SAP E-Commerce development on open source

How To Guide

Contents

E-Commerce Dev Environment Setup

NOTE: Please respect our Intellectual Property and do not distribute any parts of this documentation or referred source code and other content without prior approval. We take no responsibility of any issues that might arise from following these instructions from a technical, legal or any other aspect.

This document steps through how to turn the skeleton into a completed local development environment.

Background

- The use of Eclipse is not required, but it is what we use. So the notes are in terms of Eclipse as they have been extracted from existing documentation or will become internal documentation.
- The installation locations of tools such as Eclipse, JDKs, etc are not generally important, they are stated/suggested in the documentation then assumed in the remaining documentation and in included configuration files.
- The use of JAD /JadClipse is not necessary, but it is convenient as a fair amount of interesting/problematic libraries do not ship with readily accessible source.
- Some attempts have been made to make it possible to run multiple local development environments on a single machine, things like port numbers and temporary file locations are relatively easily changed.
- Much of the local development environment has evolved over time, so some things are not as clean as they could be.
- We are running filesystem based XCM, whereas our target environments are DB based, this is partially
    - for historic reasons (weren't permitted the required database access)
    - for separation of developer environments
- The skeleton contains no SAP source, libraries or other artifacts, these will all need to be retrieved from your environment.
- The web shop installations we have been dealing with run against CRM backends.
- Multiple versions of some libraries will exist (e.g Servlet API, the NetWeaver AS one that is built against and the more recent Jetty one which is used locally at runtime)

Required Software

Ensure the following software is installed:

- Sun JDK 1.6.0 - for Eclipse, latest version (`C:\Tools\jdk\1.6.0_21`)
- Sun JDK 1.4.2 - for Web Shop Jetty, `1.4.2_19` is presumed (`C:\Tools\jdk\1.4.2_19`), consistent with the NetWeaver AS host JVM
- Eclipse 3.6 - Helios simultaneous release, Java EE (`C:\Tools\eclipse\3.6`)
- JAD (`C:\Tools\jad`)

Windows System Configuration

Add the following line to `%WinDir%\system32\drivers\etc\services` (on Windows XP `%WinDir%` evaluates to `C:\WINDOWS`)

```
sapms<SID>     36<INST>/tcp

sapmsXYZ       3600/tcp
```

Initial Eclipse Configuration

- Check that the Ecipse installation is using an appropriate JVM
  - The file `C:\Tools\eclipse\3.6\eclipse.ini` should contain
    - The following lines before the `-vmargs` line:

      ```
      -vm

      C:\Tools\jdk\1.6.0_21\jre\bin
      ```

    - Note that if the 1.6.0_21 JDK is installed the following line should be added after the `-vmargs` line:

      ```
      -XX:MaxPermSize=256m
      ```

- Create a new workspace:
  - `C:\Work\SISU\workspaces\workspace-b2b`

- Installed JREs
  - Menu: Window / Preferences
  - Tree: Java / Installed JREs
  - Should already have an entry named `1.6.0_21`
  - Button: Add
    - Item: Standard VM
    - Button: Directory... (next to the `JRE home` field)
    - Select: `C:\Tools\jdk\1.4.2_19`
    - Fields will be filled automatically
    - Ensure the `JRE name` is set to `1.4.2_19`
    - Button: Finish
  - Check the box next to the `1.4.2_19` item (this sets the default workspace JRE)
  - Button: OK

- Java compliance level
  - Menu: Window / Preferences
  - Tree: Java / Compiler
  - Set `Compiler compliance level` to `1.4`
  - Note that if you did not close the Preference dialog after the setting up the JREs you will see a warning message.

