HOW-TO

A quick user guide

for an End2End PoC

Authors

Stefano Rossini (Head of Italy iCSD)

Angelo Muresu (Italy Devonfw Local Expert)

Jaime Diaz Gonzalez (Devonfw Core Team)

INDEX

1 Introduction

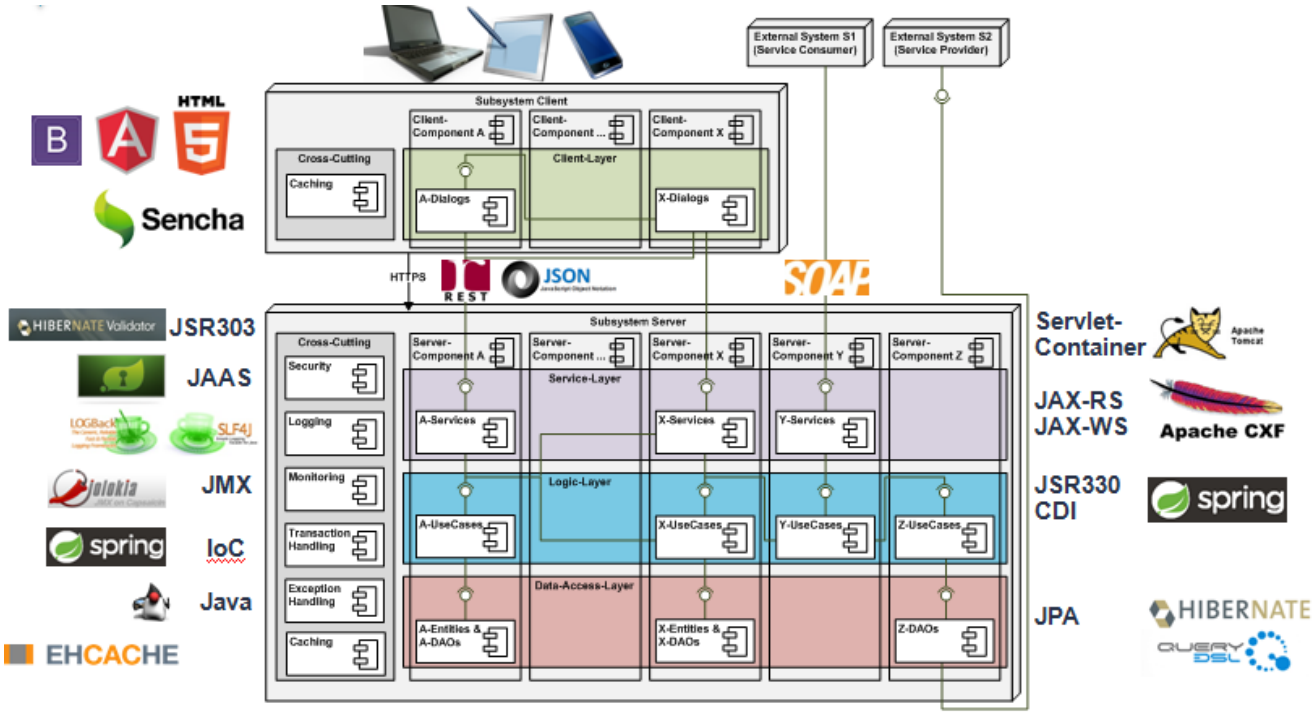
1.1 What's Devonfw

Capgemini Apps2 SBU uses the Java-based standard platform open source **Devonfw** as an industrialized approach to efficiently deliver CSD-projects to our customers. This platform is aimed to engagements where the client is flexible in the use of technology or uses outdated technology,
It can offer a modern technology approach using our experience as a group. The main idea is to not create a monolithic framework but to provide proven patterns.

Devonfw provides a solution to building applications which combine best-in-class frameworks and libraries as well as industry proven practices and code conventions. It massively speeds up development, reduces risks and helps you to deliver better results.

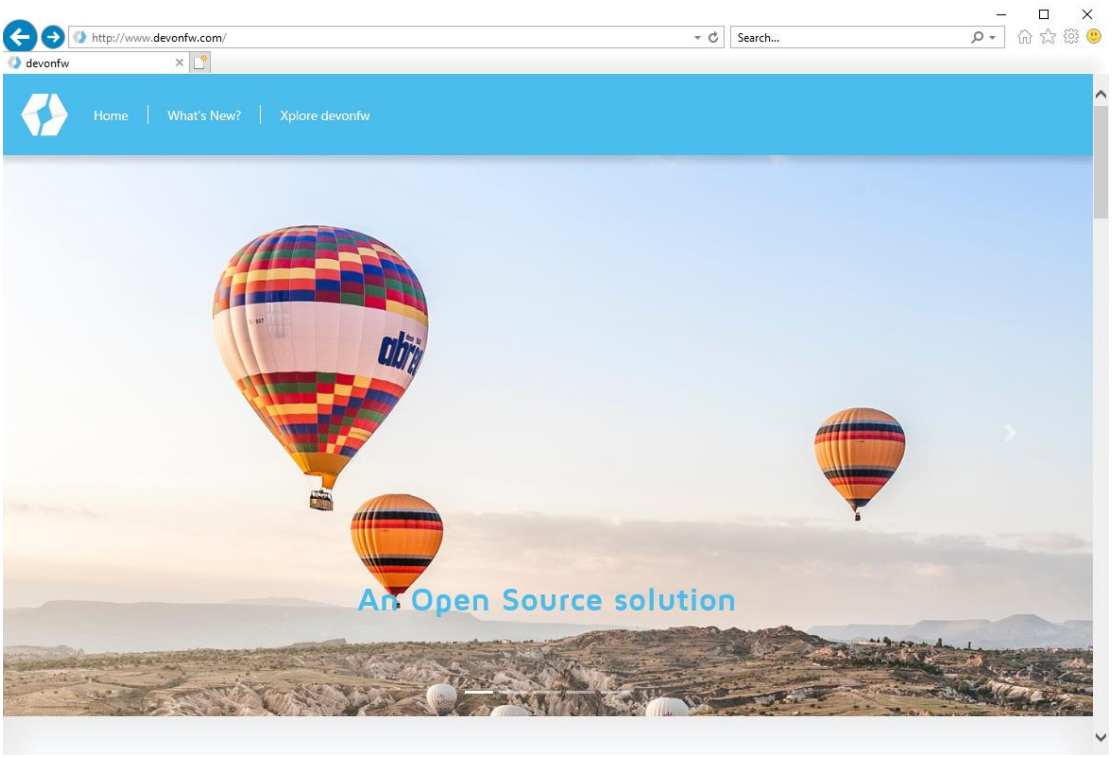
The current version of the platform is oriented to develop single-page web applications based on the Java EE programming model using the spring framework as the default implementation.

