**ISPyB**

**Developers Guide**

**Version 5.2 – New Server WildFly 10.1.0**

**Version 5.1 – Use JSON files to import data from User Portal**

**Version 5.0 – Using GitHub**

**Version 4.0 – New Server WildFly 8.2 and using Maven**

**Version 3.7 – New eclipse configuration**

**Version 3.6 – archiving data at ESRF added**

**Version 3.5 - folder structure updated with js, ant targets updated and authentication with the web services added**

**Version 3.4 – default timeout updated**

**Version 3.3 - use of the DoNotCommit.properties files to store sensible data**

**Version 3.2 - new server pyproserv - new logging**

[**1.**](#_1fob9te) **Software required 4**

[**2.**](#_3dy6vkm) **Installation 5**

[*2.1.*](#_1t3h5sf) *Checkout Ispyb from GitHub 5*

[*2.2.*](#_4d34og8) *Install on development platform WILDFLY 10.1 6*

[*2.3.*](#_2s8eyo1) *Database connection - installation 6*

[2.3.1.](#_17dp8vu) Database connection 6

[2.3.2.](#_3rdcrjn) Database installation 6

[*2.4.*](#_26in1rg) *Configuring standalone.xml 7*

[*2.5.*](#_35nkun2) *Using HDF libraries (only needed for BioSaxs) 8*

[*2.6.*](#_1ksv4uv) *Increase JVM memory size 8*

[2.6.1.](#_1664s55) Change the JBoss transaction timeout (may be skipped) 9

[**3.**](#_2jxsxqh) **installation on ispyb (“pyproserv”) machine@ESRF 9**

[**4.**](#_z337ya) **Using Maven 11**

[*4.1.*](#_3j2qqm3) *Install maven 3.1.1 11*

[*4.2.*](#_1y810tw) *Import ISPyB 11*

[*4.3.*](#_4i7ojhp) *Webservices 12*

[*4.4.*](#_2xcytpi) *Site and deployment customization 12*

[**5.**](#_1ci93xb) **Project Structure TODO 14**

[*5.1.*](#_1ci93xb) *Overview 14*

[*5.2.*](#_3whwml4) *Folder structure 14*

[5.2.1.](#_2bn6wsx) Ispyb-ejb 14

[5.2.2.](#_qsh70q) Ispyb-ui 14

[5.2.3.](#_1pxezwc) Ispyb-ws 16

[*5.3.*](#_49x2ik5) *Menu Structure 16*

[5.3.1.](#_2p2csry) Database Model 16

[5.3.2.](#_147n2zr) How it works 16

[**6.**](#_3o7alnk) **First steps 17**

[*6.1.*](#_23ckvvd) *Server Side: First CMPs 17*

[*6.2.*](#_ihv636) *Client Side: First WebPages 17*

[6.2.1.](#_32hioqz) Creating a simple page 17

[6.2.2.](#_1hmsyys) Creating a form 18

[6.2.3.](#_41mghml) Validation and Errors 18

[**7.**](#_2grqrue) **Profiles: site specific files and configuration 20**

[**8.**](#_vx1227) **Naming Conventions TODO 21**

[*8.1.*](#_3fwokq0) *Packages 21*

[8.1.1.](#_1v1yuxt) Server Side 21

[8.1.2.](#_4f1mdlm) Client Side 21

[*8.2.*](#_2u6wntf) *Actions 21*

[*8.3.*](#_19c6y18) *Tiles 21*

[**9.**](#_3tbugp1) **Security 22**

[*9.1.*](#_28h4qwu) *Security with JBOSS 22*

[*9.2.*](#_nmf14n) *ESRF LDAP Structure 22*

[*9.3.*](#_37m2jsg) *Simple authentication 23*

[*9.4.*](#_1mrcu09) *Description of Roles 24*

[*9.5.*](#_46r0co2) *ISPyB Authorization and Authentication Model 24*

[*9.6.*](#_2lwamvv) *Web service authentication 24*

[**10.**](#_111kx3o) **Link with User Portal 25**

[*10.1.*](#_111kx3o) *Update ISPyB from SMIS database@ESRF 25*

[10.1.1.](#_3l18frh) Manual update 25

[10.1.2.](#_206ipza) WSClient to update automatically ISPyB from SMIS 25

[10.1.3.](#_4k668n3) Timeout 25

[*10.2.*](#_2zbgiuw) *Update ISPyB from another Site - TODO 25*

[**11.**](#_1egqt2p) **Link with file systems and Archiving 26**

[*11.1.*](#_3ygebqi) *Link with file systems - TODO 26*

[**12.**](#_2dlolyb) **UML data model 26**

[**13.**](#_sqyw64) **Using junit testing TODO 26**

[*13.1.*](#_3q5sasy) *standalone mode 26*

[*13.2.*](#_3cqmetx) *testing 26*

[Write tests extending the EJB3Test. 26](#_1rvwp1q)

