

SHARE Session #8370: z/OS Tomcat Exploring

Introduction

Welcome! We hope that this lab session will be not only instructional (and challenging), but also fun! Please don't hesitate to ask questions during the lab, and please fill out an evaluation. We appreciate your comments as they will help us to improve future offerings of this lab.

Objectives

1. Configure and customize your own Tomcat instance on z/OS, using the SDK 5.0 “shared classes” feature.
2. Manage Tomcat using MVS console commands
3. “Hot deploy” Java web apps from a workstation IDE (Eclipse) to your running Tomcat instance.
4. Setup a Tomcat JDBC connection pool for a DB2 or Apache Derby database.
5. Install an open source, web-based, collaboration application – JspWiki
6. (Optional) Customize Tomcat to use SAF (RACF) security

Prerequisites

Lab participants will need to be comfortable with using z/OS, including ISPF, SDSF, JCL, and a very basic familiarity with z/OS Unix System Services. Participants should also be familiar with running Java batch jobs using the JZOS batch launcher (see Lab #8369). If you don't have these skills, *please team up with someone* who does; see an instructor and we'll help find you a lab partner.

It will also be helpful if you have a basic understanding of Java and Java web applications, but this is not strictly required.

Some Details

- This lab will use the IBM SHARE LPAR, with is running z/OS 1.7. If you have IBM contacts, please thank them for making this invaluable resource available to SHARE.
- We will be using IBM JAVA SDK 5.0 (31-bit), although everything presented will also run under SDK 1.4.2.
- You will be using JZOS 1.2.4 (the IBM alphaWorks version). IBM has announced that the JZOS batch launcher will be integrated into an upcoming level of the z/OS Java SDK.
- We will be using an unmodified Apache Tomcat 5.5 distribution in this lab.
- The lab instructors will help you to get connected to the IBM SHARE z/OS LPAR using the 3270 emulator on your workstation.
- For more information on using Apache Tomcat on z/OS, see <http://dovetail.com>. Materials used in this lab are available at this site for download. A forum is available for your questions on running Tomcat an other Java applications on z/OS.

HFS File Systems for this Lab

Even though you will be running Java as a batch job, the Java JVM uses HFS (or zFS) filesystems for all of its classes, jars, properties files, etc. Below is an overview of the directories that you will be using in this lab.

One of the popular ways to run Tomcat is to have a single, shared copy of the Tomcat distribution and run multiple instances by making use of the CATALINA_BASE feature. This is the approach that we will take for this lab. **Note:** we have downloaded and unzipped the Tomcat 5.5 distribution, but it is not been modified or customized in any way... that will be your job :-)

Take a minute to review the table below.

/sharelab

 /tomcatlab

/apache-tomcat-5.5.15	directory where Tomcat 5.5 zip file was unloaded (aka CATALINA_HOME)
/bin	contains Tomcat “bootstrap” jars
/common	Tomcat system jars that can also be used by webapps
/server	Tomcat system jars that are not shared with webapps
/shared	place where you can put jars available to all webapps (empty)
(others...)	other directories that are part of the Tomcat distribution
/jzos	JZOS jars and dlls (aka JZOS_HOME)
/tomcat	for convenience, a symbolic link (alias) to ./apache-tomcat-5.5.15 (aka CATALINA_HOME)

/sharelab

 /share*nn*

/tomcat	home directory(s) for each lab userid. (in Unix, your home directory can be referred to as “~” (tilde)) you will create this directory, in which to setup your Tomcat instance (aka CATALINA_BASE)
/conf	directory containing “server.xml”, which configures your Tomcat instance
/logs	
/temp	
/work	
/webapps	directory containing web applications (Java servlet / JSP) applications.

Editing HFS Files on z/OS (both EBCDIC and ASCII)

The default character encoding for files on z/OS is EBCDIC.

Some files, however, are **not** in EBCDIC:

- **XML files**, by default, are in UTF-8, a special form of ASCII. XML files can be in other encodings if specified in header prologue of the file. The XML configuration files that are distributed with Tomcat use default (ASCII) encoding.
- **Java properties** files are in ASCII (ISO-8859-1) as mandated by the Java spec.
- Other files that might be created by a Java program running under an ASCII default file encoding.