As any modern java application today, Devonfw is based on a large number of technologies and standards that build the software architecture. Devonfw defines how to use these technologies in a layered component-oriented architecture to solve all the technical aspects that make the business code work.

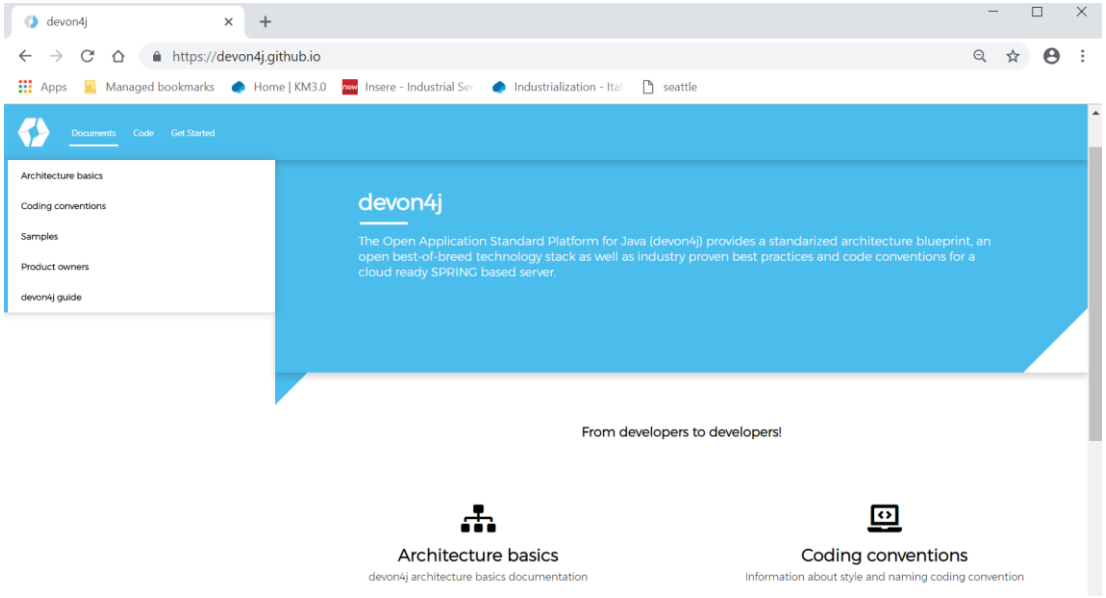


1.2 Devonfw Web site URL

The Devonfw site is available here: http://www.devonfw.com/

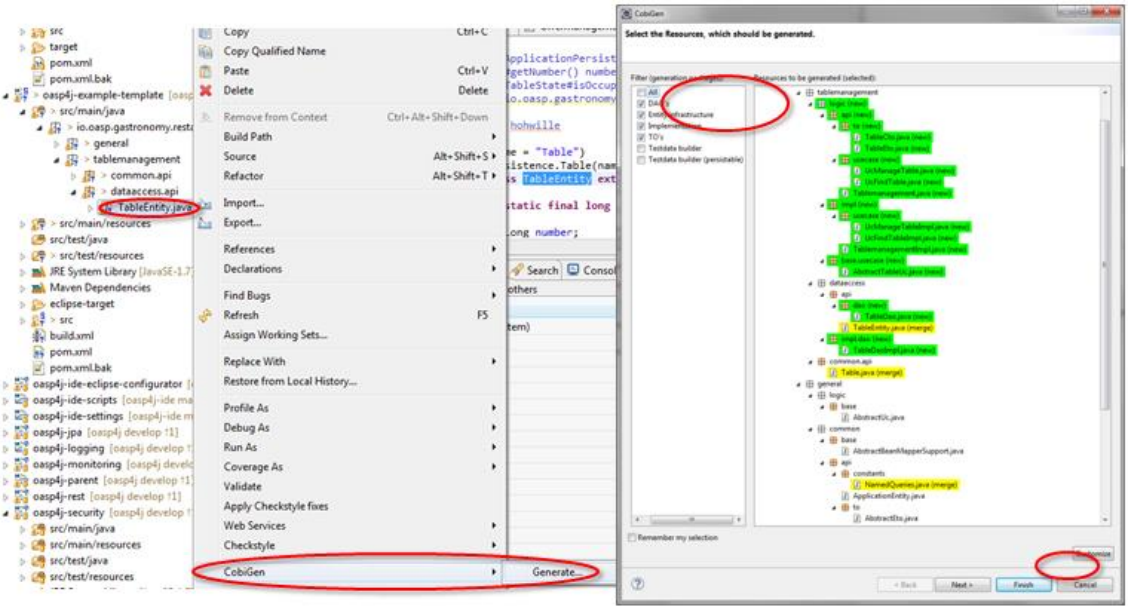


The Devonfw for JAVA is Devon4j and you can find everything here:
http://www.devonfw.com/devon4j/ and https://devon4j.github.io/



1.3 What's CobiGen

CobiGen is a high value asset that is used by **devonfw** projects to generate code across all layers of a devon-application, including the clients. It works iteratively without leaving marks or regions in the code due to its basic understanding of Java. Due to architecture patterns set in devonfw, the generator is able to support generation of higher-level concepts than just - class. It is best integrated into the provided eclipse package.



Cobigen benefits summary:

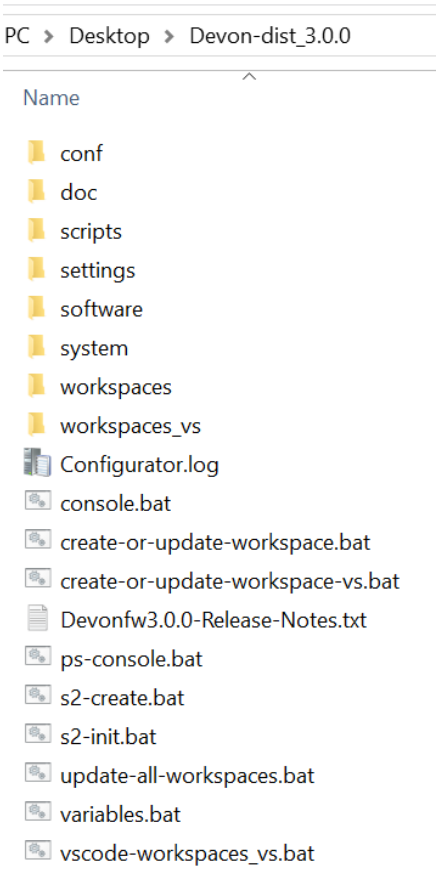
1. Can generate a whole **CRUD** application from a **single Entity class.**
2. You can save the effort for creating simple CRUD use case since CobiGen generates: **DAO**s, **DTOs**, Spring services and REST and SOAP services and even the client UI application (Angular and Ionic).
3. Agility: Boost development-speed, reduce cost, industrialize.
4. Innovation: always evolving, keep the fast pace of technology and incorporate new trends that add value for the engagements.
5. Security: Rest assured, devonfw follows best practices to secure your applications.
6. DevOps-ready: Supports development and operations with proven solutions for continuous integration, deployment and delivery.

1.4 HOW TO install Devonfw

In order to install Devonfw please make the following steps:

1. Download from http://de-mucevolve02/files/devonfw/current/ and extract the Devonfw distribution in your computer
2. **IMPORTANT:** If any, download the latest accumulative patch identified by the date in the previous link. The name of the patch file looks like this: *win_accumulative_patch_YYYYMMDD.zip*.
    a. Copy the zip in the **root** devonfw distribution folder on your machine.
    b. Extract the files and choose "overwrite all".
    c. Sometimes the patch zip contains a .bat file like *win_accumulative_patch_YYYYMMDD.bat*. If so, please execute it.

i.e: see the image below about the Devon distribution 3.0.0:



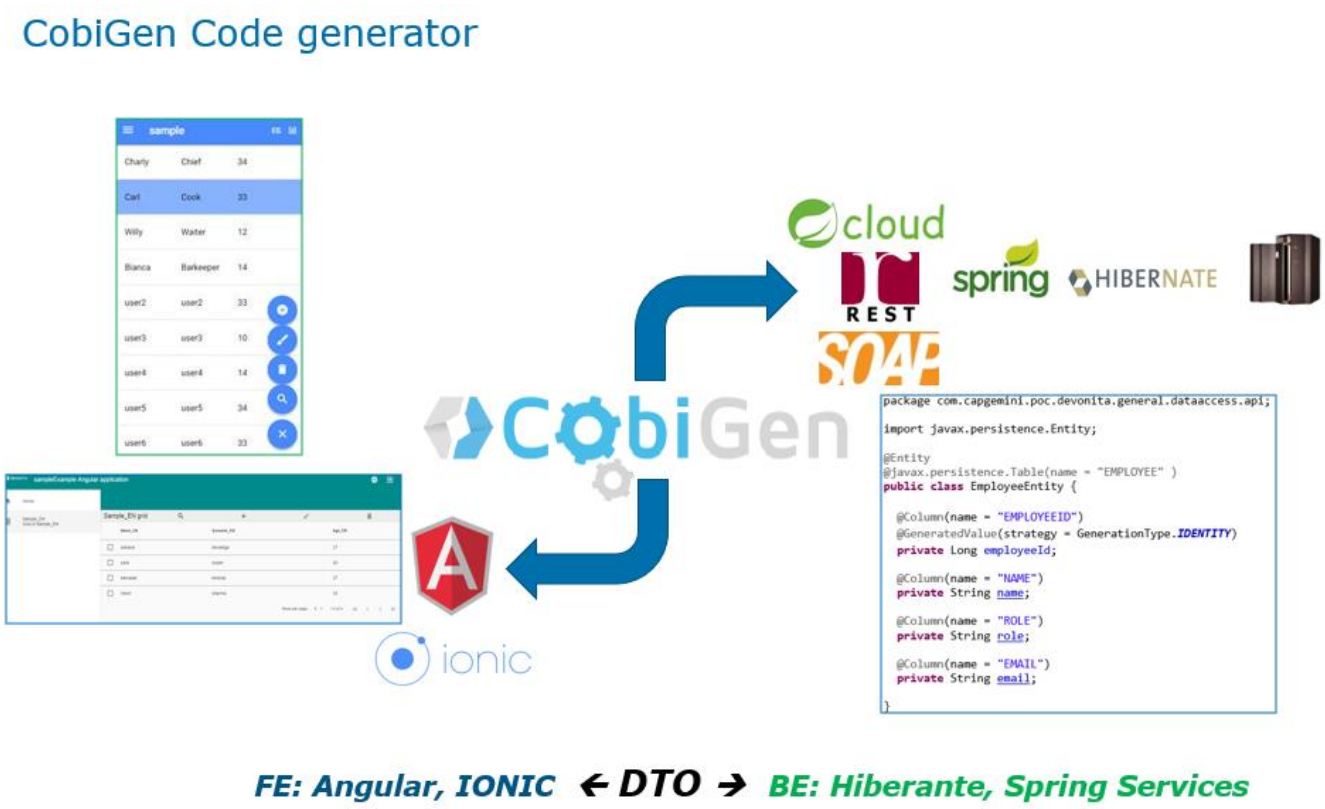
Since We're going to develop an E2E application (Mobile app + Web FE and BE services/DB) remember the following pre-conditions:

- Java
- Node and npm (https://nodejs.org/dist/v10.7.0/node-v10.7.0-x64.msi)
- Capacitor (https://capacitor.ionicframework.com/docs/getting-started/)
    - `npm install -g @capacitor/core @capacitor/cli`

2 Steps to create a Sample UI Angular4 Project through Cobigen

The HOW_TO is divided in 2 parts:

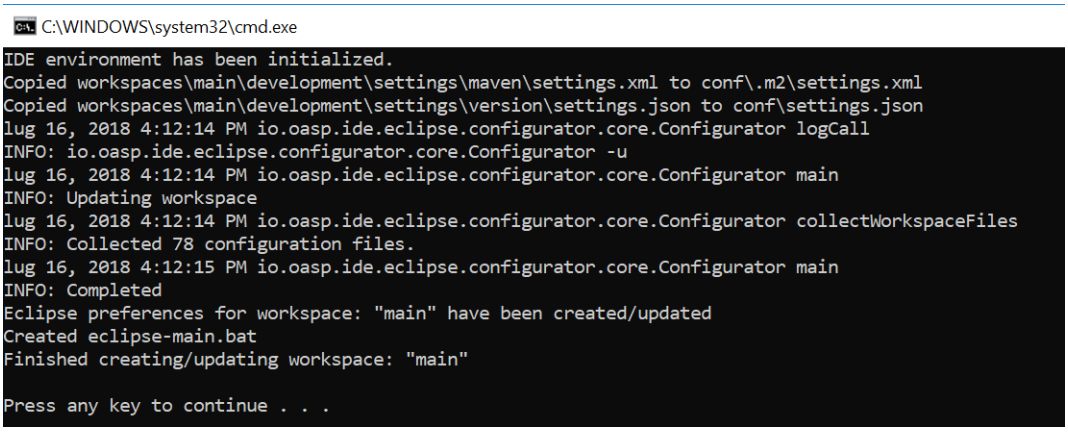
1. BE (DB + DAO + services)
2. FE (Web App Angular + Ionic App)