[**14.**](#_4bvk7pj) **Some hints 26**

[*14.1.*](#_2r0uhxc) *Start local JBoss using the machine name 26*

# **Software required**

All the software required may be directly downloaded from internet

* A Java SE Development Kit JDK 8
  + <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
* WildFly Application Server 10.1.0Final
  + <http://wildfly.org/downloads/>
* Eclipse IDE for java EE developers (last version)
  + http://www.eclipse.org/downloads/
  + from eclipse interface connect to marketplace and fetch JBOSSTOOLS.
* XDoclet 1.2.3
  + <http://xdoclet.sourceforge.net/xdoclet/install.html>
* MySQL 5.X
  + <http://dev.mysql.com/downloads/mysql/5.2.html>
* Axis1.4 libraries
  + http://axis.apache.org/axis/

# **Installation**

## ***Checkout Ispyb from GitHub***

1. Clone or fork the ISPyB repository and then clone it by typing:

git clone <https://github.com/ispyb/ISPyB.git>

1. ISPyB uses some local libraries located on /dependencies then some jars should be added to your local maven repository

cd dependencies

mvn install:install-file -Dfile=securityfilter.jar -DgroupId=securityfilter -DartifactId=securityfilter -Dversion=1.0 -Dpackaging=jar

mvn install:install-file -Dfile=securityaes.jar -DgroupId=securityaes -DartifactId=securityaes -Dversion=1.0 -Dpackaging=jar

mvn install:install-file -Dfile=jhdf.jar -DgroupId=jhdf -DartifactId=jhdf -Dversion=1.0 -Dpackaging=jar

mvn install:install-file -Dfile=jhdf5.jar -DgroupId=jhdf5 -DartifactId=jhdf5 -Dversion=1.0 -Dpackaging=jar

mvn install:install-file -Dfile=jhdf5obj.jar -DgroupId=jhdf5obj -DartifactId=jhdf5obj -Dversion=1.0 -Dpackaging=jar

mvn install:install-file -Dfile=jhdfobj.jar -DgroupId=jhdfobj -DartifactId=jhdfobj -Dversion=1.0 -Dpackaging=jar

mvn install:install-file -Dfile=Struts-Layout-1.2.jar -DgroupId=struts-layout -DartifactId=struts-layout -Dversion=1.2 -Dpackaging=jar

mvn install:install-file -Dfile=ojdbc6.jar -DgroupId=ojdbc6 -DartifactId=ojdbc6 -Dversion=1.0 -Dpackaging=jar

mvn install:install-file -Dfile=ispyb-WSclient-userportal-gen-1.3.jar -DgroupId=ispyb -DartifactId=ispyb-WSclient-userportal-gen -Dversion=1.3 -Dpackaging=jar

1. Build the project by using maven

mvn clean install

If the build has succeed a summary repost should appear:

[INFO] Reactor Summary:

[INFO]

[INFO] ispyb-parent ...................................... SUCCESS [0.251s]

[INFO] ispyb-ejb3 ........................................ SUCCESS [10.243s]

[INFO] ispyb-ws .......................................... SUCCESS [1.751s]

[INFO] ispyb-ui .......................................... SUCCESS [7.212s]

[INFO] ispyb-ear ......................................... SUCCESS [5.048s]

[INFO] ispyb-bcr ......................................... SUCCESS [2.217s]

[INFO] ispyb-bcr-ear ..................................... SUCCESS [1.806s]

## ***Install on development platform WILDFLY*** 10.1

Firstly, the softwares above (see §1) have to be installed along with the definition of the following environment variables and respective locations:

* JAVA\_HOME=/j2sdk\_root/j2sdk1.8.x.x
* JBOSS\_HOME=/jboss\_root/wildfly-10.1.0Final

Then, the project must be imported from GitHub or other folder to Eclipse and the respective paths referencing libraries corrected.

To avoid errors when using specified-type list xdoclet 1.2.3 needs a small fix:

Use xjavadoc-1.5-snapshot050611.jar to replace xjavadoc-1.1.jar in xdoclet lib directory.

(Find this jar in the files stored on the forge: https://forge.epn-campus.eu/projects/ispyb/files). Remove the xjavadoc-1.1.jar.

## ***Database connection - installation***

### **Database connection**

Copy the “mysql” folder present from ispyb-parent/configuration/mysql to the “wildfly-10.1.0.Final\modules\system\layers\base\com” folder.

### **Database installation**

Run the creation scripts present in the module ispyb-ejb, to run the scripts you will need a user “pxadmin” with full permissions.

ispyb-ejb/db/scripts/pyconfig.sql

This corresponds to the menu options, and contains both structure and data

ispyb-ejb/db/scripts /pydb.sql

This corresponds to the ISPyB metadata and contains only the database structure.

ispyb-ejb/db/scripts/schemaStatus.sql

This corresponds to the entries present in SchemaStatus table and gives an overview of the executed update scripts.

The creation scripts are normally updated for each tag, but if you are using the trunk version you may have to run the update scripts present in :

ispyb-ejb/db/scripts/ahead

Check before the entries in SchemaStatus table to know which scripts to execute.

The scripts already run for the current tag are in :

ispyb-ejb/db/scripts/passed

## ***Configuring standalone.xml***

Save the original standalone.xml from wildfly-10.1.0.Final\standalone\configuration\standalone.xml to

wildfly-10.1.0.Final\standalone\configuration\standalone.xml.orig

Copy the standalone.xml.example present in ispyb-parent/documentation to wildfly-10.1.0.Final\standalone\configuration.

Customize it with your database :

Update the datasources :

<datasource jndi-name=*"java:jboss/ispybconfigDS"* pool-name=*"ispybconfigDS"* enabled=*"true"* use-java-context=*"true"*>

<connection-url>jdbc:mysql://pydevserv.esrf.fr:3308/pyconfig</connection-url>

<driver>mysql-connector-java-5.1.21.jar</driver>

<security>

<user-name>pxuser</user-name>

<password>\*\*\*\*\*</password>

</security>

</datasource>

<datasource jndi-name=*"java:jboss/ispybDS"* pool-name=*"ispybDS"* enabled=*"true"* use-java-context=*"true"*>

<connection-url>jdbc:mysql://pydevserv.esrf.fr:3308/pydb</connection-url>

<driver>mysql-connector-java-5.1.21.jar</driver>

<security>

<user-name>pxuser</user-name>

<password>\*\*\*\*\*\*</password>

</security>

</datasource>

Change the drivers definition if needed :

<driver name=*"mysql-connector-java-5.1.21.jar"* module=*"com.mysql"*/>

Add a security domain (here an example using LDAP):

<security-domain name=*"ispyb"* cache-type=*"default"*>

<authentication>

<login-module code=*"ispyb.server.security.LdapLoginModule"* flag=*"required"*>

<module-option name=*"java.naming.factory.initial"* value=*"com.sun.jndi.ldap.LdapCtxFactory"*/>

<module-option name=*"java.naming.provider.url"* value=*"ldap://ldap.esrf.fr:389/"*/>

<module-option name=*"java.naming.security.authentication"* value=*"simple"*/>

<module-option name=*"allowEmptyPasswords"* value=*"false"*/>

<module-option name=*"principalDNPrefix"* value=*"uid="*/>

<module-option name=*"principalDNSuffix"* value=*",ou=People,dc=esrf,dc=fr"*/>

<module-option name=*"groupUniqueMember"* value=*"uniqueMember"*/>

<module-option name=*"groupAttributeID"* value=*"cn"*/>

<module-option name=*"groupCtxDN"* value=*"ou=Pxwebgroups,dc=esrf,dc=fr"*/>

</login-module>

</authentication>

</security-domain>

Add a listener for ajp if needed

<server name="default-server">

<ajp-listener name="ajp" socket-binding="ajp" max-parameters="20000"/>

<http-listener name="default" socket-binding="http" max-parameters="20000"/>

In you want to be able to edit directly the jsps without having to redeploy, change:

<jsp-config/>

To :

<jsp-config development="true"/>

Configure the logging and add the user logged to every entry :

<formatter name="PATTERN">

<pattern-formatter pattern="%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) (userId %X{userId}) %s%e%n"/>

</formatter>

## ***Using HDF libraries (only needed for BioSaxs)***

Create a module hdfgroup in :

modules/system/layers/base/org/

to have :

modules/system/layers/base/org/hdfgroup/lib64

Copy into the lib64 directory above the:

libjhdf.so libjhdf.dll

libjhdf5.so libjhdf.dll

files present in ispyb-parent/configuration/hdf5 folder.

Add the HDF5 config JVM OPTS in the standalone.conf  
-Djava.library.path=/ispyb/jboss/modules/system/layers/base/org/hdfgroup/lib64

## ***Increase JVM memory size***

In the file ${JBOSS\_HOME}\bin\standalone.conf replace

JAVA\_OPTS="$JAVA\_OPTS -Dprogram.name=$PROGNAME"

By (on local computer)

JAVA\_OPTS="$JAVA\_OPTS -Dprogram.name=$PROGNAME -Xms256m -Xmx1024m -XX:MaxPermSize=512m"

or on the production server by

JAVA\_OPTS="$JAVA\_OPTS -Dprogram.name=$PROGNAME -Xms1024m -Xmx4096m -XX:MaxPermSize=1024m"

This is to avoid the “Out of Memory” error when processing big request to the DB, per example with the update ISPyB from SMIS database process.

in Eclipse, double click on the server name in the server window to open the jboss overview, then click on "Open launch configuration" and add the above parameters in the VM arguments.

