Lab 03 – Finish and Deploy an Application

In this lab, you'll finish your version of the application featured in Lab 1. This includes creating the remaining REST calls. Test your work locally as you develop. When you're finished, you'll explore build and deployment options.

Lab 3-1: Add DELETE to the Spring Boot REST Application

Your application is nearly complete. All that remains is to write the DELETE method and add CORS support. POST and PUT methods are included in the EmployeeController class. Study these to figure out how to write your DELETE method.

- 1. Continue your project from Week 2. If you got stuck, you can download the Week 2 solution from the Lesson 2 page.
- 2. In your IDE, open EmployeeController.java. Create a DELETE method handler for a delete method. This method should:
 - a. Remove the employee whose ID is specified in the URL. For example: http://localhost:8080/employees/100
 - b. Return an appropriate error message if the employee does not exist.
- 3. Add CORS support to your application.
- **4.** Test your code locally. Commit your changes when you're satisfied.
 - a. A standalone HTML5 client is provided in the course links if you wish to test the client from the local hard disk.

Lab 3-2: Deploy your Application from the Command Line using the REST API (Optional)

In this lab, interact with the Oracle Application Container Cloud Service by creating and deploying an application. The application is provided for you, just follow these steps to deploy it.

Download Employee REST Application

First, you download the final version of the application you will build in this course. The application is contained in an application archive zip that is ready to deploy. You will deploy the application to Oracle Application Container Cloud Service and examine its features.

Oracle REST API and Required Information

Oracle Cloud provides a REST API for managing your applications in the cloud. Using the REST API, you can create scripts to upload and deploy your application. Scripts have been provided for you in the curl directory that demonstrate common operations you may perform.

Before you begin, you need several pieces of information for your scripts.

- **Username:** The user name you use to log into Oracle Cloud Service.
- Password: The password you need to log into Oracle Cloud Service.
- **Identity Domain:** Your identity domain is essentially your account name. It defined a shared cloud service space you share with your team.

- **Storage REST Endpoint:** This is the URL you use to send REST commands to the Storage Cloud Service. To get this information:
 - o Log in to your cloud account.
 - Click on Storage on the cloud service dashboard. This will show you the service details for this service.
 - A number of data fields are shown, at the end is REST Endpoint.
 - Save this URL locally. It is your Storage REST Endpoint. It has the following structure:
 - https://IdentityDomain.storage.oraclecloud .com/v1/Storage-IdentityDomain
 - Where IdentityDomain is replaced with your actual identity domain.
- **Service REST Endpoint:** This is the URL you use to send REST commands to the Oracle Application Container Cloud Service. To get this information:
 - o Log in to your cloud account.
 - Click on Application Container on the cloud service dashboard. This will show you the service details for this service.
 - Several data fields are shown, at the end is a REST Endpoint.
 - Save this URL locally. It is your REST Endpoint for this service.
 It has the following structure:
 - https://apaas.region.oraclecloud.com/
 - Where Region is replaced with your actual region.
- REST API Documentation: To find the operations you can
 perform for each cloud service, visit the documentation pages
 for each respective service. Both <u>Oracle Application Container</u>
 <u>Cloud Service</u> and <u>Storage Cloud Service</u> provide details and

examples of the REST operations that are available for the service.

Upload Your Application to Storage Cloud Service

The steps that follow perform these basic operations: list storage contents, create a storage container, and upload a application archive to a storage container.

- 1. List the contents of your storage space by creating listall.sh.
 - a. Modify the script to use your username, password, and storage REST Endpoint. The contents should look similar to the following:

curl -i -X GET -u user@example.com:password https://identitydomain.storage.oraclecloud.com/v1/Storage-identity-domain/

- b. Execute the script from a bash shell. You get a list of containers created in your storage space.
- 2. Create the apps storage container by creating the create.sh script.
 - a. Modify the script to use your username, password, and storage REST Endpoint. The contents should look similar to the following:

curl -i -X PUT -u user@example.com:password https://identitydomain.storage.oraclecloud.com/v1/Storage-identity-domain/apps

- b. Execute the script from a bash shell. You get a confirmation message that your container was created.
- 3. Run the listall.sh script to verity that your container was created. The apps directory should be included in the list.

- 4. Upload your application file your storage container by creating upload.sh.
 - a. Modify the script to use your username, password, and storage REST Endpoint. The contents should look similar to the following:

curl -i -v -X PUT -u user@example.com:password https://identitydomain.storage.oraclecloud.com/v1/Storage-identitydomain/apps/EmployeeRESTApp-1.0-dist.zip -T
/c/labs/SpringRepo/EmployeeRestApp/target/EmployeeRESTApp-1.0dist.zip

- b. Execute the script from a bash shell. You get a confirmation message that your application was uploaded.
- 5. List the contents of your apps container by creating a list.sh script that takes the folder name as a parameter. For example: list.sh apps
 - a. Modify the script to use your username, password, and storage REST Endpoint. The contents should look similar to the following:

curl -i -X GET -u user@example.com:password https://idenitydomain.storage.oraclecloud.com/v1/Storage-idenity-domain/\$1

b. Execute the script from a bash shell. You get a list of files or containers in the apps container.

Deploy Your Application

Finally, the application needs to be deployed.

6. Deploy the application stored in your storage apps container by creating deploy.sh.

a. Modify the script to use your username, password, and service REST Endpoint. The contents should look similar to the following:

```
curl -i -v -X POST -u user@example.com:password\
    -H "X-ID-TENANT-NAME:identity-domain"\
    -H "Content-Type: multipart/form-data"\
    -F "name=EmployeeRESTApp1"\
    -F "runtime=java"\
    -F "subscription=Monthly"\
    -F "deployment=@deployment.json"\
    -F "archiveURL=/c/labs/SpringRepo/EmployeeRestApp/target/EmployeeRESTApp-1.0-dist.zip"\
    -F "notes=REST app for testing"\
https://apaas.region.oraclecloud.com/paas/service/apaas/api/v1.1/apps/identity-domain
```

- b. Execute the script from a bash shell.
- c. Open the service console for Oracle Application Container Cloud.
- d. Refresh the application list a few times. Your application should appear in the list after a few seconds. The application should fully deploy in a few minutes.

Your application now deployed.

Lab 3-3: Create an Automated Build Job for your Application (Optional)

In this lab, use Oracle Developer Cloud Service to create an automated build of your project.

- 1. Create a Build Job named SpringBuild.
- 2. Configure the job to use your Git repository.

- 3. Add a Maven step to build your application.
- 4. Configure the job to save your JAR and ZIP archives.
- 5. Build your application from Developer Cloud Service.

Lab 3-4: Create an Automated Deployment for your Application (Optional)

In this lab, use Oracle Developer Cloud Service to create an automated deployment configuration for your project.

- 1. Add a build trigger to your build job to perform a build after a commit is made to your repository.
- 2. Create a Deploy Configuration named Spring Deploy.
- 3. Make the configuration automatic based on stable builds.
- Specify Oracle Application Container Cloud Service as your deployment target.
- **5.** Test your configuration.
 - a. Add a comment to your source code.
 - b. Commit the change to your local repository.
 - c. Push your changes to the cloud.
- 6. Track the process until your application is deployed.