WSO2Con AppCloud Tutorial Steps

Register in <https://cloud.wso2.com> and login in.

Create a Java web application.

Give a name like MyTravelLog.

Optionally, select an Icon for your application.

Create the app.

In the overview page, click on Deploy or Open button to check the default Hello World page.

We will be using a google place API to search places, so you can use below keys to try out the sample application.

Keys

====

AIzaSyCOuoI5b09xD6GWkETlmhycKwmnGgt-PGE

AIzaSyDPwttUXqFOVw0GoAYW9z9xv49v3x-xI6Y

AIzaSyAuDPfSxtJUnizJaNVahtjMko7EjThJh6Y

AIzaSyDQOhLqVwb5u-tH20F2SJf0RC2QegLa5-8

Let's create a property to retrieve this key.

Click on runtime configs → Properties → Add Property button.

Name the property as 'googleAPIKey' and value taken from above step.

Go to Repo and Build page and click on edit to start coding using cloud IDE.

Lets create a class with name 'PlaceAPIDataRetriever' and try to get google place api data using a open source java library.

update pom.xml with dependency.

<dependency>

<groupId>se.walkercrou</groupId>

<artifactId>google-places-api-java</artifactId>

<version>2.1.2</version>

</dependency>

create class and write business logic. The sample code is found at <https://github.com/manjulaRathnayaka/WSO2ConApp/blob/master/src/main/java/com/HelloApp/PlaceAPIDataRetriever.java>

Notice the special code segment used to read the 'googleAPIKey' value.

private String getGoogleAPIKey(String applicationKey){

String propValue = null;

String resourcePath = "/dependencies/"+applicationKey+"/googleAPIKey";

CarbonContext cCtx = CarbonContext.getThreadLocalCarbonContext();

Registry registry = (Registry) cCtx.getRegistry(RegistryType.SYSTEM\_GOVERNANCE);

try{

if (registry.resourceExists(resourcePath)) {

Resource resource = registry.get(resourcePath);

if (resource.getContent() != null) {

if (resource.getContent() instanceof String) {

propValue = (String) resource.getContent();

} else if (resource.getContent() instanceof byte[]) {

propValue = new String((byte[]) resource.getContent());

}

}

} else {

log.error("googleAPIKey property doesn't exists");

}

}catch(RegistryException e) {

log.error("Unable to read the resource content of googleAPIKey.",e);

}

return propValue;

}

Make sure to change applicationKey name in above class.

create a JSP page called ‘search.jsp’ and use ‘searchPlace’ method to retrieve search results. The sample code is found at <https://github.com/manjulaRathnayaka/WSO2ConApp/blob/master/src/main/webapp/search.jsp>

After you have completed above steps, commit the changes following below steps. If you missed to push changes, you will loose all the changes done.

1. On the File menu, click Save All
2. On the Git menu, click Add to index
3. On the Git menu, click Commit
4. On the Git menu, click Remotes / Push

Now you should be able to access your newly created search.jsp and search results.

Let's create a database to store place details searched in above page.

Go to the database section and add new database.

Provide a name such as ‘PlaDB’.

Select the ‘Development’ environment.

Provide a strong password which is required later for configuring the data source.

After you have successfully creating the DB, click on the created database and find the database url and user. Then connect to this database using a sql client and execute below commands. This will populate the table for storing the place details.

**mysql -h mysql-dev-01.cloud.wso2.com -u<user> -p**

**show databases;**

**use <your database>;**

**CREATE TABLE IF NOT EXISTS PlaceDetails ( Name VARCHAR (100), UserName VARCHAR (20), Address VARCHAR(100), Latitude DOUBLE, Longitude DOUBLE, GoogleURL VARCHAR(100), IconURL VARCHAR(100), VisitedDate DATE, Comment Text, CONSTRAINT PLACE\_PKEY PRIMARY KEY (Name, UserName,VisitedDate) );**

**insert into PlaceDetails() values("Acadia National Park", "User Foo", "Maine, United States", 44.338556, -68.273335, "https://plus.google.com/107246219091349403167/about?hl=en-US", "http://maps.gstatic.com/mapfiles/place\_api/icons/generic\_business-71.png", "0033-11-05", "comment");**

Next create a datasource with name ‘PlacesDS’ from runtime configs → Datasources → create datasource. Keep the defaults as it is and provide the password given for ‘PlaDB’.

Create PlaceDetail.java bean class. The sample code is found at <https://github.com/manjulaRathnayaka/WSO2ConApp/blob/master/src/main/java/com/HelloApp/PlaceDetail.java>

Create the DAO class for inserting and reading data from above table using the datasource. The sample code is found at <https://github.com/manjulaRathnayaka/WSO2ConApp/blob/master/src/main/java/com/HelloApp/PlaceDetailDAO.java>

Notice the special code segment used to lookup the datasource.

private DataSource getDataSource() {

DataSource dataSource = null;

Hashtable<String, String> env = new Hashtable<String, String>();

try {

InitialContext context = new InitialContext();

if (env.size() > 0) {

context = new InitialContext(env);

}

dataSource = (DataSource) context.lookup("jdbc/PlacesDS");

} catch (NamingException e) {

e.printStackTrace();

log.error("============ERROR===========",e);

}

return dataSource;

}

Update the index.jsp page to list stored place information. The sample code is found at <https://github.com/manjulaRathnayaka/WSO2ConApp/blob/master/src/main/webapp/index.jsp>

Lets add another page to store searched place details with additional information such as user name, date, comments, image links etc.

Name the page as ‘save.jsp’ and sample code can be found at [https://github.com/manjulaRathnayaka/WSO 2ConApp/blob/master/src/main/webapp/save.jsp](https://github.com/manjulaRathnayaka/WSO2ConApp/blob/master/src/main/webapp/save.jsp)

Finally for making the UI better, add the images and css directories found at <https://github.com/manjulaRathnayaka/WSO2ConApp/tree/master/src/main/webapp/css>

<https://github.com/manjulaRathnayaka/WSO2ConApp/tree/master/src/main/webapp/images>

Commit and git push all the changes done. You should be able to access your application after completing the deployment and it should looks like similar to <https://appserver.test.cloud.wso2.com/t/manjulaorg/webapps/mytravellog-1.0.0>

Now we are ready with the sample application for QA. Lets branch out and promote to QA.

Go to Repo and Build page → Create branch from trunk.

From the Overview page, click on Deploy or Open newly created version.

Next, from the Life Cycles Management page, select the newly created version and promote to Testing.

Go back to Overview page to Open the promoted application in Testing environment.

As the QA person, lets try to find few bugs, create issue tickets and demote the application back to Development.

Move to Issues page and create a new Issue. Make sure to select newly created version(1.0.0) for reporting the issues.

Everyone can access this application, not secured.

No UI validation is done…

Date field has invalid value…

Go to Life Cycle Management page, and click on Demote. Select the issues we created above and demote with a comment such as Can not proceed to production with above bugs.

Let's assume we have fixed the above issues :)

Promote back to Testing.

QA promote to Production. As the DevOps, lets create new database and update runtime configs.