



Sample Primefaces JSF Web App Report

Name: Hao Li

Email: lih13800@gmail.com

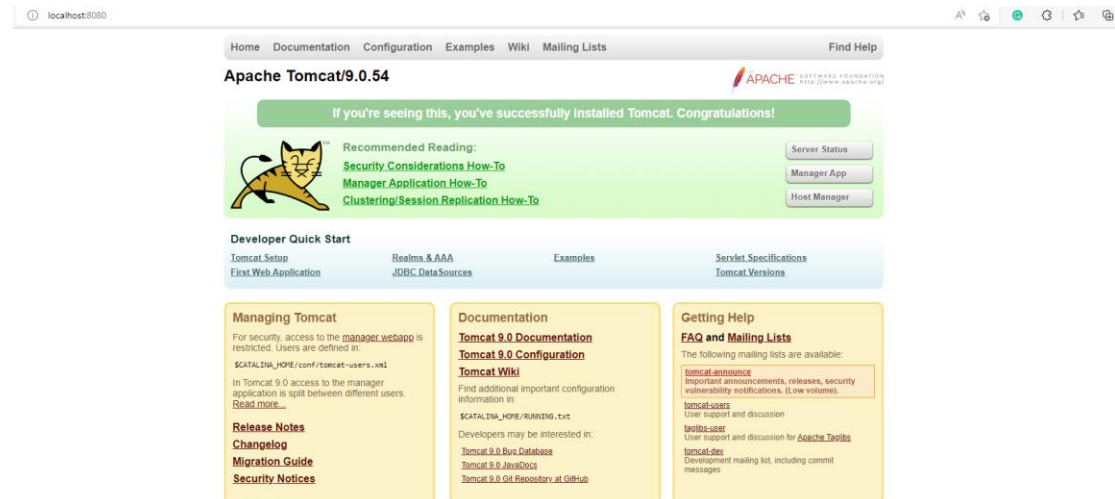
Submission Date: 15st May, 2022

URL Notation

In this project, after deployed to Tomcat, please use following URL:
http://localhost:8080/carProject/faces/index.xhtml

Environment Setup

Apache Tomcat opened on the Eclipse side successfully.



Installed JSF and Primefaces through Maven in pom.xml file

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/com.sun.faces/jsf-api -->
  <dependency>
    <groupId>com.sun.faces</groupId>
    <artifactId>jsf-api</artifactId>
    <version>2.2.18</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/com.sun.faces/jsf-impl -->
  <dependency>
    <groupId>com.sun.faces</groupId>
    <artifactId>jsf-impl</artifactId>
    <version>2.2.18</version>
  </dependency>
  <dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>7.0</version>
  </dependency>
</dependencies>
```

Setup source code in Eclipse's **resource folder**, named dataSource.xml.

carProject

Deployment Descriptor: carProject

JAX-WS Web Services

JRE System Library [JavaSE-1.8]

src/main/java

xmlDataSource

dataSource.xml

Server Runtime [Apache Tomcat v9.0]

Maven Dependencies

jsf-api-2.2.18.jar - C:\Users\Hao Li\m2\re

jsf-impl-2.2.18.jar - C:\Users\Hao Li\m2\

primefaces-7.0.jar - C:\Users\Hao Li\m2\

jstl-1.2.jar - C:\Users\Hao Li\m2\reposito

javax.inject-1.jar - C:\Users\Hao Li\m2\r

Deployed Resources

build

src

target

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<cars>

<car>

<product_name>Jeep Wrangler</product_name>

<drivetrain>AWD</drivetrain>

<manufactured>03-01-2022</manufactured>

<horsepower></horsepower>

<engine_size>1995</engine_size>

<comments></comments>

<colour>Green</colour>

<rating>5</rating>

<image_name>jEEP.png</image_name>

</car>

<car>

<product_name>Chevrolet Camaro</product_name>

<drivetrain>RWD</drivetrain>

<manufactured>01-01-2021</manufactured>

<horsepower>311</horsepower>

<engine_size>3564</engine_size>

<comments>Some comments go here</comments>

<colour>Yellow</colour>

<rating>5</rating>

<image_name>camaro.png</image_name>

</car>

<car>

<product_name>Bugatti Chiron</product_name>

<drivetrain>AWD</drivetrain>

<manufactured>06-01-2021</manufactured>

<horsepower>1480</horsepower>

<engine_size>7993</engine_size>

<comments>Most Expensive</comments>

<colour>Blue</colour>















<rating>5</rating>

<image_name>bugatti.png</image_name>

</car>

</cars>

Stored images under the webapp folder, the name of images matched corresponding image_name XML element value

- >  build
- ▼  src
 - ▼  main
 - >  java
 - ▼  webapp
 - ▼  Image
 -  bugatti.png
 -  camaro.png
 -  jeep.png
 - >  META-INF
 - >  WEB-INF
 -  index.xhtml
- >  target
-  pom.xml

Functionality

1. Read the data source through `getClass().getClassLoader().getResource()` method.

This method helps us to find resource file's location correctly. In the beginning, Eclipse could find out the xml file from source folder according to the relative path. However, when the programming running on the Tomcat, the relative path failed. The class loader which implemented Java reflection can figure out this problem. Most importantly, it would not be able to find the source data even if it is deployed in a different environment in the future.

