Dr. Uwe Breitenbücher / Daniel Vietz
Institute of Architecture of Application Systems
University of Stuttgart

Cloud Computing: Concepts and Technologies

Exercise 2 – Tomcat

Introduction

Consider you are using Infrastructure as a Service (IaaS) and you have access to a virtual machine. Coincidentally, the preinstalled operating system of the virtual machine is identical to the one installed on your local machine ☺ (All following tasks are meant to be done on your computer. Just imagine that you are working on a virtual machine.)

The main task of this exercise is to install a webserver and create a running website by using different technologies. The exercise won't give step by step instructions, you rather have to find out important parts by yourself.

Task 1 - Setup Tomcat

Apache Tomcat is an implementation for Java Servlets and JavaServer Pages (JSP). It is a webserver that can host dynamic websites written in HTML/Java. Your first task is to download and run a Tomcat server on your local machine.

You can install it or just take the packaged versions (zip/tar) to easily remove it afterwards. (Maybe you have to set the JAVA_HOME environment variable)

After the installation, familiarize yourself with the structure of the Tomcat installation. What purpose do the several directories bin, conf, webapps, lib, logs, and work have?

Configure your tomcat as follows:

- Change the running port of the application to 80
- Activate automatic reload
- If you want to: enable directory listing (Warning: Not for productional environments)

Start your server and familiarize yourself with the webinterface.

Task 2 - Access Tomcat Manager App

If everything worked, you will see the web console of tomcat where you can find documentation and even some preconfigured working examples. The Manager App although is not accessible yet. Figure out how to access it.

(Hint: You have to create user and role in the configuration)

Task 3 - Add First HTML

Tomcat can be used as a simple webserver. Create a HTML-file and add it to a new webapp. View it in your browser to check if it worked.

Task 4 - Write a Servlet

A servlet is a .java-file that contains plain Java. It extends the abstract class **`HttpServlet`** to be able to produce HMTL. You can take the following snippet as a starting point.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Test extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            // TODO
        } finally {
            out.close();
        }
    }
}
```

At first translate your HTML-file from task 3 into a corresponding servlet. In a second step add the current date to your servlet so that every time a user visits your webapp, the current date will be displayed. Compile the servlet and deploy it on your Tomcat.

(Hints: Compile as follows: **`java -cp TOMCAT_HOME/lib/servlet-api.jar [java-file]`**
Under **`WEB-INF`** you have to create a **`web.xml`** file that maps your servlet to a specific url-path)

Task 5 - Add a JSP File

JavaServer Pages (JSP) is basically a HTML-file that gets extended by Java code. Your task is to create a JSP-webapp that contains a hit-counter. Every time a user visits your site, the counter is increased by one.

Furthermore, add the possibility to reset the hit-counter via a parameter:

`http://localhost/[name-of-your-webapp]/?reset=true`

Add a button/form to your JSP that resets the counter, using this parameter.

Task 6 - Automated Restart

Consider there is an error somewhere and you have to restart your Tomcat. Maybe you have already done this for the previous tasks. Although the effort for restarting is not that big, you should write a script that automates this process.

Hint: Simply create a script that calls the corresponding start and stop scripts in `TOMCAT_HOME/bin`