThingID **getID** ()
- char * **getName** ()
- char * **getPath** ()
- char * **getThingType** ()
- char * **getMarkerType** ()
- char * **getMarkerFileName** ()
- char * **getMarkerSize** ()
- DoubleSequence * **getMarkerPosition** ()
- DoubleSequence * **getMarkerOrientation** ()
- Thing_ptr **getParent** ()
- ThingID **getParentID** ()
- void **changeParentToID** (ThingID newParent)
- StringSequence * **getChildrenNames** ()
- StringSequence * **getChildrenPaths** ()
- ThingIDSequence * **getChildrenIDs** ()
- void **addChild** (const char *name)
- Position **getAbsolutePosition** ()
- Position **getPositionToParent** ()
- Position **getPositionToID** (ThingID toID)
- Position **getPositionToPath** (const char *toPath)
- void **setPosition** (const Position &pos)
- DWARF::PropertySeq * **getProperties** ()
- CORBA::Boolean **hasProperty** (const char *prop)
- CORBA::Any * **getProperty** (const char *prop)
- void **setProperty** (const char *prop, const CORBA::Any &value)
- void **deleteProperty** (const char *prop)
- void **destroy** ()

9.102.1 Detailed Description

This class implements a Thing from the WorldModel.

Author:
Benjamin Herrmann (herrmabe@in.tum.de)

Date:
05.02.2002

The documentation for this class was generated from the following files:

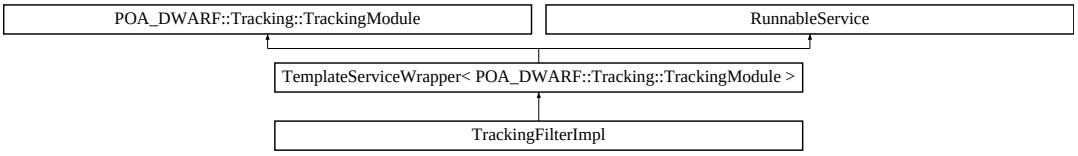
- **WorldModel.hpp**
- **mpl/WorldModel.hpp**
- Thing.cpp
- mpl/Thing.cpp

9.103 TrackingFilterImpl Class Reference

the implementation of the tracking filter service.

`#include <trackingfilter.h>`

Inheritance diagram for TrackingFilterImpl::



Public Methods

- **TrackingFilterImpl** ()

  default constructor.

- ~**TrackingFilterImpl** ()

  default destructor.

- void **describe** ()

  describes the service to the ServiceMgr.

- bool **mainLoop** ()

  does the filtering.

- char * **getName** ()
- char * **getStatus** ()
- PoseSourceCapabilities **getCapabilities** ()

  get the Filters' capabilities.

- PoseData **getPosition** (const Time &t)

  read the current position.

- void **calibrate** (const PoseData &currentposition)

  The calibrating method orders the filter to renew the filter models and re-parse the XML file, thus rebuilding the models.

Protected Methods

- void **queryTrackerStateVectors** ()

  updates the mTrackerStates matrix.

- void **updatePredictedPose** ()

  updates the mCurPose data.

- void **pushPoseData** ()

pushes the current pose estimation.

- virtual void **onImportChange** (const char *needName)
  is called if new objects are registered.

Protected Attributes

- int **mTrackerCount**
- ObjectVarVector * **mTrackers**
  counts the registered trackers.
- TrackingFilterSAXHandler * **mXmlParser**
  vector containing the tracker references.
- PoseData **mCurPose**
  reference to the XML parser.
- **PoseDataModelMapping** * **mModels**
  current Pose estimation.
- Matrix * **mTrackerStates**
  maps the PoseData entries to EKFs.
- omni_mutex **mSema**
  contains the current states.

9.103.1 Detailed Description

the implementation of the tracking filter service.

Author:
Andreas Krause This class aggregates the XML parser and the **PoseDataModelMapping** (p. 207)

9.103.2 Member Function Documentation

9.103.2.1 PoseSourceCapabilities TrackingFilterImpl::getCapabilities ()

get the Filters' capabilities.

Returns:
description of filtering precision, defined in Tracker.idl

9.103.2.2 char * TrackingFilterImpl::getName ()

Returns:
the service name

9.103.2.3 PoseData TrackingFilterImpl::getPosition (const Time & *t*)

read the current position.

Parameters:

time The time for which the position in desired. Only used for prediction filters, so check the time member of the returned position! THIS IS NOT CURRENTLY IMPLEMENTED

Returns:

current Pose estimation

9.103.2.4 char * TrackingFilterImpl::getStatus ()

Returns:

the service status

9.103.2.5 bool TrackingFilterImpl::mainLoop () [virtual]

does the filtering.

Returns:

false if service should stop

Reimplemented from **RunnableService** (p. 233).

The documentation for this class was generated from the following files:

- **trackingfilter.h**
- trackingfilter.cpp

9.104 DWARF::TrackingFilterSAXHandler Class Reference

XML parser for construction the **PoseDataModelMapping** (p. 207).

`#include <trackingfilterxmlparser.h>`

Public Methods

- **TrackingFilterSAXHandler** ()
- ~**TrackingFilterSAXHandler** ()

  the default constructor.

- void **updateModelFromXML** (ObjectVarVector *trackers, **PoseDataModelMapping** *models, const char *xmlFile)

  removes ALL mapped EKFs and remaps them according to the most suitable (first is best in XML file) model definitions.

- void **parse** (const char *xmlFile)

  starts the SAX parser.

- void **startDocument** ()
- void **endDocument** ()

  callback for the SAX handler.

- void **startElement** (const XMLCh *const name, AttributeList &attributes)

  callback if a new element is being parsed.

- void **endElement** (const XMLCh *const name)

  callback if an element is being closed.

- void **error** (const SAXParseException &exception)

  error handling.

- void **fatalError** (const SAXParseException &exception)
- void **warning** (const SAXParseException &exception)

9.104.1 Detailed Description

XML parser for construction the **PoseDataModelMapping** (p. 207).

Author:

Andreas Krause defines a class implementing a SAX based XML parser for parsing and constructing the filter<->PoseData mapping

9.104.2 Member Function Documentation

9.104.2.1 void TrackingFilterSAXHandler::endElement (const XMLCh *const *name*)

callback if an element is being closed.

Parameters:

name name of the Tag

9.104.2.2 void TrackingFilterSAXHandler::parse (const char * *xmlFile*)

starts the SAX parser.

Parameters:

xmlFile name of the XML file to be parsed

9.104.2.3 void TrackingFilterSAXHandler::startElement (const XMLCh *const *name*, AttributeList & *attributes*)

callback if a new element is being parsed.

Parameters:

name name of the Tag

attributes list of attributes supplied

9.104.2.4 void TrackingFilterSAXHandler::updateModelFromXML (ObjectVarVector * *trackers*, PoseDataModelMapping * *models*, const char * *xmlFile*)

removes ALL mapped EKFs and remaps them according to the most suitable (first is best in XML file) model definitions.

Parameters:

trackers vector referencing the connected trackers

models reference to the filter<-> PoseData mapping

xmlFile name of the XML file to be parsed

The documentation for this class was generated from the following files:

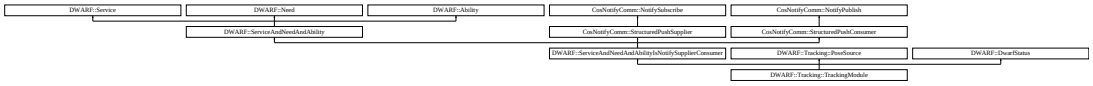
- **trackingfilterxmlparser.h**
- trackingfilterxmlparser.cpp

9.105 DWARF::Tracking::TrackingModule Interface Reference

A complete tracking module service.

`import "Tracker.idl";`

Inheritance diagram for DWARF::Tracking::TrackingModule::



9.105.1 Detailed Description

A complete tracking module service.

The documentation for this interface was generated from the following file:

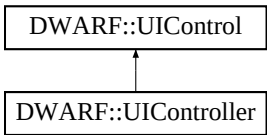
- **Tracker.idl**

9.106 DWARF::UIControl Interface Reference

This is the interface for the UIController.

`import "UserInterface.idl";`

Inheritance diagram for DWARF::UIControl::



Public Methods

- void **configureUIController** (in string pathwdl)

9.106.1 Detailed Description

This is the interface for the UIController.

9.106.2 Member Function Documentation

9.106.2.1 void DWARF::UIControl::configureUIController (in string *pathwdl*)

a pathwdl this is the WDL file for the UI controller

The documentation for this interface was generated from the following file:

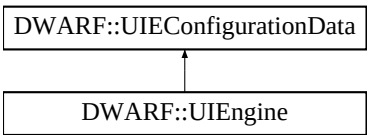
- **UserInterface.idl**

9.107 DWARF::UIEConfigurationData Interface Reference

This is the interface for the **UIEConfigurationData** (p. 265).

```
import "UserInterface.idl";
```

Inheritance diagram for DWARF::UIEConfigurationData::



Public Methods

- void **renderCUIML** (in string url)

9.107.1 Detailed Description

This is the interface for the **UIEConfigurationData** (p. 265).

9.107.2 Member Function Documentation

9.107.2.1 void DWARF::UIEConfigurationData::renderCUIML (in string *url*)

a url this is the CUIML file to render

The documentation for this interface was generated from the following file:

- **UserInterface.idl**

9.108 DWARF::UserAction Struct Reference

UserAction (p. 266) events are sent to the application depending on the user action.

```
import "UserInterface.idl";
```

Public Attributes

- long **actionID**
- string **actionValue**
- long **dest**

  dest used to distinguish where this action should go 0 means garage, 1 means wearable.

9.108.1 Detailed Description

UserAction (p. 266) events are sent to the application depending on the user action.

The documentation for this struct was generated from the following file:

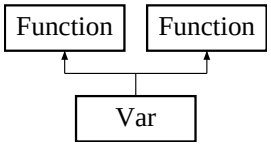
- **UserInterface.idl**

9.109 Var Class Reference

wraps a variable indexed in the constructor.

`#include <function.h>`

Inheritance diagram for Var::



Public Methods

- **Var** (int idx)
- virtual ~**Var** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual int **getIndex** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const
- **Var** (int idx)
- virtual ~**Var** ()
- virtual **Function** * **diff** (int varNo) const

  differentiates the function with respect to variable varNo.

- virtual **Function** * **simplify** ()

  does some VERY simple simplification tricks (mainly neutral/zero elements used in addition/multiplication).

- virtual double **operator()** (Matrix **values**) const
- virtual int **getIndex** () const
- virtual **Function** * **clone** () const
- virtual void **print** (ostream &ostrm) const

  used to print the function.

- virtual int **getType** () const

9.109.1 Detailed Description

wraps a variable indexed in the constructor.

Author:
Andreas Krause

See also:
Fct (p. 123)

9.109.2 Member Function Documentation

9.109.2.1 virtual Function* Var::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

9.109.2.2 Function * Var::diff (int *varNo*) const [virtual]

differentiates the function with respect to variable varNo.

Parameters:
varNo the index (0=first) of the variable to be derived for

Reimplemented from **Function** (p. 131).

The documentation for this class was generated from the following files:

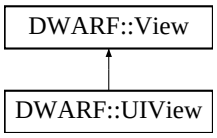
- **MovementSimulator/function.h**
- **TrackingFilter/function.h**
- MovementSimulator/function.cpp
- TrackingFilter/function.cpp

9.110 DWARF::View Interface Reference

This is the interface for the HTMLView, IPAQView, FlashView.

`import "UserInterface.idl";`

Inheritance diagram for DWARF::View::



Public Methods

- void **configureUIView** (in string pathview, in string viewtype)
- void **changeView** (in string index)

  @Param: index is the Task index.

9.110.1 Detailed Description

This is the interface for the HTMLView, IPAQView, FlashView.

9.110.2 Member Function Documentation

9.110.2.1 void DWARF::View::configureUIView (in string *pathview*, in string *viewtype*)

a pathview this is the path for View to render

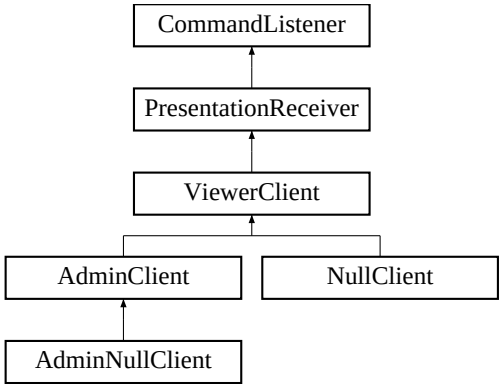
The documentation for this interface was generated from the following file:

- **UserInterface.idl**

9.111 ViewerClient Class Reference

The ViewerClient Class is the Applet (derived from **PresentationReceiver** (p. 219)), which is executed on the viewer's computer.

Inheritance diagram for ViewerClient::



Public Methods

- void **init** ()

  This method initializes the applet and creates the User Interface.

- void **start** ()

  Starts the Applet.

- void **stop** ()

  Stops the Applet.

- void **destroy** ()

  Destroys the Applet and closes Quicktime.

Protected Methods

- void **newMessage** (String message)

  Displays message from Presentation Manager to the client.

- void **regUser** ()

  pop up a window and ask for the users name.

- void **createPresentationFrame** ()

  Creates a window, in which a video stream is displayed to the viewer.

- void **newSlide** (URL url)

  the servers says, that a new slide is now being displayed the applet shows the URL in the frame with the name "slides".

- void **privateMessage** (String person, String text)

  a private message from the server has arrived, it is only displayed to a user with a certain name.

- int **h** ()

  the current pixel height.

- int **w** ()

  the current pixel width.

Protected Attributes

- Panel **pvUI**
- Panel **pvUI2**
- Panel **pvUI3**
- TextArea **taChat**
- Label **lReACT**
- Label **lEnterName**
- TextField **tfEnterName**
- Button **bEnterQ**
- Button **bChat**
- Button **bMessage**
- Frame **fReg**
- **Dialog regDialog**
- String **name**
- String **question**
- URL **actSlide**
- boolean **sDetached**
- GridBagLayout **gbl**
- GridBagConstraints **gbc**
- GridBagLayout **gbl2**
- GridBagConstraints **gbc2**
- TextArea **taChat2**
- Checkbox **eSlides**

9.111.1 Detailed Description

The ViewerClient Class is the Applet (derived from **PresentationReceiver** (p. 219)), which is executed on the viewer's computer.

It is responsible for displaying the A/V Streams (with Quicktime) and the slides (in the browser) to the user. It also should give the user the possibility to interrupt the presentation and ask a question.

Author:
Wolfgang Sprung

9.111.2 Member Function Documentation

9.111.2.1 void ViewerClient::newMessage (String *message*) [inline, protected]

Displays message from Presentation Manager to the client.

Parameters:
message to display as String

Reimplemented from **PresentationReceiver** (p. 219).

9.111.2.2 void ViewerClient::newSlide (URL *url*) [inline, protected]

the servers says, that a new slide is now being displayed the applet shows the URL in the frame with the name "slides".

Parameters:
url where the slide can be retrieved

Reimplemented from **PresentationReceiver** (p. 220).

9.111.2.3 void ViewerClient::privateMessage (String *person*, String *text*) [inline, protected]

a private message from the server has arrived, it is only displayed to a user with a certain name.

Parameters:
person, the name of the adressed user
text, the message as a String

Reimplemented from **PresentationReceiver** (p. 220).

9.111.2.4 void ViewerClient::start () [inline]

Starts the Applet.

Content is given to the QuicktimeCanvas. Applet connects to **PresentationManager** (p. 217)

The documentation for this class was generated from the following file:

- ViewerClient.java

9.112 WearableApplication Class Reference

main class of this file(what else?).

Public Methods

- **WearableApplication** (String corbaloc)

  Initializes and describes our service.

- void **startCalibration** ()

  starts the receiving of GestureData.

- void **startGestureData** ()

  starts the receiving of GestureData.

- void **startMarkerInfo** ()

  starts receiving MarkerInfo.

- void **stopMarkerInfo** ()

  stops receiving MarkerInfo.

- void **stopGestureData** ()

  stops receiving GestureData.

- String **getStatus** ()

  implements a function that returns the status of this service.

- String **getName** ()

  implements a function that returns the name of this service.

- void **transferMessage** (int id, String mes)

  message Interface for intra Application communication.

- void **sendAppMessage** (int id, String mes)

  sends our messages messages.

- int **triggerTaskflowTransition** (idval Taskflowoption)

  sends a list with instructions to Taskflow Engine.

- int **getTaskflowAction** (idval data)

  Interface for Taskflow engine to transmit Taskflow options.

- void **sendUIEConfigurationData** (String path)

  sends adress of a xml-file where configuration-data is listed.

- boolean **checkStatus** ()

Static Public Methods

- void **main** (String args[])
  main-method of **GarageServerApplication** (p. 133).

Static Public Attributes

- final String **DEFLOCATION** = "corbaloc::localhost:39273/ServiceManager"
  the Default location where we expect the service manager to listen.

9.112.1 Detailed Description

main class of this file(what else?).

9.112.2 Constructor & Destructor Documentation

9.112.2.1 WearableApplication::WearableApplication (String *corbaloc*) [inline]

Initializes and describes our service.

Parameters:
corbaloc the url where we find the service manager

9.112.3 Member Function Documentation

9.112.3.1 String WearableApplication::getName () [inline]

implements a function that returns the name of this service.

Precondition:

Postcondition:

Author:
meierb

Date:
23.1.02

9.112.3.2 String WearableApplication::getStatus () [inline]

implements a function that returns the status of this service.

Precondition:

Postcondition:

Author:
meierb

Date:
23.1.02

9.112.3.3 int WearableApplication::getTaskflowAction (idval *data*) [inline]

Interface for Taskflow engine to transmit Taskflow options.

Parameters:
data the CUIML path

Precondition:
a Taskflow was triggered

Postcondition:
The path for a ∗.TUIML file has been send to UIController

Author:
hilliges

Date:
23.1.02

9.112.3.4 void WearableApplication::main (String *args*[]) [inline, static]

main-method of **GarageServerApplication** (p. 133).

Parameters:
args

Author:
meierb

Date:
22.1.02

9.112.3.5 void WearableApplication::sendAppMessage (int *id*, String *mes*) [inline]

sends our messages messages.

Parameters:
id the message id

mes the message string

Precondition:

Postcondition:

Author:
meierb

Date:
22.1.02

9.112.3.6 void WearableApplication::sendUIEConfigurationData (String *path*) [inline]

sends adress of a xml-file where configuration-data is listed.

Parameters:
path the xml file path

Precondition:

Postcondition:

Author:
meierb

Date:
22.1.02

9.112.3.7 void WearableApplication::startCalibration () [inline]

starts the receiving of GestureData.

starts the Calibration of the gesture recognition device

Precondition:

Postcondition:

Author:
fischerc

Date:
23.1.02

9.112.3.8 void WearableApplication::startGestureData () [inline]

starts the receiving of GestureData.

Precondition:

Postcondition:

Author:
fischerc

Date:
23.1.02

9.112.3.9 void WearableApplication::startMarkerInfo () [inline]

starts receiving MarkerInfo.

Precondition:

Postcondition:

Author:
hilliges

Date:
25.1.02

9.112.3.10 void WearableApplication::stopGestureData () [inline]

stops receiving GestureData.

Precondition:

Postcondition:

Author:
meierb

Date:
23.1.02

9.112.3.11 void WearableApplication::stopMarkerInfo () [inline]

stops receiving MarkerInfo.

Precondition:

Postcondition:

Author:
hilliges

Date:
25.1.02

9.112.3.12 void WearableApplication::transferMessage (int *id*, String *mes*) [inline]

message Interface for intra Application communication.

Parameters:
id the message id

mes the transferred string message

Precondition:

Postcondition:

Author:
meierb

Date:
22.1.02

9.112.3.13 int WearableApplication::triggerTaskflowTransition (idval *Taskflowoption*) [inline]

sends a list with instructions to Taskflow Engine.

list of instructions contains an ID and value in each element

Parameters:
Taskflowoption Next Taskflowstep

Precondition:

Postcondition:

Author:
meierb

Date:
22.1.02

The documentation for this class was generated from the following files:

- UserInputSender.java
- **WearableApplication.java**

9.113 DWARF::WorldModelImpl Class Reference

This class the WorldModel.

`#include <WorldModel.hpp>`

Inheritance diagram for DWARF::WorldModelImpl::

Public Methods

- **WorldModelImpl** ()
- ~**WorldModelImpl** ()
- void **addThingToMaps** (**ThingImpl** *thing)
- void **deleteThingFromMaps** (**ThingImpl** *thing)
- **ThingImpl** * **getThingImplFromID** (ThingID id)
- void **sendStructuredEvent** (StructuredEvent *sevnt)
- virtual void **describe** ()

  has to be implemented and is called by MultipleServiceRunner to give the service a chance to describe its needs and abilities.

- bool **mainLoop** ()

  must be concretized by the inheriting class!

- char * **getName** ()
- char * **getStatus** ()
- void **SetNumberOfMarkers** ()
- ThingIDSequence * **getAllMarkers** ()
- char * **getMarkerFileNameFromID** (ThingID id)
- char * **getMarkerSizeFromID** (ThingID id)
- char * **getMarkerTypeFromID** (ThingID id)
- DoubleSequence * **getMarkerPositionFromID** (ThingID id)

  NOT USED:.

- DoubleSequence * **getMarkerOrientationFromID** (ThingID id)
- Thing_ptr **getWorld** ()
- Thing_ptr **getThingFromID** (ThingID id)
- Thing_ptr **getThingFromPath** (const char *path)
- ThingID **getIDFromPath** (const char *path)
- CORBA::Any * **getPositionEventFromId** (ThingID id)
- void **setMinHopCount** (CORBA::Short count)
- void **addContents** (const char *url, ThingID parent)
- void **startService** ()

  Service interface - used by the ServiceManager - DO NOT call these directly!

- void **stopService** ()

  Service interface - used by the ServiceManager - DO NOT call these directly!

- void **connectNeed** (const char *needName, CORBA::Short reqinstance, Connector_ptr connector)

  Need interface - used by the ServiceManager - DO NOT call these directly!

- void **disconnectNeed** (const char *needName, CORBA::Short reqinstance, Connector_ptr connector)

  Need interface - used by the ServiceManager - DO NOT call these directly!

- void **connectAbility** (const char *abilityName, Connector_ptr connector)
- void **disconnectAbility** (const char *abilityName, Connector_ptr connector)
- void **subscription_change** (const EventTypeSeq &added, const EventTypeSeq &removed)

  StructuredPushSupplier interface - used by the ServiceManager - DO NOT call these directly!

- void **disconnect_structured_push_supplier** ()

  StructuredPushSupplier interface - used by the ServiceManager - DO NOT call these directly!

- void **push_structured_event** (const CosNotification::StructuredEvent &event)

  this virtual method is called if this service recieves a pushed event Must be concretized if subclass wants to receive pushed events.

- void **disconnect_structured_push_consumer** ()

  StructuredPushConsumer interface - used by the ServiceManager - DO NOT call these directly!

- void **offer_change** (const CosNotification::EventTypeSeq &added, const CosNotification::EventTypeSeq &removed)

  StructuredPushConsumer interface - used by the ServiceManager - DO NOT call these directly!

- **WorldModelImpl** ()
- ~**WorldModelImpl** ()
- void **addThingToMaps** (**ThingImpl** *thing)
- void **deleteThingFromMaps** (**ThingImpl** *thing)
- **ThingImpl** * **getThingImplFromID** (ThingID id)
- void **sendStructuredEvent** (StructuredEvent *sevnt)
- virtual void **describe** ()

  has to be implemented and is called by MultipleServiceRunner to give the service a chance to describe its needs and abilities.

- bool **mainLoop** ()

  must be concretized by the inheriting class!

- char * **getName** ()
- char * **getStatus** ()
- void **SetNumberOfMarkers** ()
- ThingIDSequence * **getAllMarkers** ()
- char * **getMarkerFileNameFromID** (ThingID id)
- char * **getMarkerSizeFromID** (ThingID id)
- char * **getMarkerTypeFromID** (ThingID id)
- DoubleSequence * **getMarkerPositionFromID** (ThingID id)
- DoubleSequence * **getMarkerOrientationFromID** (ThingID id)
- Thing_ptr **getWorld** ()

- Thing_ptr **getThingFromID** (ThingID id)
- Thing_ptr **getThingFromPath** (const char *path)
- ThingID **getIDFromPath** (const char *path)
- CORBA::Any * **getPositionEventFromId** (ThingID id)
- void **setMinHopCount** (CORBA::Short count)
- void **addContents** (const char *url, ThingID parent)
- void **startService** ()

  Service interface - used by the ServiceManager - DO NOT call these directly!

- void **stopService** ()

  Service interface - used by the ServiceManager - DO NOT call these directly!

- void **connectNeed** (const char *needName, CORBA::Short reqinstance, Connector_ptr connector)

  Need interface - used by the ServiceManager - DO NOT call these directly!

- void **disconnectNeed** (const char *needName, CORBA::Short reqinstance, Connector_ptr connector)

  Need interface - used by the ServiceManager - DO NOT call these directly!

- void **connectAbility** (const char *abilityName, Connector_ptr connector)
- void **disconnectAbility** (const char *abilityName, Connector_ptr connector)
- void **subscription_change** (const EventTypeSeq &added, const EventTypeSeq &removed)

  StructuredPushSupplier interface - used by the ServiceManager - DO NOT call these directly!

- void **disconnect_structured_push_supplier** ()

  StructuredPushSupplier interface - used by the ServiceManager - DO NOT call these directly!

- void **push_structured_event** (const CosNotification::StructuredEvent &event)

  this virtual method is called if this service recieves a pushed event Must be concretized if subclass wants to receive pushed events.

- void **disconnect_structured_push_consumer** ()

  StructuredPushConsumer interface - used by the ServiceManager - DO NOT call these directly!

- void **offer_change** (const CosNotification::EventTypeSeq &added, const CosNotification::EventTypeSeq &removed)

  StructuredPushConsumer interface - used by the ServiceManager - DO NOT call these directly!

Public Attributes

- int **NumberOfMarkers**

9.113.1 Detailed Description

This class the WorldModel.

Author:
Benjamin Herrmann (herrmabe@in.tum.de)

Date:
05.02.2002

9.113.2 Member Function Documentation

9.113.2.1 bool DWARF::WorldModelImpl::mainLoop () [virtual]

must be concretized by the inheriting class!

Returns:
false if service should stop

Reimplemented from **RunnableService** (p. 233).

9.113.2.2 bool WorldModelImpl::mainLoop () [virtual]

must be concretized by the inheriting class!

Returns:
false if service should stop

Reimplemented from **RunnableService** (p. 233).

9.113.2.3 void DWARF::WorldModelImpl::push_structured_event (const CosNotification::StructuredEvent & *event*) [virtual]

this virtual method is called if this service recieves a pushed event Must be concretized if subclass wants to receive pushed events.

Parameters:
event reference to the received structured event

Reimplemented from **TemplateServiceWrapper** (p. 251).

9.113.2.4 void WorldModelImpl::push_structured_event (const CosNotification::StructuredEvent & *event*) [virtual]

this virtual method is called if this service recieves a pushed event Must be concretized if subclass wants to receive pushed events.

Parameters:
event reference to the received structured event

Reimplemented from **TemplateServiceWrapper** (p. 251).

The documentation for this class was generated from the following files:

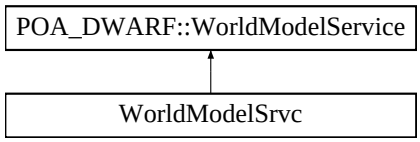
- **WorldModel.hpp**
- **mpl/WorldModel.hpp**
- WorldModel.cpp
- mpl/WorldModel.cpp

9.114 WorldModelSrvc Class Reference

class WorldModelSrvc.

`#include <worldmodelsrvc.h>`

Inheritance diagram for WorldModelSrvc::



Public Methods

- **WorldModelSrvc** (const char *name, RegisterServices_ptr r)

  the default constructor.

- virtual ~**WorldModelSrvc** ()
- void **connectAbility** (const char *offername, Connector_ptr conn)
- void **connectNeed** (const char *reqname, CORBA::Short reqinstance, Connector_ptr conn)
- void **disconnectAbility** (const char *offername, Connector_ptr conn)
- void **disconnectNeed** (const char *reqname, CORBA::Short reqinstance, Connector_ptr conn)
- void **startService** ()
- void **stopService** ()
- void **subscription_change** (const EventTypeSeq &added, const EventTypeSeq &removed)
- void **disconnect_structured_push_supplier** ()
- void **push_structured_event** (const CosNotification::StructuredEvent &event)
- void **disconnect_structured_push_consumer** ()
- void **offer_change** (const CosNotification::EventTypeSeq &added, const CosNotification::EventTypeSeq &removed)
- void **run** ()

  run method constantly pushes position events.

- string **getClassDesc** ()

  class description for debugging.

- char * **getName** ()
- char * **getStatus** ()

Static Public Methods

- void * **threadStartRoutine** (void *data)

  just for thread allocation.

- char * **getClassName** ()

  returns the Classname for debugging.

Protected Attributes

- RegisterServices_var **registrar**
  variable for registering services.
- string **myName**
  naming of the service.
- bool **hasPosConsumer**
  indicates whether there is a consumer available or not.
- StructuredProxyPushConsumer_var **notifyPosConsumer**
  the posConsumer.
- int **numEvents**
  just for counting.

9.114.1 Detailed Description

class WorldModelSrvc.

Author:
Andreas Krause

9.114.2 Constructor & Destructor Documentation

9.114.2.1 WorldModelSrvc::WorldModelSrvc (const char * *name*, RegisterServices_ptr *r*) [inline]

the default constructor.

Parameters:
name Name for the service
r reference to registrar

The documentation for this class was generated from the following files:

- **worldmodelsrvc.h**
- worldmodelsrvc.cpp

9.115 XMLLogger Class Reference

this class is responsible for the whole server logging all messages will be printed in XML format.

Static Public Methods

- void **log** (String message)

  log any normal String.

- void **log** (String type, String message)

  log something with a well defined message type.

- void **main** (String[] args)

  testing only.

Static Public Attributes

- final String **ERROR** = "ERROR"

  all the messages types the XMLLogger knows.

- final String **CHG_SLIDE** = "CHANGESLIDE"

  all the messages types the XMLLogger knows.

- final String **CHAT_MSG** = "CHATMESSAGE"

  all the messages types the XMLLogger knows.

- final String **ASKQUESTION** = "ASKQUESTION"

  all the messages types the XMLLogger knows.

- final String **ANSWERQUESTION** = "ANSWERQUESTION"

  all the messages types the XMLLogger knows.

- final String **TIMELINE** = "TIMELINE"

  all the messages types the XMLLogger knows.

- final String **LOG** = "LOG"

  all the messages types the XMLLogger knows.

9.115.1 Detailed Description

this class is responsible for the whole server logging all messages will be printed in XML format. Each log entry has to contain a timestamp for eventual later reconstruction

9.115.2 Member Function Documentation

9.115.2.1 void XMLLogger::log (String *type*, String *message*) [inline, static]

log something with a well defined message type.

Parameters:

type one of the message types this class defines

message the log entry

9.115.2.2 void XMLLogger::log (String *message*) [inline, static]

log any normal String.

Parameters:

message any log entry

The documentation for this class was generated from the following file:

- XMLLogger.java

Chapter 10

TRAMP File Documentation

10.1 Airguide.java File Reference

implements the Airguide startservice.

Compounds

- class **Airguide**

10.1.1 Detailed Description

implements the Airguide startservice.

Needs & Abilities:

Needs:

Abilities:

- status
- startAirguide

Author:
The TRAMP wearable Team

Version:

Revision:
1.1

Date:

Date:
2002/02/01 18:44:31

last author:

Author:
hilliges

last changed:

Date:
2002/02/01 18:44:31

10.2 Camera.idl File Reference

IDL file for the Camera service.

```
import "DWARF/Time.idl";
import "DWARF/ServiceCommNotify.idl";
import "DWARF/StartStop.idl";
```

Namespaces

- namespace **DWARF**

10.2.1 Detailed Description

IDL file for the Camera service.

Author:
Daniel Pustka

This file belongs to the TRAMP Context Services

10.3 datareceiver.cpp File Reference

this class is a connector for structured event communication.

`#include "datareceiver.h"`

10.3.1 Detailed Description

this class is a connector for structured event communication.

Date:
during TRAMP1 2001/02

Author:
Network team

Revision:
1.3

10.4 datasender.cpp File Reference

this class is a connector for structured event communication.

```
#include "datasender.h"
```

10.4.1 Detailed Description

this class is a connector for structured event communication.

Date:
developed during TRAMP1 2001/02

Author:
Network team

Revision:
1.3

10.5 ExpertSystem.java File Reference

implements the Expert System.

Compounds

- class **ExpertSystem**
  implements the ExpertSystem.

10.5.1 Detailed Description

implements the Expert System.

Author:
Session-Team

Version:

Revision:
1.13

Date:

Date:
2002/02/06 12:07:14

last changed:

Date:
2002/02/06 12:07:14

10.6 FileWrapper.cpp File Reference

```
#include "FileWrapper.h"
```

10.6.1 Detailed Description

Author:

Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Implementation of class: **FileWrapper** (p. 125)

Id:

FileWrapper.cpp,v 1.3 2002/02/06 18:51:35 riedel Exp

Revision:

1.3

10.7 FileWrapper.h File Reference

`#include <stdio.h>`

`#include <iostream.h>`

Compounds

- class **FileWrapper**

  A class that wrapps access to a file. The file can be given by name and path and is remembered by the FileWrapper for further use. Arbitrary open and close operations can be done by the FileWrapper on the wrapped file. The FileWrapper can also be used as interface for wrapping more complex data sources. (e.g. serail ports, raw devices, sockets).

10.7.1 Detailed Description

Author:

Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: **FileWrapper** (p. 125)

Id:

FileWrapper.h,v 1.4 2002/02/06 18:51:35 riedel Exp

Revision:

1.4

10.8 GarageServerApplication.java File Reference

implements the GarageServer.

Compounds

- class **GarageServerApplication**
  implements the GarageServer.
- class **GarageServerApplication::UserActionUser**
  creates ActionUser for sending structured Events.

10.8.1 Detailed Description

implements the GarageServer.