    - Button: OK

- Install JadClipse
    - Download: http://jadclipse.sourceforge.net (current version 3.3)
    - Shutdown Eclipse
    - Drop the JAR file into the Eclipse plugins directory
        - `C:\Tools\eclipse\3.6\plugins`
    - Create a temporary directory (the creation of this directory will be done in a later stage by a build script)
        - `C:\Work\SISU\tmp\b2b\jad`
    - Restart Eclipse

- Configure JadClipse
    - Menu: Window / Preferences
    - Tree: Java / JadClipse
        - Path to decompiler: `C:\Tools\jad\jad`
        - Directory for temporary files: `C:\Work\SISU\tmp\b2b\jad`
    - Tree: Java / JadClipse / Debug
        - Check: Output original line numbers as comments
        - Check: Align code for debugging
    - Button: OK

Eclipse project setup

To avoid having to open up our Subversion repository to all potential users of this document we decided to distribute a compressed archive containing the Eclipse workspace. You can find the archive from the same page where you downloaded this document from.

SAP JCo Library

Download the latest `2.1.x` version of the SAP Java Connector for the development platform, all our development to date has been on 32 bit Windows platforms. Later versions of the JCo libraries, such as `3.0.x`, have been *repackaged* and are inconsistent with what the web shop expects.

Unpack the JCo distribution file (`sapjco-ntintel-2.1.9.zip`)

- Copy `sapjco.jar` to:
    - `webshop-b2b/lib/local/sapjco`
    - `jco-logging/lib`
- Copy `sapjcorfc.dll` and `librfc32.dll` to:
    - `webshop-b2b/jetty-6.1.25/native`
- Check that the following files exist, they are MS Visual Studio 2003 runtime libraries, which should have been distributed, but weren't. Chances are other software will have installed them already.
    - `%SystemRoot%\system32\msvcr71.dll`
    - `%SystemRoot%\system32\msvcp71.dll`

Retrieve Web Shop Deployment Archive

Locate the deployment archive that contains the web shop, it should have a name like:

- `SAPSHRAPP*.SCA`

The `*.SCA` file is a ZIP archive, extract it, it should include the file:

- `DEPLOYMENTARCHIVES\crm~b2b.sda`

The `*.sda` file is a ZIP archive, extract it, it should contain the following:

- `META-INF/`
- `sap.com~crm~isa~web~b2b.war`
- `src.zip`

Copy the contents of the extracted `*.sda` into:

- `webshop-b2b/etc/src/sap.com/b2b`

Unpack `sap.com~crm~isa~web~b2b.war` into `webshop-b2b/war`

Copy the following files from `webshop-b2b/war` to the equivalent location under `webshop-b2b/deploy/war`

- `WEB-INF/xcm/customer/configuration/config-data.xml`
- `WEB-INF/xcm/customer/configuration/scenario-config.xml`
- `WEB-INF/xcm/sap/system/bootstrap-config.xml`

Move the contents of `webshop-b2b/war/WEB-INF/classes` to `webshop-b2b/src-properties`

Remove the now empty directory `webshop-b2b/WEB-INF/classes`

Copy the file `webshop-b2b/war/WEB-INF/web.xml` as `webshop-b2b/jetty-6.1.25/config/sap-b2b-web.xml`

Copy the contents of `webshop-b2b/etc/src/sap.com/b2b/META-INF` to `webshop-b2b/deploy/META-INF`

Retrieve Dependencies

The dependencies were determined:

- Initially suggested by the contents of `application-j2ee-engine.xml` (see: `webshop-b2b/etc/src/sap.com/b2b/META-INF`)
- Then by locating all the unresolved classes

To retrieve the dependencies:

- Open the `webshop-b2b` project
- Go to: `etc/dependencies`
- If the hosting CRM platform is Windows based:
  - Map the `SAPMNT` share on the CRM host to a drive letter
  - Set the following parameters either by editing the `build-getdeps.xml` script or creating the `local-build-degdeps.properties` file, see the build file for documentation:
    - `sap.sid`
    - `sap.j2ee_instance`
    - `source.base.dir`
  - Execute the Ant script: `build-getdeps.xml`
  - A structure will be built under `etc/dependencies/local`
- If the hosting CRM platform is something else:
  - Look at the file lists in: `etc/dependencies/sap`
  - Look at what the Ant file does
  - Basically:
    - Process the list files, striping everything after the first `*` encountered on the line