Editing using Unix “vi” editor

If you are comfortable with using the Unix “vi” editor, you can use it by opening a regular telnet (non-3270) shell. You can start a non-3270 telnet shell on Windows by using the “Putty” icons which should be on your lab desktop. The one for connecting to the SHARE LPAR using “SSH” is the best option, since it is more secure than the regular Telnet protocol.

(for more information on z/OS SSH, see Session #2912 “Securing Your z/OS UNIX Network Access with OpenSSH”)

To edit ASCII files: you can use the “**avi**” command
(in /sharelab/tomcatlab/atools, which should already be in your default search PATH).

Editing using ISPF

If you are more comfortable using the ISPF editor, you can open a Unix shell using the “TSO OMVS” command. This opens a special shell that runs under a 3270 session, under which you can use the “**oedit**” and “**obrowse**” commands to edit HFS files.

To edit ASCII files: you can use the “**aoedit**” and “**aobrowse**” command
(in /sharelab/tomcatlab/atools, which should already be in your default search PATH).

The atools “a-<command>s” work just like their normal counterpart, but automatically convert the file to/from EBCDIC. There are some other options for editing HFS files that we won't cover in this lab. more information see:

<http://dovetail.com/docs/jzos/editascii.html>

Customize and Execute Apache Tomcat

In this section, you will be setting up your own Tomcat instance directory, configuring the ports that your instance will be using, customizing the JZOS batch launcher JCL, and starting Tomcat.

The lab instructors will demonstrate and assist you with logging into the SHARE z/OS LPAR using a 3270 emulator. Some students may also prefer to use the Unix System Services shell from a Telnet session rather than using TSO “OMVS”; the PUTTY telnet client installed on your workstation may be used for this purpose.

Your TSO ID for this lab will be **SHAREnn**, where nn is your assigned unique number. This lab does **not** use the ids SHAREAnn or SHAREBnn. In the exercises below, replace <HLQ> with your assigned TSO ID.

Setup and configure HFS files for your Tomcat instance

In the first set of steps following, you will setup your own Tomcat HFS filesystem under your home directory:

1. Get into a Unix shell (either “TSO OMVS” or a PUTTY non-3270 shell).
Create a directory on your home filesystem to hold the CATALINA_BASE components:

```
mkdir ~/tomcat
```

2. Change to this new directory and make the following subdirectories:

```
cd ~/tomcat
mkdir logs
mkdir temp
mkdir work
```

3. Copy the required components from the primary tomcat installation (don't forget the trailing dot):

```
cp -R /sharelab/tomcatlab/tomcat/conf .
cp -R /sharelab/tomcatlab/tomcat/webapps .
```

4. Edit your copy of server.xml. This file is in directory: ~/tomcat/conf

This is an ASCII file, so use “aoedit” or “avi” as described in the previous section.

- Change the Server control port. Locate the the element `<Server port="8005"` and change to `<Server port="85nn"` where nn is your share lab id.
- Change the HTTP port. Locate the element `<Connector port="8080"` and change to `<Connector port="80nn"` where nn is your share lab id. This is the port that you will point your browser to when you test your Tomcat instance.
- Disable the AJP connector. This connector is used to allow Apache web server interaction with Tomcat, which we won't need for this lab. Locate the element `<Connector port="8009"` and comment out the entire element with an XML style comment:

```
<!-- Commented out for SHARE lab
<Connector port="8009"
           enableLookups="false" redirectPort="8443"
           protocol="AJP/1.3" />
-->
```

- Disable “auto deployment” for the default host. We will be using Eclipse and the Tomcat JMX management facilities to modify and redeploy our web applications, so this feature will not be needed, and if not disabled will result unneeded IO operations in z/OS.

Locate the element: `<Host name="localhost"` and change the attribute value for `autoDeploy` from `"true"` to `"false"`. Save your changes and exit.

Review the previous section, “HFS Filesystems Used in this Lab” to make sure you understand how things are laid out. As always, feel free to ask questions!

Customize JCL to run Tomcat

Your TSO ID for this lab will be **SHAREnn**, where nn is your assigned unique number. This lab does **not** use the ids SHAREAnn or SHAREBnn.

In the exercises below, replace **<HLQ>** with your assigned TSO ID.

1. From ISPF option 6, issue the following command to make your own copy of the sample JCL:

```
copy 'kirk.tomcat.jcl' tomcat.jcl nonum
```

Note: this JCL contains mixed upper/lower case and contains USS shell scripts which cannot have line numbers!

