SHARE Session #8369: z/OS Tomcat in an Hour

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.

Objectives

1. Become familiar with how to run z/OS batch jobs under the JZOS Batch Java launcher.

2. Configure and customize your own Tomcat instance on z/OS, using the SDK 5.0 "shared classes" feature.

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. Customize Tomcat to use SAF (RACF) security

6. (optional) Install an open source collaboration application – JspWiki

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.   If you don't have these skills, ***please team up with someone*** who does; see an instructor and we'll help find you a 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 1.2.3 (the IBM alphaWorks version).
  As mentioned in the previous session (#8368), 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.

- If you were not able to attend the previous session: #8368 "Java: z/OS Topics - Batch Integration, Tomcat, and Eclipse", please see if any extra handouts are available, since they contain some helpful background for this lab.

- 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.

Copyright © 2006, Dovetailed Technologies LCC

Hello World

This lab section demonstrates how to run a simple Java application on z/OS under the JZOS batch toolkit. When complete, you will have verified that JZOS and Java were installed correctly and that you can run Java programs from a z/OS batch environment.

NOTE: Your TSO ID for this lab will be SHAREnn, where nn is your assigned unique number. 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
   ```

2. Edit the PDS member **<HLQ>.TOMCAT.JCL(RUN50)**

3. Tailor the jobcard for this JCL, changing <HLQ> to your SHARE*nn* userid:
   ```
   //<HLQ>R JOB (),'TOMCAT',MSGCLASS=H,NOTIFY=&SYSUID
   ```

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

   - JZOS_HOME should be set to: `/sharelab/tomcatlab/jzos`

   - JAVA_HOME should be set to: `/usr/lpp/java15/J5.0`

   - Set: `JAVACLS='com.dovetail.jzos.sample.HelloWorld'`

   - You don't need to customize the CLASSPATH, since the HelloWorld class is in jzos.jar.

   - Uncomment (remove the "#") from the following line to enable SDK 5.0 shared classes:

     ```
     #IJO="$IJO -Xshareclasses:name=$groupname,groupAccess"
     ```

   - *Note:* this JCL contains mixed upper/lower case; the //STDENV shell script is case sensitive, and cannot have line numbers.

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:
     ```
     PREFIX SHAREnn*
     ```

6. Locate your job in the list, and bring up the individual DDNAME list via the "?" command. Browse the each of the following DDs and confirm that they exist and contain the expected output.

DDNAME	Description
SYSOUT	Contains the output from the JZOS batch launcher. Confirm that the launcher finished with an exit code of 0.
STDOUT	Contains the redirected Java standard output stream (System.out). This DD should contain the text: `Hello World! (stdout)`
STDERR	Contains the redirected Java standard error stream (System.err). This DD should contain the text: `Hello World! (stderr)`

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)

/share*nn*	home directory(s) for each lab userid. (in Unix, your home directory can be referred to as "~" (tilde))
/tomcat	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://jzos.com/docs/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.

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 on z/OS.
        Locate the element: `<Host name="localhost"` and change the attribute value for autoDeploy from "true" to "false".

5. Edit the PDS member **<HLQ>.TOMCAT.JCL(TC55)** and tailor the JCL as outlined in the comment at the top of the member.

6. Submit your tailored JCL. You can view the startup by looking at the STDOUT and STDERR DDs for your job.

7. Test Tomcat from a browser by visiting: `http://mvs1.centers.ihost.com:80`**nn** where "nn" is your share lab id.

Deploy 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 TomcatLab" icon on your desktop. You will be starting with a new workspace and will need to import the TomcatSample project to begin the lab:

> - Close the welcome window
> - Menu: File+Import...
> - In the dialog, select "Existing Projects into Workspace"
> - Press "Next >"
> - Press "Browse..." (the current workspace will already be selected)
> - Press "OK" (You should see the project "TomcatSample" checked)
> - Press "Finish"

2. Locate the TomcatSample project in the Package Explorer and expand it.
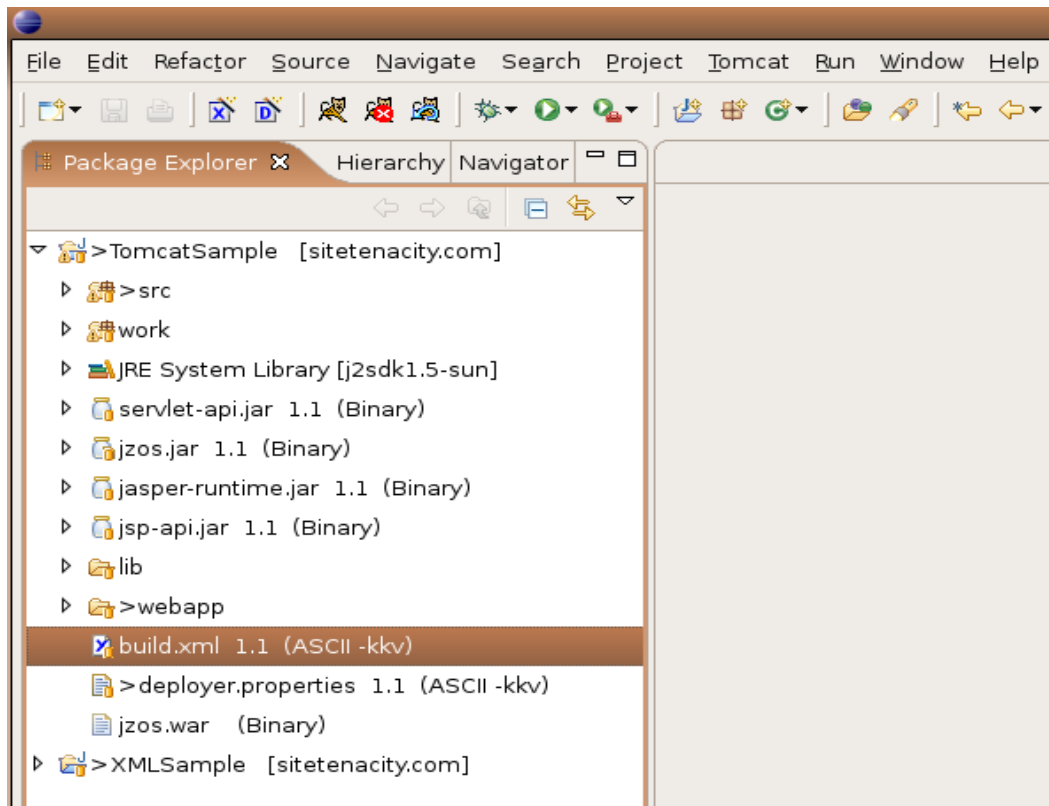
3. Edit the file **deployer.properties** and customize for your specific Tomcat installation. You will need to supply the URL that Eclipse will use to communicate with Tomcat:
    `url=http://mvs1.centers.ihost.com:80nn/manager`
    where '80nn' is the http Port that your Tomcat instance is listening on.

    *   Supply a userid and password that are authorized to manage Tomcat. Unless configured with different security parameters (see the optional labs) 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 project, which will use the customized properties you set in the previous step to deploy your web application. If you are familiar with Ant, feel free to spend some time examining this file.

    > *   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"

    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
    ```

5. 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`
    (the same url that the deployer uses). Supply the userid and password from the previous step when prompted. Once authenticated, you should see a web application at path `/jzos.`

6. 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.

7. From Eclipse, locate the HelloWorld servlet java source code. You can find it by expanding the servlet package under the project's src folder. <<screen shot?>>.

8. Edit the file `HelloWorldServlet.java` and change the text:
   `<html><h1>Hello World!</h1></html>`
   to
   `<html><h1>Hello SHARE Seattle!</h1></html>`

9. Save your changes (this will cause the java file to be compiled)

10. Redeploy your web app.
    A one-click shortcut is to click the Eclipse external tools "run" icon:
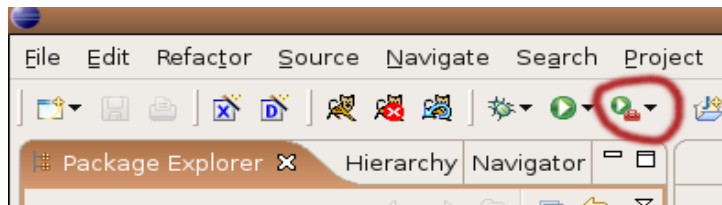
11. Visit your web application and re-run the servlet. You should see your changes.

12. 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 SHARE**nn** 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 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. Redeploy the webapp by running the build.xml ant script, as before.

3. 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!
   ```

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

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).

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
    <?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.

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. Enter your z/OS SHARE**nn** 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`