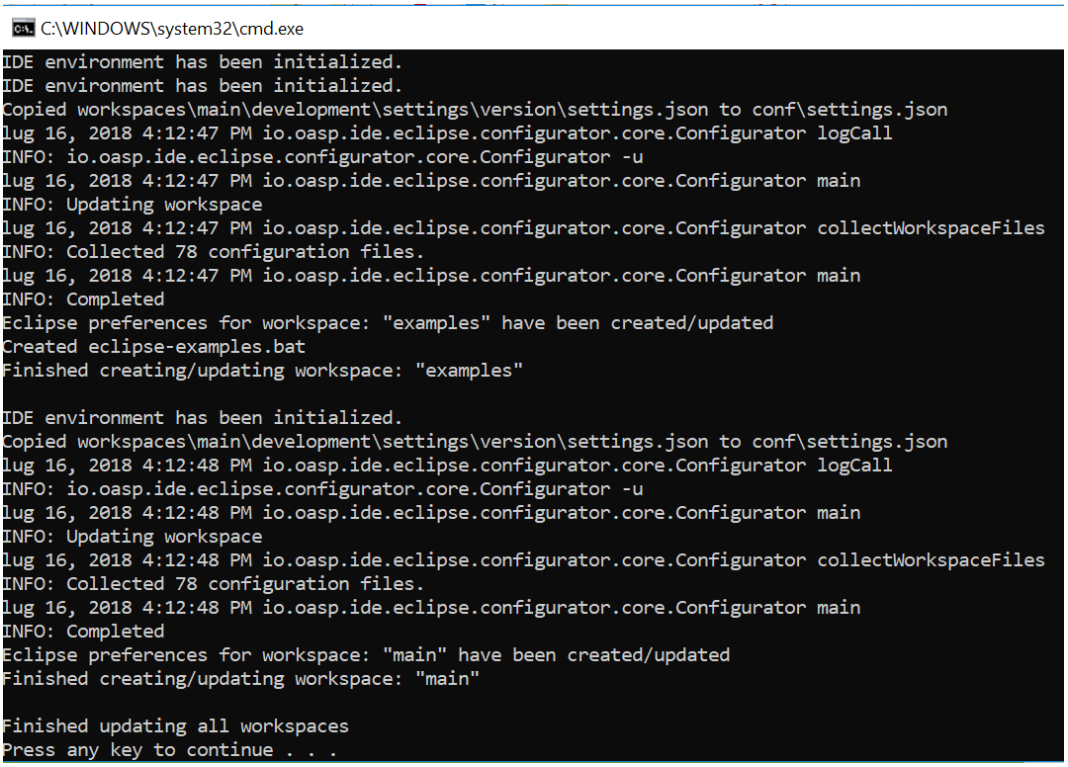
So, ready to go ! We're going to start from the BE part …

2.1 Back End (Services, DTO, DAO, DB)

1. Run the **create-or-update-workspace.bat** present within the Devon-dist_N.n.m folder (i.e: Devon-dist_3.0.0)



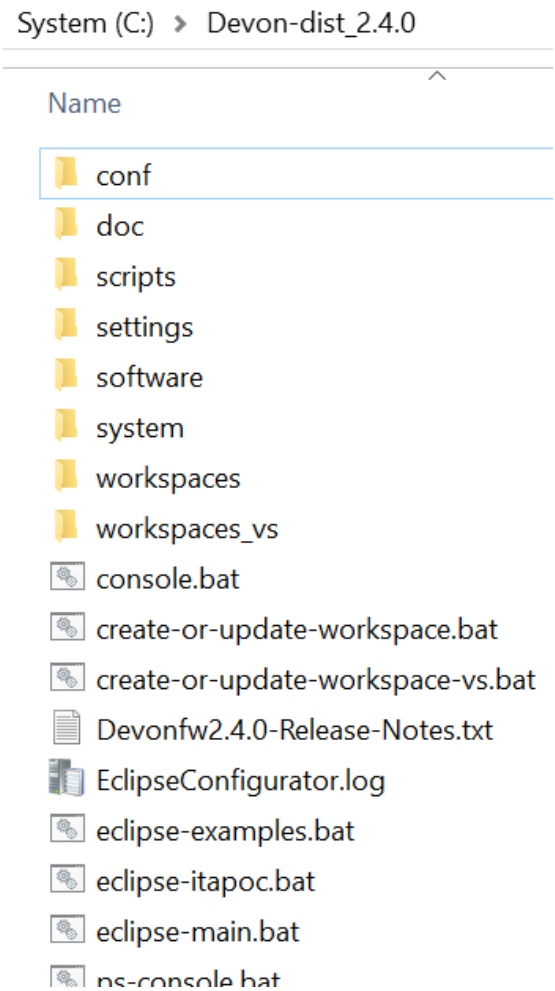
2. Run the **update-all-workspaces.bat**, again present within the Devon-dist_N.n.m folder of your path folder location (i.e: Devon-dist_2.4.0)



3. **Copy** the Devon-dist_3.0.0\workspaces\examples**jwtsample** directory in a new dir (i.e: for thisPoC WE will use Devon-dist_3.0.0\workspaces**itapoc)**

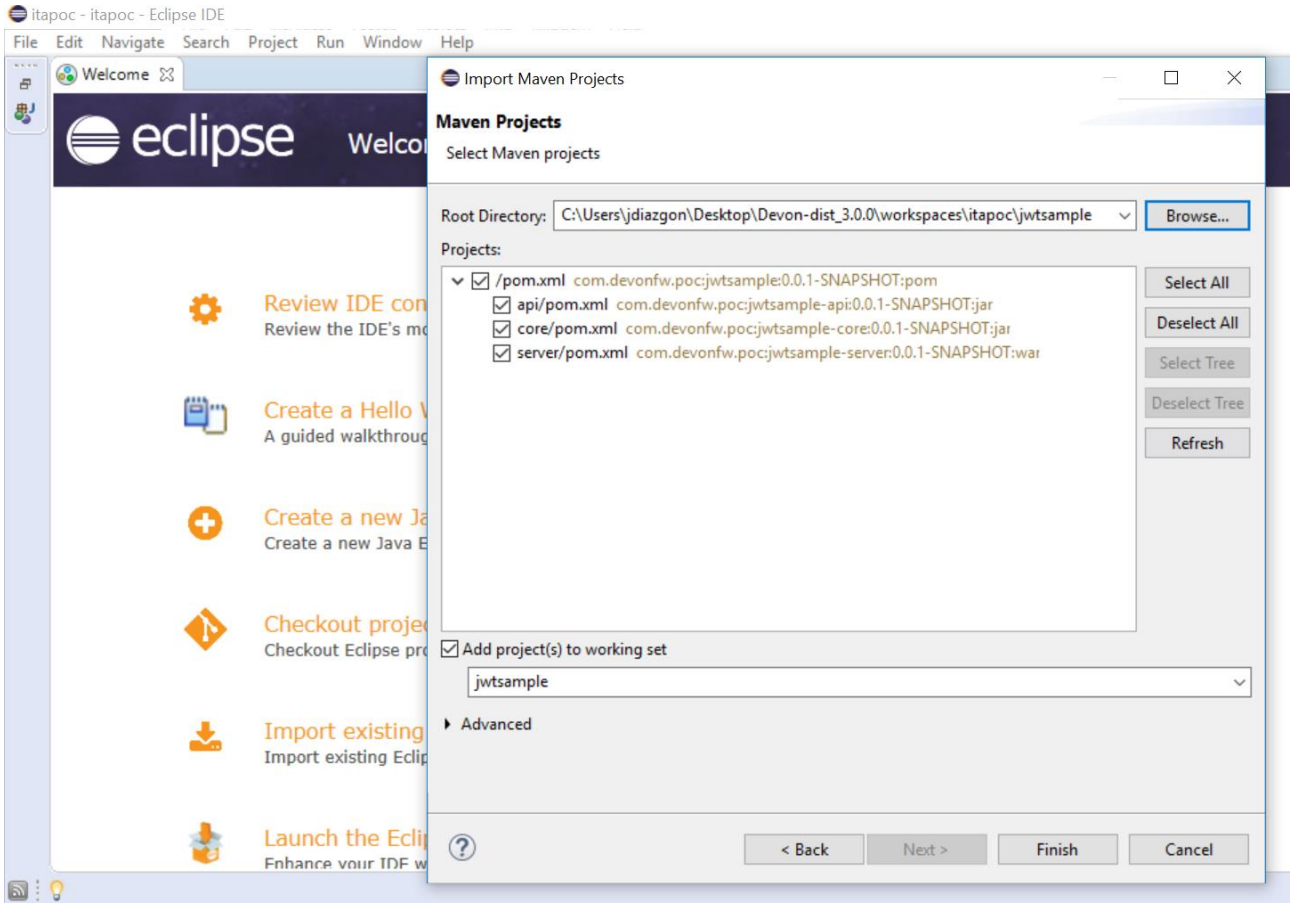
4. Now run **update-all-workspaces.bat**

5. Execute **eclipse-<<directory name>>.bat** from Devon-dist_N.m.m folder (i.e: eclipse-**itapoc**.bat)



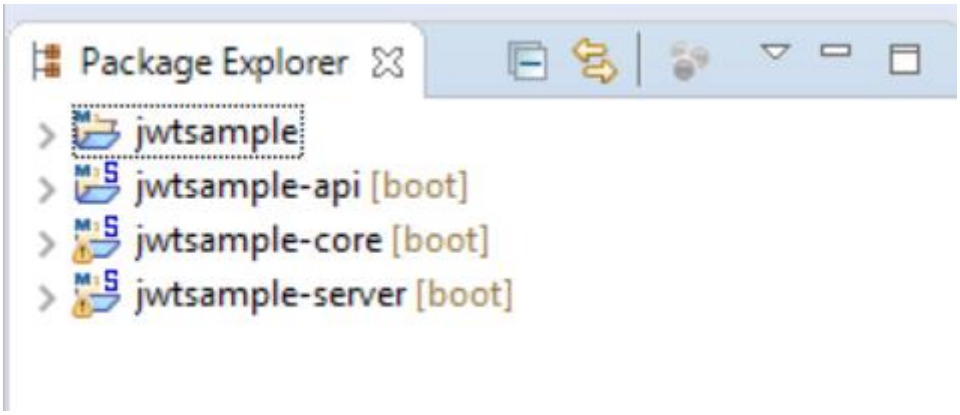
It will open eclipse for you

6. Now import the Projects pointing to the Devon-dist_3.0.0\workspaces**itapoc**\jwtsample directory



Click **FINISH**

Now We have the following 4 projects.

