



openwms.org
take the lead!

User Documentation

Heiko Scherrer

Tina Russell

Frank Lauer

Michael Schmut

Florian Gyger

0.0.1-SNAPSHOT

Copyright © 2005-2011

Copies of this document may be made for your own use and for distribution to others, provided that you do not charge any fee for such copies and further provided that each copy contains this Copyright Notice, whether distributed in print or electronically.

Preface	iv
1. Scope	iv
2. Target Audience	iv
3. Project and Sub Projects	v
I. Architecture	1
1. The Big Picture	2
2. Presentation Layer	3
2.1. Architecture of the Presentation Layer	3
3. Backend Tier	5
3.1. Architecture of the Backend Tier	5
3.2. The Service Layer	6
3.3. The Data Access Layer	6
II. Installation and Deployment	7
4. Prerequisites for Installation	8
4.1. SpringSource dmServer™	8
4.1.1. Installation of SpringSource dmServer™	8
4.2. PostgreSQL Database Server	9
4.2.1. Post Installation Steps	9
4.3. Using Other Databases	10
5. Download and Installation	11
5.1. Download OpenWMS.org	11
5.2. Unpack and Install OpenWMS.org	11
6. The First Start	12
6.1. Running OpenWMS.org	12
6.2. Logout or Lock the Application	13
III. The CORE module	14
7. Module Management	15
7.1. Overview	15
7.2. Defining new Application Modules	15
7.3. Change existing Application Modules	16
7.4. Manually Loading and Unloading Application Modules	17
8. User Management	18
8.1. Overview	18
8.2. User Details	19
8.3. Change User's Image	20
8.4. Change User's Password	20
8.5. System User Account	21
8.6. User Preferences	21
9. Role Management	22
9.1. Overview	22
9.2. Role Management Screen	22
9.3. Creating a new Role	24
9.4. Modifying an existing Roles	24
9.5. Assigning Grants to a Role	24
9.6. Assigning Users to a Role	25
10. Preference Management	26
10.1. Overview	26
10.2. Preference Management Screen	27
10.3. Create a new Preference	28

10.4. Modify an existing Preference	29
10.5. Delete an existing Preference	29
A.	30

Preface

*The show must go on.
(Queen)*

Welcome to the User Documentation of OpenWMS.org.

This part of the documentation shall be the first manual where you start reading to understand the key concepts of OpenWMS.org warehouse management system and to become an enthusiastic user or a Project Engineer who appreciates the huge amount of work we have done so far to build an open, free and hopefully widely used warehouse management system.

The license agreement (LGPL) is chosen very friendly, so that anyone can download, customize, build and sell the software like he or she wants to. Best open source projects can only survive with a maximum of feedback they earn, also OpenWMS.org relies on your feedback or improvement requests. Feel free to take part here.

1. Scope

The purpose of this manual is to explain the basic concept of OpenWMS.org, how the installation process works and how to customize basic settings regarding arbitrary runtime environments. It is not intended to explain a deeper system configuration or project customization respectively extensions. These topics are explained in more detail in the developer guide.

2. Target Audience

This documentation is written for all users of the OpenWMS.org application. Users, in that context, are not solely actors who work with the application to manage their daily business. But also Project Engineers who are deploying and customizing the system. All of them are addressed to read this manual. Part I and Part II describe the architecture and the installation process hence these two chapters are dedicated to Project Engineers, all other parts are tend to be read from all stakeholders. A list of stakeholders describes different roles and may not be mixed with terms of the applications Role Management that is specified later on.

Following types of users are addressed:

- **Project Engineer:** Stakeholders which are primarily customizing and installing the application. They are in the role of a project consultant with a great knowledge about the application and the technologies behind. Customization and configuration is almost done by themselves. A person acting in this role is the main contact person for the final customer in case of change or support requests. Project Engineers are invited to share their experience in the OpenWMS.org forum and are requested to provide all kind of feedback to wikis or bug trackers. You are the guys who know best of customers business, so please tell us more!
- **Operator:** An Operator is an user who operates on the application to do the daily work. This kind of stakeholder belongs almost always to the final customer. He or she has restricted responsibility to the functionality of the application, that means she is not allowed to access all existing functions of the GUI. She often belongs to a *Role* with limited user rights.

- **System Operator:** The System Operator is a special kind of Operator who has advanced user rights but does not have the same rights like a Project Engineer has. Like an Operator the System Operator belongs to the customer and can inquire change requests or bug reports to a Project Engineer.

3. Project and Sub Projects

OpenWMS.org is an application with the purpose to build warehouse projects. The application itself is split into several sub projects that can be combined together to build up the whole application. The delivered sub projects cover most of the concerns each warehouse project has.

- An **OpenWMS.org CORE** project: This sub project is totally out of scope of any warehouse activities or requirements. It includes all the functionality around an enterprise application like security & user management, connectivity and integration capabilities. The core project is solely independent and can also be used in combination with other projects (not only OpenWMS.org).
- The second sub project is called **OpenWMS.org COMMON** project and aims to be the base for all other sub projects. This project includes common domain classes that exist in all warehouse projects that were built with OpenWMS.org - think about Locations and TransportUnits for example. Every warehouse deals with these terms, probably with different characteristics but all projects know about them and their relationship to each other. Not only a common domain model but also common services that provide business functionality are part of this sub project.
- The third one is called **OpenWMS.org TMS**. Many warehouse projects are built as automatic warehouses where PLCs are used to control the transportation of TransportUnits from a Location A to a Location B. In contrast to manual warehouses, where transports are carried out by the staff directly. All operations that are common in automatic warehouse projects are implemented in the OpenWMS.org TMS sub project. It includes additional domain classes and services that can be extended to project needs as well. Compared to latter sub projects it is not mandatory to use this one.
- The last sub project is called **OpenWMS.org WMS** and targets all top-level warehouse management requirements like order- inventory handling or stock management. It is not mandatory to use this sub project, because there is no need for this in many warehouse projects. Several projects already have an ERP system on top, that is capable to handle WMS functionality.

Each of these sub projects consists of a backend part, that cares about the implementation of the business functionality, and a client part that implements the UI use cases and is used by operators. How all these sub projects (modules) work together is part of the next chapters.

Part I. Architecture

*Vieles das groß ist könnte auch klein sein.
Dieses Gebirge könnte ein Stein sein,
vieles das weit ist könnte so nah sein,
vieles ist Einsicht, wieviel ist Anschein?
(Xavier Naidoo)*

To have a common understanding of the architecture and the principles of application layering, it is important to clarify some terms before we start going into more detail. We often talk about modules, bundles or components in different contexts. Regarding the architecture of OpenWMS.org we want to distinguish between these terms and exactly define what each of these notions means.

- **Bundle (techn.)**: Our definition of a Bundle is of completely technical nature and defines in particular an OSGi bundle. This is naturally spoken a jar file which contains Java classes and can be deployed and started in an OSGi container. Typically each OSGi bundle includes a MANIFEST.MF file with mandatory OSGi header entries - otherwise it would be a simple Java jar file.
- **Module (log.)** A Module is per definition a collection of Bundles and other artifacts, to combine related parts of an Application that belong together logically. A Module is not an artifact that exists physically, it is more or less a definition to group artifacts. The Core Framework could be seen as a Module as well as the TMS or the WMS Module, or something else you define as a logical group of items.
- **Flex Module (techn.)** This kind of artifact is specific to the Flex Framework. Adobe Flex defines a Module as an ActionScript or MXML component that implements the mx.modules.IModule interface. A Flex Module can't run solely and needs to be loaded by the main Flex Application. This is the concept of Adobe Flex modularization and shall be mentioned here for clarification. Each of our delivered Modules includes a Flex Module - so far it needs to have an UI part.
- **Flex Application (techn.)** The Flex Application is the main UI application that may exist only once and can load other Flex fragments like Flex Modules or Flex Libraries. The Flex Application is deployed within a Bundle (OSGi Web Bundle), packaged together with other Flex artifacts.
- **Flex Library (techn.)** A Flex Library is a collection of ActionScript or MXML components that is compiled into the Flex Application or into Flex Modules. Flex Libraries can also be linked at runtime instead of compiling them statically. Doing it that way decreases the application size but slows down the application startup time.
- **Application (log.)** The term Application implies the sum of artifacts be runnable as a single logical unit that can be deployed in the runtime environment. Beside our own artifacts, 3rd party libraries belong to the Application as well. The term Application does not suggest that it is a single technical deployment unit (like a WAR or EAR file), it is just a definition for a group of artifacts.

