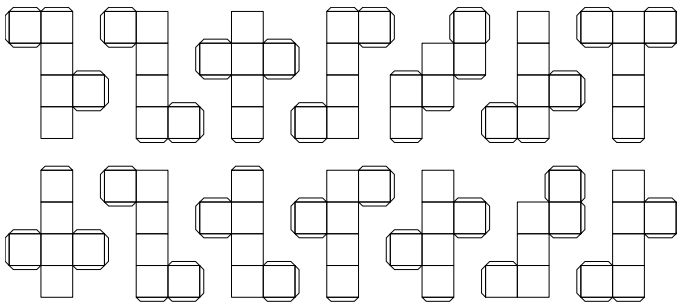


## DeveloperGuide

ProjectForge® 2011



**Version:** 3.6.1

**Date:** 2011-05-27

**Project:** ProjectForge® 2011

**URL:** [www.projectforge.org](http://www.projectforge.org)

**Author:** Kai Reinhard  
 [k.reinhard@me.com](mailto:k.reinhard@me.com)

## Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                                  | <b>4</b>  |
| <b>2</b> | <b>Setting up the environment</b>                    | <b>4</b>  |
| 2.1      | Preparation of Eclipse                               | 4         |
| 2.1.1    | Installation of Maven2 plugin                        | 4         |
| 2.1.2    | Installation of WTP plugin                           | 4         |
| 2.1.3    | Setting up the maven path                            | 4         |
| 2.1.4    | Installation of the Groovy plugin                    | 4         |
| 2.1.5    | Creation of Eclipse project settings                 | 4         |
| 2.2      | Useful eclipse commands                              | 5         |
| 2.3      | Coding style and templates                           | 5         |
| 2.4      | Develop mode   | 5         |
| 2.5      | More convinient use of PostgreSQL                    | 5         |
| <b>3</b> | <b>Building a new release</b>                        | <b>7</b>  |
| 3.1      | Changing test data                                   | 7         |
| 3.2      | Clover   | 8         |
| <b>4</b> | <b>Configuration</b>                                 | <b>8</b>  |
| 4.1      | WTP plugin   | 8         |
| 4.1.1    | server.xml (apache tomcat 6.0)                       | 8         |
| 4.1.2    | m2eclipse WTP-Plugin                                 | 9         |
| 4.1.3    | Install hsql data-base                               | 9         |
| 4.1.4    | Keystore   | 9         |
| <b>5</b> | <b>Concepts</b>                                      | <b>10</b> |
| 5.1      | Hibernate history                                    | 10        |
| 5.1.1    | ShortDisplayNameCapable                              | 10        |
| 5.2      | Wicket vs. Stripes                                   | 10        |
| 5.3      | Wicket pages (context.xml)                           | 10        |
| 5.3.1    | Strip Wicket tags in development mode                | 10        |
| 5.3.2    | i18n   | 10        |
| 5.3.3    | Cooking book   | 11        |
| 5.3.4    | SearchPage   | 11        |
| 5.4      | Lucene analyzer PFTokenizer.jflex                    | 12        |
| 5.5      | Core DO's and Dao's                                  | 12        |
| 5.5.1    | Adding a new Dao                                     | 12        |
| 5.6      | Hibernate search                                     | 12        |
| 5.6.1    | Dependent DO's                                       | 12        |
| 5.6.2    | Adding search fields of child DO's                   | 14        |
| 5.7      | Web optimizations                                    | 14        |
| 5.7.1    | Caching of images, css, java scripts and flash files | 14        |

CONTENTS

3

---

|          |                              |           |
|----------|------------------------------|-----------|
| 5.7.2    | Image dimensions             | 14        |
| 5.8      | Excel export                 | 15        |
| 5.9      | Custom skins for Spaces      | 16        |
| 5.9.1    | Customizing your skin        | 16        |
| 5.9.2    | Building your skin           | 17        |
| 5.9.3    | Installing your skin         | 17        |
| 5.9.4    | Applying your skin           | 17        |
| <b>6</b> | <b>Plugins</b>               | <b>18</b> |
| 6.1      | My first plugin in one hour! | 18        |

## List of Figures

|   |  |   |
|---|--|---|
| 1 | <a href="#">Eclipse XML settings</a> . . . . . | 6 |
|---|--|---|

## 1 Introduction

This document contains some useful information for developers of ProjectForge. Issues on how to set up the database etc. you will find in the AdministrationGuide.

## 2 Setting up the environment

### 2.1 Preparation of Eclipse

#### 2.1.1 Installation of Maven2 plugin

Add <http://m2eclipse.codehaus.org/> as new RemoteSite and install the maven plugins.

#### 2.1.2 Installation of WTP plugin

Add <http://download.eclipse.org/webtools/updates/> as new RemoteSite and install the WTP plugins.

#### 2.1.3 Setting up the maven path

Eclipse -> Preferences -> Java -> Build Path -> Classpath:

Adding variable: M2\_REPO=/Users/kai/.m2/repository

#### 2.1.4 Installation of the Groovy plugin

[Groovy-Plugin Howto \(groovy.codehaus.org/Eclipse+Plugin\)](http://groovy.codehaus.org/Eclipse+Plugin)

#### 2.1.5 Creation of Eclipse project settings

- At first you have to clean up your eclipse project settings, automatically generated by eclipse when you're using the eclipse svn plugin to get the sources: `mvn eclipse:clean`
- After this is done, build a new eclipse project configuration with: `mvn -DdownloadSources=true eclipse:eclipse`
- NOTE: `mvn eclipse:eclipse` won't override an existing `.project` file.

## 2.2 Useful eclipse commands

- `mvn clean install`
- `mvn -DdownloadSources=true eclipse:eclipse`
- `mvn site`

## 2.3 Coding style and templates

Use the coding style for this project: `misc/eclipse/codingstyle.xml` (Window->Preferences->Java->Code Style->Formatter import).

Use the coding templates for this project (last update: 2010-01-31): `misc/eclipse/codetemplates.xml` (Code Style->Code Templates import).

For XML files use the following settings:

- Line width: 150
- Preserve whitespace in tags with PCDATA content: yes
- Indent using white spaces: 2

Useful template (Eclipse -> Preferences -> Java -> Templates:

**Name:** `log4j`

**Description:** `Log4j Logger declaration`

**Pattern:** `private static final org.apache.log4j.Logger log = org.apache.log4j.Logger.getLogger`

## 2.4 Develop mode

1. Please edit `META-INF/context.xml` and set the property `development` to true.  
The development parameter can be get over `BaseActionBean.isDevelopmentSystem()`.  
It's actually used for system administration menu for dumping data base to xml.
2. Please edit `log4j.properties` and set the debug level for all or the special categories you want to debug.

## 2.5 More convinient use of PostgreSQL

For accessing the database it's useful to create a database user which is same to the unix user:

```
1 Archon:~ admin$ createuser -U postgres kai
```

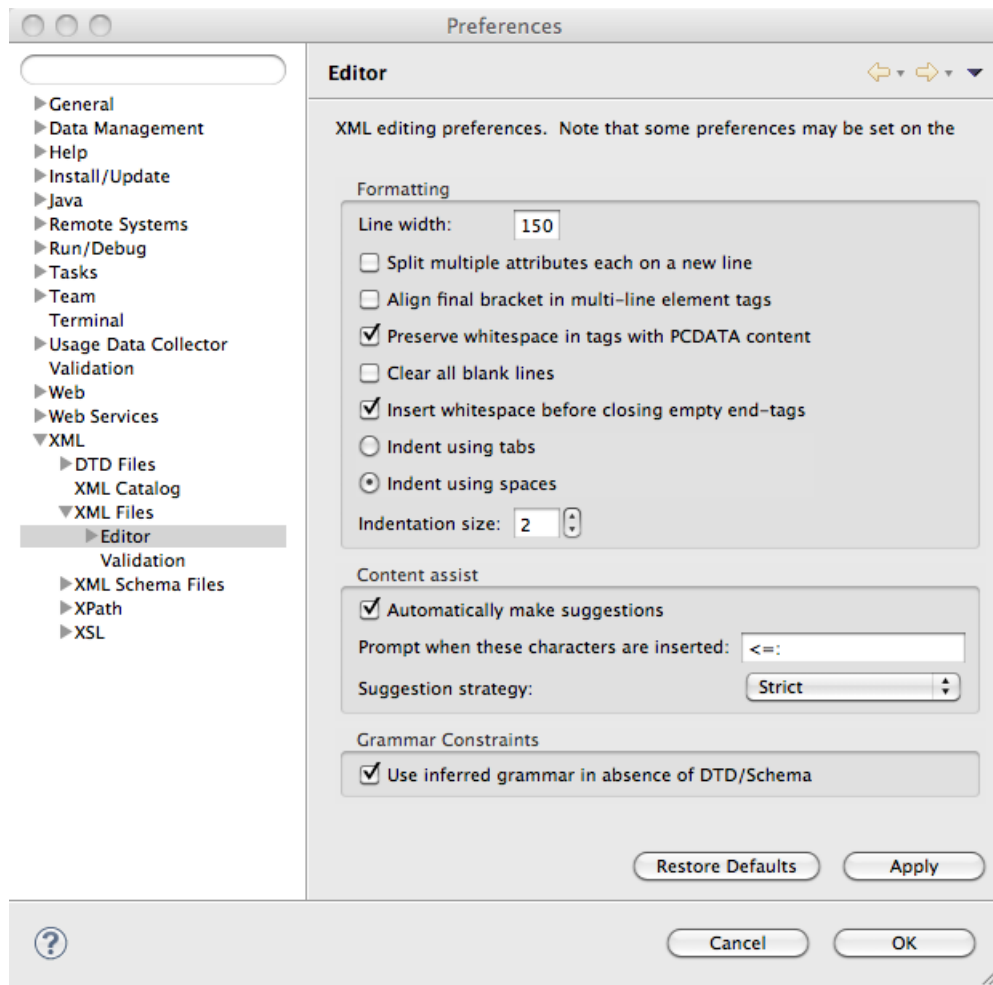


Figure 1: Eclipse XML settings.

```
2 psql -U postgres
3 grant projectforge to kai
```

Afterwards you can use `dropdb`, `createdb` and `psql` directly from the shell (if not, check the AdministrationGuide for configuration of `pg_hba.conf`).

### 3 Building a new release

#### 1. Update version in `build.xml` and `pom.xml`

Increase version id:

```
1 <property name="version" value="3.6.1"
2           />

, <version>3.6.1</version>
```

#### 2. `ant pre-dist`

generates version files: `Version.java`, `version.xml`, `version.dtd`.

#### 3. `mvn clean install site`

builds the web archive (war): `target/ProjectForge.war` and `target/site`. Use `mvn -o install site` if not all maven repositories are available or if you are working off line. If any test case was failed the archive is not built. Please fix the failed test cases before continue.

#### 4. `ant dist`

builds a zip archive without `WEB-INF/lib` and `META-INF`. `META-INF` is not contained of avoiding the overwriting of settings in `META-INF/context.xml`.

tbd.: (testing, deployment, javadoc, doc etc.)



#### Hint

Don't forget to check the user preferences in the admin web. Because if any changes in the user preferences objects are done, all user preferences are lost, because the deserialization of the xml-objects does not work.

In that case, please update and test the method `UserPreferencesMigrator`.

### 3.1 Changing test data

The ProjectForge release contains a data base dump file with test data for initializing an empty database (see AdministrationGuide). The test data file is `src/main/resources/data/init-test-data.xml.gz`. If you want to change it, start ProjectForge with the standard `context.xml` (hsqldb) with parameter

development=true with an empty database (maybe you have to delete the hypersonic database files before starting). Login with initialize/test as described in the AdministrationGuide and do the modifications as wanted via ProjectForge. Dump the data base via System administration menu (only visible in development mode). Modify the result xml file:

1. Clean-up the resulting xml file by removing the following sections: UserPreferencesDO and PFUserDO/timezone .
2. gzip projectfordedump\_xxx.xml
3. Replace the existing init-test-data.xml.gz and check-in.
4. Check the InitDatabaseDaoWithTestDataTest for running successfully (maybe some modifications are needed to pass through).

## 3.2 Clover

```
mvn clover2:instrument clover2:aggregate clover2:clover
```

# 4 Configuration

## 4.1 WTP plugin

### 4.1.1 server.xml (apache tomcat 6.0)

**File:** server.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <Server port="8005" shutdown="SHUTDOWN">
3    <Listener SSLEngine="on" className="org.apache.catalina.core.AprLifecycleListener"/>
4    <Listener className="org.apache.catalina.core.JasperListener"/>
5    <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"/>
6    <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
7    <GlobalNamingResources>
8      <Resource auth="Container" description="User database that can be updated and saved"
factory="org.apache.catalina.users.MemoryUserDatabaseFactory" name="UserDatabase" pathname="conf/
type="org.apache.catalina.UserDatabase"/>
9    </GlobalNamingResources>
10   <Service name="Catalina">
11     <Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>
12     <Connector SSLEnabled="true" acceptCount="100" clientAuth="false" disableUploadTimeout="true
enableLookups="true" keystoreFile="${user.home}/.keystore" keystorePass="changeit" maxSpareThreads
maxThreads="200" minSpareThreads="5" port="8443" scheme="https" secure="true" sslProtocol="TLS"/>
13     <Engine debug="0" defaultHost="localhost" name="Standalone">
14       <!--Realm className="org.apache.catalina.realm.UserDatabaseRealm"

```

```
15     resourceName="UserDatabase" /-->
16     <Host appBase="webapps" debug="0" name="localhost" unpackWARs="true">
17         <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" pattern="c
prefix="localhost_access_log." suffix=".txt"/>
18         <Context docBase="ProjectForge" path="/ProjectForge" reloadable="true" source="org.eclipse
19     </Engine>
20 </Service>
21 </Server>
```

#### 4.1.2 m2eclipse WTP-Plugin

Since Eclipse 3.6 you probably need this plugin installed for starting ProjectForge as WTP module.

#### 4.1.3 Install hsql data-base

Copy hsqldb-<version>.jar into your Apache Tomcat lib dir.

#### 4.1.4 Keystore

ProjectForge is configured with SSL connector as default. Please create your keystore-file:

```
1 keytool -genkey -keystore tomcat-localhost.keystore -alias tomcat -keyalg RSA -keysize
1024 -keypass changeit -storepass
2 changeit
```

## 5 Concepts

### 5.1 Hibernate history

The Micromata's hibernate history is used for historizing data base objects.

#### 5.1.1 ShortDisplayNameCapable

Mark your data object as `ShortDisplayNameCapable` for manipulating the history output of a data object (e. g. `TaskDO` implements this interface). Implement the interface method in your data object:

```
1 public class TaskDO extends DefaultBaseDO implements ShortDisplayNameCapable
2 {
3     ...
4     @Transient
5     public String getShortDisplayName()
6     {
7         return this.getName() + " (#" + this.getId() + ")";
8     }
9     ...
10 }
```

### 5.2 Wicket vs. Stripes

All Stripes pages are replaced by Wicket pages.

### 5.3 Wicket pages (context.xml)

#### 5.3.1 Strip Wicket tags in development mode

During the development the output of Wicket tags are most times a little bit annoying. At default, the Wicket tags are stripped also in development mode. If you need the output of the Wicket tags set `stripWicketTags` to `false` in `context.xml`. In productive (aka deployment) mode and in mobile pages Wicket tags are always stripped.

#### 5.3.2 i18n

- Standard set of i18n keys (prefix is specified in constructor of the list and edit pages):  
[prefix].title.add, [prefix].title.edit, [prefix].title.list

### 5.3.3 Cooking book

- Register your Wicket pages in `WicketApplication` as bookmarks if used by the menu (bookmarking is also useful for shorter bookmarks for the users).

### 5.3.4 SearchPage

The requirements of all objects which should be part of the `SearchPage` are:

- The Dao should be added to the Registry (see `DaoRegistry`).
- The registered `ListPage` must implement the interface `IListPageColumnsCreator`. You should also support `returnToPage` and `sortable` (the tables should not be sortable on `SearchPage`).

**File:** `AddressListPage.java`

```

1  @SuppressWarnings("serial")
2  public List<IColumn<AddressDO>> createColumns(final WebPage returnToPage, final boolean
   sortable)
3  {
4      ...
5      view.add(new ListSelectActionPanel(view.newChildId(), rowModel, AddressEditPage.class
   address.getId(), returnToPage, ...
6      ...
7      columns.add(new CellItemListenerPropertyColumn<AddressDO>(new Model<String>(getString("
   getSortable("lastUpdate", sortable), "lastUpdate", ...

```

- The Dao method `getList(BaseSearchFilter)` should support the generic `BaseSearchFilter` if the super method is overwritten:

**File:** `AddressDao.java`

```

1  @Override
2  public List<AddressDO> getList(BaseSearchFilter filter)
3  {
4      final AddressFilter myFilter;
5      if (filter instanceof AddressFilter) {
6          myFilter = (AddressFilter) filter;
7      } else {
8          myFilter = new AddressFilter(filter);
9      }
10     QueryFilter queryFilter = new QueryFilter(myFilter);
11     ...
12 }

```

Therefore the Filter should have the constructor which copies all fields of the `BaseSearchFilter` to the special filter (if exist).

**File:** `AddressFilter.java`

```

1  public AddressFilter()
2  {
3  }
4
5  public AddressFilter(BaseSearchFilter filter)
6  {
7      super(filter);
8      ...
9  }

```

## 5.4 Lucene analyzer PFTokenizer.jflex

In ProjectForge a customized Analyzer derived from StandardAnalyzer is used. After modifying the PFTokenizerImpl.jflex file you need to rebuild the PFTokenizerImpl.java:

```
jflex --nobak src/main/java/org/projectforge/lucene/PFTokenizerImpl.jflex
```

The JFlex version 1.4.2 is used from [jflex.de](http://jflex.de).

## 5.5 Core DO's and Dao's

Please refer the plugins chapter for handling plugins. This chapter does only concern core DO's and Dao's.

### 5.5.1 Adding a new Dao

1. Add the Dao class to:

```
org.projectforge.registry.DaoRegistry.java
```

2. Add the Dao to the spring configuration: applicationContext-business.xml.

3. Add the DO to hibernate configuration in:

```
org.projectforge.database.HibernateCoreEntities.java
```

## 5.6 Hibernate search

### 5.6.1 Dependent DO's

Hibernate does not update the index of dependent objects automatically. In ProjectForge exists an hot fix, which should be implemented in each Dao managing DO's which depends on other DO's.



#### Example

This code is an example for DO's containing one dependent DO:

```

1  public ProjektDao()
2  {
3      super(ProjektDO.class);
4      baseDaoReindexRegistry.registerDependent(KundeDO.class, this);
5  }
6
7
8  @Override
9  public void reindexDependents(BaseDO< ? > obj)
10 {
11     if (obj instanceof KundeDO) {
12         @SuppressWarnings("unchecked")
13         List<ProjektDO> list = getHibernateTemplate().find("from ProjektDO p where
p.kunde.id=?", ((KundeDO) obj).getId());
14         reindex(list, null);
15     } else {
16         super.reindexDependents(obj);
17     }
18 }

```



### Example

This code is an example for DO's containing more than one dependent DO:

```

1  public EmployeeDao()
2  {
3      super(EmployeeDO.class);
4      baseDaoReindexRegistry.registerDependent(PFUserDO.class, this);
5      baseDaoReindexRegistry.registerDependent(Kost1DO.class, this);
6  }
7
8  @Override
9  public void reindexDependents(BaseDO< ? > obj)
10 {
11     Set<Serializable> alreadyReindexed = new HashSet<Serializable>(); // For
avoiding multiple re-indexing.
12     if (obj instanceof PFUserDO) {
13         @SuppressWarnings("unchecked")
14         List<EmployeeDO> list = getHibernateTemplate().find("from EmployeeDO e
where e.user.id=?", ((PFUserDO) obj).getId());
15         reindex(list, alreadyReindexed);
16     } else if (obj instanceof Kost1DO) {
17         @SuppressWarnings("unchecked")
18         List<EmployeeDO> list = getHibernateTemplate().find("from EmployeeDO e
where e.kost.id=?", ((Kost1DO) obj).getId());
19         reindex(list, alreadyReindexed);
20     } else {
21         super.reindexDependents(obj);
22     }

```

```
23 }
```

### 5.6.2 Adding search fields of child DO's

## 5.7 Web optimizations

### 5.7.1 Caching of images, css, java scripts and flash files

Https normally does not allow to cache any content in the client's browser. It's annoying, that for every request all java script, css, image and flash files will be loaded from the server. ProjectForge uses the ResponseFilter of Micromata WebUtils. This filter modifies the response header of all in web.xml registered files for caching that resource files by the client's browser. This speed-ups ProjectForge!

### 5.7.2 Image dimensions

Setting all image dimensions in the html markup results in faster rendering processes of the client's browser. In ProjectForge is a test case defined which runs automatically every time during building a new release:

**File:** GetImageDimensionsTest.java

```
1  ...
2  private static final String PATH = "src/main/webapp/images";
3
4  private static final String DIMENSION_FILE = "src/main/resources/" + WebConstants.FILE_IMAGE_DIMENSIONS;
5
6  private static final String[] IMAGE_SUFFIXES = new String[] { "png", "gif", "jpg"};
7
8  @Test
9  public void doit() throws IOException
10 {
11     log.info("Create dimension file of all webapp images.");
12     @SuppressWarnings("unchecked")
13     final Collection<File> files = FileUtils.listFiles(new File(PATH), IMAGE_SUFFIXES,
14 true);
15     final File absolutePathFile = new File(PATH);
16     final String absolutePath = absolutePathFile.getAbsolutePath();
17     final List<ImageDimension> dimensions = new ArrayList<ImageDimension>();
18     for (final File file : files) {
19         final Image image = Toolkit.getDefaultToolkit().getImage(file.getAbsolutePath());
20         final ImageIcon icon = new ImageIcon(image);
21         final String filename = file.getAbsolutePath().substring(absolutePath.length() +
22 1);
23         final ImageDimension dimension = new ImageDimension(filename, icon.getIconWidth(),
24 icon.getIconHeight());
25         dimensions.add(dimension);
26     }
27 }
```

```

23     }
24     final FileWriter writer = new FileWriter(DIMENSION_FILE);
25     final XStream xstream = new XStream();
26     xstream.alias("images", List.class);
27     xstream.alias("image", ImageDimension.class);
28     String xml = xstream.toXML(dimensions);
29     writer.append(xml);
30     IOUtils.closeQuietly(writer);
31     log.info("Creation of dimension file done: " + DIMENSION_FILE);
32 }
33 ...

```

The file uses the Java AWT toolkit for calculating the geometry of all images. The resulting image dimension file is:

**File:** src/main/resources/imageDimensions.xml

```

1 <images>
2   <image>
3     <path>accept.png</path>
4     <width>16</width>
5     <height>16</height>
6   </image>
7   <image>
8     <path>add.png</path>
9     <width>16</width>
10    <height>16</height>
11  </image>
12  ...
13 </images>

```

Inside ProjectForge for all images rendered by Wicket the width and height will be set in the markup automatically.

## 5.8 Excel export

Excel downloads are quite simple:

**File:** Kost1ListPage.java

```

1 private enum Col
2 {
3     STATUS, KOST, DESCRIPTION;
4 }
5
6 void exportExcel()
7 {
8     ...

```

```

9   final ExportWorkbook xls = new ExportWorkbook();
10  final ContentProvider contentProvider = new XlsContentProvider(xls);
11  xls.setContentProvider(contentProvider);
12  final ExportSheet sheet = xls.addSheet(sheetName);
13  final ExportColumn[] cols = new ExportColumn[] { //
14  new I18nExportColumn(Col.KOST, "fibu.kost1", 10), // Id, i18n key, length
15      new I18nExportColumn(Col.DESCRPTION, "description", 30),
16      new I18nExportColumn(Col.STATUS, "status", 10)};
17  sheet.setColumns(cols);
18  // Insert here cell formats if needed.
19  final PropertyMapping mapping = new PropertyMapping();
20  for (final Kost1DO kost : kost1List) {
21      mapping.add(Col.KOST, kost.getFormattedNumber());
22      mapping.add(Col.STATUS, kost.getKostentraegerStatus());
23      mapping.add(Col.DESCRPTION, kost.getDescription());
24      sheet.addRow(mapping.getMapping(), 0);
25  }
26  DownloadUtils.setDownloadTarget(xls.getAsByteArray(), filename);
27 }

```

If you need your own cell formats, please try something like this:

```

1  final ContentProvider sheetProvider = sheet.getContentProvider();
2  sheetProvider.putFormat(Col.START_TIME, "yyyy-MM-dd HH:mm");
3  sheetProvider.putFormat(Col.STOP_TIME, "HH:mm");
4  sheetProvider.putFormat(Col.DURATION, "[h]:mm");
5  sheetProvider.putFormat(Col.ID, "0");
6  sheetProvider.putFormat(Col.BETRAG, "#,##0.00;[Red]-#,##0.00");

```

## 5.9 Custom skins for Spaces

Normally all spaces comes with the same skin: ProjectForge

If you like to customize the look of a specific Space, you need to build a new GWiki skin. To get a generally idea of GWiki skins, have a look to it's documentation at <http://labs.micromata.de/gwiki/gwikidocs/howtos>

For ProjectForge needs, an example skin already exists (gwiki-style-ProjectForge-example), so you can adopt it's project structure and code to build your own skin. You can download the skin from the ProjectForge svn repository.

### 5.9.1 Customizing your skin

All relevant customizations can be made in the files `standardhead.gspt` and `standardfoot.gspt`



**Hint**

You should include the original ProjectForge-include files, e.g.: `<@include file="inc/ProjectForge/headM`  
`@>` Actually there is no reason to try another approach.

### 5.9.2 Building your skin

After you have customized the files `gwikiplugin.xml` and `pom.xml` you can build the skin with the command `mvn install`

### 5.9.3 Installing your skin

Installing the skin is very easy:

1. Navigate your browser to `ProjectForge -> Spaces`
2. In the Admin Menu choose `Installed Plugin Index` . Here you can see and manage all installed Plugins.
3. Upload your skin / plugin (choose the zip file).
4. In the Admin Menu choose `Wiki Control`
5. Reload the whole GWiki Web
6. Activate the skin via `Plugin Admin` , if nesaccery

### 5.9.4 Applying your skin

GWiki pages inherits their skin property by it's parents. In this case the Spaces skin.

1. Browse to the space you wish to edit
2. In the Page menu choose `Edit`
3. Choose the `Settings` tab and click on the `Content` area
4. Enter your skin-name as specified `Skin` (e.g.: `ProjectForge-example`)

## 6 Plugins

Extend ProjectForge with your own plugins or third party plugins. This chapter describes how easy it is to write own plugins. Enable the following features inside your plugins with a few lines of code:

- **Data-base objects**

There is a convenient mapping from your Java classes to data-base entries.

- **Full-text index**

For all your data-base entries a full-text engine is automatically enabled for fast full-text search.

- **History of changes**

If required all changes of your data-base entries are persisted in a history of changes containing the user, time stamp, change (old and new value) etc.

- **E-Mail templating**

Send e-mail with the ProjectForge's built-in template mechanism.

- **Mobile pages**

It's so easy to provide web pages in your plugin which are optimized for mobile devices (iPhone, Android, BlackBerry, Windows phones etc).

- **Access management**

You can define your own access management. Therefore only those users are able to see or modify data they are authorized to. You can define rules or add your own rights to the central user management.

- **Updating mechanisms**

ProjectForge provides a convenient update mechanism. Every time the administrator starts a new version of ProjectForge or your plugin a check will be done during the start-up phase. If required, the administrator is able to update your data-base schema or required migration scripts by simply clicking the update button.

- **Scripting**

Your new plugin data are automatically available inside ProjectForge's scripting functionality.

### 6.1 My first plugin in one hour!

Every step should be very easy to understand with an high re-use opportunity inside your own plugins. Gather your experience of the technologies such as Spring, Wicket and Hibernate by developing your plugins step by step.

**1. Download examples.**

Download [resources/plugins.zip \(resources/plugins.zip\)](#) for getting example sources.

**2. Create your data base object.**

See `MemoDO.java` for seeing how easy it is to define persistent data models.

**3. Create your data access object.**

Refer `MemoDao.java` as an example on writing own data access objects. Data access objects are responsible for reading and writing objects from and into the data-base. Your Dao is available by adding a `pluginContext.xml` in your resource path (see example).

**4. Define the access rights.**

Refer `MemoRight.java` for defining which user should have access to your data objects.

**5. Define your data-base setup and update scripts.**

See `MemoPluginUpdates.java` for seeing how easy it is to define your data-base setup and update scripts for any further release of your plugin.

**6. Write list pages with filters.**

See `MemoListPage.java`, `MemoListForm.java` and `MemoListPage.html` for implementing list pages with filters and full text search engine support.

**7. Write edit pages.**

See `MemoFormRenderrer.java`, `MemoEditForm.java` and `MemoEditPage.java` for implementing edit formular pages with insert, update, delete functionality.

**8. Putting the stuff together.**

Refer `MemoPlugin.java` for putting all the stuff together. This class is used to register all your components (data base object, data access object, menu entries and web pages) as well as the i18n resource bundle (e. g. `MemoI18nResources*.properties`).

**9. Register your plugin to ProjectForge.**

Edit your `config.xml` by adding your plugins (coma separated):

**File:** `config.xml`

```
1 <config>
2   ...
3   <pluginMainClasses>
4     org.projectforge.plugins.todo.ToDoPlugin,
5     org.projectforge.plugins.memo.MemoPlugin
6   </pluginMainClasses>
7   ...
8 </config>
```

10. **Ready, run, enjoy it.**

Add your plugin as jar to your WEB-INF/lib directory and restart ProjectForge.

An example plugin with more features is the ToDoPlugin (e-mail templating etc.).