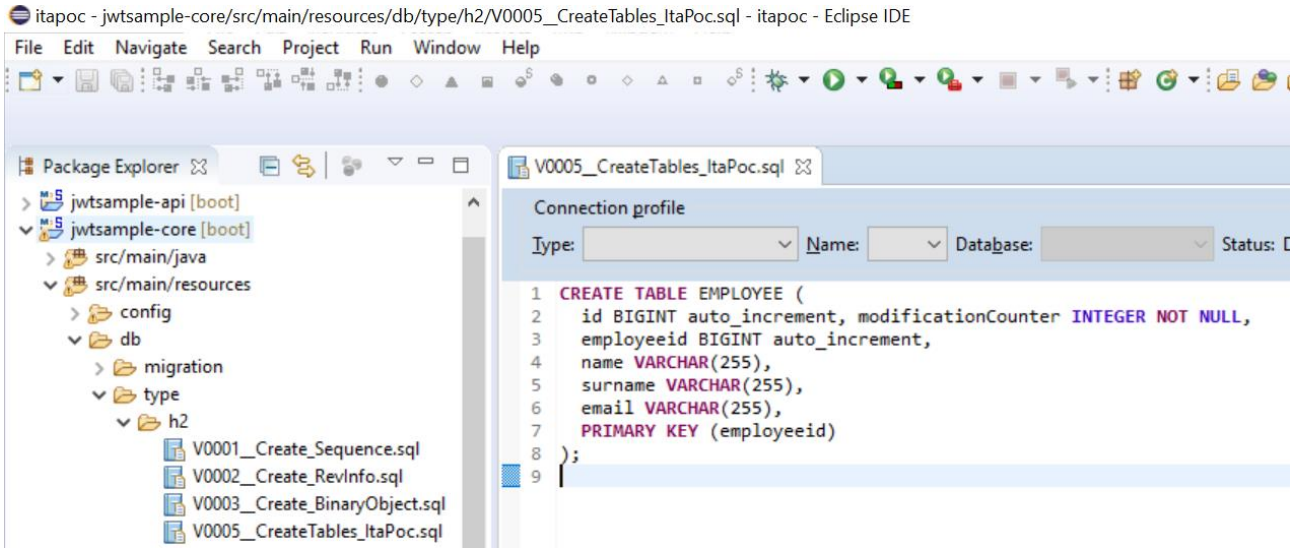


BEFORE to start to create an Entity class, remember to create the tables !

7. Create a new **SQL file** (i.e: V0005__CreateTables_ItaPoc.sql) inside *jwtsample*-core and insert the following script:

```
CREATE TABLE EMPLOYEE (
  id BIGINT auto_increment, modificationCounter INTEGER NOT NULL,
  employeeid BIGINT auto_increment,
  name VARCHAR(255),
  surname VARCHAR(255),
  email VARCHAR(255),
  PRIMARY KEY (employeeid)
);
```

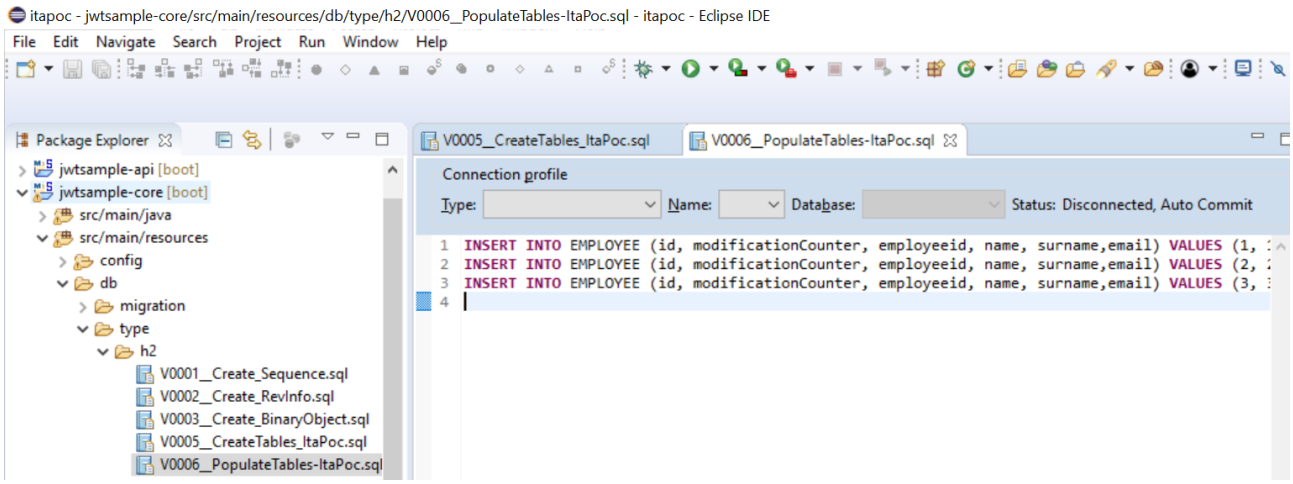
WARNING: please note that there are 2 underscore in the name !