Chapter 1. The Big Picture

Figure 1.1, “Architecture of the OpenWMS.org Application” is a sketch of the whole Application that represents how all different artifacts fit together and how they interact with each other. Important to notice is the fact that we deal with two different kinds of layers. In the vertical direction the layering is more technical oriented whereas the horizontal layering is driven by the business domain. An application with too many layers implies a high degree of complexity, therefore we reduced the amount of technical layers to a minimum. It's up to you how you structure your new Modules and it is not required to follow this concept strictly. More details about the technical layering follow in the next chapters.

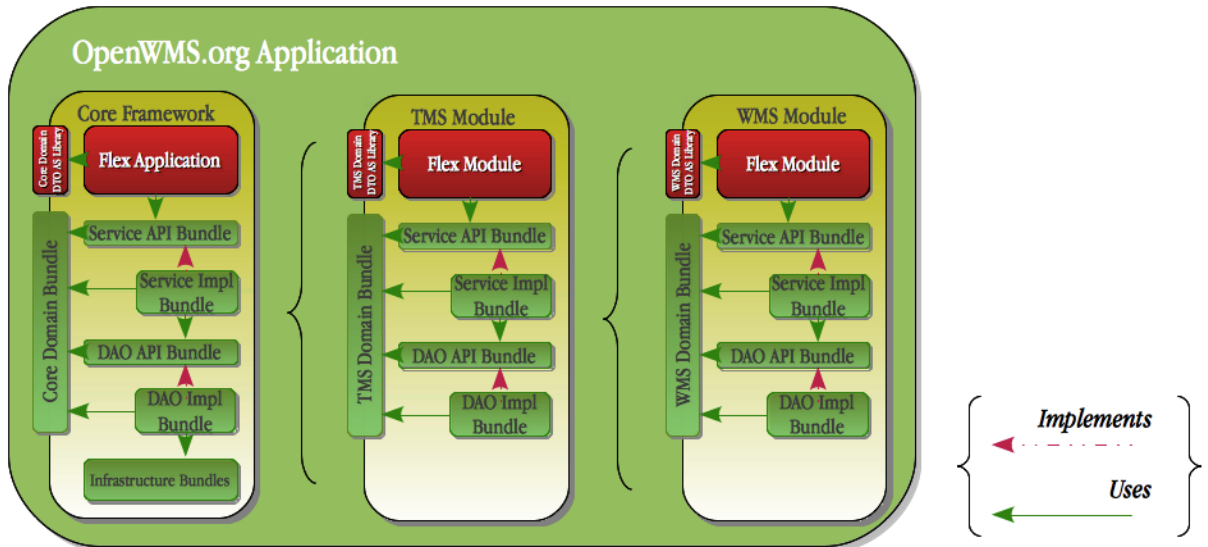


Figure 1.1. Architecture of the OpenWMS.org Application

In horizontal direction the Modules have dependencies on each other. This is important to consider whenever you assemble the Application or deploy/undeploy single Bundles. Module dependencies between the standard Modules and the Core Framework are shown in Figure 2. One of the mayor functional requirements is decoupling the WMS part of the application from the TMS part and vice versa. The Core Framework is the one and only Module where all other Modules rely on.

Chapter 2. Presentation Layer

In this chapter we'll focus on the presentation layer of the existent architecture and find out how the Flex frontend application is combined with the service backend. It is essential to understand how both worlds (Flex and Java) fit together and where things can be configured and customized. We don't talk about the software design of the presentation layer, that's done in the developers reference and shall not be mentioned here.

2.1. Architecture of the Presentation Layer

To better understand the presentation layer, we consider the Web Application Bundle as well, even it is part of the backend tier. Let's start with some details about the architecture of the presentation layer, most important for Project Engineers who setup and customize the system. On the upper right side in Figure 2.1, "Architecture - Presentation Tier" we identify our client-side application logic in form of Flex MXML components and ActionScript classes. Both of these artifacts are compiled into ActionScript classes together with the GraniteDS Service Declaration that is a MXML component as well. It's a simple Flex component that solely includes GraniteDS RemoteObject declarations. Each declared object refers to a Flex Service, configured in the standard Flex Service Configuration file (on the upper left side). In turn a Flex Service definition consists of several Destinations with unique identifiers which are mapped to an AMF Channel. This Channel is the port to the backend application and is configured with an endpoint on the server (URL).

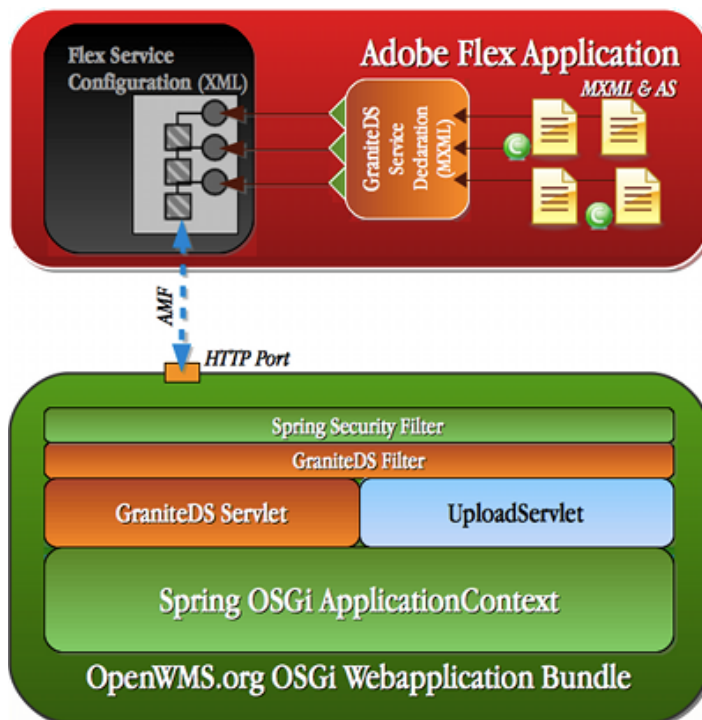


Figure 2.1. Architecture - Presentation Tier

Because we are talking about a web application it is more as natural that a Servlet is mapped to a particular URL pattern on the server side. Two Servlet Filters are registered to scan incoming requests before the GraniteDS Servlet does handle them. The first Servlet Filter is responsible for the integration of Spring Security and delegates to a chain of standard Spring Security handler beans for authentication and authorization purpose. The second Filter in

the chain is the GraniteDS context filter which handles deserialization of incoming AMF3 requests, context population and serialization of the response respectively. Afterwards the request can be processed by GraniteDS' Servlet that resolves a proper Service Invoker from its Service Factory and delegates processing accordingly. We don't go into the details of GraniteDS here hence we pursue with the OSGi service binding. At all, GraniteDS knows about the Flex Service Configuration and the configured Destinations and tries to reference a Spring bean within the Spring ApplicationContext by name. The name of the bean is part of the Destination configuration and must match an injected OSGi service reference. There is no explicit bean instantiation done in the Spring XML bean configuration, all services are imported from the OSGi service registry exclusively.

Chapter 3. Backend Tier

Rushing forward to the backend part of the application. Its quite more interesting hence we spend multiple sub sections for each of the layers. How OpenWMS.org is structured is more important for Project Engineers who assemble the application rather than for Operators and other users. If you are implementing own services or whole Modules than you should join the project and become a developer to access the developer section and read more about the technical internals. In the following sections we explain how the actual layering is set up - less about the design concept and implementation concepts.

3.1. Architecture of the Backend Tier

As already mentioned in the main concept on the website as well as in the overview section we strongly rely on the traditional layering architecture, splitting the business tier into a service and a data access layer and bundling domain objects together to be used in all layers. Persistent Domain Objects like Users, Modules, TransportUnits or Orders are part of the cross-cutting Business Domain Object layer. Each Module comes with a set of it's own domain classes suitable for the Module's domain. Splitting technical layers into domain Modules is a trade-off between high cohesion and lose coupling, thats why each set of domain classes is specific to it's Module.

