

## USER FEEDBACK CODE

### Open 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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.3</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.user</groupId>
  <artifactId>Feedback</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Feedback</name>
  <description>Create a Spring Boot project that will capture user feedback using a
REST endpoint. The REST resource will take in parameters using HTTP POST. The
feedback data will be then added to a database table.</description>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
```

```
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jersey</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
```

```
        <optional>true</optional>
    </dependency>
    <!-- this Dependency helps make sure that pathing works correct-->
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <scope>provided</scope>
    </dependency>
```

```
    <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
    </dependency>
```

```
    <dependency>
        <groupId>org.javassist</groupId>
        <artifactId>javassist</artifactId>
        <version>3.25.0-GA</version>
    </dependency>
```

```
</dependencies>
```

```
<build>
```

```
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
```

```
        </plugins>
    </build>
```

```
</project>
```

## **src/main/java**

```
package com.user.Feedback;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class FeedbackApplication {

    public static void main(String[] args) {

        SpringApplication.run(FeedbackApplication.class, args);

    }

}
```

## **FeedbackController.java**

```
package com.user.Feedback.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.MediaType;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;
```

```
import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.bind.annotation.RestController;

import com.project.Feedback.entities.Feedback;

import com.project.Feedback.services.FeedbackService;

@RestController

public class FeedbackController {

    @Autowired

    FeedbackService feedbackService;

    @GetMapping("/feedback")

    public Iterable<Feedback> getAllFeedbacks(){

        return feedbackService.GetAllFeedback();

    }

    @PostMapping(path="/feedback", consumes=
{MediaType.APPLICATION_JSON_VALUE})

    public Feedback addNewFeedback(@RequestBody Feedback fb) {

        Feedback newFb = new Feedback(fb.getComments(), fb.getRating(),
fb.getUser());

        feedbackService.addNewFeedback(newFb);

        return newFb;

    }
```

```
}
```

### **TestFormController.java**

```
package com.user.Feedback.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import com.project.Feedback.entities.Feedback;

import com.project.Feedback.services.FeedbackService;

@Controller

public class TestFormController {

    @Autowired

    FeedbackService feedbackService;

    @GetMapping("/test_form")

    public String showTestForm(ModelMap model) {

        model.addAttribute("test", new Feedback());

        return "testformjsp";

    }

}
```

```

        @PostMapping("/test_form")

        public String submitTestForm(@ModelAttribute("testUser") Feedback fb,
ModelMap m) {

            feedbackService.addNewFeedback(fb);

            m.addAttribute("test", fb);

            return "post";

        }

    }

    //RestTemplate restTemplate = new RestTemplate();

    //URL testForm = new URL("http://localhost:8090/feedbacks/{feedback}");

    //ResponseEntity<String> response = restTemplate.getForEntity(testForm + "/7",
String.class);

    //ObjectMapper mapper = new ObjectMapper();

    //JsonNode root = mapper.readTree(response.getBody());

    //JsonNode name = root.path("name");

    //model.addAttribute(name);

    //String result =
    restTemplate.getForObject("http://localhost:8090/feedbacks/{feedback}", String.class,
7);

```

### **FeedbackRepository.java**

```

package com.user.Feedback.repositories;

import org.springframework.data.repository.CrudRepository;

```

```
import org.springframework.stereotype.Repository;

import com.project.Feedback.entities.Feedback;

@Repository

public interface FeedbackRepository extends CrudRepository<Feedback, Integer> {

    public Feedback findByUser(String feedback);

}
```

### **Feedback.java**

```
package com.user.Feedback.entities;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.NotNull;

import lombok.Data;

@Entity

@Data

public class Feedback {

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)
```



```
@Column(name="id")
```

```
@NotNull
```

```
private Integer id;
```

```
@Column(name="comments")
```

```
private String comments;
```

```
@Column(name="rating")
```

```
@NotNull
```

```
private int rating;
```

```
@Column(name="user")
```

```
private String user;
```

```
public Feedback() {
```

```
    super();
```

```
}
```

```
public Feedback(String comments, Integer rating, String user) {
```

```
    this.comments = comments;
```

```
    this.rating = rating;
```

```
    this.user = user;
```

```
}
```

```
/*
```

```
 * Needed the setters and getters to be able to add name and comments
```

```
otherwise
```

\* they are nulls when entering the SQL DB

\*/

```
public String getComments() {
```

```
    return comments;
```

```
}
```

```
public void setComments(String comments) {
```

```
    this.comments = comments;
```

```
}
```

```
public Integer getRating() {
```

```
    return rating;
```

```
}
```

```
public void setRating(Integer rating) {
```

```
    this.rating = rating;
```

```
}
```

```
public String getUser() {
```

```
    return user;
```

```
}
```

```
public void setUser(String user) {
```

```
    this.user = user;
```

```

    }

    @Override

    public String toString() {

return "Feedback [id=" + id + ", comments=" + comments + ", rating=" + rating + ", user=" + user
+ "];"

    }

}

```

### **FeedbackService.java**

```

package com.user.Feedback.services;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.project.Feedback.entities.Feedback;

import com.project.Feedback.repositories.FeedbackRepository;

@Service

public class FeedbackService {

    @Autowired

    FeedbackRepository feedbackRepo;

    public Iterable<Feedback> GetAllFeedback() {

        return feedbackRepo.findAll();

    }

    public Feedback addNewFeedback(Feedback fb) {

```

```
        return feedbackRepo.save(fb);
    }
}
```

## Src/main/resources

Create folder static and create testform.html and testform.js testform.html

```
<!DOCTYPE html>
<html>
<head>
<script src="testform.js">
</script>
</head>
<body>
<!-- This is a form that is used for testing on the client
side using a client-side code form -->
<h2>Feedback Test Form</h2>

<form onsubmit="SubmitTestForm()">
  <label for="user">User:</label><br>
  <input type="text" id="user" name="user" placeholder="John"><br>
  <label for="comments">Comments:</label><br>
  <input type="text" id="comments" name="comments"
placeholder="Doe"><br><br>
  <input type="submit" value="Submit">
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page called
"/action_page.php".</p>
```

```
</body>
```

```
</html>
```

### **testform.js**

```
function SubmitTestForm() {
```

```
    //TODO: gather fields from form
```

```
    //TODO: Jsonify form fields
```

```
    //TODO: Call postFormDataAsJson to
```

```
    http://localhost:8090/your/endpoint
```

```
    alert("The form was submitted");
```

```
}
```

```
/**
```

```
 * Helper function for POSTing data as JSON with fetch.
```

```
 *
```

```
 * @param {Object} options
```

```
 * @param {string} options.url - URL to POST data to
```

```
 * @param {FormData} options.formData - `FormData` instance
```

```
 * @return {Object} - Response body from URL that was POSTed to
```

```
 */
```

```
async function postFormDataAsJson({ url, formData }) {
```

```
    /**
```

```
     * We can't pass the `FormData` instance directly to `fetch`
```

```
     * as that will cause it to automatically format the request
```

```
     * body as "multipart" and set the `Content-Type` request header
```

```
     * to `multipart/form-data`. We want to send the request body
```

```
     * as JSON, so we're converting it to a plain object and then
```

\* into a JSON string.

\*

\* @see <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST>

\* @see [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Object/fromEntries](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/fromEntries)

\* @see [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/JSON/stringify](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify)

\*/

**const** plainFormData = Object.fromEntries(formData.entries());

**const** formDataJsonString = JSON.stringify(plainFormData);

**const** fetchOptions = {

/\*\*

\* The default method for a request with fetch is GET,

\* so we must tell it to use the POST HTTP method.

\*/

method: "POST",

/\*\*

\* These headers will be added to the request and tell

\* the API that the request body is JSON and that we can

\* accept JSON responses.

\*/

headers: {

  "Content-Type": "application/json",

  "Accept": "application/json"

},

/\*\*

\* The body of our POST request is the JSON string that

```

        * we created above.
        */
        body: formDataJsonString,
    };

    const response = await fetch(url, fetchOptions);

    if (!response.ok) {
        const errorMessage = await response.text();
        throw new Error(errorMessage);
    }

    return response.json();
}

```

### **application.properties**

```

spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/work
spring.datasource.username=root
spring.datasource.password=12345

logging.level.org.springframework.web: DEBUG
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
server.port=8080

```

### **src/main/webapp/WEB-INF/jsp**

#### **index.jsp**

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

```

```

    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>

<html>
<head>
<meta charset="ISO-8859-1">
<title>Welcome Page</title>
</head>
<h2>Landing Page</h2>
<body>
<a href="test_form">Test Form</a><br/><br/>
<a href="feedback">See all Feedbacks</a><br/><br/>

<!-- Can only use these (below) if you have jersey dependency -->
<br/><br/>
<p>Can only use these link below if you have the jersey dependency added to
this dependency.
Jersey has been added to this project so it can use the links below.</p>

<a href="feedbacks">See all feedbacks as Json format</a><br/><br/>
<a href="profile/feedbacks">See Json's in profile</a>
</body>
</html>

```

### Create post.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>

```



```
<meta charset="ISO-8859-1">
<title>Post test</title>
</head>
<body>
Successfully added: ${testUser.toString()}
</body>
</html>
```

### testformjsp.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Spring test App</title>
</head>
<body>
<form:form action="/test_form" method="post" commandName="testUser">
  <label for="user">User:</label><br>
  <input type="text" id="user" name="user" placeholder="John"><br>
  <label for="comments">Comments:</label><br>
  <input type="text" id="comments" name="comments"
placeholder="Doe"><br><br>
  <input type="submit" value="Submit">
  <label for="rating">Rating:</label><br>

  <input type="range" name="rating" id="rating" min="0" max="10" value="5"
class="slider">
</form:form>
</body>
```

</html>