### **Change the JBoss transaction timeout (may be skipped)**

The default value is 300, if you need to change :

Login To Wildfly Management Console (localhost:9990 by default)

Go To Configuration > container > Timeout and set the new value (for example 900)

But a best way is to set an annotation directly on the method:

@TransactionTimeout(value = 900, unit = TimeUnit.SECONDS)

This is to avoid the transaction timeout when running the cron job in charge of the database update (WSClient).

# **installation on ispyb (“pyproserv”) machine@ESRF**

Same as §2.1 except for xdoclet. There is no need to install them.

* jboss is on /ispyb/jboss

Special ispyb commands exist and should be used to start, stop, copy .ear, launch database update from smis.

It is possible to have their full list and how to use them by typing : ispyb help

For example :

* to stop jboss server: ispyb stop jboss
* to start jboss server: ispyb start jboss
* to copy new .ear : ispyb deploy ispyb.ear

Only files in jboss/server/default/conf and in jboss/server/default/deploy can be updated

Backups repositories exist which store old versions of copied and replaced files : /ispyb/backup-deploy and /ispyb/backup-conf

Differents logs can be find on /usr/local/logs

* jboss\_console.log -> to see it use : tail –f jboss\_console.log
* jboss\_run.log : keeps track of stops and restarts of jboss
* ispyb\_run.log : keeps track of ispyb commands

New folder for uploaded files in : /nobackup.uploads/files (up to 4 Go)

# **Using Maven**

## ***Install maven 3.1.1***

<!-- RFC: ispyb.site is already on pom.xml and could be moved to ispyb-parent/pom.xml -->

Configure your own settings.xml (copy the one from maven/conf and update them for you)

A part of the site profile are set in this file, see an example in documentation/settings.xml.example.

## ***Import ISPyB***

Import the ispyb-parent project  
in the project ispyb-parent" you will find the 6 projects :

* ispyb-ear : contains the 3 followings modules as jars and wars, has to be deployed on the server
* ispyb-ejb : contains all the ejbs + some others classes like security, commons, hdf5, it contains also all the properties and the Constants class which will be used by ispyb-ws and ispyb-ui modules.
* ispyb-ui : contains the "web client" part with all the web user interfaces , struts actions/forms, tiles, jsp, js, ...
* ispyb-ws : contains all the webservices
* ispyb-bcr: contains the “web client” part of the bar code reader application used for the dewar tracking
* ispyb-bcr-ear: contains the preceeding module as war and is linked to the ispyb-ejb through a dependency in the pom.xml, this module can be deployed or not separately from ispyb.ear.

In Eclipse, select "import project", then select the maven project and point to the pom.xml of the ispyb-parent

If some projects are missing, select once more "import project", maven project, and point to the pom of the missing module.

Run maven install on the 4 modules.

You will need to install missing jars (maven can not find them in maven repository) picking them in the jboss6 ispyb installation :

Added all these libraries in ispyb-parent/configuration/maven/jar as well as a script (installCustomJavaLibrariesToMaven.sh) that will install them on the maven local repository

mvn install:install-file -Dfile=xxx\ISPyB\_project\server\lib\securityfilter.jar -DgroupId=securityfilter -DartifactId=securityfilter -Dversion=1.0 -Dpackaging=jar  
mvn install:install-file -Dfile=xxx\ISPyB\_project\server\lib\securityaes.jar -DgroupId=securityaes -DartifactId=securityaes -Dversion=1.0 -Dpackaging=jar  
mvn install:install-file -Dfile=xxx\server\default\lib\jhdf.jar -DgroupId=jhdf -DartifactId=jhdf -Dversion=1.0 -Dpackaging=jar  
mvn install:install-file -Dfile=xxx\server\default\lib\jhdf5.jar -DgroupId=jhdf5 -DartifactId=jhdf5 -Dversion=1.0 -Dpackaging=jar  
mvn install:install-file -Dfile=xxx\server\default\lib\jhdf5obj.jar -DgroupId=jhdf5obj -DartifactId=jhdf5obj -Dversion=1.0 -Dpackaging=jar  
mvn install:install-file -Dfile=xxx\server\default\lib\jhdfobj.jar -DgroupId=jhdfobj -DartifactId=jhdfobj -Dversion=1.0 -Dpackaging=jar  
mvn install:install-file -Dfile=xxx\ISPyB\_project\client\lib\struts-layout-1.2.jar -DgroupId=struts-layout -DartifactId=struts-layout -Dversion=1.2 -Dpackaging=jar

**To use the BCR for dewar tracking :**  
mvn install:install-file -Dfile=dewarAPI.jar -DgroupId=ispyb -DartifactId=dewarAPI -Dversion=1.0 -Dpackaging=jar

**To use the jar built from SMIS WS client: (the jar is found in ispyb-ejb/lib)**

mvn install:install-file -Dfile= ispyb-WSclient-userportal-gen-1.1.jar -DgroupId=ispyb -DartifactId= ispyb-WSclient-userportal-gen -Dversion=1.1 -Dpackaging=jar

Of course this jar contains ESRF User Portal classes but at least, the compilation will be OK on every site, and then a custom generation of WS client corresponding to the site can be developed.