Needs & Abilities:

Needs:

- FindTaskflow
- UserAction
- ApplicationMessage
- Status

Abilities:

- UserInput
- ControlDetectMarker
- UIEConfigurationData
- TriggerTaskflowTransition
- ApplicationMessage

Communication Standards:

UserAction:

- dest=0;
- actionName="problem";
- actionValue="turnlight";

Status:

- getName="<The Exact name of your Service here>"
- getStatus="OK"

getTaskflowAction(idval data):

- data.id="constants.TF_DOCUMENT"
- data.val="The path to the TUIML file"

Author:
The TRAMP Application Team

Version:

Revision:
1.59

Date:

Date:
2002/02/01 18:43:16

last author:

Author:
hilliges

last changed:

Date:
2002/02/01 18:43:16

10.9 GestureData.idl File Reference

contains the struct GestureData.

```
import "DWARF/ServiceCommNotify.idl";
import "DWARF/Time.idl";
import "DWARF/StartStop.idl";
```

Namespaces

- namespace **DWARF**

Compounds

- struct **DWARF::GestureData**
  contains information about the type of time of a recognized gesture.

10.9.1 Detailed Description

contains the struct GestureData.

Author:
Sven Hennauer (hennauer@in.tum.de)

10.10 gesturerecognition.cpp File Reference

Implementation of the gesture recognition service.

`#include "gesturerecognition.h"`

`#include "../MathUtils.h"`

`#include <math.h>`

Functions

- int **main** (int argc, char **argv)
  additional main method.

10.10.1 Detailed Description

Implementation of the gesture recognition service.

Author:
Sven Hennauer (`hennauer@in.tum.de`)

Id:
gesturerecognition.cpp,v 1.21 2002/02/05 14:24:01 hennauer Exp

10.10.2 Function Documentation

10.10.2.1 int main (int *argc*, char ** *argv*)

additional main method.

Initializes Corba, adds and starts the service

10.11 gesturerecognition.h File Reference

DWARF (p. 59) service for gesture recognition.

```
#include "DWARF/GestureData.h"
#include "DWARF/PoseData.h"
#include "../templateservicewrapper.h"
#include "../dwarfutil.h"
#include <queue>
```

Namespaces

- namespace **std**

Compounds

- struct **gestureEvent**

  struct containing data from the inertial tracker (converted due to easier use for the gesture recognition).

- class **GestureRecognitionImpl**

  Main class for the gesture recognition service.

Typedefs

- typedef **TemplateServiceWrapper**< POA_DWARF::StartStopService > **GestureRecognitionParent**

10.11.1 Detailed Description

DWARF (p. 59) service for gesture recognition.

This service analyses the orientation data from an inertial tracker mounted on the user's head and recognizes if the user nodded (meaning 'yes') or shook his head (meaning 'no'). To use this service, register the needs 'GestureData' and 'StartStop' with the service manager and call start(). As soon as gesture is detected, a structured event is pushed, containing a 'GestureData' struct. This struct consists of the type of the gesture (yes or no) and a timestamp.

Please stop the gesture recognition with the method stop() as soon as you don't need it anymore.

See also:
GestureRecognitionImpl (p. 142)

Author:
Sven Hennauer (hennauer@in.tum.de)

Id:
gesturerecognition.h,v 1.15 2002/02/05 14:24:01 hennauer Exp

10.12 ManualCalibration.h File Reference

This file is an implementation of the manual calibration service Distributed Wearable Augmented Reality Framework - www.augmentedreality.de.

```
#include "../templateservicewrapper.h"
```

```
#include "DWARF/Tracker.h"
```

```
#include "DWARF/ManualCalibration.h"
```

Compounds

- class **ManualCalibrationImpl**

  the implementation of the ManualCalibration service.

Typedefs

- typedef **TemplateServiceWrapper**< POA_DWARF::ManualCalibrationService > **MCS-Parent**

  The template classes are used to tell the service wrapper, which **DWARF** (p. 59) *interface to inherit to add new interfaces, only the include line has to be added to the templateservicewrapper.h file.*

10.12.1 Detailed Description

This file is an implementation of the manual calibration service Distributed Wearable Augmented Reality Framework - www.augmentedreality.de.

Author:
Andreas Krause - `krausea@in.tum.de`

REMARK: The service name of the service to collect the PoseData from has been hardcoded as 'nmeatrackingmodule'

Id:
ManualCalibration.h,v 1.4 2002/02/06 17:05:55 krausea Exp

Revision:
1.4

10.13 markertracker.cpp File Reference

```
#include "markertracker.h"
#include <DWARF/DetectMarkers.h>
#include <DWARF/StartStop.h>
#include <AR/ar.h>
#include <AR/matrix.h>
#include <iostream>
#include "../MathUtils.h"
```

Functions

- ostream & **operator**<< (ostream &out, double mat[3][4])
- ostream & **operator**<< (ostream &out, double v[4])
- int **main** (int argc, char **argv)

  This is a simple, sample main routine which demonstrates the usage of the ServiceRunner to describe services, their abilities and run the service.

Variables

- double **quatRot0** [] = { 0, 0, 0, 1 }
- double **quatRotX90** [] = { 0.707106781, 0, 0, 0.707106781 }
- double **quatRotZ90** [] = { 0, 0, 0.707106781, 0.707106781 }
- double **quatRotZ180** [] = { 0, 0, 1, 0 }
- double **quatRotZ270** [] = { 0, 0, -0.707106781, 0.707106781 }

10.13.1 Detailed Description

Author:
Daniel Pustka Calculate current position from markers @id

Id:
markertracker.cpp,v 1.11 2002/02/05 17:47:56 pustka Exp

@revision

Revision:
1.11

10.14 MathUtil.java File Reference

implements common Math procedures.

Compounds

- class **MathUtil**
  implements common Math procedures.

10.14.1 Detailed Description

implements common Math procedures.

Author:
The TRAMP Context Team, Martin Wagner

Version:

Revision:
1.11

Date:

Date:
2002/02/02 13:59:53

last author:

Author:
wagnerm

last changed:

Date:
2002/02/02 13:59:53

10.15 MathUtils.cpp File Reference

source file for **MathUtils** (p. 163) class, a collection of mathematical functions.

```
#include "MathUtils.h"
#include <math.h>
#include <iostream.h>
```

10.15.1 Detailed Description

source file for **MathUtils** (p. 163) class, a collection of mathematical functions.

Author:
Gerhard Reitmayr

Header:
/cvs/tramp/tramp1/src/cpp/services/MathUtils.cpp,v 1.12 2002/02/04 12:32:34 pustka Exp

10.16 MathUtils.h File Reference

Header file for **MathUtils** (p. 163) class, a collection of mathematical functions.

Compounds

- class **MathUtils**

  This class implements some utility classes for matrix and quaternion calculations.

Defines

- #define **PI** 3.141592654

10.16.1 Detailed Description

Header file for **MathUtils** (p. 163) class, a collection of mathematical functions.

Author:
Gerhard Reitmayr

Header:
/cvs/tramp/tramp1/src/cpp/services/MathUtils.h,v 1.9 2002/02/04 12:32:34 pustka Exp

10.17 MessageAlerter.java File Reference

implements the **MessageAlerter** (p. 168) Service.

Compounds

- class **MessageAlerter**
  implements the MessageAlerter Service.
- class **MessageAlerter::MessageAlertUser**
  creates MessageAlertUser for receiving structured Events.

10.17.1 Detailed Description

implements the **MessageAlerter** (p. 168) Service.

Needs:

- MessageEvent

Abilities:

Author:
The TRAMP Context Team, Martin Wagner

Version:

Revision:
1.4

Date:

Date:
2002/01/29 18:28:45

last author:

Author:
wagnerm

last changed:

Date:
2002/01/29 18:28:45

10.18 MessageWindow.java File Reference

implements MessageWindows of the **MessageAlerter** (p. 168) Service.

Compounds

- class **MessageWindow**
  implements the Message Windows for the **MessageAlerter** (p. 168) *Service.*

10.18.1 Detailed Description

implements MessageWindows of the **MessageAlerter** (p. 168) Service.

Author:
The TRAMP Context Team, Martin Wagner

Version:

Revision:
1.3

Date:

Date:
2002/01/29 18:28:45

last author:

Author:
wagnerm

last changed:

Date:
2002/01/29 18:28:45

10.19 NMEAConfiguration.cpp File Reference

```
#include "NMEAConfiguration.h"
```

10.19.1 Detailed Description

Author:
Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Implementation of class: **NMEAConfiguration** (p. 178)

Id:
NMEAConfiguration.cpp,v 1.3 2002/02/06 18:51:35 riedel Exp

Revision:
1.3

10.20 NMEAConfiguration.h File Reference

```
#include <util/PlatformUtils.hpp>
#include <util/TransService.hpp>
#include <sax2/DefaultHandler.hpp>
#include <sax2/SAX2XMLReader.hpp>
#include <sax2/XMLReaderFactory.hpp>
#include <framework/XMLFormatter.hpp>
#include "FileWrapper.h"
#include "SerialWrapper.h"
#include "UTM.h"
```

Compounds

- class **NMEAConfiguration**

  An NMEAConfiguration stores all data needed to configure a **NMEATrackingModule** (p. 194). *Since the main purpuse of this class is to wrapp configuration data, those data are simply stored in public attributes. They can be read and changed directly. For details see the discription of the specific attributes. NMEAConfiguration also provides the method* **readConfigFile***()* (p. 180), *which alows to read the whole configuration or parts of it from a XML based configuration file. Therefore it implements the SAX2 interfaces ContentHandler and ErrorHandler.*

10.20.1 Detailed Description

Author:
Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: **NMEAConfiguration** (p. 178)

Id:
NMEAConfiguration.h,v 1.3 2002/02/06 18:51:35 riedel Exp

Revision:
1.3

10.21 NMEAInterpreter.h File Reference

```
#include <stdio.h>
#include <iostream.h>
#include <DWARF/Tracker.h>
#include "../dwarfutil.h"
#include "NMEASentence.h"
#include "NMEASentence_GGA.h"
#include "NMEASentence_HDG.h"
```

Compounds

- class **NMEAInterpreter**

  This class is parser for NMEA data. It has setter methods to specify a input source, and a PoseData which shall be updated according to the NMEA data. Then the read method can be used to parse the next NMEA sentence, and update the PoseData. The implementation of this class is generated by lex. This tool takes care of all the token recognition, and makes the implementation easily adjustable to different NMEA sentences.