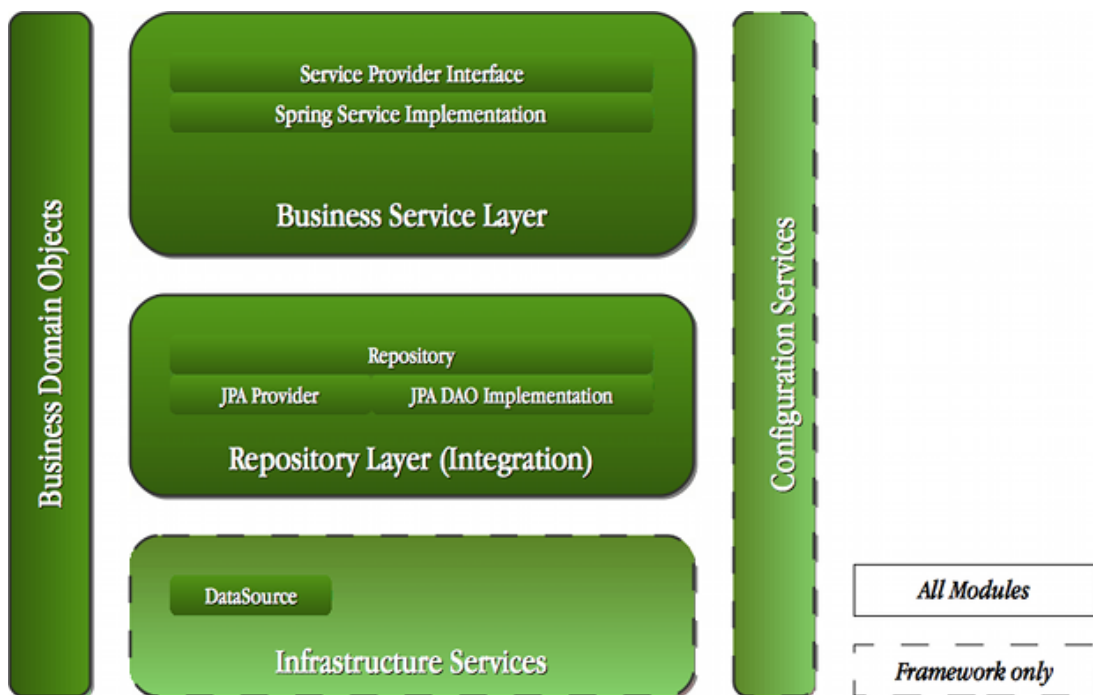


Figure 3.1. Architecture - Backend Tier

Configuration Services on the right side are used in all layers to access global configuration data. A global scope spans all layers in all Modules, not only the Core Framework Module. It is a central service that comes with the framework. Providing an access point to the persistent storage is part of the Infrastructure Service layer that is arranged at the bottom of Figure 3.1, "Architecture - Backend Tier" and exists only once in the whole application - but can be referenced by all other Modules not only the framework.

3.2. The Service Layer

The Service Layer (on top of Figure 3.1, “Architecture - Backend Tier”) provides functionality regarding the business requirements - coarse grained services with transactional behavior. These services interact with other services within the same Module or with the underlying Repository Layer (Data Access Layer) to perform data access operations. A common goal is to minimize dependencies between services, also it is not always possible to eliminate them at all. Services offer their business functionality via a thin interface layer to outer clients (e.g. the web application) and shield the internal implementation. In the past we implemented two different service implementations, one with the Spring Framework and a second in EJB technology. It was quite convenient to switch between both service layer implementations without the need to change something on the client application. Note that Figure 3.1, “Architecture - Backend Tier” does not reflect the deployment concept and does not suggest different Bundles for both parts of the Business Service Layer even if it is so. It is up to you how you structure your new Modules, probably it is more feasible for you to combine service interface and implementation. However it's a good practise to implement against interfaces. No matter how you build your deliverables later on.

3.3. The Data Access Layer

The Data Access Layer, or more fashioned the Repository Layer is a so called Integration Layer because it encapsulates the actual data access operations from the Service Layer and thus increases robustness of the implemented business logic. In former days it was essential to cut-off this functionality from the intrinsic business logic. Lately we already have an abstraction with the standardized Java Persistence API, so you could argue to remove this layer. Our suggestion is, if you feel you write duplicated code or code that just delegates to JPA's EntityManager, then you can omit this layer in your own Modules without doubts. Like in the Business Service Layer we code against an interface definition (Repository) to offer data access functionality. The implementation itself (JPA DAO Implementation) is specific to the JPA standard specification and does not depend on y particular JPA provider (like Hibernate or EclipseLink). But at runtime we need to have an underlying JPA provider, that's why there is another part in the Repository Layer - the JPA Provider. This is always a 3rd party library that comes with OpenWMS.org - by default we are using Hibernate as provider. But it shall also be possible to switch to EclipseLink or another JPA compliant implementation.

Part II. Installation and Deployment

Chapter 4. Prerequisites for Installation

To install and run OpenWMS.org, an application server and a database is needed. You should already have installed a version of Java SE 6 before you follow the next steps.

Server Requirements:

Install everything that is needed to run SpringSource dmServer™, for more information have a look at the SpringSource dmServer™ website [<http://www.springsource.com/products/dmserver/>].

Client Requirements:

Because the client part of OpenWMS.org is implemented with the Adobe Flex framework, the client must be able to install the Adobe Flash Player Plugin [<http://get.adobe.com/flashplayer/>] in a minimum release of version 9.0.x. It is supposed to run the application within a web browser, that is capable to install this plugin. For more information about this please visit Adobe Flash framework [<http://get.adobe.com/flashplayer/>] website.

Most modern web browsers are able to install the Adobe Flash Plugin [<http://get.adobe.com/flashplayer/>]. On handheld devices like smartphones or label scanners it is not a standard at all. Please check the list of supported devices on Adobes website [<http://www.adobe.com/products/flashplayer/systemreqs/>] directly. Nevertheless if you have to support a device that is not capable with the Flash Player Plugin, you can rely on the OpenWMS.org backend services and implement the necessary views in a technology of your choice.

Never forget: Implementation of screens, views and dialogs must not be time consuming and should be done with the fewest effort as possible, because frontend technology often changes ;-)

4.1. SpringSource dmServer™

OpenWMS.org is dedicated to run on SpringSource dmServer™ application server (for simplification called dmServer from now). This server guarantees a stable and reliable environment to run OSGi applications on. It makes life easier when you have to identify OSGi dependency problems and it comes with sophisticated tracing and failure capture (FFDC) mechanism. It is delivered as a pre-configured OSGi container that saves your time compared to setting up a base OSGi implementation like Eclipse Equinox or Knopplerfish. All necessary bundles are already included, like Apache Tomcat web container, Spring Dynamic Modules for OSGi and obviously the Spring Framework itself. SpringSource dmServer™ support different application deployment formats like PAR, Plans and Library Definition files that are mandatory to run OpenWMS.org and will hopefully become a standard in the next OSGi specification.

OpenWMS.org is currently tested on version 2.0.5.RELEASE.

4.1.1. Installation of SpringSource dmServer™

To download and install dmServer please visit the download site [<http://www.springsource.org/dmserver/>] or click the direct link to version 2.0.6.RELEASE [<http://dist.springframework.org/release/DMS/springsource-dm-server-2.0.6.RELEASE.zip>]. Just

unzip the downloaded ZIP file to an arbitrary location on your harddrive to finish the installation. SpringSource propose to create an environment variable called `SERVER_HOME` that points to the installation directory (this variable is referenced in the rest of this document):

```
$ export $SERVER_HOME=<PATH TO UNZIPPED DM SERVER>
```

Now you should be able to run dmServer:

```
$ cd $SERVER_HOME/bin
$ ./startup.sh
```

To stop a running dmServer instance press CTRL-C.