## ***SOAP Webservices from User Portal***

Import the project ispyb-WSclient-userportal-gen to generate a jar containing the classes built from wsdl  
Customize the bindings.xml file in src/main/resources to the correct wsdl location

The ispyb-WSclient-userportal-gen-1.1.jar is created with this project.

## ***Site and deployment customization***

In the pom.xml of ispyb-ui and ispyb-ejb you will find profiles, there are for now 2 types of profiles :

* **Deployment profiles**

DEV, ALT, and PROD define the way the application is deployed : for example with DEV, the javascript is not minimized

* **Site configuration**

ESRF, EMBL, SOLEIL, MAXIV contains the properties previously defined in the ISPyB\_XXX.properties,

you will have to add your own entries taking in example the ESRF ones and copying the properties from your files ISPyB\_XXX.properties, which are still present in ispyb-ui/src/main /resources/OldProperties\_obsolete.

These profiles with their properties are defined in the ispyb-ejb/pom.xml only because the other modules are dependent on this one.

* **DoNotCommit.properties**

These properties have been put outside of the project, in the maven settings.xml, so you can safely define them, and get rid of the DoNotCommit.properties

# **Project Structure**

## ***Overview***

The project structure is divided in 3 big areas (the main folders):

* Ispyb-ui – which contains all the client side files.
* Ispyb-ejb – which contains all the server side files + commons + properties
* Ispyb-ws – which contains all the web services.

Besides that, there is one more folder in the project root:

* Ispyb-parent – This folder is the parent folder, containing also some documentation and configuration files

## ***Folder structure***

### **Ispyb-ejb**

\src\main\java

Source classes for server side: *Entity EJBs (CMPs)* and *Session Façade EJBs*.

Source classes for common classes

\src\main\resources\META-INF

persistence.xml info:

* persistence.xml

\sql\_scripts

Folder to store the sql scripts to generate , update the database structure and data.

### **Ispyb-ui**

\client\src

Source classes for client side: *Struts* *Actions* and *Forms* and others classes.

\client\build

Compiled classes from \client\src forder.

\client\dist

War client file, which contains \client\build, \client\generate\meta, \client\lib and \client\web folders.

\client\etc\conf

Configuration files to be deployed in *WEB-INF* web application folder. At the moment there are two files:

* authfilter.xml – Authentication configuration for this web application.
* tiles-defs.xml – Tiles definition file.

\client\etc\mergedir

Folder where are the files that will be mixed with the *XDoclet* generated tags defined in the files located in \client\src folder to create the following files:

* jboss-web.xml
* struts-config.xml
* web.xml

For additional information about the names of the files to be put in this folder, consult the files with the names above and located in \client\generate\meta folder.

\client\etc\resources

Folder where the resources properties are.

\client\generate\meta

XDoclet generated meta info. Contains web, Struts and Jboss descriptors:

* jboss-web.xml
* struts-config.xml
* validation.xml
* web.xml

\client\generate\src

This folder was not used at the moment of this manual.

\client\lib

Folder with the libraries used by the client. At the moment there are only struts framework libraries.

\client\web

This folder is the root of the web application and contains other subfolders:

* \client\web\css – Cascade style sheets files.
* \client\web\images – images.
* \client\web\js – js files:
  + \client\web\js\css – all css used
  + \client\web\js\external – external js used
  + \client\web\js\ispyb – js files for ispyb:
    - \client\web\js\ispyb\biosaxs – js files for biosaxs
    - \client\web\js\ispyb\min – js file minified
    - \client\web\js\ispyb\mx – js files for ispyb mx
    - \client\web\js\ispyb\utils – js files utils
* \client\web\tiles – *JSPs* for tiles applications:
  + \client\web\tiles\bodies – All bodies of the pages. In this folder there are several folders to better divide the different pages.
  + \client\web\tiles\common – Common *JSPs* fragments to all website:
    - \client\web\tiles\common\footer
    - \client\web\tiles\common\header
    - \client\web\tiles\common\left
  + \client\web\tiles\layouts – *JSPs* with the layout definitions of the pages.
  + \client\web\WEB-INF - Internal tld files for struts framework.

\client\applet\src

This folder contains the source of applets used in html pages. At the moment only exists one.

\client\applet\build

This folder contains the applets built.

\client\applet\lib

This folder contains all the JDLImage code modified to better fit our project and built in a jar file. This project is a courtesy of John Campbell (<http://www.ccp4.ac.uk/jwc/image_applet/ImageDisplay_ccp4.html>) and the source code is present on CVS.