10.21.1 Detailed Description

Author:

Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: **NMEAInterpreter** (p. 182)

Id:

NMEAInterpreter.h,v 1.8 2002/02/06 19:47:18 riedel Exp

Revision:

1.8

10.22 NMEASentence.cpp File Reference

```
#include "NMEASentence.h"
```

Functions

- ostream & **operator**<< (ostream &stream, **NMEASentence** &nmea)
  Prints the the data represented by this **NMEASentence** (p. 185) *to a stream.*

10.22.1 Detailed Description

Author:
Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Implementation of class: **NMEASentence** (p. 185)

Id:
NMEASentence.cpp,v 1.3 2002/02/06 18:51:35 riedel Exp

Revision:
1.3

10.22.2 Function Documentation

10.22.2.1 ostream& operator<< (ostream & *stream*, NMEASentence & *nmea*)

Prints the the data represented by this **NMEASentence** (p. 185) to a stream.

The output is structured in a manner specified by the implementing sub class.

Parameters:
stream The ostream to print to.

nmea The **NMEASentence** (p. 185) whose data to print.

10.23 NMEASentence.h File Reference

```
#include <stdio.h>
#include <iostream.h>
#include <string.h>
#include <DWARF/Tracker.h>
#include "../dwarfutil.h"
```

Compounds

- class **NMEASentence**

  Abstract class to represent the data of a NMEA sentence. Derive from this class to handle different types of NMEA sentences. This can be usefull as a helper class while parsing NMEA.

Functions

- ostream & **operator**<< (ostream &stream, **NMEASentence** &nmea)

  Prints the the data represented by this **NMEASentence** (p. 185) *to a stream.*

10.23.1 Detailed Description

Author:
Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: **NMEASentence** (p. 185)

Id:
NMEASentence.h,v 1.5 2002/02/06 18:51:35 riedel Exp

Revision:
1.5

10.23.2 Function Documentation

10.23.2.1 ostream& operator<< (ostream & *stream*, NMEASentence & *nmea*)

Prints the the data represented by this **NMEASentence** (p. 185) to a stream.

The output is structured in a manner specified by the implementing sub class.

Parameters:
stream The ostream to print to.

nmea The **NMEASentence** (p. 185) whose data to print.

10.24 NMEASentence__GGA.cpp File Reference

```
#include "NMEASentence__GGA.h"
```

10.24.1 Detailed Description

Author:

Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Implementation of class: **NMEASentence__GGA** (p. 189)

Id:

NMEASentence__GGA.cpp,v 1.9 2002/02/06 18:51:35 riedel Exp

Revision:

1.9

10.25 NMEASentence__GGA.h File Reference

```
#include <stdio.h>
#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include <DWARF/Tracker.h>
#include "NMEASentence.h"
#include "../dwarfutil.h"
#include "UTM.h"
```

Compounds

- class **NMEASentence__GGA**

  Represents a **NMEASentence** (p. 185) *with the sentence ID GGA (Global Positioning Szstem Fix Data).*

10.25.1 Detailed Description

Author:

Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: **NMEASentence__GGA** (p. 189)

Id:

NMEASentence__GGA.h,v 1.7 2002/02/06 18:51:35 riedel Exp

Revision:

1.7

10.26 NMEASentence__HDG.cpp File Reference

```
#include "NMEASentence__HDG.h"
```

10.26.1 Detailed Description

Author:
Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Implementation of class: **NMEASentence__HDG** (p. 192)

Id:
NMEASentence__HDG.cpp,v 1.6 2002/02/06 18:51:35 riedel Exp

Revision:
1.6

10.27 NMEASentence__HDG.h File Reference

```
#include <stdio.h>
#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include <DWARF/Tracker.h>
#include "NMEASentence.h"
#include "../dwarfutil.h"
#include "UTM.h"
```

Compounds

- class **NMEASentence__HDG**

  Represents a **NMEASentence** (p. 185) *with the sentence ID HDG (Heading - Deviation & Variation).*

10.27.1 Detailed Description

Author:

Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: **NMEASentence__HDG** (p. 192)

Id:

NMEASentence__HDG.h,v 1.4 2002/02/06 18:51:35 riedel Exp

Revision:

1.4

10.28 nmeatrackingmodule.cpp File Reference

```
#include "nmeatrackingmodule.h"
```

Functions

- int **main** (int argc, char **argv)

  This is a simple, sample main routine which demonstrates the usage of the ServiceRunner to describe services, their abilities and run the service.

10.28.1 Detailed Description

Author:
Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Implementation of class: **NMEATrackingModule** (p. 194)

