Java servlet send image tutorial

In Java servlet send image tutorial, we create a classic web application in Java using a servlet. The servlet sends an image to the client. The web application is deployed on Jetty server.

Like 0 Share Tweet

Java servlet

Servlet is a Java class which responds to a particular type of network request - most commonly an HTTP request. Servlets are used to implement web applications. They run in a servlet container such as Tomcat o Jetty. In modern-day Java web development programmers use frameworks that are built on top of servlets

Eclipse Jetty is a Java HTTP server and Java Servlet container. Jetty can be easily embedded in devices, tools, frameworks, application servers, and clusters.

Java servlet image example

The following web application sends an image to the client. The web application uses a Java servlet.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.zetcode</groupId>
    <artifactId>sendimageservlet</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>12</maven.compiler.source>
        <maven.compiler.target>12</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>

    </dependencies>

    <build>
        <plugins>

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.2</version>
            </plugin>

            <plugin>
                <groupId>org.eclipse.jetty</groupId>
                <artifactId>jetty-maven-plugin</artifactId>
                <version>9.4.14.v20181114</version>
            </plugin>
        </plugins>
    </build>

</project>
```

The `javax.servlet-api` dependency is a library for building Java servlets. The `maven-war-plugin` collec all artifact dependencies, classes and resources of the web application and packages them into a web application archive (WAR). The `jetty-maven-plugin` plugin is useful for rapid development and testing with Jetty server.

```
pom.xml
src
├───main
│   ├───java
│   │   └───com
│   │       └───zetcode
│   │               SendImageServlet.java
│   ├───resources
│   └───webapp
│       │   index.html
│       └───images
│               sid.jpg
└───test
    └───java
```

This is the project directory structure.

`webapp/index.html`

```
<!DOCTYPE html>
<html>
    <head>
        <title>Servlet image</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <a href="image">Get image</a>
    </body>
</html>
```

The `index.html` file is the home page of our application. It has a link that calls a servlet which servers an image file.

com/zetcode/SendImageServlet.java

```
package com.zetcode;

import javax.servlet.ServletContext;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;


@WebServlet(name = "SendImageServlet", urlPatterns = {"/image"})
public class SendImageServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {

        ServletContext sc = getServletContext();

        try (InputStream is = sc.getResourceAsStream("/images/sid.jpg")) {

            // it is the responsibility of the container to close output stream
            OutputStream os = response.getOutputStream();

            if (is == null) {

                response.setContentType("text/plain");
                os.write("Failed to send image".getBytes());
            } else {

                byte[] buffer = new byte[1024];
                int bytesRead;

                response.setContentType("image/png");

                while ((bytesRead = is.read(buffer)) != -1) {

                    os.write(buffer, 0, bytesRead);
                }
            }
        }
    }
}
```

The `SendImageServlet` servlet returns an image file to the client.

```
@WebServlet(name = "SendImageServlet", urlPatterns = {"/image"})
```

The `@WebServlet` annotation maps the request with `image` URL pattern to the `SendImageServlet` servlet

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
```

The request is a GET request, so we serve it in the `doGet()` method.

```
ServletContext sc = getServletContext();
```

We get the `ServletContext`, which contains a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.

```
try (InputStream is = sc.getResourceAsStream("/images/sid.jpg")) {
```

We get the image resource stream with `getResourceAsStream()`.

```
OutputStream os = response.getOutputStream();
```

We get the servlet output stream. We write image data to this stream. It is the responsibility of the contain
to close servlet output stream.

```
if (is == null) {

    response.setContentType("text/plain");
    os.write("Failed to send image".getBytes());
} else {
```

If we fail to open an image input stream, we send an error message back to the client.

```
response.setContentType("image/png");
```

The image has PNG format; therefore, we set the content type of the response to `image/png`.

```
byte[] buffer = new byte[1024];
int bytesRead;

response.setContentType("image/png");

while ((bytesRead = is.read(buffer)) != -1) {

    os.write(buffer, 0, bytesRead);
}
```

If we successfully opened the image input stream, we read the data and write it to the servlet output strean
We set the response content type to `image/png`.

```
$ mvn jetty:run
```

We run the Jetty server and navigate to `localhost:8080`.

In Java servlet send image tutorial, we have used a Java servlet to send an image to the client.

You might also be interested in the following related tutorials: Java servlet check box tutorial, Java Servlet PDF tutorial, Java Servlet chart tutorial, Servlet FreeMarker JdbcTemplate tutorial, Serving image file in Spring Boot, Java tutorial, or jQuery DatePicker tutorial.

List Java Servlet tutorials.

Home Top of Page