Servlet JSON

Raúl Estrada

Octubre 2020

1. Introduction

In this quick tutorial, we'll create a small web application and explore how to return a JSON response from a *Servlet*.

2. Maven

For our web application, we'll include *javax.servlet-api* and Gson dependencies in our *pom.xml*:

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>${javax.servlet.version}</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>${gson.version}</version>
</dependency>
```

The latest versions of the dependencies can be found here: javax.servlet-api and gson.

We also need to configure a Servlet container to deploy our application to. This article is a good place to start on how to deploy a WAR on Tomcat.

3. Creating an Entity

Let's create an *Employee* entity which will later be returned from the *Servlet* as JSON:

```
public class Employee {

    private int id;

    private String name;

    private String department;

    private long salary;

    // constructors
    // standard getters and setters.
}
```

4. Entity to JSON

To send a JSON response from the *Servlet* we first need to **convert the *Employee* object into its JSON representation**.

There are many java libraries available to convert an object to there JSON representation and vice versa. Most prominent of them would be the Gson and Jackson libraries. To learn about the differences between GSON and Jackson, have a look at this article.

A quick sample for converting an object to JSON representation with Gson would be:

```
String employeeJsonString = new Gson().toJson(employee);
```

5. Response and Content Type

For HTTP Servlets, the correct procedure for populating the response:

1. Retrieve an output stream from the response
2. Fill in the response headers
3. Write content to the output stream
4. Commit the response

In a response, a *Content-Type* header tells the client what the content type of the returned content actually is.

For producing a JSON response the content type should be *application/json:*

```
PrintWriter out = response.getWriter();
response.setContentType("application/json");
response.setCharacterEncoding("UTF-8");
out.print(employeeJsonString);
out.flush();
```

Response headers must always be set before the response is committed. The web container will ignore any attempt to set or add headers after the response is committed.

Calling *flush()* on the *PrintWriter* commits the response.

6. Example Servlet

Now let's see an example *Servlet* that returns a JSON response:

```
@WebServlet(name = "EmployeeServlet", urlPatterns = "/employeeServlet")
public class EmployeeServlet extends HttpServlet {

    private Gson gson = new Gson();

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

        Employee employee = new Employee(1, "Karan", "IT", 5000);
        String employeeJsonString = this.gson.toJson(employee);

        PrintWriter out = response.getWriter();
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        out.print(employeeJsonString);
        out.flush();
    }
}
```

7. Conclusion

This article showcased how to return a JSON response from a Servlet. This is helpful in web applications that use Servlets to implement REST Services.