Create a Java Web Application Using Embedded Tomcat

Last updated 13 November 2015

Table of Contents

This tutorial will show you how to create a simple Java web application using embedded Tomcat.

Follow each step to build an app from scratch, or skip to the end get the source for this article.

If you have questions about Java on Heroku, consider discussing them in the **Java on Heroku forums (https://discussion.heroku.com/category/java)**.

Sample code for the **embedded Tomcat demo (https://github.com/heroku/devcenter-embedded-tomcat)** is available on GitHub.

Prerequisites

- Basic Java knowledge, including an installed version of the JVM and Maven.
- Basic Git knowledge, including an installed version of Git.

Create your pom.xml

Create a folder to hold your app and create a file called pom.xml in the root of that folder with the following contents:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLS
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.heroku.sample</groupId>
  <artifactId>embeddedTomcatSample</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>embeddedTomcatSample Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <properties>
    <tomcat.version>8.0.28</tomcat.version>
  </properties>
  <dependencies>
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-core</artifactId>
        <version>${tomcat.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-logging-juli</artifactId>
        <version>${tomcat.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <version>${tomcat.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.tomcat</groupId>
        <artifactId>tomcat-jasper</artifactId>
        <version>${tomcat.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.tomcat</groupId>
        <artifactId>tomcat-jasper-el</artifactId>
        <version>${tomcat.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.tomcat</groupId>
        <artifactId>tomcat-jsp-api</artifactId>
        <version>${tomcat.version}</version>
    </dependency>
  </dependencies>
  <build>
    <finalName>embeddedTomcatSample</finalName>
    <plugins>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>appassembler-maven-plugin</artifactId>
```

```
            <version>1.1.1</version>
            <configuration>
                <assembleDirectory>target</assembleDirectory>
                <programs>
                    <program>
                        <mainClass>launch.Main</mainClass>
                        <name>webapp</name>
                    </program>
                </programs>
            </configuration>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>assemble</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
  </build>
</project>
```

This pom.xml defines the dependencies that you'll need to run Tomcat in an embedded mode.

The last 3 entries are only required for applications that use JSP files. If you use this technique for an application that doesn't use JSPs then you can just include the first 3 dependencies.

There is also a single plugin defined. The appassembler plugin generates a launch script that automatically sets up your classpath and calls your main method (created below) to launch your application.

Add a launcher class

Create a file called Main.java in your src/main/java/launch directory and put the following in it:

```
package launch;

import java.io.File;

import org.apache.catalina.WebResourceRoot;
import org.apache.catalina.core.StandardContext;
import org.apache.catalina.startup.Tomcat;
import org.apache.catalina.webresources.DirResourceSet;
import org.apache.catalina.webresources.StandardRoot;

public class Main {

    public static void main(String[] args) throws Exception {

        String webappDirLocation = "src/main/webapp/";
        Tomcat tomcat = new Tomcat();

        //The port that we should run on can be set into an environment variable
        //Look for that variable and default to 8080 if it isn't there.
        String webPort = System.getenv("PORT");
        if(webPort == null || webPort.isEmpty()) {
            webPort = "8080";
        }

        tomcat.setPort(Integer.valueOf(webPort));

        StandardContext ctx = (StandardContext) tomcat.addWebapp("/", new File(webappDirLocation
        System.out.println("configuring app with basedir: " + new File("./" + webappDirLocat

        // Declare an alternative location for your "WEB-INF/classes" dir
        // Servlet 3.0 annotation will work
        File additionWebInfClasses = new File("target/classes");
        WebResourceRoot resources = new StandardRoot(ctx);
        resources.addPreResources(new DirResourceSet(resources, "/WEB-INF/classes",
                additionWebInfClasses.getAbsolutePath(), "/"));
        ctx.setResources(resources);

        tomcat.start();
        tomcat.getServer().await();
    }
}
```

Add a Servlet

Create a file called HelloServlet.java in the src/main/java/servlet directory and put the following into it:

```
package servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(
        name = "MyServlet",
        urlPatterns = {"/hello"}
    )
public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        ServletOutputStream out = resp.getOutputStream();
        out.write("hello heroku".getBytes());
        out.flush();
        out.close();
    }

}
```

This is simple Servlet that uses annotations to configure itself.

Add a JSP

Create a file called index.jsp in the src/main/webapp directory:

```
<html>
    <body>
        <h2>Hello Heroku!</h2>
    </body>
</html>
```

Run your application

To generate the start scripts simply run:

```
$ mvn package
```

And then simply run the script. On Mac and Linux, the command is:

```
$ sh target/bin/webapp
```

On Windows the command is:

```
C:/> target/bin/webapp.bat
```

That's it. Your application should start up on port 8080. You can see the JSP at http://localhost:8080 and the servlet and http://localhost:8080/hello

Deploy your application to Heroku

Create a Procfile

You declare how you want your application executed in `Procfile` in the project root. Create this file with a single line:

```
web: sh target/bin/webapp
```

Learn more about procfile (https://devcenter.heroku.com/articles/procfile).

Deploy to Heroku

You can either deploy to Heroku by using the Heroku Maven plugin (https://devcenter.heroku.com/articles/deploying-java-applications-with-the-heroku-maven-plugin) or you can deploy using Git. The latter is described in this article.

Commit your changes to Git:

```
$ git init
$ git add .
$ git commit -m "Ready to deploy"
```

Create the app:

```
$ heroku create
Creating high-lightning-129... done, stack is cedar-14
http://high-lightning-129.herokuapp.com/ | git@heroku.com:high-lightning-129.git
Git remote heroku added
```

Deploy your code:

```
$ git push heroku master
Counting objects: 227, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (117/117), done.
Writing objects: 100% (227/227), 101.06 KiB, done.
Total 227 (delta 99), reused 220 (delta 98)

-----> Heroku receiving push
-----> Java app detected
       ...
       [INFO] ------------------------------------------------------------------------
       [INFO] BUILD SUCCESS
       [INFO] ------------------------------------------------------------------------
       [INFO] Total time: 36.612s
       [INFO] Finished at: Tue Aug 30 04:03:02 UTC 2011
       [INFO] Final Memory: 19M/287M
       [INFO] ------------------------------------------------------------------------
-----> Discovering process types
       Procfile declares types -> web
-----> Compiled slug size is 62.7MB
-----> Launching... done, v5
       http://pure-window-800.herokuapp.com deployed to Heroku
```

Congratulations! Your web app should now be up and running on Heroku. Open it in your browser with:

```
$ heroku open
```

This will show your your JSP and then you can navigate to /hello to see your servlet.

Clone the source

If you want to skip the creation steps you can clone the finished sample:

```
$ git clone git@github.com:heroku/devcenter-embedded-tomcat.git
```