4.2. PostgreSQL Database Server

PostgreSQL Database Server is the one we have chosen to develop OpenWMS.org. Of course OpenWMS.org is database independent so you can switch to a database of your choice. We have chosen PostgreSQL DB because it is a mature and proven database server formerly branched from IngresDB, that is now opensource and comes with a proper administration tool. Installation and running the server is described in the PostgreSQL documentation [<http://www.postgresql.org/docs/8.4/static/index.html>] . Simply download [<http://www.postgresql.org/download/>] the latest version (9.x) and install it on your machine.

4.2.1. Post Installation Steps

When you have installed the database server and it starts up correctly than you have to add a database user and an instance to run OpenWMS.org. Extend your environment `$PATH` settings to the `<POSTGRES_INSTALL_DIR>/bin` directory or change your current directory to that location in order to create all the resources.

Create a new LOGIN ROLE with name and password set to `OPENWMS` (The standard PostgreSQL database username is set to `postgres`. If you have changed this before, you must also change the `-U` parameter in the following commands).

```
$ ./createuser -Upostgres -W -P OPENWMS
```

First of all you are prompted to specify a new password for the login role `OPENWMS`, set it to `OPENWMS` and proceed. The new role doesn't need to inherit superuser grants.

As next we are going to create a database instance within your server installation. Probably you will install other instances e.g. for a test environment on the same server, but for now we are only creating one instance.

```
$ ./createdb -Upostgres -W -OOPENWMS OPENWMS
```

Executing this command will create a new database instance with the name `OPENWMS` owned by the recently created user `OPENWMS`. Congratulations, the database is now installed properly to run OpenWMS.org!

The values of username, password and database name used in the commands above are the default values. Of course you can choose different names and configure the application to take that ones.

4.3. Using Other Databases

By default OpenWMS.org is configured to connect to a PostgreSQL Database Server. Other databases are supported as well but have to be configured in the OSGi Configuration Service properties file that is shipped with the application. All application properties are stored in one Java properties file (`($SERVER_HOME/repository/usr/org.openwms.common.infrastructure.configuration-x.y.z.properties)`). Properties regarding the database connectivity are prefixed with 'jdbc.'

```
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql:OPENWMS
jdbc.username=OPENWMS
jdbc.password=OPENWMS
jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect
jdbc.database=POSTGRESQL
jdbc.properties=hibernate.connection.compatible=7.1
jdbc.persistenceUnit=OpenWMS-test-durable
...
```

Detailed information about the intention of each property can you find in the developer manual.

Chapter 5. Download and Installation

5.1. Download OpenWMS.org

All versions of OpenWMS.org can you find here [<http://www.openwms.org/docs/downloads.html>] (<http://www.openwms.org/docs/downloads.html>)

Please visit the download page and download the latest version of OpenWMS.org.

5.2. Unpack and Install OpenWMS.org

Unzip the downloaded file to your harddrive.

Inside the unzipped folder you find two directories containing all the bundles that have to be installed in dmServer. Please copy all files of the `bundles` directory to dmServer's user repository (usually to `$SERVER_HOME/repository/usr`). Within the `pickup` directory you find the OpenWMS.org web application that has to be copied to dmServer's hot-deployment directory (usually to `$SERVER_HOME/pickup`).

```
$ cp bundles/* $SERVER_HOME/repository/usr
```

```
$ cp pickup/* $SERVER_HOME/pickup
```

In a third directory you find the license agreement files of all third party libraries that are shipped with OpenWMS.org. Please read and accepts the license agreements before using OpenWMS.org.

Last but not least, you have to tell dmServer to start all bundles listed in the OSGi Plan file `org.openwms.app.plan`. Therefor you must add this plan to the list of plans in dmServers config file `com.springsource.kernel.userregion.properties`.

Open `$SERVER_HOME/config/com.springsource.kernel.userregion.properties`

Scroll down to the last line in this file where the `initialArtifacts` are listed and add the `org.openwms.app.plan` file. It should looks like:

```
initialArtifacts=..., repository:plan/org.openwms.app.plan, ...
```

Without this entry dmServer will not start our bundles in the repository directory, but will start the Web Application. Due to the lack of backend services the OSGi container won't be able to resolve the service dependencies of the frontend application and will throw an exception.

Chapter 6. The First Start

6.1. Running OpenWMS.org

If you've completed all steps successfully you should now be able to start SpringSource dmServer™ with the installed OpenWMS.org application.

```
$ ./$SERVER_HOME/bin/startup.sh
```

(We assume that you have installed the server and the database with default configuration).

After dmServer's Kernel is loaded, all application specific bundles are installed and shall come up without any errors. When everything is started correctly you should see at least the line:

```
Started bundle 'org.openwms.zz.client.flex.wrapper' version 'x.y.z'
```

This means, dmServer was able to install and start the web application bundle correctly. Open a web browser and switch to <http://localhost:8080/openwms> to verify that your installation is successful. The login dialog of OpenWMS.org should appear now.



Figure 6.1. Application Login Dialog

6.2. Logout or Lock the Application

After you have successfully logged in the application you see some vertical oriented buttons in the header part of the portal. These buttons will not change, doesn't matter which screen you open. The 'Logout' button is used to logout the currently logged in *User* from the application. All screens are closed, non-saved data is discarded and the *User* is logged out and forwarded to the Login screen (Figure 6.1, "Application Login Dialog") again. Another way to leave the application is to just lock the application for other *Users*. In this place, an opened screen closes as well, but you can continue your work when logging in again. Unsaved data is still there after login. Notice, that only the *User* who locked the screen can unlock it again, other *Users* can not login to a locked application.



Figure 6.2. Application Header

Part III. The CORE module

The OpenWMS.org CORE module provides management functionality regarding the base framework requirements like *User*, *Role* and *Module* management. All enterprise applications need to have some kind of security management. Authentication and authorization are essential features that have to be handled by the main application itself. In particular, fine-grained security level adjustment must be possible as well.

Chapter 7. Module Management


7.1. Overview

From the main application actions bar open Application->Modules to display the Module Management view. On this screen a System Operator or Project Engineer defines all *Modules* of the project. One of the most important requirements on OpenWMS.org is modularization. That's why we often talk about modules either in the frontend part of the application or in the service layer (backend). An *Application Module* is a defined set of views, services and domain objects that belong logically together. Some modules have dependencies to other modules where they operate on.



Figure 7.1. Module Management View

7.2. Defining new Application Modules

On the main Module Management View press the button New  to define a new Application Module. Now you have to enter a minimum set of information that is necessary to locate and run the Application Module. Required fields like the Modulename and the URL are marked




with a red asterisk and must be provided to start up the module correctly. After you have done your changes click Save  to save the entries in the database.

Table 7.1. Definition of Module properties

Input field	Description
Modulename	The module name is an unique identifier of the <i>Module</i> within the application. Please provide a proper name that describes the <i>Module</i> in a short term. For example: COMMON Module
URL	The URL is mandatory to locate and load the <i>Module</i> . It is not mandatory that a <i>Module</i> is shipped with the application, it can also be loaded from another application domain. If the <i>Module</i> is packaged with the application, just use the filename of the Shockwave (SWF) file, e.g. org.openwms.common.client.flex.swf, including the file extension (.swf). If your <i>Module</i> is stored in a different application domain like the main application than you have to provide the full URI (unique resource identifier), e.g. http://www.myCompany.com/openwms/org.openwms.common.flex.client.swf. Under these circumstances please notice that the application security policy must allow access to your foreign domain (see Crossdomain configuration)
Description	The description field is not required, but you should provide short information about the Application Module, e.g. for what it is needed and what dependencies it has to other modules.
Load on Startup	Check this field if you want to start your <i>Module</i> the next time the application starts up.




7.3. Change existing Application Modules

In the same view you can do changes on existing module definitions. Just choose the Application Module you want to change and the detail information of the chosen module is shown in the text fields next to the list of modules. Change the properties and press Save  to write the information to the persistent storage. Be aware of uniqueness of the Modulename and the URL identifier.

Removing an existing module can be done that way too, just choose the module to delete and press the Delete  button.

Your Changes will take effect the next time you reload the application.

7.4. Manually Loading and Unloading Application Modules

To try out if a module is able to be loaded on startup, you should try to load it manually before. Just select the Module you want to load and press the Load  button. If the module was already loaded, the buttons icon changed  and is used to Unload. Pressing the Unload button  unloads the current selected module. After an Application Module was successfully loaded into the application domain you should notice that the main application actions bar is populated with a couple of menu items provided by the new module. What you can't recognize so far is that the list of views was updated, too. After unloading a module, the main application actions bar is reorganized and only show the menu items of all currently loaded modules.