      ```
      sed 's%\*.*$%%g' <filename.lst>
      ```

    - Use those files as input lists to `tar` on the CRM host
    - Map the paths in the following manner (this will result in the path under `webshop-b2b/lib`):

List File	Source Path Prefix to Remove	Destination Path Prefix to Add
`application.lst`	`apps`	`lib/references/application`
`extra.lst`	*no change*	`lib/extras`
`interface.lst`	`bin/interfaces`	`lib/references/interface`
`library.lst`	`bin/ext`	`lib/references/library`
`services.lst`	`bin/services`	`lib/references/service`
`wars.lst`	*remove all*	`wars`

Install the dependencies:

- Copy the `lib/references` directory retrieved above over `webshop-b2b/lib/references`
- Copy the `lib/extras` directory retrieved above over `webshop-b2b/lib/extras`
- Copy the `wars/htmlb.war` file retrieved above into `webshop-b2b/webapps/com.sapportals.htmlb`
- Most of the skeleton `lib` directories contain marker files (*`filename`*`.jar.txt`)
  - a `*.jar` file is expected to exist for each `*.jar.txt` file (although there may be some exceptions due to version differences)
  - these marker files can be removed

- Additionally an example JAR list is can be found as **Appendix 1** at the end of this document, which contains the state of the `lib` directory at the completion of the setup (so will include JARs that are yet to be placed at this point in the instructions)

Duplicate dependencies into other projects

- Copy the following files from `webshop-b2b/lib` to `sap-patches/lib`
  - `extra/server/bin/system/logging.jar`
  - `references/library/com.sap.km.trex.client/trex.jc_api.jar`
  - `references/library/com.sap.security.api.sda/com.sap.security.api.jar`

SAP Patches

Update: `TicketVerifier`

Class: `com.sap.security.api.ticket.TicketVerifier`

Obtain the source from `lib\references\library\com.sap.security.api.sda\com.sap.security.api.jar` either:

- The source will be included in the JAR (in was in our version) *or*
- Decompile the class

Place the source under: `sap-patches/src`

Remove all references to `iaik.x509.X509Certificate` all the methods that use it are deprecated and do not appear to be used, so remove the methods.

Update: `TNSClient`

Class: `com.sapportals.trex.tns.TNSClient`

Obtain the source from `lib\references\library\com.sap.km.trex.client\trex.jc_api.jar` either:

- The source might be included in the JAR (it was not in our version) or
- Decompile the class

Place the source under: `sap-patches/src`

Edit the following method

```
  private synchronized Transferable sendRequest(Transferable
request) throws TrexException
```

Ensure that the String variable used to store the master name server (in our version it is named `masterNS`, decompilation results may vary) is never `null`.

The value is populated with the result of the `TrexConfigManager.getMasterNameServer()` call.

Something along the lines of the following is all that needs to be added:

```
if (masterNS == null) {

  masterNS = "";

}
```

Build

Build the `sap-patches` project using the Ant `build.xml` script.

Copy the resulting JAR, renaming:

- From: `sap-patches/build/local-sap-patches-*.jar`
- To: `webshop-b2b/lib/local/custom/local-sap-patches.jar`

Note: at this point you should be able to successfully build all the projects, except `webshop-b2b`, using the Ant `build.xml` scripts, however the other build artifacts are already present in `webshop-b2b`.

Retrieve Web Shop Configuration

Retrieve the XCM configuration from the deployed (and functioning) *vanilla* shop (an unmodified SAP B2B shop), should exist at a path like:

- `/usr/sap/${sap.sid}/JC${sap.j2ee_instance}/j2ee/cluster/server0/apps/sap.com/crm~b2b/servlet_jsp/b2b/root/WEB-INF/xcm`

Overwrite the following files in `webshop-b2b/war/WEB-INF/xcm` with those retrieved from the *vanilla* shop:

- `customer/configuration/config-data.xml`
- `customer/configuration/scenario-config.xml`

XCM Configuration

Edit the file: `webshop-b2b/war/WEB-INF/xcm/customer/configuration/config-data.xml`

Find all the sections that look like:

```
<params id="sap_logon_part2">

  <param name="user" value="BACKEND_USERNAME" savevalue=""/>

  <param name="passwd" value="#securestorage#xxxxxxx#encrypt"
savevalue=""/>

</params>
```

The second `param` element's `value` attribute contains the encrypted password used to connect to the backend, replace it like the following:

```
<params id="sap_logon_part2">

  <param name="user" value="BACKEND_USERNAME" savevalue=""/>

  <param name="passwd"
value="#securestorage#DevEnv:PlainText:BACKEND_PASSWORD#encrypt"
savevalue=""/>

</params>
```

Where `BACKEND_PASSWORD` is the password associated with the `BACKEND_USERNAME` user.

Edit the file: `webshop-b2b/war/WEB-INF/xcm/sap/system/bootstrap-config.xml`

- Change the value of the `target-datastore` parameter from `DB` to `FS`

Web Configuration

Edit the file `webshop-b2b/jetty-6.1.25/config/sap-b2b-web.xml`

Note that the Jetty instance is configured not to use `WEB-INF/web.xml` in the web application as a number of local development environment specific changes need to be made, that should not be deployed elsewhere.

Make the following changes:

- Add a context parameter:
  - Name: `customer.config.path.xcm.config.isa.sap.com`
  - Value: `/WEB-INF/xcm/customer`
- Add the following listeners:
  - `nz.co.sisu.jco.JCoLoggingContextListener`
  - `nz.co.sisu.dev.listener.UMEInitializer`
  - `nz.co.sisu.dev.listener.UserSessionDataAttributeListener`

You may also make the following changes that are unlikely to impact behaviour and may reduce the number of logged warnings:

- Remove the following registered filters and associated mappings
  - `TeaLeafCapture`
- Remove the following registered servlets and associated mappings
  - `monitoringservlet`
  - `smartstream`
  - `download`

TREX Configuraton

Edit the file `webshop-b2b/jetty-6.1.25/resources/trexjavaclient.properties`

Set the hostname and port of the TREX instance to use.

Log4j Configuration

The log4j properties file is:

- `webshop-b2b/jetty-6.1.25/resources/log4j.properties`

Jetty Configuration

In theory you should not need to change anything in the Jetty configuration file:

- `webshop-b2b/jetty-6.1.25/config/jetty.xml`

Local realm configuration exists in:

- `webshop-b2b/jetty-6.1.25/config/realm.properties`

SSL Certificate

A self signed certificate exists for the development environment in:

- `webshop-b2b/jetty-6.1.25/security/keystore`

A new certificate can be generated with the following command:

```
  keytool -keystore keystore -alias jetty -genkey -keyalg RSA -
validity 14610
```

Note:

- the `keytool` tool is part of the JDK, use the one from 1.4.2
- the passwords for the keystore, etc need to be updated in the `SslSocketConnector` configuration within `webshop-b2b/jetty-6.1.25/config/jetty.xml` (currently these are all `password`)

Webshop B2B Local Classes

Update: `IsaLocationSla`

Class: `com.sap.isa.core.logging.sla.IsaLocationSla`

Obtain the source from `webshop-b2b\etc\src\sap.com\b2b\src.zip`

Place the source under: `webshop-b2b/src-local`

This is a logging class, we want to make it log to log4j so that everything ends up in a single easily managed location.

Lots of changes here, so these notes are started and are by no means complete:

- Drop the existing fields:

```
    private Location location
    private static Map mSessions
    private statid Map mLocations
```

- Add a new field to hold a log4j logger

```
    private org.apache.log4j.Logger location
```

- Modify the constructor to initialise the logger

```
    location = Logger.getLogger(name);
```

- Fix all the errors that now exist, it should be a fairly straightforward translation to the log4j logger
- It's probably worth fixing the implementation of the following method so that you get full information even if there are no available message resources

```
    private String getResourceBundleString(Object key, Object
args[])
```

Update: `SecurityUtil`

Class: `com.sap.isa.core.security.SecurityUtil`

Obtain the source from `webshop-b2b\etc\src\sap.com\b2b\src.zip`

Place the source under: `webshop-b2b/src-local`

The following works around our local shop not being able to decrypt stored passwords.

Add the following constants:

```
    private static final String DEV_PASSWORD_PLAINTEXT =
"DevEnv:PlainText:";

    private static final String DEV_PASSWORD_PROPERTY =
"DevEnv:Property:";

    private static final String DEV_PASSWORD_FAIL = "DevEnv:Fail:";
```

Edit the following method:

```
  public static Serializable decryptFromBase64(String encryptedStr)
```

After the argument checks, replace the remaining method body with the following:

```
            if (encryptedStr.startsWith(DEV_PASSWORD_PLAINTEXT)) {

                log.warn("Using plaintext password value");

                String result =
encryptedStr.substring(DEV_PASSWORD_PLAINTEXT.length());

            if (result.length() == 0) {

                    throw new ISASecurityException(encryptedStr + "
is invalid");

                }

                return result;

            }

            else if (encryptedStr.startsWith(DEV_PASSWORD_PROPERTY))
{

                log.warn("Using password from system property");



                String name =
encryptedStr.substring(DEV_PASSWORD_PROPERTY.length());

                if (name.length() == 0) {

                    throw new ISASecurityException("No password
property name set");

                }
```

```
                String result = System.getProperty(name);

                if (result.length() == 0) {

                    throw new ISASecurityException("No value set for
password system property '" + name + "'");

                }



                return result;

            }

            else if (encryptedStr.startsWith(DEV_PASSWORD_FAIL)) {

                log.warn("Deliberately failing");



                String result =
encryptedStr.substring(DEV_PASSWORD_FAIL.length());

                if (result.length() == 0) {

                    result = "Deliberate failure retrieving
password";

                }

                throw new ISASecurityException(result);

            }

            else {

                throw new ISASecurityException("Cannot decrypt
password");

            }
```

Update: **`RequestProcessor`**

Class: `com.sap.isa.core.RequestProcessor`

Obtain the source from `webshop-b2b\etc\src\sap.com\b2b\src.zip`

Place the source under: `webshop-b2b/src-local`

Edit the method:

```
  public void process(HttpServletRequest  request,
HttpServletResponse response)
```

In the `catch (Throwable)` block comment out the entire `if-else` block, **except** the `log.error` call

This just means that errors will always get logged to the log file, rather than to standard error.

Note: `StubSessionSecurityAuth`

If the `com.sap.isa.user.backend.ume.ISessionSecurityAuth` interface does not exist (it was introduced in a very recent version) then:

- Remove the class: `nz.co.sisu.dev.security.StubSessionSecurityAuth` (in `webshop-b2b/src-local`)
- Remove the class: `nz.co.sisu.dev.listener.UserSessionDataAttributeListener` (in `webshop-b2b/src-local`)
- Remove the listener registration for the above class from `webshop-b2b/jetty-6.1.25/config/sap-b2b-web.xml`

Webshop B2B Deployment Metadata

Update: `application.xml`

File: `webshop-b2b/deploy/META-INF/application.xml`

Replace the element text with the following Ant replacement parameters:

Element	Replacement Parameter
`display-name`	`@appl_name@`
`web-uri`	`@appl_weburi@`
`context-root`	`@appl_ctxroot@`

Update: `MANIFEST.MF`

File: `webshop-b2b/deploy/META-INF/MANIFEST.MF`

Replace the property values with the following Ant replacement parameters:

Property Name	Replacement Parameter
`Implementation-Title`	`@appl_name@`
`Implementation-Version`	`@key_counter@`

Update: `SAP_MANIFEST.MF`

File: `webshop-b2b/deploy/META-INF/SAP_MANIFEST.MF`

Remove all the `dtr-*` properties

Replace the property values with the following Ant replacement parameters:

Property Name	Replacement Parameter
keyname	@appl_name@
keylocation	@key_location@
keycounter	@key_counter@

Remove the following attributes from the `componentelement` property XML value:

- servertype
- sourceserver
- changenumber

Replace the attribte values with the following Ant replacement parameters in the `componentelement` property XML value::

Property Name	Replacement Parameter
name	@appl_name@
location	@key_location@
counter	@key_counter@
updateversion	@update_version@

Project Configuration

- Import user libraries
  - Menu: Window / Preferences
  - Tree: Java / Build Path / User Libraries
  - Button: Import
  - Browse: `C:\Work\SISU\workspaces\workspace-b2b\webshop-b2b\etc\eclipse\b2b-libs.userlibraries`
  - The following libraries should be listed and selected:
    - B2B_Dev
    - B2B_Extra
    - B2B_Jetty
    - B2B_Ref_Application
    - B2B_Ref_Interface
    - B2B_Ref_Library
    - B2B_Ref_Service
    - B2B_WebApp
  - Button: OK
  - Note
    - you may have to cleanup the library definitions to include new files and exclude file that did not exist

        - once the workspace has rebuilt there should be no errors

- Configure workspace variables
    - Menu: Window / Preferences
    - Tree: Run/Debug / String Substitutions
    - Button: New...
    - Create a new variable using the following values:
        - Variable: `b2b_working_base`
        - Value: `C:\Work\SISU\tmp\b2b`
        - Description: `B2B base working directory`

- Create temporary directory (the creation of this directory will be done in a later stage by a build script)
    - `C:\Work\SISU\tmp\b2b\jetty\temp`

Project Initialisation

- Create `webshop-b2b` project temporary directories
    - Menu: Run / External Tools / External Tools Configurations...
    - Tree: Ant Build / ProjectInit-Webshop
    - Button: Run

Native Library Version Workaround

Check if the following file exists, if it does not this section should be skipped:

- `%WinDir%\system32\librfc32.dll`

At the moment there is an issue with native library versioning, to ensure that the correct versions are loaded:

- Copy: `C:\Work\SISU\workspaces\workspace-b2b\webshop-b2b\jetty-6.1.25\native\librfc32.dll`
- Into: `C:\Tools\jdk\1.4.2_19\bin` (i.e. the directory the JVM executable exists in)

There are better ways to do this.

Status of the Local Development Environment

At this point you should have a **mostly working** local webshop installation.

The following issues are known:

- Many JSPs in the webshop contain errors that are silently ignored by the NetWeaver JSP engine (generally with less than desirable results) or the pages are just not used. Running `JspC` over the webapp is informative, if not reassuring. The bulk of these fall into the following categories:

    - Malformed taglib usage (misspelt tag name, unbalanced tag elements)
    - Unresolved classes (the classes just don't seem to exist in our environment)
- A number of pages generate result in Jasper generating `_jspService` method bodies which exceed the maximum size (~64KB). The approach we have taken to make things work is to either split large pages up or alter the inclusion mechanism, `<jsp:include>` is the workaround. The following pages are known to cause issues in the current release (`2010-08-20`):
    - `webshop-b2b/war/b2b/order.jsp`
    - `webshop-b2b/war/catalog/ProductDetails.inc.jsp`
    - `webshop-b2b/war/catalog/ScComponents.inc.jsp`
- Several exceptions may be seen during startup, these are not fatal, can be fixed with stub implementations (not included) and relate to:
    - The lack of an available JNDI implementation at runtime
    - The lack of an available JMX implementation at runtime
    - The lack of an available JMS implementation at runtime

Running the B2B Shop

- Start the development instance
    - Menu: Run / Debug Configurations...
    - Tree: Java Application / JettyTemplate
    - Button: Debug
    - The application should start, messages will be logged to the Eclipse console
    - Start a web browser and go to: https://localhost:8081/sap-b2b
    - Note
        - the webapp is ready when the log output stops and you see a line like

```
2010-09-23 07:45:41,116 [main] INFO
org.mortbay.log [] Started
SelectChannelConnector@localhost:8080

2010-09-23 07:45:41,163 [main] INFO
org.mortbay.log [] Started
SslSocketConnector@0.0.0.0:8081
```

        - currently during startup several ignorable exceptions are logged, these will be cleaned up in the near future
        - the browser may complain about the self signed server certificate

- The web shop is contained in the `webshop-b2b` project.

Building the Deployment Archive

To build the EAR file to deploy the web shop to a NetWeaver AS instance, run the Ant build script:

- `webshop-b2b/build-deploy.xml`

Project Overview

A quick overview of the projects follows.

Supporting Projects

- `jasper-sap-compat`
  - Purpose: Attempts to make the JSP compiler more compatible with the NetWeaver AS JSP implementation
    - Additional default page imports
      - See: `org.apache.jasper.Constants`
    - Modified `IterationTag` generation, to permit the dubious use of the `continue` statement
      - See: `org.apache.jasper.compiler.Generator`
  - Produces:
    - `webshop-b2b/jetty-6.1.25/lib/jsp-2.0/local-jasper-compiler-5.5.15.jar`
    - `webshop-b2b/jetty-6.1.25/lib/jsp-2.0/local-jasper-runtime-5.5.15.jar`
  - Replaces:
    - `webshop-b2b/jetty-6.1.25/lib/jsp-2.0/jasper-compiler-5.5.15.jar`
    - `webshop-b2b/jetty-6.1.25/lib/jsp-2.0/jasper-runtime-5.5.15.jar`

- `jco-logging`
  - Purpose: Unified JCo logging output
  - Produces:
    - `webshop-b2b/lib/local/custom/local-jco-logging.jar`

- `jetty-utils`
  - Purpose: Class path construction and class loader implementation.
  - Produces:
    - `webshop-b2b/jetty-6.1.25/lib/local-jetty-utils.jar`

- `sap-patches`
  - Purpose: Patches to standard externally referenced SAP libraries.
  - Produces:
    - `webshop-b2b/lib/local/custom/local-sap-patches.jar`

Webshop Project

- `webshop-b2b`
  - `src` - source which will form part of the production web shop deployment
  - `src-local` - source which supports the local dev environment
  - `src-properties` - standard web shop properties files
  - `deploy` - files used to build the deployment archive (EAR)
  - `etc/dependencies` - used to retrieve dependent libraries

  - `etc/eclipse` - Eclipse configuration and support files
  - `etc/src` - source for referenced libraries
  - `jetty-6.1.25` - the local web container
  - `lib/extra` - web shop dependences either:
    - assumed to exist in a NetWeaver AS environment *or*
    - required by libraries in `lib/references`
  - `lib/local` - libraries to support the local dev environment (not used during deployment build)
  - `lib/references` - web shop dependences referenced by `application-j2ee-engine.xml`
  - `war` - the root of the web application
  - `webapps` - other web applications referenced or linked to by the web shop at runtime

- Jetty
  - Startup system properties
    - Set
      - `jco.trace_level` - sets logging level in the JCo library
      - `working.dir` - working directory used by Jetty (set to Eclipse variable `${b2b_working_base}`)
    - Available
      - `jetty.host` (default `localhost`) - Jetty server name
      - `jetty.port` (default `8080`) - Jetty HTTP port
      - `jetty.ssl.port` (default `8081`) - Jetty HTTPS port
  - Class loader structure
    - Bootstrap (JVM)
    - System (JVM)
    - Jetty, classpath is dynamically built at start, but is mainly `webshop-b2b/jetty-6.1.25/lib/**/*.jar` and `webshop-b2b/jetty-6.1.25/resources`
    - Services, custom classloader (`nz.co.sisu.tools.jetty.util.ServicesClassLoader`) essentially loads almost everything under `webshop-b2b/lib` (see `jetty.xml` for configuration). A rough approximation of NetWeaver AS classloading.
    - WebApp, standard web application classloader supplied by Jetty