Id:
nmeatrackingmodule.cpp,v 1.18 2002/02/06 18:51:35 riedel Exp

Revision:
1.18

10.29 nmeatrackingmodule.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include <DWARF/Tracker.h>
#include "../templateservicewrapper.h"
#include "../dwarfutil.h"
#include "../MathUtils.h"
#include "FileWrapper.h"
#include "SerialWrapper.h"
#include "NMEAInterpreter.h"
#include "NMEAConfiguration.h"
```

Compounds

- class **NMEATrackingModule**

  This is the main class implementing the NMEA Tracking service. NMEATracking module is implemented as a **DWARF** (p. 59) *service, and describes its needs & abilities according to the* **DWARF** (p. 59) *standard: a: PoseSource a: poseData a: status Since NMEATrackingModule implements the TrackingModule interface, its main purpuse is to calculate PoseData and deliver them to other* **DWARF** (p. 59) *services. Therefore it uses several helper classes to set up its configuration, parse an input source, interpret NMEA messages and transform them to PoseDate in a target coordinate system. All this is part of the method* **mainLoop***()* (p. 194) *which has to be called frequently from a ServiceRunner. Additionally the method* **getPosition***()* (p. 196) *can be invoke by remote* **DWARF** (p. 59) *services, to get the PoseData currently tracked.*

Defines

- #define **INFINITY** -1

Typedefs

- typedef **TemplateServiceWrapper**< POA_DWARF::Tracking::TrackingModule > **TrackerParent**

10.29.1 Detailed Description

Author:

AndreasKrause, Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: **NMEATrackingModule** (p. 194)

Id:

nmeatrackingmodule.h,v 1.11 2002/02/06 18:51:35 riedel Exp

Revision:
1.11

10.30 PersistentDataStorage.java File Reference

implements the perstistent data storage server and proxy in one class.

Functions

- **PersistentDataStorage** (String corbaloc)

  Constructor-method of **PersistentDataStorage** (p. 204)*, registers needs and abilities according to its function as proxy or server. In short, the proxy needs the server, both services register the ability* **PersistentDataStorage** (p. 204) *for being called by other services needing file access.*

- synchronized int **getDataObject** (String id, PersistentDataStorage_dataHolder data)

  retrieves a file as a byte stream inside the second param variable data.value.

- synchronized int **storeDataObject** (String id, byte[] data)

  stores a file given as a byte stream inside the second param variable data.value to the filesystem, using the relative path name in id.

- synchronized int **appendData** (String id, String data)

  appends the given data string to the given file (useable for protcolwriting and logging).

- synchronized int **removeDataObject** (String id)

  removes the addressed file from the data storage.

- synchronized String **getAbsFilename** (String relfname)

  returns the absolute path name of a given relative one.

- synchronized int **getFileList** (String relpath, PersistentDataStorage_dataHolder data)

  used for synchronization with the persistent data storage server.

- String **getStatus** ()

  Status Interface: returns the status of this Service (not registered right now).

- String **getName** ()

  Status Interface: returns the name of this service (not registered right now).

- void **run** ()

  The run methods initially tries to synchronize with the PDS-server in proxy mode.

Variables

- boolean **registered** = false
- **ObjectImporter PdsImporter**

  class instance of PersistentDataStorageImporter.

- **ObjectExporter PdsExporter**

  class instance of PersistentDataStorageExporter.

- **CorbaService cs**
- Thread **terminator**

  This terminator will take care of unregistering the service in case we are killed NOTE: The macos-x VM appear to "forget" to call this sometimes...

10.30.1 Detailed Description

implements the perstistent data storage server and proxy in one class.

Needs:

- **PersistentDataStorage** (p. 204) (in proxy mode to connect from proxy to server)

Abilities:

- PersistentDataStorageProxy (in proxy mode)
- **PersistentDataStorage** (p. 204) (in server mode)

Author:
The TRAMP Session Team

Version:

Revision:
1.18

Date:

Date:
2002/01/31 13:06:29

last author:

Author:
sager

last changed:

Date:
2002/01/31 13:06:29

10.30.2 Function Documentation

10.30.2.1 synchronized int appendData (String *id*, String *data*)

appends the given data string to the given file (useable for protcolwriting and logging).

Parameters:
id see getDataObject for description. If the relative path doesn't exist, the file is created including parent directories - the data string is added. If the file exists the string is added.

data contains the string to append

Returns:
DATA_* values are returned. Please see constant definitions for descriptions.

Author:
sager

Date:
22.1.02

10.30.2.2 synchronized String getAbsFilename (String *relfname*)

returns the absolute path name of a given relative one.

Parameters:
relfname relative pathname

Returns:
absolute path name

getAbsFilename returns the absolute filename of a string representing a relative filename. This is implemented for avoiding to convert a file into a bytestream which is converted back to a file at the recipient's side. (The recipient might access the file itself)

Author:
groher

Date:
20.1.02

10.30.2.3 synchronized int getDataObject (String *id*, PersistentDataStorage_dataHolder *data*)

retrieves a file as a byte stream inside the second param variable data.value.

Parameters:
id the identifier for the data file to transfer. In the current implementation this id is a relative path to DEFFSROOT. It can be either given with a leading slash or without. The id is concatenated to DEFFSROOT to address the requested file inside the persistent data storage.

data is of type PersistentDataStorage_dataHolder which includes the variable 'value' of type byte[], which is used as the file container.

Returns:
DATA_* values are returned. Please see constant definitions for descriptions.

Author:
sager

Date:
22.1.02

10.30.2.4 synchronized int getFileList (String *relpath*, PersistentDataStorage_dataHolder *data*)

used for synchronization with the persistent data storage server.

Parameters:
relpath contains the relative path inside which all files should be listed

data contains the variable data.value used for the file list to be returned

Returns:
DATA_* values are returned. Please see constant definitions for descriptions.

The list of filenames is relative to DEFFSROOT. Be aware that no directories will be returned.

Author:
sager

Date:
20.1.02

10.30.2.5 PersistentDataStorage (String *corbaloc*)

Constructor-method of **PersistentDataStorage** (p. 204), registers needs and abilities according to its function as proxy or server. In short, the proxy needs the server, both services register the ability **PersistentDataStorage** (p. 204) for being called by other services needing file access.

Parameters:
corbaloc the url where we find the service manager

Author:
sager

Date:
2001/01/29

10.30.2.6 synchronized int removeDataObject (String *id*)

removes the addressed file from the data storage.

Parameters:
id see getDataObject for description. The given file is removed.

Returns:
DATA_* values are returned. Please see constant definitions for descriptions.

Author:
sager

Date:
22.1.02

10.30.2.7 void run ()

The run methods initially tries to synchronize with the PDS-server in proxy mode.

Afterwards proxy and server are open for requests.

10.30.2.8 synchronized int storeDataObject (String *id*, byte *data*[])

stores a file given as a byte stream inside the second param variable data.value to the filesystem, using the relative path name in id.

Parameters:

id see getDataObject for description. If the relative path doesn't exist, the file is created including parent directories. If the file exists it is overwritten.

data is of type PersistentDataStorage_dataHolder which includes the variable 'value' of type byte[] containing the file content.

Returns:

DATA_* values are returned. Please see constant definitions for descriptions.

Author:

sager

Date:

22.1.02

10.30.3 Variable Documentation

10.30.3.1 Thread terminator

Initial value:

```
new Thread() {
       public void run() {
               Debug.out(4, "Shutting down");
               if(registered)
                       cs.unregisterService();
       }
   }
```

This terminator will take care of unregistering the service in case we are killed NOTE: The macos-x VM appear to "forget" to call this sometimes...

10.31 posdatasender.cpp File Reference

this class is a wrapper for sending events in event based communication.

`#include "posdatasender.h"`

10.31.1 Detailed Description

this class is a wrapper for sending events in event based communication.

Date:
developed during TRAMP1 2001/02

Author:
Network team

Revision:
1.3

10.32 posedatamodelmapping.h File Reference

needed for combining the PoseData struct members with KalmanFilters Traveling Repair And Maintenance Platform - tramp.globalse.org.

```
#include "kalmanfilter.h"

#include "../templateservicewrapper.h"

#include <DWARF/PoseData.h>

#include "matrixhelper.h"
```

Compounds

- struct **MeasureMap**

  defines the Mapping for each member of the PoseData struct defines the entries of the ModelMapping.

- class **PoseDataModelMapping**

  wrapper to make the PoseData struct 'intelligent' it combines each member of PoseData with a **KalmanFilter** (p. 149) *if Trackers providing the variable are available.*

Defines

- #define **MEASUREDVARIABLES** 6

Enumerations

- enum **variance** { **INFINITELY_UNACCURATE** = 100000000 }
- enum **statevectorrows** { **ABSOLUTEVALUE** = 0, **VARIANCE** = 1, **DISCRETEDIFFERENCE** = 2 }

  determines the semantix of the values in the mTrackerStates matrix.

- enum **values** { **varX** = 0, **varY** = 1, **varZ** = 2, **phi1** = 3, **phi2** = 4, **phi3** = 5 }

  enumerates the measured and filtered variables, elements of the PoseData struct varX..varZ correspond to position[0..2] phi1..phi3 correspont to euler angles of quaternion in orientation [0..3].

10.32.1 Detailed Description

needed for combining the PoseData struct members with KalmanFilters Traveling Repair And Maintenance Platform - tramp.globalse.org.

Author:
Andreas Krause - krausea@in.tum.de

Id:
posedatamodelmapping.h,v 1.6 2002/02/06 17:05:08 krausea Exp

Revision:
1.6

Implementation of class: **PoseDataModelMapping** (p. 207)

10.32.2 Enumeration Type Documentation

10.32.2.1 enum statevectorrows

determines the semantix of the values in the mTrackerStates matrix.

Enumeration values:

VARIANCE absolute value returned by tracker.

DISCRETEDIFFERENCE the variance returned by the tracker.

10.33 SerialWrapper.cpp File Reference

```
#include "SerialWrapper.h"
```

10.33.1 Detailed Description

Author:

Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Implementation of class: **SerialWrapper** (p. 234)

Id:

SerialWrapper.cpp,v 1.6 2002/02/06 18:51:35 riedel Exp

Revision:

1.6

10.34 SerialWrapper.h File Reference

```
#include <stdio.h>
#include <iostream.h>
#include <fcntl.h>
#include <unistd.h>
#include <termios.h>
#include "FileWrapper.h"
#include "../dwarfutil.h"
```

Compounds

- class **SerialWrapper**

  Implements a **FileWrapper** (p. 125) *for SerialPorts. Many additional properties can be used to configure the SerialPort properly.*

10.34.1 Detailed Description

Author:
Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: **SerialWrapper** (p. 234)

Id:
SerialWrapper.h,v 1.5 2002/02/06 18:51:35 riedel Exp

Revision:
1.5

10.35 shmreader.cpp File Reference

this class is a wrapper for retreiving a key to the control block of a shared memory segment created by an existing shmwriter, the reader then uses the key to get read access to the shm segment.

`#include "shmreader.h"`

10.35.1 Detailed Description

this class is a wrapper for retreiving a key to the control block of a shared memory segment created by an existing shmwriter, the reader then uses the key to get read access to the shm segment.

Date:
Created on January 23, 2001

Author:
fischerj and richter

Revision:
1.3

Id:
shmreader.cpp,v 1.3 2002/02/01 14:12:20 fischerj Exp

10.36 shmreader.h File Reference

The header file of the shared memory reader wrapper class.