Chapter 8. User Management



8.1. Overview





From the main application actions bar choose Application->Users to open the User Management View. After the view is loaded a list of *Users* appears on the left-hand side. In the right panel all details of a selected *User* are shown. In a completely new installation no *Users* are defined, hence the list and the detail form are both empty. The User Management is used to add new *Users* and to modify or delete existing ones. Furthermore you can set User details (*UserDetails*), change a *User's* password, or add an image to an *User*.



Figure 8.1. User Management View

Table 8.1. Actions bar of the User Management View

Icon	Description
	Add a new <i>User</i> .
	Delete an existing <i>User</i> , or an <i>User</i> entry that was created but not saved before.

Icon	Description
	After changing an <i>User's</i> data you can save your changes by pressing the save button.
	Reload and refresh all <i>Users</i> from the persistent storage.
	Change the password of an existing <i>User</i> . The operation cannot be done on an unsaved <i>User</i> .
	Change the image of an existing <i>User</i> .

8.2. User Details

By default some detail information can be attached to each *User* profile. Required fields are marked with a red asterisk and must be set to successfully save the changes in the persistent storage. The Username is a system-wide unique identifier that can be changed at any time but must be set initially.

Table 8.2. Description of User Details input fields

Input field	Description
Username	Unique identifier of the <i>User</i> .
Fullname	Fullname of the <i>User</i> .
Description	Description text of the <i>User</i> .
Phone	The <i>Users</i> phone number.
IM	The <i>Users</i> IM adress.
Department	The department the <i>User</i> is working for.
Office	The office the <i>User</i> is working in.
Sex	Male or female.
Comment	Some comment stored with the <i>User</i> .
Enabled	The <i>User</i> is only able to login when he or she is enabled. This way it is possible for a System Operator to forbid an <i>User</i> to login.
Locked	An <i>User</i> is being locked by the application automatically after he or she tries to login with an invalid password several times. The number of invalid retries is set in the system properties.
Expiration Date	A System Operator can define an expiration date for an <i>User</i> to allow login for a defined

Input field	Description
	period of time. When this field is left empty the account never expires.
Roles	Usually an <i>User</i> is assigned to <i>Roles</i> . Assigned <i>Roles</i> are listed in this list but cannot be changed here. Changing an <i>User's Roles</i> is done with the Role Management Screen.

8.3. Change User's Image

After clicking the Change Button (📁) a popup window appears and you are prompted to enter a valid path to an image file, that is uploaded to the server afterwards. When the upload succeeds the new image file is directly assigned to the selected *User*, there is no need to save the *User* anymore. The width of the image is resized to max. 100 pixel and a height to max. 150 pixel.

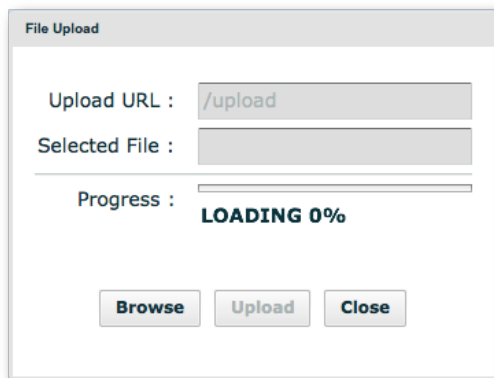


Figure 8.2. Changing the Users image

8.4. Change User's Password

To change the *Users* password, select an *User* and choose Change Password (🔑) from the actions bar. In the dialogue Figure 8.3, "Changing the Users password" reset the password and click on Change.

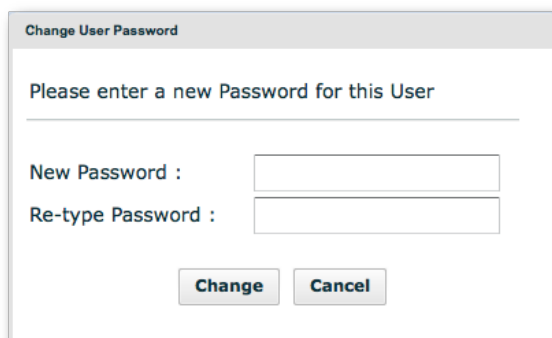


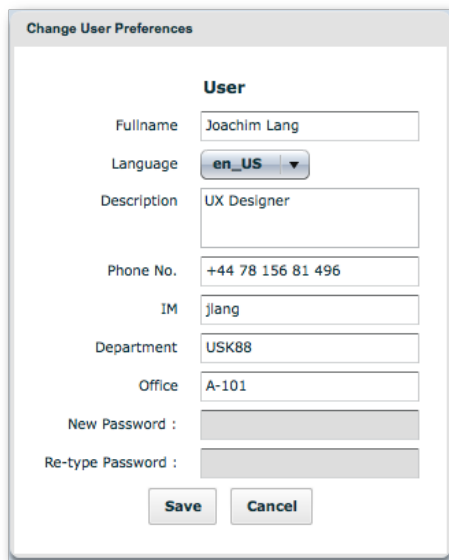
Figure 8.3. Changing the Users password

8.5. System User Account

A *System User* account exists by default. This particular *User* profile is especially dedicated to Project Engineers and shall only be used for installation and maintenance purpose. The *System User* has full access to the application without any security restrictions. By default Username and password of the *System User* are both set to `OPENWMS`. How to change these credentials is described in the Developer Manual.

8.6. User Preferences

Once you have logged in the application, your username is displayed in the upper right corner of the application header. Click on your username to open a dialogue with your personal user settings (Figure 8.4, “Change User Preferences”). All *Users*, beside the *SystemUser*, can manage their own settings, like setting a default language or change the *UserDetails*. Changing the current password to login is possible as well.



The screenshot shows a dialog box titled "Change User Preferences" for a user named "Joachim Lang". The dialog contains the following fields and values:

- Fullname: Joachim Lang
- Language: en_US (dropdown menu)
- Description: UX Designer
- Phone No.: +44 78 156 81 496
- IM: jlang
- Department: USK88
- Office: A-101
- New Password: (empty field)
- Re-type Password: (empty field)

At the bottom of the dialog are "Save" and "Cancel" buttons.

Figure 8.4. Change User Preferences

Chapter 9. Role Management

9.1. Overview

Generally the term *Role* is a synonym for a group of *Users*. Having some *Roles* in each project is always recommended. For example, you could define a separate *Role* for Operators, System Operators and Project Engineers. Doing so you could assign *Users* to *Roles* and afterwards assign *Security Objects*, like *Grants* to each *Role*. Doing it that way is much more convenient and requires less administration than assigning each *Grant* to each *User*. New *User* accounts can easily tied to already existing *Roles*. Defining a set of *Grants* to a *Role* is done once per *Role*. This is standard security management in multi-user enterprise applications.

The *Grants* in Table 9.1, “Table of Grants regarding Role Management” determine the actions an *User* can perform on *Roles*. All *Grants* are stored in the `secured-objects.xml` file of the main Flex Application Module.

Table 9.1. Table of Grants regarding Role Management

Grant Key	Description
APP_Role_Management	Permission to open the Role Management Screen.
APP_add_roles_button	Ability to add new <i>Roles</i> .
APP_save_role_button	Ability to save changes on a <i>Role</i> .
APP_remove_role_button	Ability to remove an existing <i>Role</i> .
APP_assign_grants_button	Assign to or remove <i>Grants</i> from a <i>Role</i> .
APP_assign_users_button	Assign to or remove <i>Users</i> from a <i>Role</i> .

9.2. Role Management Screen

From the main application actions bar click Application->Roles to open the Role Management view. Purpose of this management view is to declare security *Roles* and assign individual *Grants* and *Users* to each of them.

