# Development of Distributed Applications
# Seminars 2 and 3 – Development of components deployable in Web servers: Java Servlets

13th October 2016
Davinia Hernández-Leo

## Description

**The aim of this seminar is to develop an example of a component deployable in a web server using Java Servlets technology. The implemented servlet will be deployed in the Apache Tomcat servlet container.**

**You are free to define the functionality of your application but its operation should include:**

1. **Use at least one HTTP header to implement its functionality.**
2. **Use at least one HTML form to implement its functionality.**

## Objectives

- See in practice an example of a component-oriented middleware technology: Java servlets.
- Build Web applications whose server-side is based on a servlet deployed in a servlet container (Apache Tomcat).
- Take advantage of HTTP headers to implement the functionality of the servlet.
- Practice the use of HTML and HTML forms in the context of Java servlet development.

## Before the seminar session

Review the documentation of Unit 4. Component-Oriented Middleware.

## Development

Next sections help you to build the application[1].

- Build an HTML page.
- Build a Java servlet.
- Deploy HTML page and Java servlet in Tomcat.

### *Build an HTML page*

This is an example of HTML page linking to a servlet (`TestPage.html`).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
       <HEAD><TITLE>Test</TITLE></HEAD>
       <BODY BGCOLOR="#FDF5E6">
              <H1 ALIGN="CENTER">Test</H1>
              Click <A HREF="/TestServlet">here</A>
              to visit the servlet.
        </BODY>
</HTML>
```

[1] The steps are explained using Windows as operating system and Eclipse as the IDE.

## *Build a Java servlet*

It is necessary to use the **`servlet-api.jar`** library to compile the servlet. It can be found in the lib folder of the Tomcat installation directory. Add it to the library path of the Java project.

The following code is an example of a Java servlet. It returns a web with the text “Hello!”.

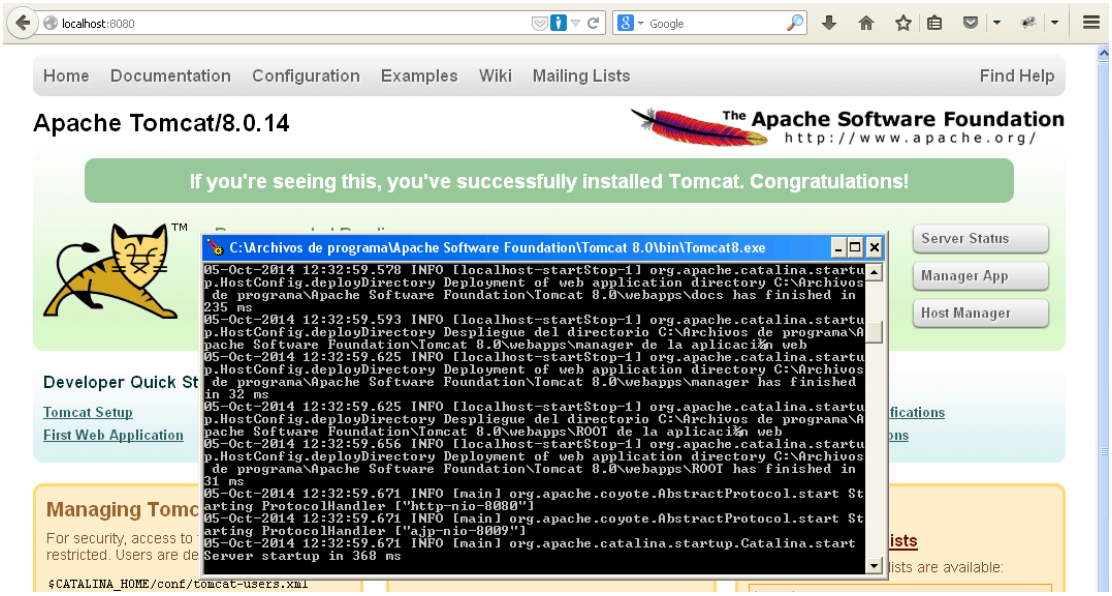
```
public class TestServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
              throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType = "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">\n";
    out.println(docType +
                "<html>\n" +
                "  <head><title>Test Servlet</title></head>\n" +
                "<body>\n" +
                "  <h1>Hello!</h1>\n" +
                "</body></html>");
    }
}
```

## *Deploy HTML web form and Java servlet in Tomcat*

After building the web form and the servlet, it is necessary to deploy both in the Apache Tomcat[2]. Launch Tomcat using **`Tomcat8.exe`** file. It will be available at http://localhost:8080/

If the launching fails, set the environment variables:
JAVA_HOME=C:\Program Files\Java\jdk1.8.0_45
CATALINA_HOME=C:\Program Files (x86)\apache-tomcat-8.0.23

Then, launch the service in CMD:
C:\Program Files (x86)\apache-tomcat-8.0.23\bin>service install
Launch Tomcat using **`Tomcat8.exe`** file. It will be available at http://localhost:8080/




[2] By default, Tomcat 8 is installed in the computer rooms. In case you need to install it, it can be downloaded from http://tomcat.apache.org/download-80.cgi

We will use the default configuration and the `ROOT` application folder of Tomcat `webapps` directory, where all applications must be deployed:

- Copy HTML web form in the `ROOT` folder, for example in Windows: `C:\Archivos de programa\Apache Software Foundation\Tomcat 8.0\webapps\ROOT`
  The web form will be available at: http://localhost:8080/TestForm.html
- Copy the class file of the Servlet (TestServlet.class) in the `classes` folder of `WEB-INF` directory: `C:\Archivos de programa\Apache Software Foundation\Tomcat 8.0\webapps\ROOT\WEB-INF\classes` (create it if it does not exist)
- To deploy the servlet, you need to add it to the deployment descriptor. For web applications, the deployment descriptor must be called `web.xml` and must reside in the `WEB-INF` directory. Tomcat already has a deployment descriptor in the `ROOT/WEB-INF` folder, so you must add the following definition inside the **<web-app> </web-app>** tags:

```
<!-- Servlet definitions for the servlets that make up your web application.
-->
<servlet>
     <servlet-name>TestServlet</servlet-name>
     <servlet-class>TestServlet</servlet-class>
</servlet>

<!-- Define mappings that are used by the servlet container to translate a
     particular request URI (context-relative) to a particular servlet.
-->
<servlet-mapping>
     <servlet-name>TestServlet</servlet-name>
     <url-pattern>/TestServlet</url-pattern>
</servlet-mapping>
```

- Now, the servlet will be available at: http://localhost:8080/TestServlet

You can use the `TestPage.html` to test the servlet.

Classes copied in the `WEB-INF/classes` folder are not automatically reloaded. You must restart Tomcat to load the new classes or modify the `web.xml` file to reload the context.

# Seminar assessment

**For the assessment of the seminar, it is necessary to show the operation of the application to the professor.**

**The operation can be shown along the duration of Seminar 3.**

**The professor will also monitor progress during Seminar 2.**