SAP / cloud-businessrules-samples

Branch: master ▾ cloud-businessrules-samples / cf-apps / README.md Find file Copy path

ArchanaShukla Modified README afb374c on May 28, 2018

1 contributor

189 lines (152 sloc) 7.29 KB

Business Rule in Cloud Foundry Sample Application

Prerequisites

To make this sample application work, please make sure you have:

- An account on SAP Cloud Platform.
  - For Example, you may sign up for a trial account. For more information about the Sign up process, see this blog
- A basic knowledge of Java/Spring
- A basic knowledge of Maven
  - Download, Install and Setup Maven in your system
  - All components are built with Apache Maven
- A basic knowledge of Node.js
  - Install and Setup NPM in your system
  - For more information about working with NPM registry, see this blog

Running the Application in Cloud Foundry Environment

1. Download the Shopping Cart Application

Download and extract the shopping cart application under **cf-apps** folder

2. Adding required security libraries

To secure the application we have to add Spring Security to the classpath. By configuring Spring Security in the application, Spring Boot automatically secures all HTTP endpoints with BASIC authentication. Since we want to use OAuth 2.0 together with Java Web Tokens (JWT) instead, we need to add the Spring OAUTH and Spring JWT dependencies as well. To enable offline JWT validation the SAP XS Security Libraries need to be added as well. The latest version can be downloaded from the Service Marketplace.

- Create **libs** folder inside the extracted **cf-shoppingcart** folder in step 1
- Unzip the file downloaded from the service marketplace to the *libs* folder
- Install SAP XS Security Libraries to your local maven repo by executing this command:

```
cd cf-shoppingcart/libs
mvn clean install
```

- Once the libraries are successfully installed, you will see the following dependencies are added to `pom.xml` file:

Note: You may need to adapt the version number in your `pom.xml` in case you are using a newer version of the SAP XS Security Libraries. You can get that version number from the downloaded libraries, and accordingly change them in `pom.xml` for all `groupIds` that starts-with `sap.com` respectively.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.security.oauth</groupId>
    <artifactId>spring-security-oauth2</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-jwt</artifactId>
</dependency>
<dependency>
    <groupId>com.sap.xs2.security</groupId>
    <artifactId>security-commons</artifactId>
    <version>0.28.6</version>
</dependency>
<dependency>
    <groupId>com.sap.xs2.security</groupId>
    <artifactId>java-container-security</artifactId>
    <version>0.28.6</version>
</dependency>
<dependency>
    <groupId>com.sap.xs2.security</groupId>
    <artifactId>java-container-security-api</artifactId>
    <version>0.28.6</version>
</dependency>
<dependency>
    <groupId>com.unboundid.components</groupId>
    <artifactId>json</artifactId>
    <version>1.0.0</version>
</dependency>
<dependency>
    <groupId>com.sap.security.nw.sso.linuxx86_64.opt</groupId>
    <artifactId>sapjwt.linuxx86_64</artifactId>
    <version>1.1.19</version>
</dependency>
<dependency>
    <groupId>com.sap.security.nw.sso.ntamd64.opt</groupId>
    <artifactId>sapjwt.ntamd64</artifactId>
    <version>1.1.19</version>
</dependency>
<dependency>
    <groupId>com.sap.security.nw.sso.linuxppc64.opt</groupId>
    <artifactId>sapjwt.linuxppc64</artifactId>
    <version>1.1.19</version>
</dependency>
<dependency>
    <groupId>com.sap.security.nw.sso.darwinintel64.opt</groupId>
    <artifactId>sapjwt.darwinintel64</artifactId>
    <version>1.1.19</version>
</dependency>
```

3. Configure NPM

Execute the following command:

```
npm config set @sap:registry https://npm.sap.com
```

4. Build the application

Build Java App

Execute the following command:

```
cd cf-shoppingcart/app
mvn clean install
```

```
cd ..
```

Build Web App

```
cd cf-shoppingcart/web
npm install
cd ..
```

5. Deploy application in Cloud Foundry

Access your cloud foundry API endpoint

```
cf api https://api.cf.eu10.hana.ondemand.com
```

Login to your account

```
cf login
```

If you have access to more than 1 org or space, execute the following command:

```
cf target -o ORG -s SPACE
```

Update the application

URLs (or better host names) must be unique within the domain used by Cloud Foundry installation.

To prevent URL collisions with other employees who are trying the same sample application in the trial org of Cloud Foundry, you need to update the application manifest-cf-factory.yml.

For that you need to **replace** in the manifest file **all [trial user id]** occurrences with your user id. Hint: Use find/replace or sed for reliable results.

Create Service Instance

```
cf create-service business-rules lite java-ruleservice
cf create-service xsuaa application java-rule-sample-uaa -c ./security/xs-security.json
```

Push the application

Ensure that you have assembled the application with Maven as described above.

```
cd cf-shoppingcart
cf push -f manifest-cf-factory.yml
```

Assign Business Rules Roles

Assign the user with the following Roles

Application Identifier	Role Name	Role Template
bpmrulebroker!b<tenant no>	RuleRuntimeSuperUser	RuleRuntimeSuperUser
rule-sample-java!t<tenant no>	Editor	Editor
rule-sample-java!t<tenant no>	Viewer	Viewer

Follow the HowTo JAM page for assigning the roles to a user

Map Routes to application

Execute the following command in order to create a tenant-specific route for the application:

- Replace the `[subdomain]` with your cloud foundry subdomain

```
cf map-route java-rule-sample-web cfapps.eu10.hana.ondemand.com -n [subdomain]-java-rule-sample-web
```

6. Access the application

URL: http://[subdomain]-java-rule-sample-web.cfapps.eu10.hana.ondemand.com

Authors

Archana Shukla

Copyright and License