```
private void unMarshalingExample() throws JAXBException, URISyntaxException {
    JAXBContext jaxbContext = JAXBContext.newInstance(Cars.class);
    Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

    URL resource = getClass().getClassLoader().getResource("dataSource.xml");
    // System.out.println(resource.toURI().toString());

    Cars cars = (Cars) jaxbUnmarshaller.unmarshal(new File(resource.toURI()));
    // Cars cars = (Cars) jaxbUnmarshaller.unmarshal(new File("xmlDataSource/dataSource.xml"));

    for (Car car : cars.getCars()) {
        System.out.println(car.getImage_name());
        System.out.println(car.getProduct_name());
        System.out.println(car.getManufactured());
    }
    setCars(cars.getCars());
}
```

Use `jaxbUnmarshaller` to unmarshal the xml data source file.

Each group of data is stored in Car Object.

2. Created label 'Products' and a drop down with value from `product_name` XML element. Once the page loads successfully, the data of the first xml element is loaded on the page.

Products: Jeep Wrangler

Product Name: Jeep Wrangler

Drivetrain: AWD

Manufactured: Mar 01, 2022

Horsepower: 0

Engine Size: 1995

Comments

Color ☐ Rating | ⭐⭐⭐⭐⭐

Validate Reset

3. When users change the drop down manual, it updated the panel with corresponding value successfully.

The image displays two screenshots of a web browser showing a car selection form. The top screenshot shows the 'Chevrolet Camaro' selected in the 'Products' dropdown, with fields for Drivetrain (RWD), Manufactured (Jan 01, 2021), Horsepower (311), Engine Size (3564), and a Comments box. A yellow Camaro image is shown on the right. The bottom screenshot shows 'Bugatti Chiron' selected, with fields for Drivetrain (AWD), Manufactured (Jun 01, 2021), Horsepower (1480), Engine Size (7993), and a Comments box. A blue Chiron image is shown on the right. Both forms have 'Validate' and 'Reset' buttons at the bottom.

4. Implemented page level and Bean (model) validation to enforce these constraints
- "Product Name" and "Drivetrain" are text fields which must not be empty, and they can be up to 254 characters as maximum length. The error message would appear if the constraints were violated.

```
@NotNull(message = "Name cannot be null")
@Size(max = 254, message="No more than 254 characters")
private String selectedCar;

@NotNull(message = "Drivetrain cannot be null")
@Size(max = 254, message="No more than 254 characters")
private String selectedDrivetrain;
```

Browser

Products: Jeep Wrangler

Product Name:


Drivetrain:

Manufactured:

Horsepower:

Engine Size:

Comments:



Color ☐ Rating | ☐ ☐ ☐ ☐ ☐

Product Name Error
Product Name cannot be Empty.

Browser

Products: Jeep Wrangler

Product Name:


Drivetrain:

Manufactured:

Horsepower:

Engine Size:

Comments:



Color ☐ Rating | ☐ ☐ ☐ ☐ ☐

Drivetrain Error
Drivetrain cannot be Empty.

- “Manufactured” field was setup a date picker, and its formatted meets the requirement.

```
<div>
  <h:panelGrid columns="2" cellpadding="5" >
    <h:outputText value="Manufactured:" />
    <p:datePicker id="date" value="#{cpt.date}" pattern="MMM dd, yyyy" placeholder="#{cpt.selectedDate}">
  </h:panelGrid>
</div>
```

Manufactured:

Horsepower:

Engine Size:

Comments:

May 2022						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

- Setup “number only” constraint for “Engine size” and “Horsepower” text filed. And the number should not less than 0. The error message would appear if any non-number character appeared.

```
@Min(value = 0, message = "Horsepower should not be less than 0")
private Integer selectedPower;

@Min(value = 0, message = "Engine size should not be less than 0")
private Integer selectedEngine;
```

Browser

Products:

Product Name:


Drivetrain:

Manufactured:

Horsepower:

Engine Size:

Comments



Color Rating |

Engine Size Error
Please Input Number
Greater than 0.

- Implemented the characters constraint on the multirow test area. Also, add on hard constraint on the page side. The text area would not receive any input when it achieved to its limitation.

```
@Size(max = 200, message="No more than 200 characters")
private String selectedComment;
```

```
<div>
  <h:panelGrid columns="2" cellpadding="5" style="width: 116%">
    <h:outputText value="Comments" />
    <p:inputTextarea id="comment" value="#{cpt.selectedComment}" rows="4" cols="50" maxlength="200" autoResize="false" />
  </h:panelGrid>
</div>
```

Horsepower:

Engine Size:

Comments

5. Clicked the color block, it would open a colorpicker

```
<h:outputText value="Color" />
<p:colorPicker value="#{cp.popupColor}" />
```




- Rating component would use corresponding rating XML element value as its default value. Those lit stars could be canceled by mouse click. Also, click the little button on the left side of stars would extinguish all stars.

```
<h:outputText value="Rating |" />
<p:rating id="rate" value="#{cpt.selectedRating}" />
```



- Validating all constraints by using “Validation” button and show error message on the top of the UI. The error message would disappear until the text filed match the constraint and press the “Validation” button again.

Browser

Product Name Error

Drivertrain Error

Horsepower Error

Engine Size Error

Products: Jeep Wrangler

Product Name:

Drivertrain:

Manufactured: Mar 01, 2022

Horsepower: 0

Engine Size: 0

Comments

Color

Rating | ★ ★ ★ ★ ★

Validate

Reset

Product Name Error
Product Name cannot be Empty.

Drivertrain Error
Drivertrain cannot be Empty.

Horsepower Error
Please Input Number Greater than 0.

Engine Size Error
Please Input Number Greater than 0.

8. “Reset” button would reset those changed of current selected car.

Browser

Products:

Product Name: x

Drivetrain:

Manufactured:

Horsepower:

Engine Size:

Comments:

Products:

Product Name: x

Drivetrain:

Manufactured:

Horsepower:

Engine Size:

Comments: