# EnhydraDM application

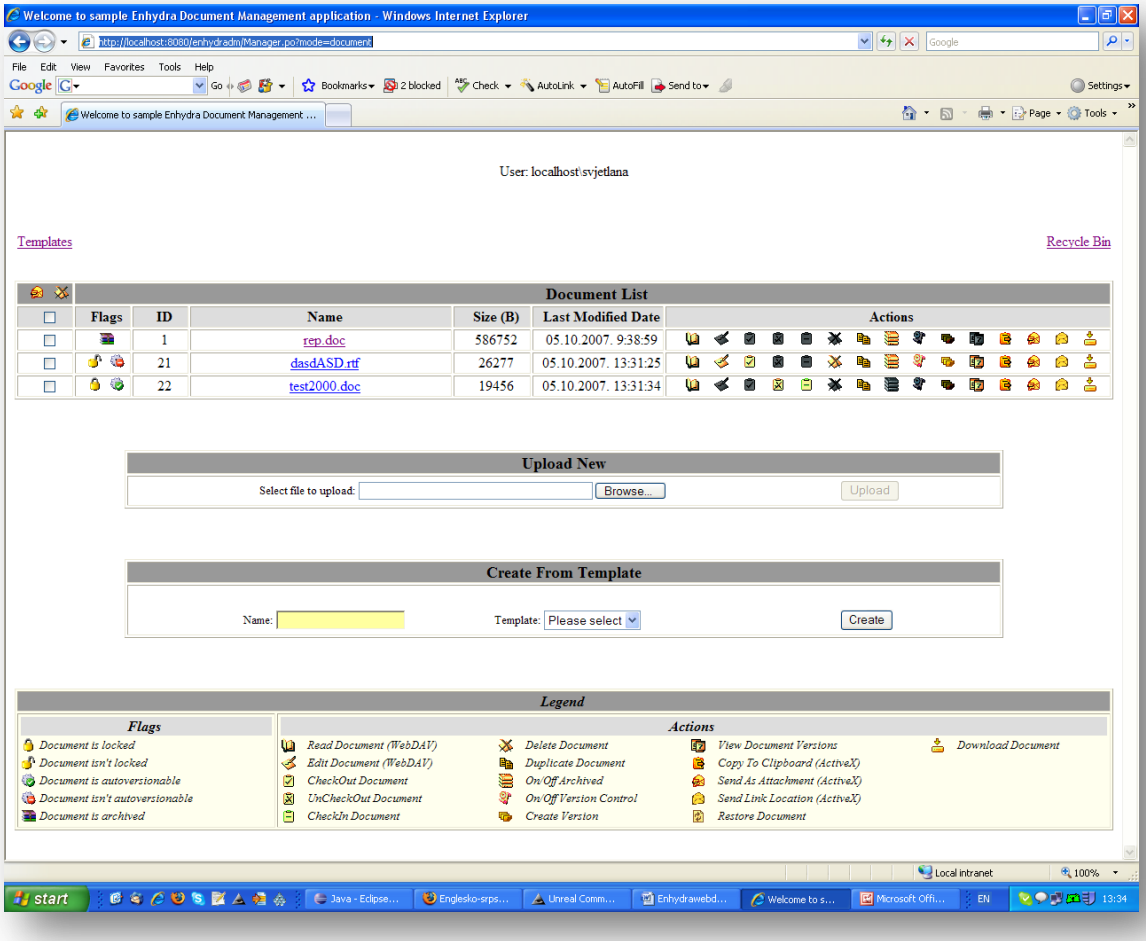
This is demo application for document management.

On application startup two servlet classes are initialized EnhydraDM and WebDavServlet. They behave like two separated applications that share business, specification and data layer. They're also using the same configuration file (web.xml file) with separated parameters for every application. It works with HSQL DB.

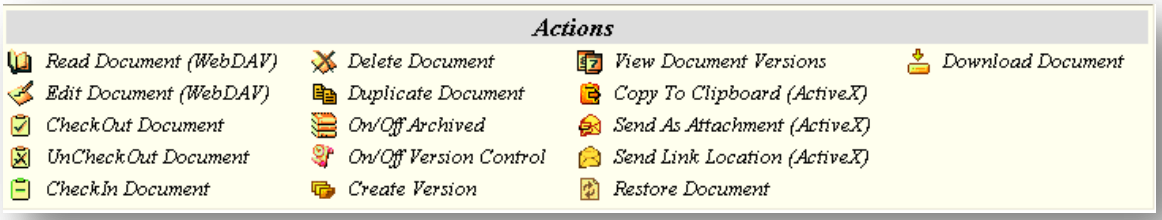
EnhydraDM application is typical Enhydra application. It implements presentation logic with PO objects. The most of actions in application go throughout this presentation layer except **Read** and **Write** actions that are handled by WebDavServet.

Application uses NTLM authentication, by default, but FORM authentication is supported, too. That is meter of configuration in web.xml file.

The main page can be opened in two different modes: Document and Template. Default is Document. In Document mode the main page shows list of documents with list of possible actions, and in Template mode it shows list of templates.




**Picture 1: Main page (Manager.po)**




**Picture2: Actions**

In the top left corne on the main page there are link for changing mode **Document/Template,** And in the top right corner there is **Recycle Bin** link for reviewing of deleted documents with **restore** action for every of them.

User can upload a document, and create new document from template. There are limitations of file extensions that can be uploaded, and file size, too. By default allowed file extensions for upload are: **doc, docx, xls, xlsx, ppt, pptx, pps, pptx, pps, ppsx, jpeg, msg, pdf, zip, gif, bmp, txt, tif, png**. Maximum file size is 2MB, by default.

There is a list of all possible actions for every document depends of its current state. Only **Read** and **Edit** actions are executed trought WebDAV protocol.

**Read** action is available for every document. Document will be opened in read only mode in correspondent default interpreter program for document mime type (Word, Excel, AdobeReader ..) , or in browser.

In application property file **editablemimetypes.properties** are defined document extensions and correspodent mime types for all editable document types. That means that only documents with extension stored in this property file can be opened in **Edit** mode.

For **Check In** functionality there is separete page where user can browse for file that want to Check In.

There are adequate flags for every document state: **locked**, **unlocked**, **autoversionable**, **not autoversioable** and **archived**. If document is **archived** there is no possibility to change document any more so the only flag for this document is **archived** flag**.**

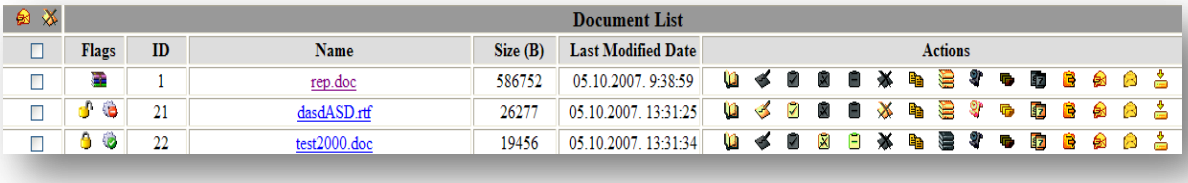
If one user locks the document (**Check Out** or **Edit**), actions like **Check In**, **Delete**, **Uncheck Out**, **On/Off Archived**, **On/Off Autoversionable**, **Create Version**, **Set as Current Version** (in Versions part) are disabled for all other users, except him, until he does unlock.

If document is archived actions: **Edit**, **Check Out**, **Uncheck Out, Check In**, **Delete**, **On/Off Autoversionable**, **Create Version**, **View Document Versions**, **Set as Current Version** are disabled.

If **autoversionable** flag for document is **false** no version history is saved for that document (during period that **autoversionable** is false), but in that case document version can be manually created using **CreateVersion** action which is available for **not autoversionable** documents.

Actions: **Read**, **Duplicate Document**, **Copy To Clipboard, Send As Attachment, Send Link Location, Download Document,** and **Restore Document** (from Recycle Bin) are always available.