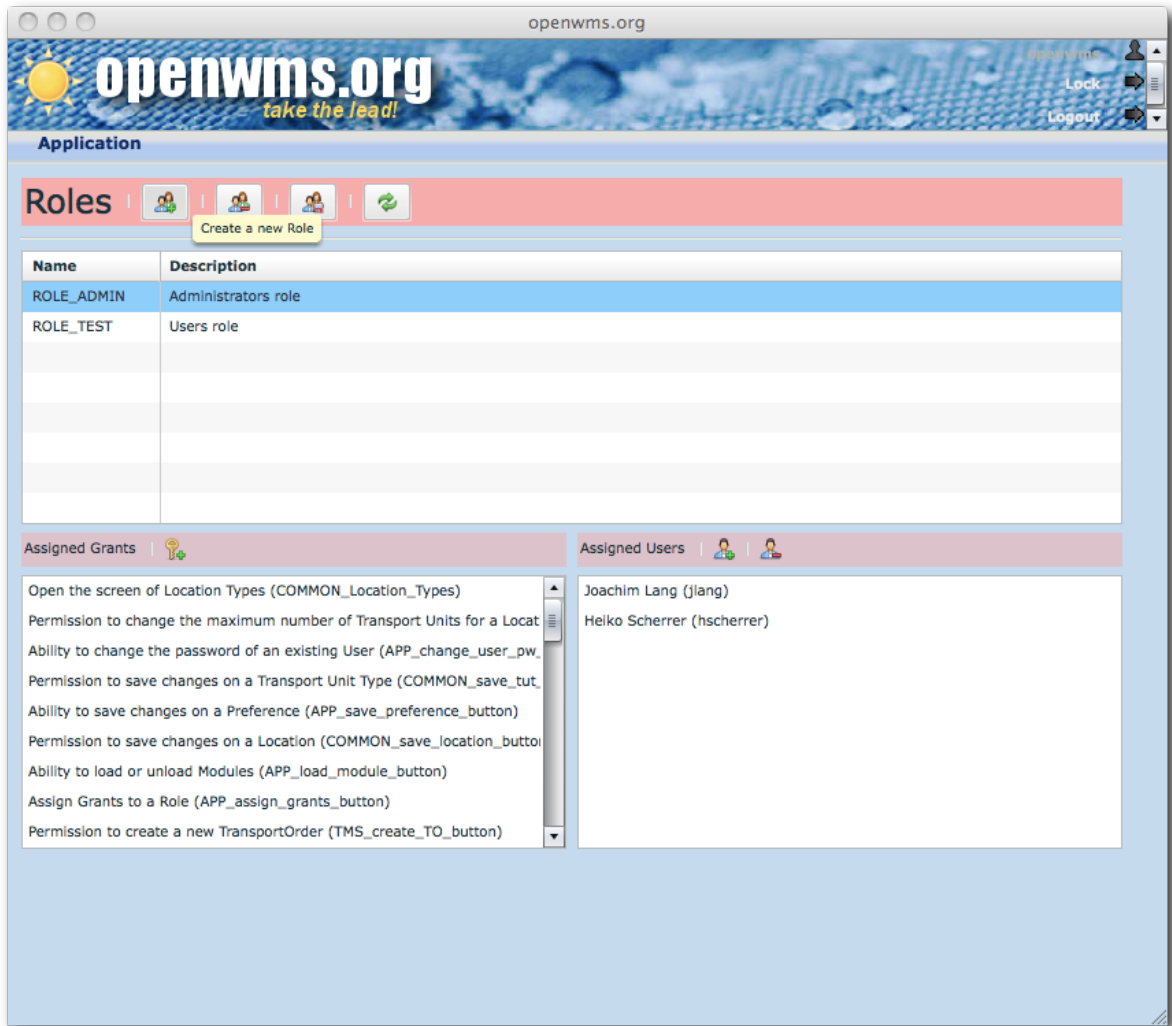




Figure 9.1. Role Management View

Table 9.2. Actions bar of the Role Management View

Icon	Description
	Open a dialogue to create a new <i>Role</i> with name and description.
	Delete an existing <i>Role</i> .
	After double-clicking a <i>Role</i> you can change data and press Save to save your changes.
	Reload and refresh <i>Role</i> information from the persistent storage.
	Assign <i>Users</i> to a selected <i>Role</i> . Opens a dialogue to add <i>Users</i> to the <i>Role</i> .
	Select already assigned <i>Users</i> you want to remove from the selected <i>Role</i> and press this button to remove their <i>Role</i> membership immediately.

Icon	Description
	Select a <i>Role</i> and press this button to assign one or more <i>Grants</i> to the <i>Role</i> . An dialogue opens to assign or remove <i>Grants</i> from a <i>Role</i> .

9.3. Creating a new Role

To create a new *Role* press the 'Create' button of the actions bar (). In a simple dialogue, you have to provide the name of the new *Role* and an optional descriptive text. After the *Role* is created the roleName is prefixed with 'ROLE_'.

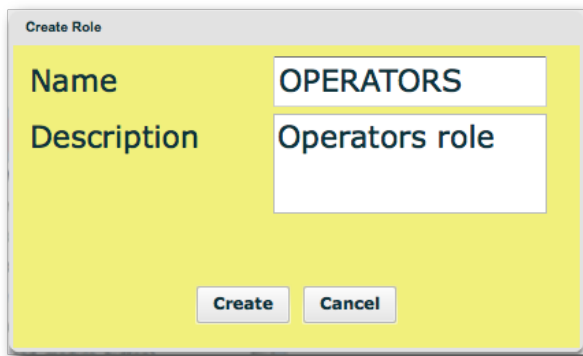


Figure 9.2. Create a Role

9.4. Modifying an existing Roles

Existing *Roles* can also be modified. To change the role name or description, just double click the *Role* to open a dialogue, like shown in Figure 9.3, "Modify a Role", where you can change the values as desired.

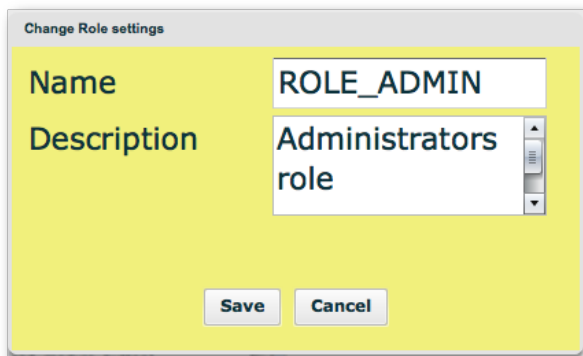



Figure 9.3. Modify a Role

9.5. Assigning Grants to a Role

If you already have a *Role* defined, you are now able to assign *Grants* to this *Role*. Just select the *Role* and press the 'Assign Grants' button (). A dialogue opens that lists all non-

assigned *Grants* on the left side and all currently assigned *Grants* on the right side. Choose the *Grants* you want to add or remove to a *Role* and press one of the shift buttons in the middle. After you confirm the dialogue you have to save the *Role*. Your changes do not take affect without saving the *Role* explicitly, because you could have done changes to the *Role* before.

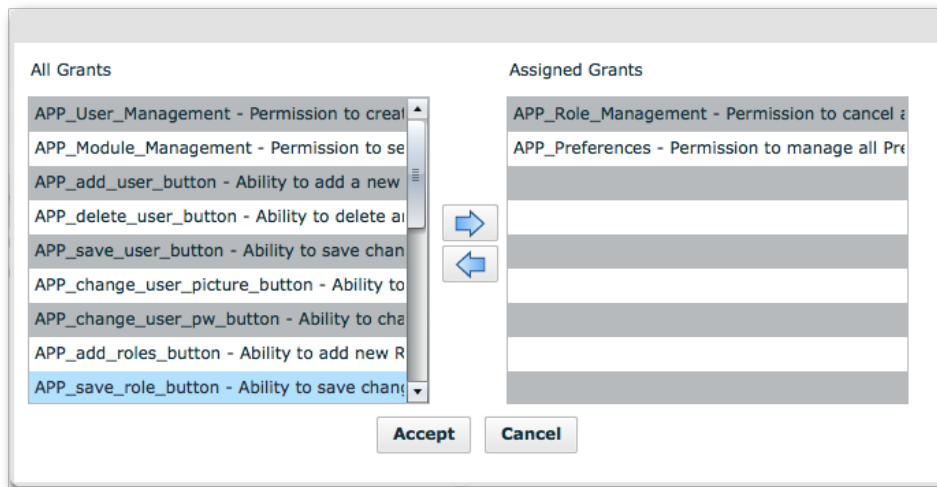


Figure 9.4. Assign Grants to a Role

9.6. Assigning Users to a Role

Role Management does only make sense when you assign *Users* to *Roles* and manage access control through *Roles*. So go ahead and add some *Users* to a defined *Role*. Press the 'Assign Users' button (👤➕) and do it like you did before. Nearly the same dialogue opens where you can add or remove one or more *Users* from a selected *Role*.

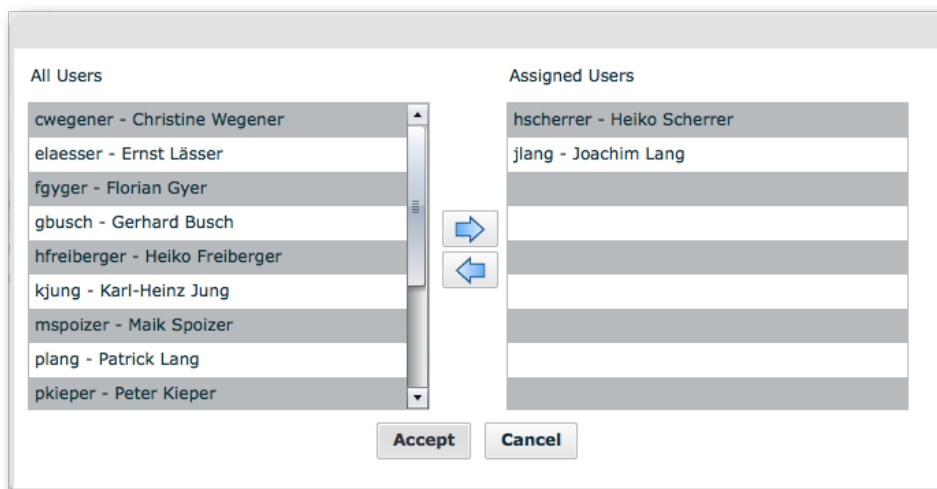


Figure 9.5. Assign Users to a Role

Chapter 10. Preference Management

10.1. Overview

All enterprise applications have some kind of configurable application preferences or settings, nevertheless whether they come from a file, a database or somewhere else. In OpenWMS.org we call these setting parameters *Preferences*. A *Preference* is always in a certain scope, valid for a particular part of the application. So far, we have defined four different types of *Preferences*:

- *ApplicationPreference*

An *ApplicationPreference* is in global application scope. It is visible from all parts of the application and valid for all entities until it is overruled by a more particular *Preference*

- *ModulePreference*

A *ModulePreference* is only visible and valid for the *Module* it is assigned to.

- *RolePreference*

This kind of *Preference* is only valid for a certain *Role*. Each *Role* can have a set of *RolePreferences* defined.

- *UserPreference*

An *User* can have her own *Preferences*, too. This type of *Preference* which is assigned to a particular *User* is a so called *UserPreference*.

An authorized *User* is able to create new *Preferences*, modify or delete existing ones. The *Grants* in Table 10.1, “Table of Grants regarding Preferences Management” determine the actions an *User* can perform. All *Grants* are stored in the `secured-objects.xml` file of the main Flex Application Module.

Table 10.1. Table of Grants regarding Preferences Management

Grant Key	Description
APP_Preferences	Permission to open the Preferences Management Screen and manage all <i>Preferences</i> .
APP_add_preference_button	Ability to add new <i>Preferences</i> .
APP_delete_preference_button	Ability to remove <i>Preferences</i> .
APP_save_preference_button	Ability to save changes made on a <i>Preference</i> .

10.2. Preference Management Screen

From the main application actions bar click Application->Preferences to open the Preference Management view. On the left-hand side of the screen all Preferences are grouped by their type in an accordion container. Details of a selected Preference are shown right beside.

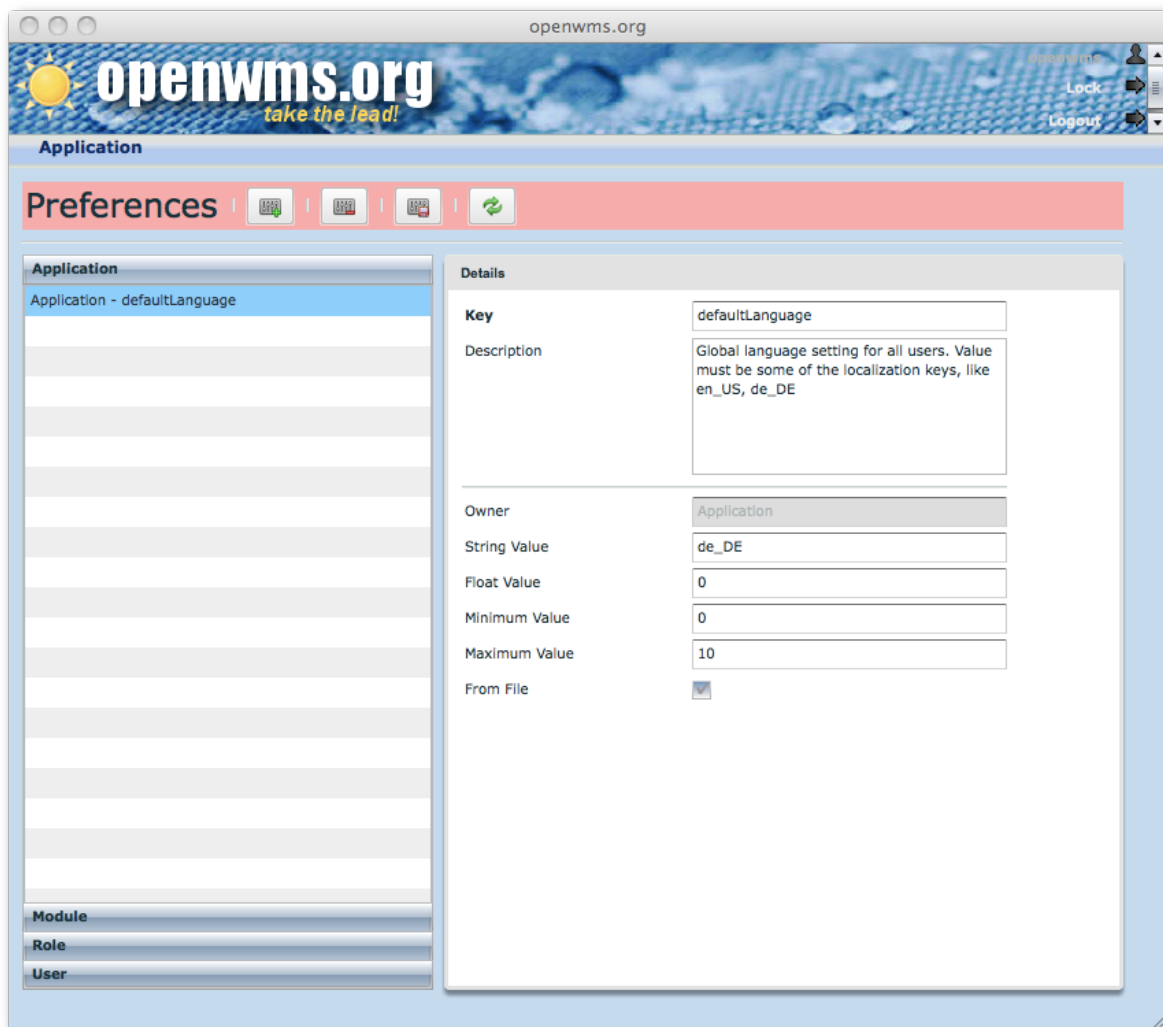


Figure 10.1. Preference Management View

Table 10.2. Actions bar of the Preference Management View


Icon	Description
	Open a dialogue to create a new <i>Preference</i> with all the required information like the preference key, a value and an owner (like <i>Application</i> , <i>Module</i> , <i>Role</i> or <i>User</i>).
	Delete an existing <i>Preference</i> .
	Save changes you did on the currently chosen <i>Preference</i> .
	Reload and refresh all <i>Preferences</i> from the persistent storage.

