Advanced Internet Technology Lab

Lab # 3

Servlet Basics

&

Apache Tomcat Server

Eng. Haneen El-masry

March, 2014
Objectives

To be familiar with:

- Servlets Concept.
- Servlet API.
- Servlet Life Cycle.
- Apache Tomcat Server.
- Web Applications.
- Eclipse IDE.

Introduction

What is Java Servlets?

A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP.

Servlet Responsibilities

- Read explicit data sent by client (form data).
- Read implicit data sent by client (request headers).
- Generate the results (Talking with a database).
- Send the explicit data back to client (HTML).
- Send the implicit data to client (cookies and response headers).
Servlet API

The **Servlet interface**: defines methods to initialize a servlet (init method), to receive and respond to client requests (service method), and to destroy a servlet and its resources (destroy method). init, service and destroy are known as life-cycle methods.

The **GenericServlet class**: implements the Servlet interface. It provides simple versions of the life-cycle methods init, service and destroy.

The **HttpServletRequest subclass**: abstract class extends GenericServlet class. It determines request type and sends to doGet(), doPost().

**MyServlet**: (which you create) extends HttpServlet and overrides doGet and/or doPost methods. It provides the specific implementation of your servlet.

The Servlet Life Cycle

The life cycle of a servlet can be categorized into four parts:

1- **Loading and Instantiation**: The servlet container loads the servlet during startup or when the first request is made. After loading of the servlet, the container creates the instance of the servlet.

2- **Initialization**: After creating the instances, the servlet container calls the init() method and passes the servlet initialization parameters to the init() method. The init() must be called by the servlet container before the servlet can service any
request. The initialization parameters persist until the servlet is destroyed. The init() method is called only once throughout the life cycle of the servlet.

3- Servicing the Request: After successfully completing the initialization process, the servlet will be available for service. Servlet creates a separated thread for each request. The servlet container calls the service() method for servicing any request. The service() method determines the type of request and calls the appropriate method (doGet() or doPost()) for handling the request and sends response to the client using the methods of the response object.

4- Destroying the Servlet: If the servlet is no longer needed for servicing any request, the servlet container calls the destroy() method. Like the init() method this method is also called only once throughout the life cycle of the servlet.

What is Servlet Container?

Servlet container (also known as servlet engine) is a runtime environment, which implements servlet API and manages life cycle of servlet components.

Container is responsible for instantiating, invoking, and destroying servlet components.

One example of container is Apache Tomcat which is an open source container.

Apache Tomcat Server

Apache Tomcat is an open source software that implements Java Servlet and Java Server Pages, enabling sites to run Java servlets and Java-based dynamic content.

Apache Tomcat Server Installation and Configuration:

1. Install the Java Software Development Kit (SDK).
2. Download and Extract Tomcat.
Web Application

Servlets, JSP pages, HTML files, utility classes, beans, tag libraries, etc. are bundled together in a single directory hierarchy or file.

Structure of a Web Application

```
webAppDirectory
  |- SomeFile.html
    (URL: http://host/webAppPrefix/SomeFile.html)
  `- SomeFile.jsp
    (URL: http://host/webAppPrefix/SomeFile.jsp)
      `- Images, Style Sheets, etc.

WEB-INF
  `- web.xml

classes
  `- SomeServlet.class
    (URL: http://host/webAppPrefix/servlet/SomeServlet)
  `- SomeUtility.class

somePackage
  `- AnotherServlet.class
    (URL: http://host/webAppPrefix/servlet/somePackage.AnotherServlet)
  `- AnotherUtility.class

lib
  `- SomeClasses.jar
  `- MoreClasses.jar

tlds
  `- SomeTags.tld
  `- MoreTags.tld

META-INF
  `- MANIFEST.MF
```
Apache Tomcat Server Configuration:

The Tomcat configuration files are located in the "conf" sub-directory of your Tomcat installed directory. There are 4 configuration XML files:

1. server.xml
2. web.xml
3. context.xml
4. tomcat-users.xml

"conf\server.xml": Set the TCP Port Number.

The default TCP port number configured in Tomcat is 8080, we want to change it to the default port number of HTTP server that is 80.

```
<Connector port="8080" redirectPort="8443" connectionTimeout="20000"
protocol="HTTP/1.1"/>
```

Change it to:

```
<Connector port="80" redirectPort="8443" connectionTimeout="20000"
protocol="HTTP/1.1"/>
```

"conf\web.xml": Enabling Directory Listing

We enable directory listing by changing "listings" from "false" to "true" for the "default" servlet.

```
<servlet>
  <servlet-name>default</servlet-name>
  <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <init-param>
    <param-name>listings</param-name>
    <param-value>true</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```
"conf\context.xml": Enabling Automatic Reload

We add the attribute reloadable="true" to the <Context> element to enable automatic reload after code changes.

```xml
<Context reloadable="true">
    ....
</Context>
```

Eclipse IDE

Eclipse is a free open source IDE. Support for Java, Android, HTML, CSS, JavaScript, C++, PHP, JSF, servlets, and more.

Eclipse Installation and Configuration:

1- Extract the downloaded file.
2- Run eclipse.exe.
3- Click on “Workbench” icon.
4- Tell Eclipse about Java version.
   a- Window → Preferences
   b- Java → Installed JREs → Press “Add”
c- Choose Standard VM.

d- Navigate to JDK folder → OK
e- Finish >> OK.
5- Tell Eclipse about Tomcat.
   a- Click on Servers tab at bottom.

b- R-click in window → New → Server.
c- Choose Apache → Tomcat v7.0 → Next.

d- Navigate to server folder.
e- Finish.
f- Start Tomcat Server: R-click on Tomcat v7.0, choose "Start".

g- Eclipse forgets to copy the default apps (ROOT, examples, etc.) when it creates a Tomcat folder inside the Eclipse workspace. Copy the ROOT folder. Then go to Eclipse workspace, go to the workspace\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps (or .../tmp1/wtpwebapps if you already had another server registered in Eclipse), and paste ROOT.
h- Go to your browser, open http://localhost/

^_^ Congrats ^_^