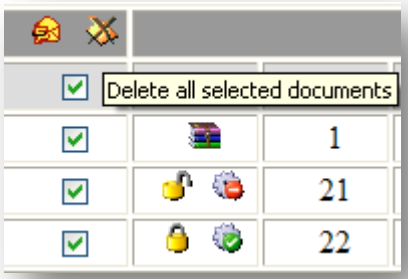
Action **Duplicate Document** creates copy of document with default values for lock and autoversionable.





**Picture 3 Flags**

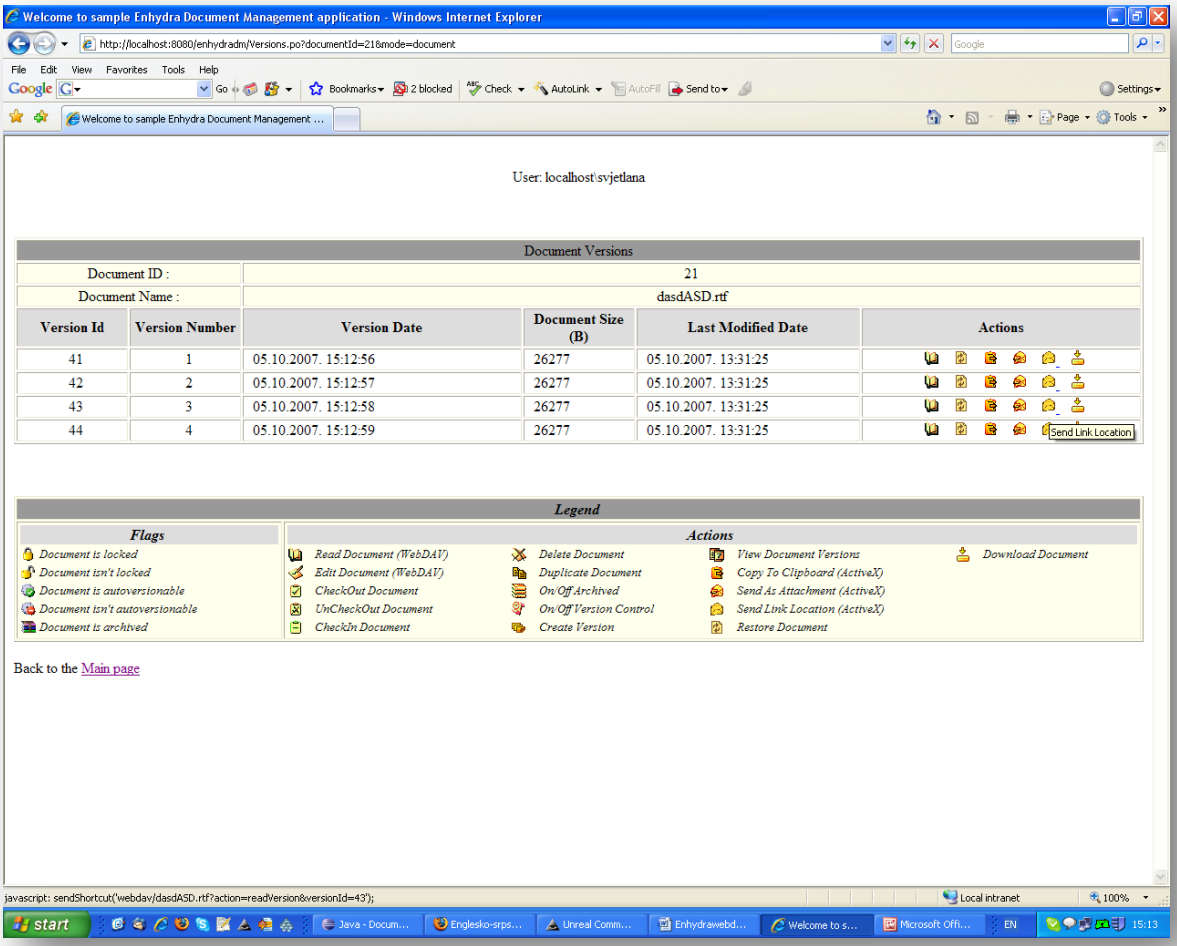
Actions **Delete** and **Send as Attachment** is possible to execute on multiple documents. Check Box is used to select documents that are candidates for group actions execution.




To create document from template user have to enter document name and choose template from drop down list in Document mode on the main page. If one choose XSL template, above name field it will appear radio buttons for choosing language (English or German language) and on create, page with FoEditor in chosen language, will be opened. Document created this way is PDF document

Creating template from template is similar to creating document from template with one exception. Creating XML template from template isn't allowed, and XML templates will newer appear in mentioned drop down list.

**DocumentVersions** page shows list of document/temlate versions.

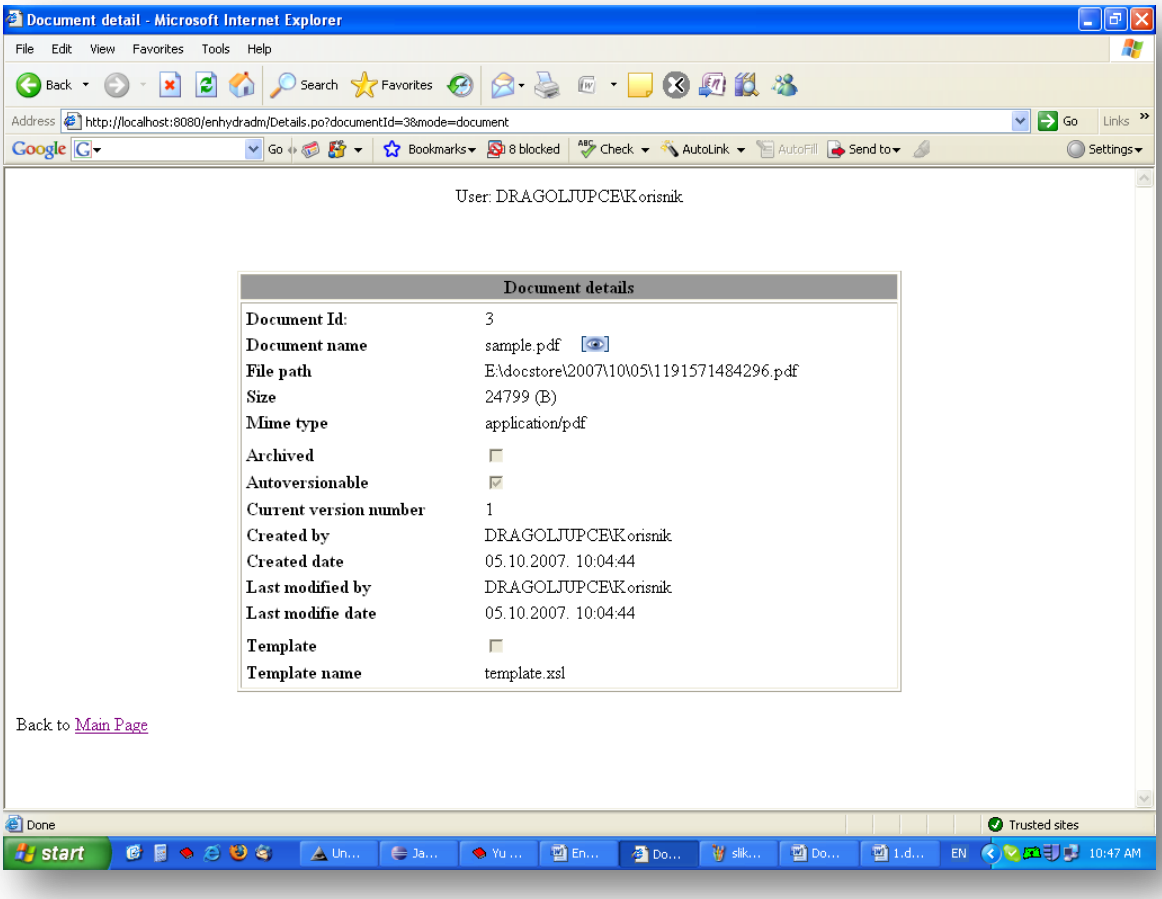



**Picture4: Versions**

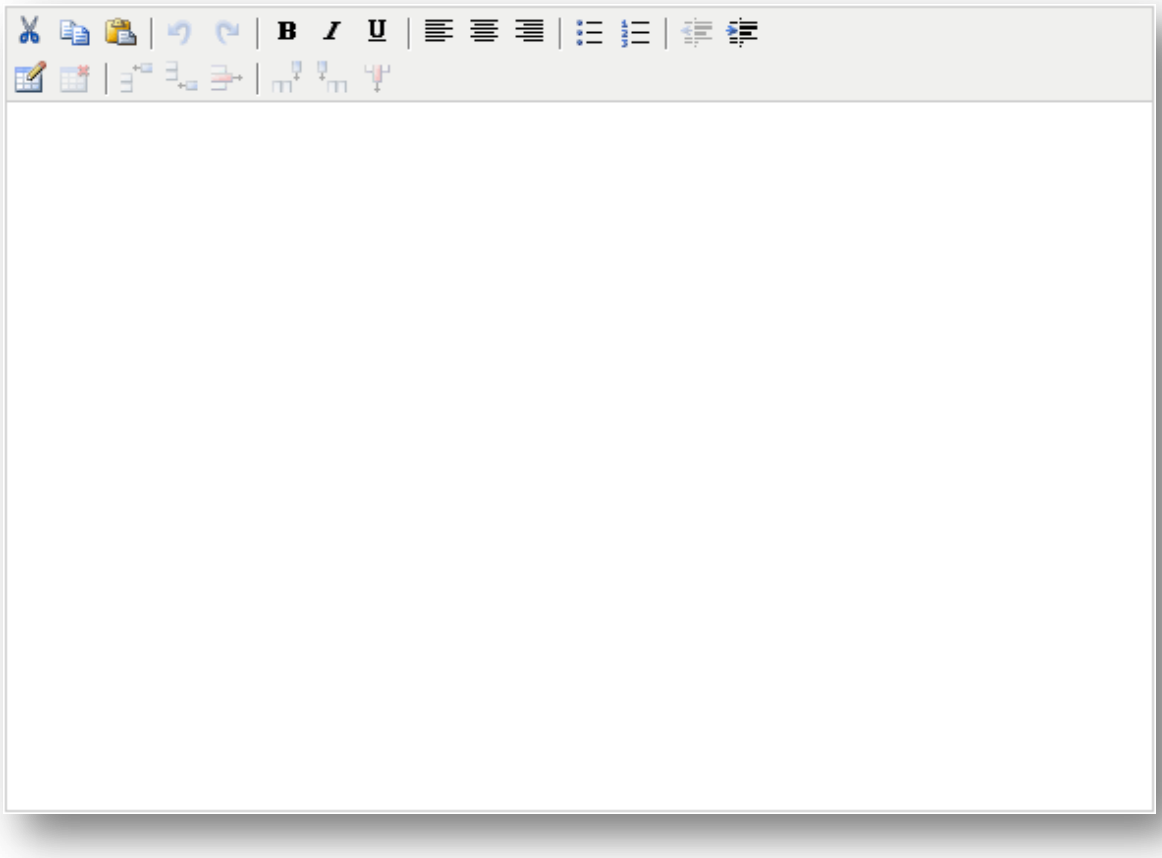
The list of possible actions (**Read Document Version, Set as Current Version, Copy To Clipboard, Send As Attachment, Send Link Location and Download Document**):