Each *Preference* must have at least a key and a type assigned. *Preferences* of type *ModulePreference*, *RolePreference* and *UserPreference* additionally need to have an owner. The key and owner fields are unique along all defined *Preferences* and cannot be declared twice - be aware of that. For example, it is not allowed to define two *Preferences* with a key 'defaultLanguage' within the same *Module* 'CORE', but it is still allowed to have this *Preference* defined for two separate *Modules* 'CORE' and 'Common'.

Table 10.3. Description of Preference Details

Input field	Description
Key	Unique identifier of the <i>Preference</i> within the type of <i>Preference</i> .
Description	Description text of the <i>Preference</i> .
Owner	An owner where the <i>Preference</i> belongs to.
String Value	An alphanumeric value.
Float Value	A numeric value.
Minimum Value	A possible minimum value. Used to define a range.
Maximum Value	A possible maximum value. Used to define a range.
From File	Indicates, whether the <i>Preference</i> is defined in a file or created by the <i>User</i> .

10.3. Create a new Preference

Clicking the button to create a new *Preference* () opens the dialogue shown in Figure 10.2, "Create a Preference" to add a *Preference* and store it in the database. Actually the same fields can be set like in the details page of the overview screen Figure 10.1, "Preference Management View".

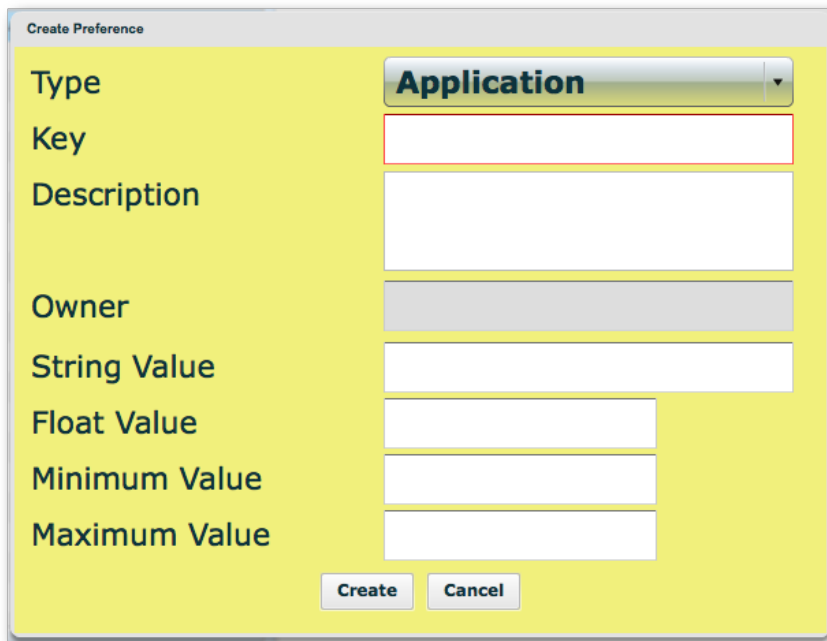




Figure 10.2. Create a Preference

10.4. Modify an existing Preference

To make changes on an existing *Preference*, select that one from list on the left side of the screen. On the details page you can modify the values of the selected *Preference*. Notice that the key of an existing *Preference* cannot be changed after it is created. To save your work back to the system click on the 'Save' button () in the upper actions bar.

10.5. Delete an existing Preference

Removing an existing *Preference* is simple as well. Just select the *Preference* to remove and click on the 'Remove' button (). No confirmaton will be shown, hence be careful with deletion of existing entries.

Appendix A.

Glossary

ApplicationPreference	An ApplicationPreference is used to store a configuration setting in application scope. The table model of an ApplicationPreference spans an unique key over the columns C_TYPE and C_KEY. It's counterpart in the context of JAXB is the applicationPreference element.
Barcode	A Barcode is a printable item with an unique identifier to label TransportUnits. The identifier has a defined number of characters whereas these characters are aligned either left or right. Non filled positions of a Barcode are padded with a so called padding character.
Email	An Email represents the email address of an User.
Grant	A Grant gives permission to access some kind of application object. Grants to security aware application objects can be permitted or denied for a certain Role, depending on the security configuration. Usually Grants are assigned to a Role and on or more User s are assigned to each Roles. A Grant is security aware, that means it is an concrete SecurityObject. Permissions to UI actions are managed with Grants.
Location	A Location, represents some physical as well as virtual place in a warehouse. Could be something like a storage location in the stock or a location on a conveyer. Also virtual or error locations can be modeled with a Location entity. Multiple Locations are grouped to a LocationGroup.
LocationGroup	A LocationGroup is a logical group of Locations, grouping together Locations with same characteristics.
LocationGroupState	A LocationGroupState defines possible states used for LocationGroups.
LocationType	A LocationType is the type of Locations with same characteristics.
Module	A Module represents an Adobe Flex Module and is used to store some basic information about that module, i.e. a name, an URL where the module from, or whether the Adobe Flex Module should be loaded on application startup.
ModulePreference	A ModulePreference is used to store configuration settings in Module scope. The table model of an ModulePreference spans an unique key over the columns C_TYPE, C_OWNER

	and C_KEY. It's counterpart in the context of JAXB is the modulePreference element.
Preferences	An instance of a Preferences represents the root of a preferences XML file and aggregates all other types of preference.
Problem	A Problem is used to signal an occurred failure.
Role	A Role is a group of Users. Basically more than one User belong to a Role. Security access policies are assigned to Roles instead of Users.
RolePreference	A RolePreference is used to provide settings specific to an Role . These kind of Preferences is valid for the assigned Role only. Users assigned to a Role inherit these RolePreferences but a RolePreference can be overruled by an UserPreference. RolePreferences can be defined within a preferences file but also be created with the UI.
Rule	A Rule used as marker interface.
SecurityObject	A SecurityObject is the generalization of Roles and Grants and combines common used properties of both.
SystemUser	A SystemUser is granted with all privileges and omits all defined security constraints. Whenever a SystemUser logs in, she is assigned to a virtual Role with the name ROLE_SYSTEM. Furthermore this kind of Role is immutable and it is not allowed for the SystemUser to change her UserDetails or UserPassword. Changing the UserPassword has to be done in the application configuration when the project is setup.
Target	A Target is either a physical or a logical endpoint of any kind of order in a warehouse. A TransportOrder has a Target set, to where a TransportUnit has to be moved to.
TransportUnit	A TransportUnit is an item like a box, a toad, a bin or a palette that is moved around within a warehouse and can carry goods. Used as container to transport items like LoadUnits. It can be moved between Locations.
TransportUnitState	A TransportUnitState defines a set of states for TransportUnits.
TransportUnitType	A TransportUnitType is a type of a certain TransportUnits. Typically to store some static attributes of TransportUnits, such as the length, the height, or the weight of TransportUnits. It is possible to group and characterize TransportUnits.
TypePlacingRule	A TypePlacingRule is a Rule that defines which types of TransportUnits (TransportUnitTypes) can be put on which type

	<p>of Location (LocationType). A privilegeLevel is defined to order a list of allowed LocationTypes.</p>
TypeStackingRule	<p>A TypeStackingRule is a Rule that defines which TransportUnitType can be stacked on other types. Additionally a maximum number of TransportUnits can be defined.</p>
Unit	<p>A definition of any kind of unit used in the application. In general, Units are defined by a particular type of UnitType and a value. For example 42 grams is a weight, whereas weight is the Unit.</p>
UnitType	<p>An UnitType is the type definition of an Unit. Each UnitType defines a base Unit of it's character. For example a UnitType of weights can define grams, kilograms or tons.</p>
User	<p>An User represents a human user of the system. Typically an User is assigned to one or more Roles to define security constraints. Users can have their own configuration settings in form of UserPreferences and certain user details, encapsulated in an UserDetails object that tend to be extended by projects.</p>
UserPassword	<p>Is a representation of an User together with her password. When an User changes her password, the current password is added to a history list of passwords. This is necessary to omit Users from setting formerly used passwords.</p>
UserPreference	<p>An UserPreference is used to store settings specific to an User. It is always assigned to a particular User and not accessible from, nor valid for, other Users. UserPreferences cannot be overruled by any other type of Preferences.</p>