(../..)

Getting started

This lab is more challenging than the previous labs. We will only describe at a fairly high level what needs to happen for the replatforming to be completed. Do not hesitate to do some additional research if you feel your are stuck.

Clone the repository:

```
git clone https://github.com/platform-acceleration-lab/apps-platform-acceleration-struts-code.git
```

Introduce the spring application

- Update the `pom.xml`
  - Add plugin `spring-boot-maven-plugin`
  - Set parent `spring-boot-starter-parent`
  - Add dependencies
    - Web starter `spring-boot-starter-web`
    - JPA starter `spring-boot-starter-data-jpa`
    - Tomcat starter `spring-boot-starter-tomcat`
    - JSP support `tomcat-embed-jasper`
    - Database driver `mysql-connector-java`
- Create application class
- Configure DB connection in `application.yml`

```
spring:
  jpa:
    generate-ddl: true
    properties.hibernate.dialect: org.hibernate.dialect.MySQL5Diale
ct

  datasource:
    url: jdbc:mysql://localhost:3306/struts?useSSL=false
    username: root
```

Setup the filters

- Configure a `FilterRegistrationBean` for each filter in the `web.xml`
- Make sure you set the order to match the order in the `web.xml` file

Setup the index page

Create an action for the index in `struts.xml`.

- action with `name=""`
- result `/index.jsp`
- create a Class for it

Insert the prelude in JSP files that use it

Insert the following line at the top of `index.jsp`, `addUserForm.jsp`, and `findUserForm.jsp`:

```
<%@ taglib prefix="s" uri="/struts-tags" %>
```

Setup `struts2-spring-plugin`

- Add the `struts2-spring-plugin` dependency to the `pom.xml`
  - Version should match the version of `struts2-core`, there is no need to upgrade the version.
- Make UserService injectable.
  - Annotate the `UserServiceImpl` class with `@Repository`.
- Make all action classes (`FindUser`, `AddUser`, etc.) injectable.
  - Annotate the class with `@Component` /
  - Make the `UserService` a field and create matching constructor/
  - Do not use JNDI anymore/

- Annotate the `execute()` function with `@Transactional` if it modifies the database content.

Update JSPs to use struts tags

- `displayUser.jsp`

```
<title>User Details</title>

<%@ taglib prefix="s" uri="/struts-tags" %>
<h2>User Details </h2>
<table>
    <tr>
        <td><b>ID</b></td>
        <td><s:property value="user.id"/></td>
    </tr>
    <tr>
        <td><b>First Name</b></td>
        <td><s:property value="user.firstName"/></td>
    </tr>
    <tr>
        <td><b>Last Name</b></td>
        <td><s:property value="user.lastName"/></td>
    </tr>
</table>
```

- `displayUsers.jsp`

```
<title>All Users</title>

<%@ taglib prefix="s" uri="/struts-tags" %>
<table>
    <tr>
        <th>ID</th>
        <th>First Name</th>
        <th>Last Name</th>
    </tr>
    <s:iterator value="users">
        <tr>
            <td><s:property value="id"/></td>
            <td><s:property value="firstName"/></td>
            <td><s:property value="lastName"/></td>
        </tr>
    </s:iterator>
</table>
```

Test it locally

- Create a database called `struts`:

```
mysql -uroot
```

```
create database struts;
```

- Run the app:

```
mvn spring-boot:run
```

Visit the web page and ensure it works as expected before moving on.

Ship it!

Deploy the application to Pivotal Cloud Foundry:

```
mvn clean package
cf push struts -p target/struts.war --random-route --no-start
cf create-service p-mysql 100mb struts-mysql
cf bind-service struts struts-mysql
cf start struts
```

Visit the web page and ensure it works as expected.

Assignment

Once you are done with the lab and the application is deployed and working on PCF, you can submit the assignment using the `submitReplatformingStruts` gradle task. It requires you to provide the `strutsAppUrl` project property. For example:

```
cd ~/workspace/assignment-submission
./gradlew submitReplatformingStruts -PstrutsAppUrl=http://my-struts-ap
p.cfapps.io
```