**Picture5: Document Version Actions**

**Details** page shows document/template details

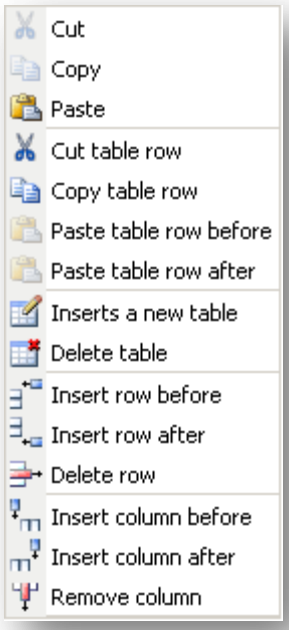



FoEditor is text editor that interprets entered text as HTML. On submit entered text will be stored in database and then transformed to PDF format. Generated PDF file then will be stored with other files from the main page. Font and text size are the same in the Fo Editor and PDF file. Font is Times New Roman and size is 12pt.



**Picture6:Fo Editor**

FoEditor supports some basic text editor functionalities: cut, copy, paste, undo, redo, text styling (bold, italic underline), bulleting, numbering and table drawing. All functionalities are available from toolbar and some of them are available from context menu too.

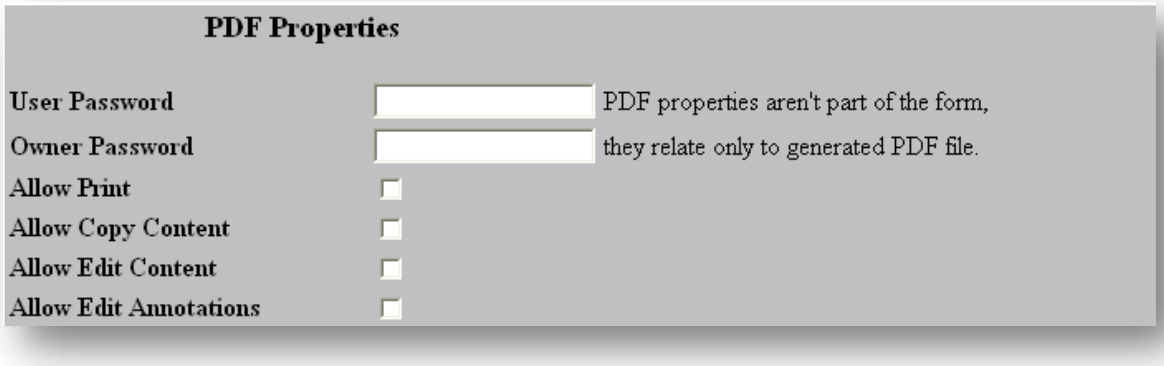



**Picture7: Context menu**

Above FoEditor, there is label with document name entered on the main page and two text fields. Name should be entered without extension. Text fields use as example how additional data can be embedded in PDF file beside FoEditor content. That data will be stored in database too.


**Pictue9: Document properties**

In the gray box on the right side of this page, there are two password fields and couple check boxes that enable user to lock created PDF file. They aren't part of the form and don't influence on data stored in database, but only on PDF file.




**Picture10: PDF Properties**

This page supports Multilanguage. FoEditor supports all required languages: English, German, Hungarian, Slovenian and Slovakian (button hints and context menu items are translated). Language setup is separate for FoEditor and labels and buttons on the page. Labels and buttons translation is stored in **foeditor_laguageId.properties** files. Those files can be changed at runtime and new properties files can be added at runtime.
Fo Editor for different languages uses different JavaScript files.
Default language is English, but it can be changed. Default language is used when no language is explicitly required or required language settings aren't found. As an example that Multilanguage works well, when user choose XSL file in the "Create from Template" table on the main page, radio buttons for choosing language will appear.

Document created this way is will be edited in FoEditor. Along with editing document content, content from text fields stored with that document will be loaded into appropriate text fields. When opens page with FoEditor for editing document, user can't choose language. In this mode language is always English.

## FoEditor implementation

On the FoEditor page naming convection for form input names and FoEditor names must be strictly followed. "Document properties" text fields and FoEditor must have **"fo."** prefix. Those text fields must have **".string"** suffix and FoEditor must have **".node"** suffix. "PDF properties" text fields and check boxes must have **".pdf"** prefix and they don't have suffix.

FoEditor is implemented using **tinyMCE** control. **TinyMCE** is open source JavaScript control developed by Moxiecode Systems AB. This contol interprets entered text as HTML content and it overalls variety of plugins with different functionalities. FoEditor uses only two of them, **table** and **contextmenu. TinyMCE** supports Multilanguage. It use different JavaScript file for different language. FoEditor supports English, German, Hungarian, Slovenian and Slovakian languages. If one require language that isn't supported with appropriate JavaScript it won't harm editors functionalities, it will only damage toolbar and contextmenu layout.

Folder with **TinyMCE** control is placed inside **resource** folder in **Presentation layer.**

**TinyMCE** control is included and configured in FoEditor.html page. All configuration options are placed within the tinyMCE.init() JavaScript call.

There are only one FoEditor on the page but there number is unlimited.

**Tested within: Microsoft Internet Explorer and Mozilla Firefox**

### *Feature 1:*

Within Mozila Firefox when user presses **Enter**, text lose style (bold, italic, underline)

### *Resolution 1:*

We still have this problem.

## Generating PDF file procedure

Generating PDF file procedure is divided into three steps, three transformations. All files necessary for transformation are placed in **application layer** into **resource/appwizard** folder. Paths to all files necessary for transformations are defined in **web.xml.**

- **Tidy transformation –** transforms FoEditor content from HTML to XHTML format. It use **tidy.properties** file, placed in **application layer** into **resource/appwizard/conf** folder. Path to this folder is defined in **web.xml.**
- **xhtml2fo transformation** – transform XHTML format to FO format. It use **xhtml2fo.xsl** template. This template file can be changed at runtime.
- **Fop transformation** – finally create PDF file. It uses **userconfig.xml** file and specialized (**XSLFast**) XSL-FO template. **XSLFast** project and additional files are placed in **XSLFast** folder. This XslFast project reports errors when preview of generated document is required, but it makes useful template. Generated template should be manual changed in order to be used for mentioned transformation. This template must be uploaded on main page (Template mode)

*Note: (only for FORM authentication): If someone uses application for the first time the first step is registration. After successful registration user can log in.*

# APPLICATION STRUCTURE

## SPECIFICATION LAYER
## (Package: org.enhydra.dm)

This layer contains all base software classes (basic API, WebDAV implementation part):

- servlet class: **WebDavServlet.java** the main servlet for WebDAV related application part

**In sub package API are stored all interfaces:**

- DocumentManager.java, DocumentStore.java, FoDocumentManager.java, Document.java, AppUser.java, DocumentVersion.java, UserManager.java, FoDocument.java and FoDocumentParam interfaces

In sub package: **api.exceptions**

- **BaseException.java**: the exception class that is all other exception classes from business part extends.

In sub package: **api.handler**

- **MethodHandler.java**: interface for al method handlers
- **AbstractHandler.java:** abstract class that implements MethodHandler.java interface with abstract **service ()** method. All handler classes extends this class.

In sub package: **api.loggers**

- **Log.java**: abstract class for logger functionality

In sub package: **api.util**

- **DesEncryption.java**: Interface for encryption and decryption.

In sub package: **api.util.filesys**

- **FileChangeListener.java**: interface
- **FileChangeListenerImpl.java**: abstract class that implements FileChangeListener.java

These two classes together with FileMonitor.java represent mechanism for monitoring and reloading file changes. Files reload period is setup in **web.xml** file. Files reload period should be set up in seconds. If files reload period is 0 or negative number or it isn't set up at all, application won't reload files at runtime.

**In sub package HANDLER are stored all WebDAV handler implementations:**

- There are placed all possible handler classes but only some of them are really implemented and used in this application (That is list of handlers is derived from Davenport project and adaptive). Implemented handlers are :

- **DefaultGetHandler.java** (for GET request)
- **DefaultHeadHandler.java** (for HEAD request)
- **DefaultOptionsHandler.java** (for OPTIONS request)
- **DefaultLockHandler.java** (for LOCK request)
- **DefaultUnlockHandler.java** (for UNLOCK request)
- **DefaultPropfindHandler.java** (for PROPFIND request)
- **DefaultPutHandler.java** (for PUT request)

**In sub package LOGGERS are stored all log (Log.java) implementations:**

- **EnhydraLog.java**
- **Log4jLog.java**

**In sub package UTIL are stored implementations of interfaces from api.util package and some additional classes for basic functionality:**

- **Base64.java :** Encodes and decodes from Base64 notation
- **DesEncryptionImpl.java :** implementation of DesEncryption.java interface
- **EnhydraDMConstants.java :** project constants class (list of all constants)
- **MimeUtility.java :** mime type utility class
- **Numerator.java**
- **TransformUtility.java (xhtml2fo.xsl)**

**In sub package UTIL.FILESYS are implementation classes for mechanism for monitoring and reloading file changes.**

- **FileMonitor.java**
- **PropertyChangeListenerImpl.java** listener for properties files.
- **TemplateChangeListenerImpl.java** listener for xsl template files.

## BUSINESS LAYER
## (Package: org.enhydra.dm.business)

This layer contains:

- Default implementations for major interfaces **DocumentManagerImpl.java**, **DocumentStoreImpl.java** and **FoDocumentManagerImpl.java, UserManagerImpl.java**

(These are default implementations. In **web.xml** file application can be configured to use some custom implementations)

Implementations for beans: **DocumentImpl.java**, **AppUserImpl.java**, **DocumentVersionImpl.java** **FoDocumentIParamImpl.java** and **FoDocumentImpl.java** interfaces (not configurable)

**In sub package EXCEPTIONS are places BasicException.java extensions:**

- **ActionNotAllowedException.java**
- **ConfigurationException.java**
- **DatabaseException.java**
- **DesEncryptionException.java**
- **DocumentStoreException.java**
- **LockException.java**
- **FileSizeException.java**

## PRESENTATION LAYER
## (Package: org.enhydra.dm)

This layer contains in package:

- **EnhydraDM.java**

In sub package: **presentation**

- PO objects implementations.

**Tested with many extensions: doc, docx, xlsx, xls, pps, ppt, msg, pdf, and the others**

### *Feature 1:*

PDF documents, opened via WebDAV, are placed inside browser window instead of related program (Adobe Reader …).

### *Resolution 1:*

Set parameter inside default PDF interpreter (Adobe Reader). This parameter is placed in Edit->Preferences window. Choose "Internet" in list and uncheck "Display PDF in browser" option.

### *Feature 2:*

Office documents, opened via WebDAV, are placed inside browser window instead of related program (Microsoft Office Word, Microsoft Office Excel …).

### *Resolution 2:*

Set parameter in Control Panel->Folders Options window in File Types tab, find needed extension (doc, xls…), click on "Advanced" button and uncheck option "Browse in same window" (In Office 2007 it is disabled and unchecked by default).

### *Feature 3:*

When you open a (e.g.) Microsoft Office XP document in Microsoft Internet Explorer, the document opens as read-only and you cannot save the changes that you make to the document.

### *Resolution 3:*

**Note:** This isn't confirmed with Windows Vista, where we still have this problem.

To open documents in Internet Explorer as read/write so that you can save changes to the document, add the "OpenDocumentsReadWriteWhileBrowsing" sub-key to the registry. To do this, follow these steps.

1. Quit any Office XP programs that are running.
2. Click **Start**, and then click **Run**, type regedit in the **Open** box, and then click **OK**.
   Locate and then click to select the following registry subkey:
3. **HKEY_CURRENT_USER\Software\Microsoft\Office\xx**(version number)**\Common\Internet**

4. On the **Edit** menu, point to **New**, and then click **DWORD Value**.
5. Type **OpenDocumentsReadWriteWhileBrowsing**, and then press ENTER.
6. Right-click **OpenDocumentsReadWriteWhileBrowsing**, and then click **Modify**.
7. In the **Value data** box, type 1, and then click **OK**.
8. On the **File** menu, click **Exit** to quit Registry Editor