Appendix 1 – sample jar list

lib/extra/server/bin/core_lib/opensqlcore.jar
lib/extra/server/bin/ext/com.sap.mobile.clientinfo/clientinfo.jar
lib/extra/server/bin/ext/com.sap.mobile.clientinfo/clientinfo_data.jar
lib/extra/server/bin/ext/opensql/opensql.jar
lib/extra/server/bin/ext/opensql/sqljapi.jar
lib/extra/server/bin/services/servlet_jsp/servlet_jsp.jar
lib/extra/server/bin/system/exception.jar
lib/extra/server/bin/system/frame.jar
lib/extra/server/bin/system/logging.jar
lib/local/custom/local-jco-logging.jar
lib/local/custom/local-sap-patches.jar
lib/local/log4j/log4j-1.2.16.jar
lib/local/sapjco/sapjco.jar
lib/references/application/sap.com/com.sap.jdo/connector/connectors/sapjdojca.rar/jdo.jar
lib/references/application/sap.com/com.sap.jdo/connector/connectors/sapjdojca.rar/sapjdo.jar
lib/references/application/sap.com/com.sap.jdo/connector/connectors/sapjdojca.rar/sapjdojca.jar
lib/references/interface/tc~sec~destinations~interface/tc_sec_destinations_interface.jar
lib/references/library/activation/activation.jar
lib/references/library/com.sap.km.trex.client/trex.jc_api.jar
lib/references/library/com.sap.mona.api/jmonapi.jar
lib/references/library/com.sap.mw.jco/jrfc.jar
lib/references/library/com.sap.security.api.sda/com.sap.security.api.jar
lib/references/library/com.sap.security.api.sda/com.sap.security.api.perm.jar
lib/references/library/com.sap.security.core.sda/com.sap.security.core.jar
lib/references/library/com.sap.security.core.sda/com.sap.security.core.tpd.jar
lib/references/library/com.sap.tc.Logging/loggingStandard.jar
lib/references/library/com.sap.util.monitor.grmg/grmg.jar
lib/references/library/com.sap.util.monitor.jarm/jARM.jar
lib/references/library/com.sap.util.monitor.jarm/jARMSat.jar
lib/references/library/com.sapportals.htmlb/htmlb.jar
lib/references/library/crm~tc~lib~corelib/sap.com~crm~tc~corelib~assembly.jar
lib/references/library/crm~tealeaf/TLFilter.jar
lib/references/library/ejb20/ejb20.jar
lib/references/library/j2eeca/connector.jar
lib/references/library/jms/jms.jar
lib/references/library/mail/javamail_library.jar
lib/references/library/mail/mail.jar
lib/references/library/sapxmltoolkit/sapxmltoolkit.jar
lib/references/library/security.class/tc_sec_compat.jar
lib/references/library/security.class/tc_sec_csi.jar
lib/references/library/security.class/tc_sec_https.jar
lib/references/library/security.class/tc_sec_jaas.jar
lib/references/library/security.class/tc_sec_jaas_test.jar
lib/references/library/security.class/tc_sec_saml_jaas.jar
lib/references/library/security.class/tc_sec_saml_service_api.jar
lib/references/library/security.class/tc_sec_saml_toolkit_api.jar
lib/references/library/security.class/tc_sec_saml_toolkit_core.jar
lib/references/library/security.class/tc_sec_saml_util.jar
lib/references/library/security.class/tc_sec_saml_xmlbind.jar
lib/references/library/security.class/tc_sec_ssf.jar
lib/references/library/security.class/tc_sec_userstore_lib.jar
lib/references/library/servlet/servlet.jar
lib/references/library/tc~jmx/com_sap_pj_jmx.jar
lib/references/service/adminadapter/adminadapter.jar

lib/references/service/applocking/applocking.jar
lib/references/service/apptracing/apptracing.jar
lib/references/service/jms_provider/sapjms.jar
lib/references/service/tc~sec~destinations~service/tc_sec_destinations_service.jar
lib/references/service/tc~sec~securestorage~service/tc_sec_securestorage_service.jar