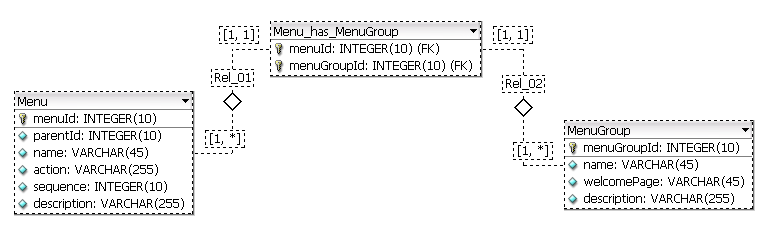
### **Ispyb-ws –TODO -**

\src\main\java

Source classes for webservices

## ***Menu Structure***

### **Database Model**



### **How it works**

There three main tables:

* Menu – contains the hierarchy of menus, menu target URL and menu name.
* MenuGroup – Contains all sort of groups existing in ISPyB.
* Menu\_has\_MenuGroup – Establish the relation m-n of the above tables.

However, just the two first tables are mapped as EJBs.

# **First steps**

## ***Server Side: First CMPs***

to be filled with ejb3 automated generation or using ispyb templates.

The eclipse templates are found on :

/server/etc/ispyb\_templates.xml

See ISPyB\_ejb3.doc

The database connection shall be set in the persistence.xml file which should be in META\_INF of ejb3.jar, it is found on server/config/ejb3/META-INF.

## ***Client Side: First WebPages – OBSOLETE with EXI***

### **Creating a simple page**

This project uses Tiles for layout presentation.

Thus, instead of creating a complete page with included tags repeatedly for every page (for instance, to include header, footer and menus), we just create one part of the page.

For most of the pages associated to business functionalities a body is just created. The rest of the layout of this page is defined in tiles configuration file.

To create a simple text page it is needed to:

1. Create a JSP with the body of the page and put it in \client\web\tiles\bodies\<content related folder>\
2. Create an entry with the page definition (definition tag with extends attribute) in \client\etc\conf\tiles-defs.xml. It is necessary to define the name of the page, which layout it uses and fill in all the values defined in that layout. You can check the layout definition in the top of this xml file. For instance:

<definition name="user.welcome.page" extends="site.main.layout" >

<put name="title" value="ISPyB" />

<put name="body" value="/tiles/bodies/welcome/user.jsp" />

<put name="links" value="null" />

</definition>

1. Then, and because we cannot request a page in the way it has been previously defined, we have to create an action to call the new page (only if is called directly by http request). To that, we must create a new entry tag called action in \client\etc\mergedir\struts-actions.xml which has two main attributes: path, the name requested in URL, and forward, the previous definition in tiles-defs.xml.

<action

path="/user/welcomeUserPage"

forward="user.welcome.page"

scope="request"

validate="false"/>

### **Creating a form**

To create a form submission page is necessary to create a page – a page with a form – in the way like described above. The JSP body of this page has to contain the following struts form tag: <html:form action="<*action name*>.do">

The action name defined in the tag above is the action to be invoked when the submit button is pressed.

This action is an **Action Class** and the form passed to it is the **Action Form** and contains the data filled in the page form. Both definitions have to be developed.

For the Form Action there are 3 possibilities to define it:

1. If you don’t need JavaScript validation before submit the form, it can be defined in \client\etc\mergedir\struts-forms.xml as a DynaValidatorForm. (See logon)
2. If you need JavaScript validation the easiest way to do it is coping the auto generated form file which is in \server\generate\form to client source folder, modify it in order to have only the fields you need and add XDoclet @struts.validator tags for validation requirements.
3. You can also simply copy one already existing

All parameters passed through Action Form belong to the request.

Only one Form : **BreadCrumbForm** is used to keep some parameters in the session. These parameters will be displayed in the breadcrumbs bar at the top of the body of the page to show useful information.

To code the action, create a new class in client source folder which extends either Action or DispatchAction class and include on the top of it the XDoclet definitions for this action:

* @struts.action
  + name – name of the form defined in struts-forms.xml
  + path – <name of the action>.do
  + input – Tiles definition page where the form is filled.
  + parameter – in case of using DispatchAction, this parameter is used to select the name of the method defined in the action class.
* @struts.action-forward
  + name – Name of the forward returned by this action.
  + path – Tiles definition page to be forwarded.

### **Validation and Errors**

Struts Layout only provides features for validation on server side.

The validation consists in putting messages in one queue (ActionErrors) which uses a different name to index every entry:

* ActionErrors.GLOBAL\_ERROR - For global errors.
* The name of the field in the form.

It's is also important to use the the saveErrors method to save only once the queue on the request: saveErrors(request, errors)

The following example is standard use of what has been described:

public ActionForward displayFromMenu(ActionMapping mapping, ActionForm actForm, HttpServletRequest request, HttpServletResponse response) {  
  
    ActionErrors errors= new ActionErrors();  
      
     try {  
  
        (...)  
        // Check Required fields populated  
        // The error queue is a parameter  
        if (!this.validate(form, errors)) {  
            ActionForward f =  mapping.findForward("shippingCreatePage");  
            saveErrors(request, errors);  
            return f;  
        }  
        (...)  
        // EXAMPLE OF GLOBAL ERROR  
        errors.add(**ActionErrors.GLOBAL\_ERROR**, new ActionError("errors.detail", "Nah Nah Nah c'est pas bon"));  
          
    } catch (Exception e) {  
        errors.add(**ActionErrors.GLOBAL\_ERROR**, new ActionError("errors.detail", e.toString()));  
        ClientLogger.getInstance().error(e.toString());  
    }  
      
    saveErrors(request, errors);  
      
    if (!errors.isEmpty()) {      
        return (mapping.findForward("error"));  
    }  
    return mapping.findForward("shippingViewPage");  
}  
   
private boolean validate(ViewShippingForm form, ActionErrors errors){  
    boolean requiredFieldsPresent = true;  
     
    if (form.getInfo().getProjectCode().length() == 0){  
        requiredFieldsPresent = false;  
        ActionError actionError = new ActionError("errors.required","Shipping Label"); // Shipping Label  
        errors.add(**"info.projectCode"**, l\_actionErrorPassword);  
    }  
    return requiredFieldsPresent;  
}

# **Profiles: site specific files and configuration**

You can set the ISPyB.site value in the profile of pom.xml in ispyb-ejb module.

This value is then used in the Constants class (SITE\_IS\_ESRF() or SITE\_IS\_SOLEIL() for example).

The properties are defined in ISPyB.properties present in the ispyb-ejb module, and the correct values depend on the SITE where it is installed.

SITE customization properties are defined in the profile in the pom.xml of ispyb-ejb module

# **Naming Conventions TODO**

## ***Packages***

### **Server Side**

All packages should have the prefix:

* ispyb.server.\*

The Entity and Façade EJBs should have the prefix:

* ispyb.server.data – for experiments database
* ispyb.server.config – for configuration database

### **Client Side**

All packages should have the prefix:

* ispyb.client.\*

## ***Actions***

All Action classes should have the suffix:

* Action

The name of the class has be the same as the action name:

* [name]Action.java
* [name].do

For example:

/\*\*

\* @struts.action path="/security/**logon**"

\*/

public class **Logon**Action extends Action {

## ***Tiles***

All definitions created on tiles-defs.xml should have as name the prefix:

* [group name].[menu name].[identification name].page

For instance:

* user.shipping.container.create.page

This **page** belongs to the **user** group, menu **shipping** and **creates** a **container**.

# **Security**

## ***Security with JBOSS***

JBoss has several predefined ways to do the authentication and authorization.

To take advantage of these features, an application-policy should be defined in standalone.xml or a new one has to be developed and configured in this file.

In the client side the policy to apply should be defined in web application jboss-web.xml which is created from compilation using the file :

\client\etc\mergedir\jbossweb-resource-env-ref.xml

<!-- Security domain defined in standalone.xml -->

<security-domain>java:/jaas/ispyb </security-domain>

<context-root>ispyb</context-root>

## ***ESRF LDAP Structure***

Because LDAP configuration of the ESRF can not be used with standard LDAP Module provided with JBOSS, a new LDAP module has been developed.

Following is described how to configure the authentication with the developed LDAP Module.

ESRF LDAP configuration:

All groups belonging to this group:

dn: ou=Pxwebgroups,dc=esrf,dc=fr

ou: Pxwebgroups

objectClass: top

objectClass: organizationalunit

Example of the User group:

dn: cn=User,ou=Pxwebgroups,dc=esrf,dc=fr

ou: Pxwebgroups

objectClass: top

objectClass: groupOfUniqueNames

cn: User

uniqueMember: uid=ifx999,ou=people,dc=esrf,dc=fr

uniqueMember: uid=mx9999,ou=people,dc=esrf,dc=fr

uniqueMember: uid=leal,ou=people,dc=esrf,dc=fr

See the entries added in standalone.xml in 2.4

In the pom.xml of ispyb-ejb, set the authentication method property value as:

<ispyb.authentication.method>LDAP</ispyb.authentication.method>

## ***Simple authentication***

One of the aims of ISPyB is the portability and therefore the possibility of using it in others facilities. The simple authentication is a built in module which provides the same authentication schema as we use in ESRF LDAP. Its name is UsersRolesLoginModule.

The policy should be added in the standalone.xml.See 2.4

<security-domain name="ispyb" cache-type="default">

<authentication>

<login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule" flag="required">

<module-option name="usersProperties" value="${jboss.server.config.dir}/users.properties"/>

<module-option name="rolesProperties" value="${jboss.server.config.dir}/roles.properties"/>

</login-module>

</authentication>

</security-domain>

Users and the respective passwords can be added in the following file wildfly-server\standalone\configuration/users.properties with the syntax: <USERNAME>=<PASSWORD>

The roles assigned to each user are defined in jboss- server\standalone\configuration/roles.properties with the syntax: <USERNAME>=<ROLE>,<ROLE>]

These are examples of both files:

* roles.properties

mx9999=User

ifx999=User

manager=User, Manager

* users.properties

mx9999=password1

ifx999=password2

manager=manager

In the pom.xml of ispyb-ejb, set the authentication method property value as:

<ispyb.authentication.method>SIMPLE</ispyb.authentication.method>

## ***Description of Roles***

**User:**

Default role used for proposal accounts like “MXyyy”

Role assigned also to a person when some proposals are defined for this person in the ProposalHasPerson table.

**Localcontact:**

Role assigned to the scientist in charge of the experiment, to give access to all the sessions / shipments where she/he is localcontact.

**Industrial:**

Role assigned to industrial users, which is nearly the same as User but with a bit less options.

**Store**

Role assign to the people at store to track the shipments/parcels/dewars

**Webservice:**

Special role to be used to connect to ISPyB by Webservice

**Manager:**

Manager role to access all data

## ***ISPyB Authorization and Authentication Model***

The authentication method in ISPyB is based in the standard tomcat servlet: j\_security\_check. See /tiles/bodies/ logon/login.jsp for more details.

The authorization method is based in URL patterns.

See /client/etc/mergedir/web-security.xml for constraints and roles defined.

## ***Web service authentication***

The web service authentication is done through ldap at ESRF.

@RolesAllowed({ "WebService", "User", "Industrial"})

@SecurityDomain("ispyb")

@WebContext(authMethod="BASIC", secureWSDLAccess=**false**, transportGuarantee="NONE")

@RolesAllowed specifies the list of roles permitted to access web services.

@SecurityDomain references the application policy defined in the login-config.xml file. (see 8.5.1)

@WebContext enables HTTP basic authentication.

# **Link with User Portal**

## ***Update ISPyB from SMIS database@ESRF***

### **Manual update**

For each individual proposal

For each proposal account there is the possibility to update the database: all sessions and samplesheets information concerning the proposal will be updated.

There is also the possibility to update only the samplesheet information.

### **WSClient to update automatically ISPyB from SMIS**

WSClient is another eclipse java project which is saved on SVN.

It contains the client of the web service created from the class “UpdateFromSMIS.java” of ISPyB project (ispyb.server.smis).

The class is “BatchUpdateFromSMIS.java”

The client is called from a cron job on the ispyb machine every night (launched at 04:12 am)

In the case this update fails, there is always the possibility to update manually the database.

### **Timeout**

It is possible to change the timeout of the web service on the ispyb server, see 2.3.4.

## ***Update ISPyB from another Site***

Either you can re-use the Webservices developed for our User Portal (SMIS) or you will have to connect your own to be able to fill the Proposal/Session/Protein tables

Note that the installed jar “ispyb-WSclient-userportal-gen-3.0.jar” contains all the generated classes for ESRF SMIS and will allow the compilation on every site, of course, then it is required to generate the appropriate clients.

The classes calling these webservices to fill the data coming from the User Portal are:

ispyb/server/smis/ScientistsFromSMIS

ispyb/server/smis/UpdateFromSMIS

According to the property set in the pom.xml:

<ispyb.userportal.link>JSON</ispyb.userportal.link>

If it is set to ‘SMIS’ then the SOAP webservices will be used.

If it is set to ‘JSON’ then these classes will use json files to import the data coming from user portal.

An example of such json files are present in ispyb-parent/documentation/UserPortalLink.

Some « test » json strings are provided in the ISPyB.properties and are called by default in the code.

The methods to use from a User Interface are ready in ispyb-ejb/ispyb/server/userportal:  
UserPortalUtils.*getSamples*(proposalName);

UserPortalUtils.*getMainProposers*(proposalName);

UserPortalUtils.*getSessions*(proposalName);

UserPortalUtils.*getLabContacts*(proposalName);

These methods read the corresponding “proposal\_json\_samples” file present in the server in a folder defined in the properties

<ispyb.upload.folder.json>/data/pyarch/pdb/json/</ispyb.upload.folder.json>

An example is written in ispyb-ws:ispyb/ws/rest/proposal/UserPortalRestWebService which loads the files present in the dedicated folder on the server and integrate them in ISPyB database.

# **Link with file systems and Archiving**

## ***Link with file systems - TODO***

/pyarch

# **UML data model**

The database model is written using DBDesigner (free download from sourceforge). Then it is possible to connect to a MySQL database to update either the database or the model, or to generate a new database.

When using dbdesigner, **do not use** standards inserts, and default values, sometimes these lead to error when synchronising database with model.

# **Using junit testing TODO**

## ***testing***

Write tests extending the EJB3Test.

# **Some hints**

jsp code : when using " inside a value = " use \" instead

## ***Start local JBoss using the machine name***

Run JBoss with the option

-b 0.0.0.0