```
#include <COS/CosNotifyChannelAdmin.hh>
#include <iostream>
#include <vector>
#include <DWARF/ServiceComm.h>
#include <DWARF/ServiceDescribe.h>
#include <DWARF/ServiceCommNotify.h>
#include <DWARF/ServiceCommShm.h>
#include <DWARF/TrackerData.h>
#include <string>
#include <omnithread.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include "../servicecomm/ringbuffer/ringbufreader.h"
```

Compounds

- class **ShmReader**

10.36.1 Detailed Description

The header file of the shared memory reader wrapper class.

Date:
Created on January 23, 2002

Author:
fischerj and richter

Revision:
1.3

Id:
shmreader.h,v 1.3 2002/02/01 14:12:20 fischerj Exp

10.37 shmwriter.cpp File Reference

this class is a wrapper for creating and writing into a shared memory segment which is identified by the key of the control block.

`#include "shmwriter.h"`

10.37.1 Detailed Description

this class is a wrapper for creating and writing into a shared memory segment which is identified by the key of the control block.

Date:
Created on January 23, 2001

Author:
fischerj and richter

Revision:
1.5

Id:
shmwriter.cpp,v 1.5 2002/02/01 16:35:17 richter Exp

10.38 shmwriter.h File Reference

The header file of the shared memory writer wrapper class.

```
#include <COS/CosNotifyChannelAdmin.hh>
#include <iostream>
#include <vector>
#include <DWARF/ServiceComm.h>
#include <DWARF/ServiceDescribe.h>
#include <DWARF/ServiceCommNotify.h>
#include <DWARF/ServiceCommShm.h>
#include <DWARF/TrackerData.h>
#include <string>
#include <omnithread.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include "../servicecomm/ringbuffer/ringbufwriter.h"
```

Compounds

- class **ShmWriter**

10.38.1 Detailed Description

The header file of the shared memory writer wrapper class.

Date:
Created on January 23, 2002

Author:
fischerj and richter

Revision:
1.4

Id:
shmwriter.h,v 1.4 2002/02/01 14:12:20 fischerj Exp

10.39 TaskflowEngine.java File Reference

@brief.

Compounds

- class **TaskflowEngine**

  public class TaskflowEngine extends TaskflowEnginePOA.

10.39.1 Detailed Description

@brief.

Needs:

- taskflowAction
- persistentDataStorage

Abilities:

- triggerTaskflowTransition
- taskflowEngineStatus

Author:
The TRAMP Session Team

Version:

Revision:
1.39

Date:

Date:
2002/02/04 13:16:32

last author:

Author:
groher

last changed:

Date:
2002/02/04 13:16:32

10.40 TestingGUI.java File Reference

implements the GUI for viewing structured events Created on January 23, 2002, 9:30 AM.

Compounds

- class **TestingGUI**

10.40.1 Detailed Description

implements the GUI for viewing structured events Created on January 23, 2002, 9:30 AM.

Needs:

- PoseData
- UserAction
- Userinput
- GestureData
- MarkerInfo

10.41 trackingfilter.h File Reference

A **DWARF** (p. 59) service implementing an XML configurable extended Kalman Filter Traveling Repair And Maintenance Platform - tramp.globalse.org.

```
#include "DWARF/Tracker.h"
#include "../templateservicewrapper.h"
#include "kalmanfilter.h"
#include "../dwarfutil.h"
#include "../MathUtils.h"
#include "trackingfilterxmlparser.h"
```

Compounds

- class **TrackingFilterImpl**

  the implementation of the tracking filter service.

Typedefs

- typedef **TemplateServiceWrapper**< POA_DWARF::Tracking::TrackingModule > **TrackingFilterParent**

10.41.1 Detailed Description

A **DWARF** (p. 59) service implementing an XML configurable extended Kalman Filter Traveling Repair And Maintenance Platform - tramp.globalse.org.

Author:
Andreas Krause - krausea@in.tum.de

Id:
trackingfilter.h,v 1.11 2002/02/06 17:05:08 krausea Exp

Revision:
1.11

Implementation of class: **TrackingFilterImpl** (p. 258)

Implemented interfaces: Defined in: Service ServiceComm.idl Ability ServiceComm.idl Structured-PushSupplier CosNotifyComm.idl TrackingModule Tracker.idl

10.42 trackingfilterxmlparser.h File Reference

A SAX based XML parser for parsing and constructing the filter<->PoseData mapping Traveling Repair And Maintenance Platform - tramp.globalse.org.

```
#include <string>
#include <stack>
#include <sax/HandlerBase.hpp>
#include <framework/XMLFormatter.hpp>
#include <util/PlatformUtils.hpp>
#include <util/TransService.hpp>
#include <parsers/SAXParser.hpp>
#include "../templateservicewrapper.h"
#include "../dwarfutil.h"
#include "posedatamodelmapping.h"
#include "DWARF/Tracker.h"
```

Namespaces

- namespace **DWARF**

Compounds

- class **DWARF::TrackingFilterSAXHandler**
  XML parser for construction the **PoseDataModelMapping** (p. 207).

Variables

- MessageSender * **ms**

10.42.1 Detailed Description

A SAX based XML parser for parsing and constructing the filter<->PoseData mapping Traveling Repair And Maintenance Platform - tramp.globalse.org.

Author:
Andreas Krause - krausea@in.tum.de

Id:
trackingfilterxmlparser.h,v 1.8 2002/02/06 17:05:08 krausea Exp

Revision:
1.8

Implementation of class: TrackingFilterSAXHandler

10.43 UTM.cpp File Reference

```
#include "UTM.h"
```

10.43.1 Detailed Description

Author:

Sven Hennauer, Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Implementation of class: UTM

Id:

UTM.cpp,v 1.4 2002/02/06 18:51:35 riedel Exp

Revision:

1.4

10.44 UTM.h File Reference

```
#include <math.h>
#include "../MathUtils.h"
```

Compounds

- class **UTM**

10.44.1 Detailed Description

Author:
Sven Hennauer, Michael Riedel - `riedel@in.tum.de` Traveling Repair And Maintenance Platform - tramp.globalse.org

Declaration of class: UTM

Id:
UTM.h,v 1.3 2002/02/06 18:51:35 riedel Exp

Revision:
1.3

10.45 WearableApplication.java File Reference

implements the **WearableApplication** (p. 273) Needs: - GestureData -MarkerData -Taskflow-Action -TriggerTaskflowTransition -User Action -ApplicationMessage Abilities: - UserInput -ApplicationMessage -Status.

Compounds

- class **WearableApplication**

  main class of this file(what else?).

- class **WearableApplication::GestureDataUser**

  creates GestureDataUser for receiving the gesture data as structured Events.

- class **WearableApplication::MarkerInfoUser**

  creates MarkerInfoUser for receiving the marker data as structured Events.

- class **WearableApplication::UserActionUser**

  creates ActionUser for receiving the user action as structured Events.

10.45.1 Detailed Description

implements the **WearableApplication** (p. 273) Needs: - GestureData -MarkerData -Taskflow-Action -TriggerTaskflowTransition -User Action -ApplicationMessage Abilities: - UserInput -ApplicationMessage -Status.

Author:
The TRAMP Application Team

Version:

Revision:
1.1

last author:

Author:
winterm

last changed:

Date:
2002/02/05 14:04:49

Chapter 11

TRAMP Page Documentation

11.1 Todo List

Class MathUtils think about using double in implementation to make it more accurate.

Member NMEAConfiguration::orientationAccuracy This is not used yet. Must be included in the specific implementations of MNEASentence.

Member NMEAConfiguration::positionAccuracy This is not used yet. Must be included in the specific implementations of MNEASentence.

Member NMEAConfiguration::targetCoordinateSystem Later a common baseclass for UTM and other coordinates should be included.

Class NMEAInterpreter No information about past or future NMEA Sentences is evaluated. Such a feature should be added, since there are some NMEA sentences consisting of more than one parts. Currently all NMEA sentences are handled singularily.

Member NMEAInterpreter::getStatus() Currently only a predefined status string is returned. Implement some usefull output here.

Member NMEAInterpreter::read() Support for more NMEA sentences can easily be added later.

Member NMEASentence::validateChecksum() Implement this method! Currently it always returns -1.

Member NMEASentence__GGA::setParam(char *paramVal) Not all parameters are evaluated yet.

Member NMEASentence__GGA::updatePose(PoseData *pose, omni_mutex *poseSemaphore) The accuracy is not calculated from the GPS-data yet, but set to a fixed value.

Before the PoseData is updated the NMEA checksum might be checked. However, this is not that necessary on modern hardware.

Member NMEASentence_HDG::updatePose(PoseData *pose, omni_mutex *poseSemaphore) The accuracy is not calculated from the compas-data yet, but set to a fixed value.

Before the PoseData is updated the NMEA checksum might be checked. However, this is not that necessary on modern hardware.

Class NMEATrackingModule Currently the PoseDate ability is turned of, so an **NMEATrackingModule** (p. 194) does not push PoseData by itself, but only delivers them on request.

Member NMEATrackingModule::describe() Currently the PoseDate ability is turned of, so an **NMEATrackingModule** (p. 194) does not push PoseData by itself, but only delivers them on request.

Member NMEATrackingModule::getPosition(const Time &time) The given time is not evaluated yet. Instead the latest PoseData is always returned.

Member NMEATrackingModule::getStatus() Make the output more individual according to the current status. Currently just a fixed message is used.