2. Edit the PDS member **<HLQ>.TOMCAT.JCL(TC55)**
3. Tailor the jobcard for this JCL, changing **<HLQ>** to your **SHAREnn** userid:

```
//<HLQ>T JOB ( ), 'TOMCAT',MSGCLASS=H,NOTIFY=&SYSUID
```

4. Tailor the rest of the JCL per the instructions included in the member

Note: these Unix path/directory names are case sensitive!

- JZOS_HOME should be set to: /sharelab/tomcatlab/jzos
- JAVA_HOME should be set to: /usr/lpp/java15/J5.0
- CATALINA_HOME should be set to: /sharelab/tomcatlab/tomcat
- CATALINA_BASE should be set to: /sharelab/sharenn/tomcat

5. Submit the job and switch to SDSF to view the job output.

Tips:

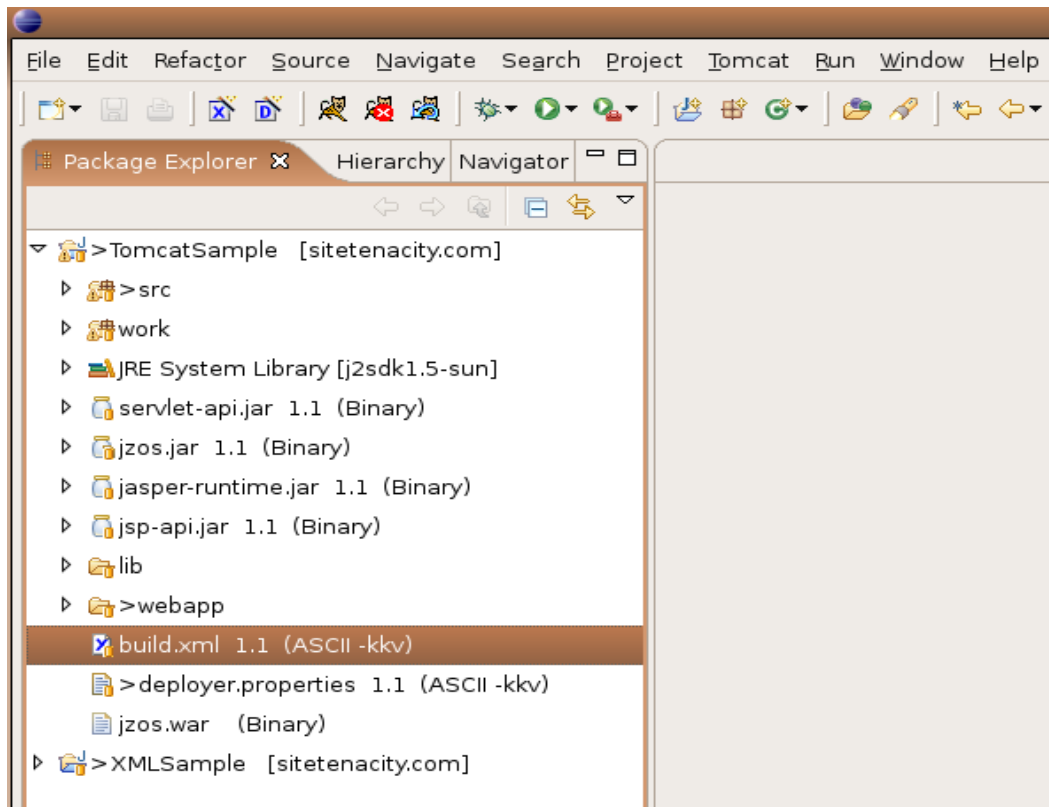
- SDSF is option “S” from the SHARE ISPF primary options menu.
- Consider splitting your ISPF screen so that you can tailor and submit from one screen and view output from the other.
- So that SDSF filters only your jobs, issue the command (note trailing asterisk!):
PREFIX SHARE**nn***

6. Test Tomcat from a browser by visiting: <http://mvs1.centers.ihost.com:80nn> where “nn” is your share lab id. You should see the Tomcat “welcome” page.

Deploy a Web Application from Eclipse to z/OS Tomcat

This lab uses Eclipse and a set of Apache Ant tasks to communicate with a running tomcat server. You will use Eclipse to build a sample web application and deploy it to a running Tomcat instance.

1. Start Eclipse on your workstation using the “Eclipse jzosLabs” icon on your desktop.
2. Locate the TomcatSample project in the Package Explorer and expand it.



3. Edit the file **deployer.properties** and customize for your specific Tomcat installation. Modify the following properties:

Property	Value
url	http://mvs1.centers.ihost.com:80 nn /manager
username	share – note that this is not your share nn id
password	<pw> - this will be supplied by the instructors

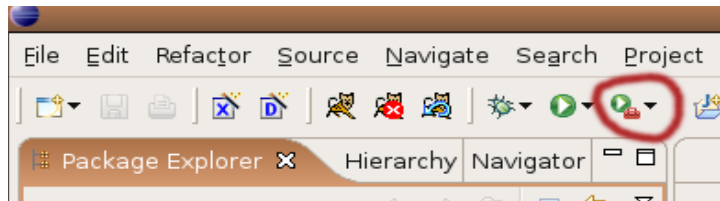
- The 'url' property names the URL for the Tomcat Manager application, which is used by the deployer task to install and update Tomcat. Use port “80**nn**”, the where “nn” is your student number.
 - Supply a userid and password that are authorized to manage Tomcat. Unless configured with different security parameters (see the optional SAF lab) ,Tomcat will authenticate and authorize user credentials by checking the file `CATALINA_BASE/conf/tomcat-users.xml`. By default, this file does not contain a manager role. However, your version has been customized to add this role. You can access it with a userid of `share` and a password given to you by the instructors. Update your `deployer.properties` file with these values.
 - Save your changes: `File+Save` or `(Ctrl-S)`.
4. This Eclipse project uses Apache's customized Ant tasks to automatically deploy a web application. The file `build.xml` contains the complete Ant script, which will use the customized properties you set in the previous step to deploy your web application.

- Left click on the Ant script "build.xml" to select
- Right click + Run As+Ant Build...
- In the dialog, locate and select the “JRE” tab
- Choose "Run in the same JRE as the workspace"
- Press "Apply"
- Press "Run"

5. After running the script, your Eclipse console should look something like this:

```
Buildfile: /home/goetze/workspace/TomcatSample/build.xml
build:
    [jar] Building jar:
/home/goetze/workspace/TomcatSample/jzos.war
deploy:
    [deploy] OK - Undeployed application at context path /jzos
    [deploy] OK - Deployed application at context path /jzos
buildAndDeploy:
BUILD SUCCESSFUL
Total time: 4 seconds
```

6. Check Tomcat to see if your web application was successfully deployed by visiting the manager web application with your browser:
`http://mvs1.centers.ihost.com:80nn/manager/html`. Supply the userid and password from the previous step when prompted. Once authenticated, you should see a web application at path `/jzos`.
7. Visit your web application, either by clicking on “jzos” in the manager application or by visiting: `http://mvs1.centers.ihost.com:80nn/jzos` and run the HelloWorld servlet.
8. From Eclipse, locate the HelloWorld servlet java source code. You can find it by expanding the servlet package under the project's `src` folder.
9. Edit the file `HelloWorldServlet.java` and change the text:
`<html><h1>Hello World!</h1></html>`
to
`<html><h1>Hello SHARE!</h1></html>`
10. Save your changes (this will cause the java file to be compiled)
11. Redeploy your web app.
A one-click shortcut is to click the Eclipse external tools “run” icon:



12. Visit your web application and re-run the servlet. You should see your changes.
13. Do a clean shutdown of your Tomcat job by using the MVS “STOP” (P) command:
From SDSF, type “DA” to get to the active jobs list. Put a “Y” prefix command in front of your Tomcat Job (SHAREnnT). This will send a MVS STOP command to the JZOS batch launcher which will cleanly quiesce and shutdown Tomcat.

Configure Tomcat for JDBC connections to DB2

In this lab section, you will define a JDBC connection pool to Tomcat. The pool will be available to your web application using JNDI (Java Naming and Directory Interface).

1. In Eclipse, in the TomcatSample project, find and open the file:

`webapp/META-INF/context.xml`

- For the “username” and “password” attributes, supply your SHAREnn userid and password.
- Java SDK 5.0 currently only supports “type-4” (pure java; tcp/ip) JDBC drivers. The IBM JDBC Universal Driver name: `com.ibm.db2.jcc.DB2Driver` is already configured. We have already copied the jars for this driver to `$CATALINA_HOME/common/lib`. These are:

```
db2jcc.jar
db2jcc_javax.jar
db2jcc_license_c.jar      ( license jar for Apache Derby database)
db2jcc_license_cisuz.jar  ( license jar for z/OS DB2)
```

- Unless there is late-breaking news, the version of DB2 on the SHARE LPAR does not have the PTFs to support the DB2 Universal Driver. However, the IBM Universal Driver is also supported by the pure Java Apache Derby database. We have setup an instance of this running on the SHARE LPAR.

To use our Derby database server, use port 8527 and database “DBSG”:

```
url="jdbc:db2://127.0.0.1:8527/DBSG"
```

- Save your changes (File/Save or Cntrl-S)
2. Restart Tomcat if it is not running.
 3. Redeploy the webapp by running the build.xml ant script, as before.
 4. Point your browser to the jzos web application:

```
http://mvs1.centers.ihost.com:80nn/jzos
```

Press the “Execute” link for “DB2 JDBC Test Servlet”. You should see:

```
Testing JDBC...
**** JDBC Statement Created
**** JDBC Result Set Created
**** Number of SYSTABLES = 17
**** JDBC Statement Closed
**** JDBC Disconnect
Test Successful!
```

5. To see the source code for this server, open the following file in Eclipse:
`src/com.dovetail.jzos.servlet/JdbcServlet`

Installation of JSPWiki on z/OS

This lab exercise was originally developed by Rob Schramm.
If you have the opportunity, please thank him for his contributions and support of SHARE and the z/OS Java community!

What is JSPWiki?

JSPWiki is a free, open-source Java Servlet/JSP application that implements a “Wiki”. It runs very nicely under Tomcat on z/OS.

What is a Wiki?

From: <http://wiki.org/wiki.cgi?WhatIsWiki>

The simplest online database that could possibly work.

Wiki is a piece of server software that allows users to freely create and edit Web page content using any Web browser. Wiki supports hyperlinks and has a simple text syntax for creating new pages and crosslinks between internal pages on the fly.

Wiki is unusual among group communication mechanisms in that it allows the organization of contributions to be edited in addition to the content itself.

Why would a z/OS Systems Programmer need a Wiki?

If you have ever answered "yes" to any of these questions...

- Ever wanted all your documentation on the mainframe?
- Would you like all of your documentation to be part of your mainframe recovery?
- Are you tired of trying to keep notes and documentation in PDS members?
- ASCII or EBCDIC drawings in PDS members come up a bit short on the usability scale?
- Needed a way to organize Word documents, Excel spread sheets etc etc with a couple of notes about each one?
- Need a demonstration platform for showing off Java and open source on the mainframe?

Get JSPWiki

1. Download the latest stable release from:

<http://www.jspwiki.org/wiki/JSPWikiDownload>

The top/right corner of the page has a link to the [stable] release, which is currently labeled:

[“JSPWiki 2.2.33 package”](#)

2. Save the .zip to your workstation
 - unzip the file
 - find the JSPWiki.war file
 - rename JSPWiki.war to zoswiki.war

Note: the reason for renaming the JSPWiki.war to zoswiki.war is that we want our wiki to be **zoswiki**. Fixing the name here makes the Tomcat application deployment very easy.

Deploy JSPWiki to your Tomcat instance

The *Tomcat Manager* application is a web application which can install and manage applications running under Tomcat. This is the web application that the Tomcat Ant “deployer” tasks also use to remotely upload and install web applications to Tomcat. We will now use the web interface to the Tomcat Manager to upload and deploy the war file for JspWiki.

1. Point your browser to the following URL to bring up the Tomcat Manager UI:

`http://mvs1.centers.ihost.com:80nn/manager/html`

Note: You may be prompted for a Tomcat userid and password to access the Manager application. As before when you used the Tomcat Manager from the Eclipse deployer, use the *Tomcat userid* “share” and the password provided by the instructors.

(Tomcat can be configured to use SAF/RACF for its password database, but thats a later exercise!)

2. In the "**WAR file to deploy**" part of the Manager page, click **BROWSE** to locate the zoswiki.war file (the one that you renamed in the previous step)
3. Click **DEPLOY** (this may take a minute or so to upload.. be patient)
4. Locate the **zoswiki** application in the "**Applications**" section of the page and click **STOP** (look to the right)

Install the JSPWiki sample pages

JSPWiki stores all of its content (pages, attachments, page history) in a regular Unix (HFS) filesystem. In this section you will create this directory under your home directory and upload the default set of starter pages distributed with JSPWiki.

In this lab, you will create a “zoswiki” directory under your z/OS Unix home directory to store JSPWiki content. The full path name of this directory will be:

```
/sharelab/sharenn/zoswiki
```

where, as usual, “nn” is your assigned lab userid number.

1. From a Windows command window, first change to the directory containing that file “**JSPWiki-samplepages.zip**” that you unzipped from the JSPWiki distribution, and then login to z/OS using ftp:

```
cd (directory containing jspwiki-samplepages.zip)
ftp mvsl.centers.ihost.com
```

Login with your sharenn userid and password.

2. Upload the jspwiki-samplepages.zip file using the following ftp commands:

```
cd /sharelab/sharenn
mkdir zoswiki
cd zoswiki
bin
put JSPWiki-samplepages.zip
```

3. From a z/OS Unix shell (TSO “OMVS” or Putty/Telnet shell), unzip the sample pages:

```
cd ~/zoswiki
jar -xvf JSPWiki-samplepages.zip
rm JSPWiki-samplepages.zip
```

Customize JSPWiki properties

JSPWiki uses a Java properties file named “`jspwiki.properties`”, for configuration purpose. There are just a couple of things that must or should be changed in this file to enhance your wiki experience.

1. From a z/OS Unix shell (either “OMVS” or a PUTTY/Telnet session), edit the properties file.

The `jspwiki.properties` file is an ASCII file, so you must use “`aoedit`” or “`avi`” as described earlier in this lab to edit it.

```
cd ~/tomcat/webapps/zoswiki/WEB-INF
avi (or aoedit) jspwiki.properties
```

2. Here is a list of things to change:

For each line below, make sure you also “uncomment” the line by removing any leading `'#'` characters.

- `jspwiki.applicationName = zoswiki`
- `jspwiki.fileSystemProvider.pageDir = /sharelab/sharenn/zoswiki/`
- `jspwiki.basicAttachmentProvider.storageDir = /sharelab/sharenn/zoswiki/`
- `jspwiki.baseURL = http://mvs1.centers.ihost.com:80nn/zoswiki/`

Here are some additional changes that you might want to make at your installation, but aren't really required for purposes of this lab exercise:

- `jspwiki.referenceStyle=relative`
- `jspwiki.translatorReader.allowHTML = true`
Warning: only use this if the wiki is being "secured" ... do not use it in an open Wiki as it may be used to compromise your system!!
- `jspWiki.interWikiRef.wiki =`
`http://mvs1.centers.ihost.com:80nn/zoswiki/Wiki.jsp?page=%s`
Note: you have to add this (as a single unbroken line)

Start “zoswiki” using Tomcat Manager

1. Point your browser again to the Tomcat Manager application:

`http://mvs1.centers.ihost.com:80nn/manager/html`

(as before, login with userid “share” and the password you were given)

2. Locate the **zoswiki** application in the "**APPLICATION**" section of the page and click **START** (look to the right)

Start Using Your Wiki (its not tricky!)

1. Just point your browser to:

`http://mvs1.centers.ihost.com:80nn/zoswiki`

and explore! Here are some suggestions of things to look at from the Main page:

- The “About” page (see links at top left)
- The “SandBox” page.
 - Try the “Edit this page” link at the bottom to change the page
- The “Find Pages” link at the left allows you to search the Wiki

JPWiki: Parting comments

- One of the things that can be done to increase the effectiveness of your Wiki is to also use the z/OS HTTP server and its “MVSDS” service. It is extremely easy to implement and will allow you to create links in your Wiki pages to MVS sequential or partitioned datasets, to be viewed as text files in your browser.

Make sure that you protect **MVSDS** URLs, as appropriate, with the HTTP server `{{ % %SAF% % }}` directive, so that datasets are accessed using the end user's security credentials.

Check the `/etc/httpd.conf` member and the "HTTP Server for z/OS v5R3 Planning, Installing and Using" for more information.

- PDF documents can be generated automatically from JSPWiki pages by using the following Plugin: <http://www.jspwiki.org/wiki/PDFPlugin>
- If you configure your Tomcat Server on z/OS to use SAF/RACF login security (see next exercise), you can manage and control access to JspWiki content using SAF/RACF.

The “Open Source Java on z/OS” forum at <http://dovetail.com/forum> is a good place to exchange questions and ideas regarding the use of open source Java applications on z/OS, including JSPWiki.

(Optional) Configure Tomcat for SAF (RACF) authentication

This lab section uses a custom Tomcat security plugin, called a “Realm”, which is a free download from <http://dovetail.com>. This security realm uses z/OS SAF (RACF) apis to authenticate user passwords and to map SAF class/entity permission rules onto Tomcat security roles (permissions).

The information in this section is also available at <http://dovetail.com>, so it is available to you later if you don't have time to complete this during the lab.

1. Normally, you would need to copy the custom realm jar to \$CATALINA_HOME/server/lib, but since we are using a shared CATALINA_HOME, we have already done this.
2. Edit ~/tomcat/conf/server.xml (using avi or aoedit)

- Insert the following line:

```
<Listener className="com.dovetail.zos.tomcat.SafLifecycleListener" />
```

Before the line:

```
<Listener  
className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"  
/>
```

- Find and comment out the following lines (using <!-- XML comments --> as shown):

```
<!--  
<Resource name="UserDatabase" auth="Container"  
type="org.apache.catalina.UserDatabase"  
description="User database that can be updated and saved"  
factory="org.apache.catalina.users.MemoryUserDatabaseFactory"  
pathname="conf/tomcat-users.xml" />  
-->
```

- After this comment, insert the following lines:

```
<Resource name="UserDatabase" auth="Container"  
type="org.apache.catalina.UserDatabase"  
factory="com.dovetail.zos.tomcat.SafRoleDatabaseFactory"  
pathname="conf/saf-roles.xml">  
</Resource>
```

- Find and comment out the following:

```
<!--  
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"  
resourceName="UserDatabase"/>  
-->
```

- After this comment, insert the following:

```
<Realm className="com.dovetail.zos.tomcat.SafRealm"  
resourceName="UserDatabase" />
```

3. We've already configured the file: `~/tomcat/conf/saf-roles.xml`
This file contains the initial mapping of SAF (RACF) entities to Tomcat/Servlet roles:

```
<?xml version='1.0' encoding='utf-8'?>
<saf-roles>
<role rolename="admin" safclass="FACILITY"
      safentity="BPX.SERVER" saflevel="READ"/>
<role rolename="manager" safclass="FACILITY"
      safentity="BPX.SERVER" saflevel="READ"/>
</saf-roles>
```

Note: that we've mapped the “BPX.SERVER” SAF/RACF permission to the Tomcat “admin” and “manager” roles. “BPX.SERVER” access has already been granted to all SHAREnn userids, since it is required to use the RACF authentication / check permissions APIs.

4. Stop Tomcat if it is running, and restart by submitting your TC55 JCL.

To stop Tomcat, use the “Y” (Stop) prefix command from SDSF / DA panel. It will take a few moments for Tomcat to shut everything down and end the job.

5. Open your browser to the Tomcat URL and click on the “Tomcat Administration” link.
- Since the “admin” webapp in Tomcat is protected by role “admin”, you will be prompted for a userid/password. Since SAF is now enabled, enter your z/OS SHAREnn userid and password.
 - The admin webapp will take a few moments to initialize the first time you use it. Eventually you will see the administration console, confirming that your SAF/RACF password was accepted and that you have permissions to use the application.
 - You can add a new role mapping from the UI by selecting “User Definition + Roles”. Add a new role by selecting “Role Actions”: “Create new Role”. Enter the new role name of your choosing, In the “Description” field, enter the SAF class, entity, and level separated by slashes, for example: `FACILITY/BPX.SERVER/READ`

Press Enter to save the new role. This will also update `conf/saf-roles.xml`

The “jzos” sample web application has a JSP (Java Server Page) that can be invoked from the jzos webapp's main menu as “SAF Security test”.

- The source for this JSP is in Eclipse path: `/webapp/SafTest.jsp`

This servlet uses the JZOS SAF API to query the authority of a user to have access to a given SAF resource.

Try out this JSP, with:

- Userid = `sharenn` (your userid)
- SAF Class = Facility
- SAF Entity = `BPX.SERVER`
- Access = Read

(You should have access. Try the same with resource “BPX.DAEMON”)