8. Now create another SQL file (i.e: V0006__PopulateTables-ItaPoc.sql) and add following script about the INSERT in order to populate the table created before

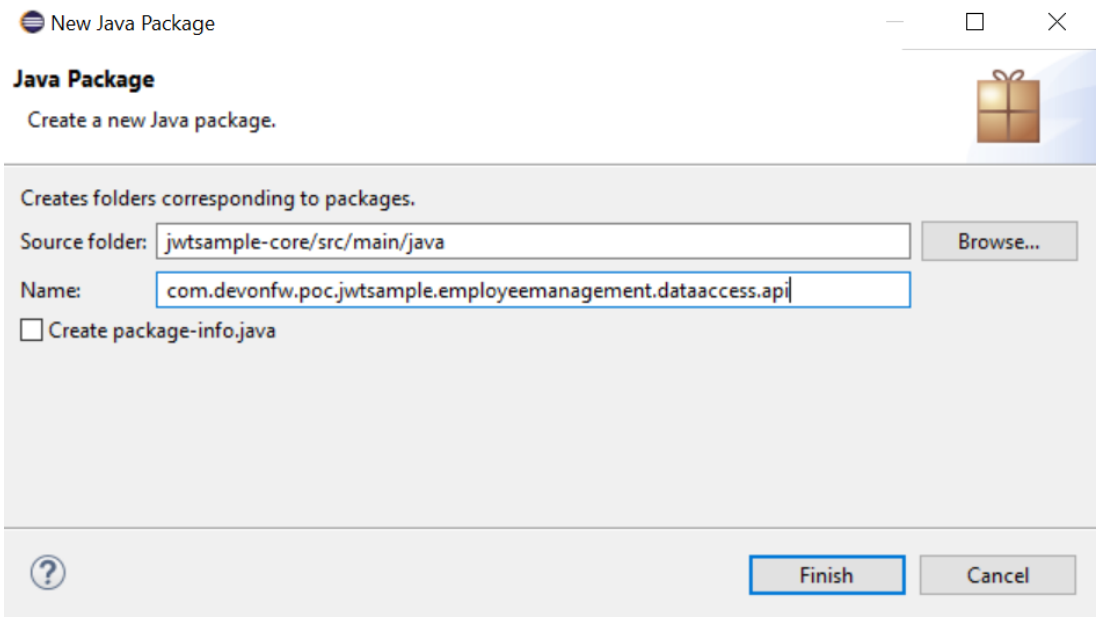
WARNING: please note that there are 2 underscore in the name !

```
INSERT INTO EMPLOYEE (id, modificationCounter, employeeid, name, surname,email) VALUES
(1, 1, 1, 'Stefano','Rossini','stefano.rossini@capgemini.com');
INSERT INTO EMPLOYEE (id, modificationCounter, employeeid, name, surname,email) VALUES
(2, 2, 2, 'Angelo','Muresu', 'angelo.muresu@capgemini.com');
INSERT INTO EMPLOYEE (id, modificationCounter, employeeid, name, surname,email) VALUES
(3, 3, 3, 'Jaime','Gonzalez', 'jaime.diaz-gonzalez@capgemini.com');
```

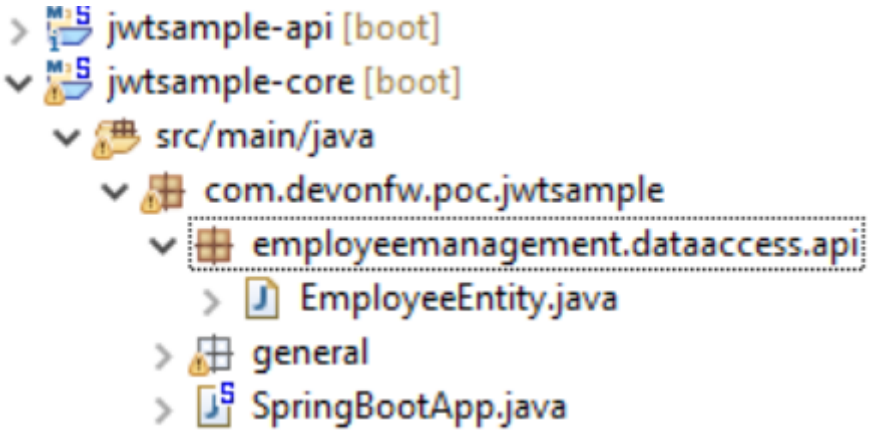


NOW you can create the Entity class 😉

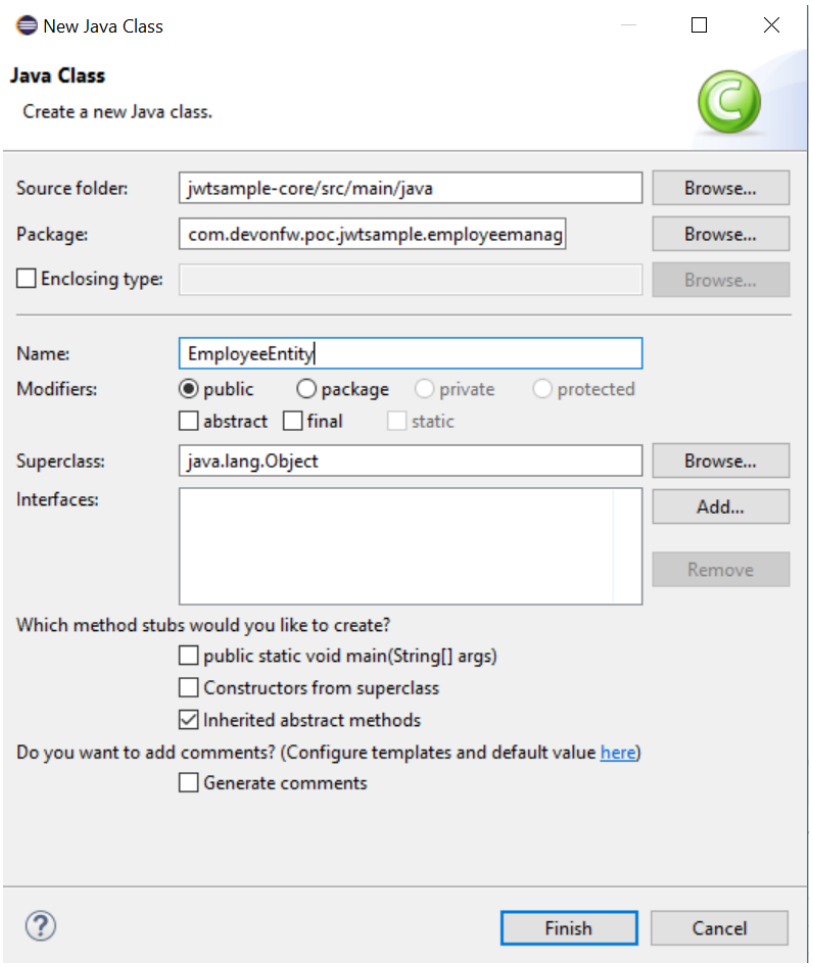
9. First of all create a package under the folder “jwtsample-core”



10. Now under this new package created



create a class (**Hibernate entity !**)

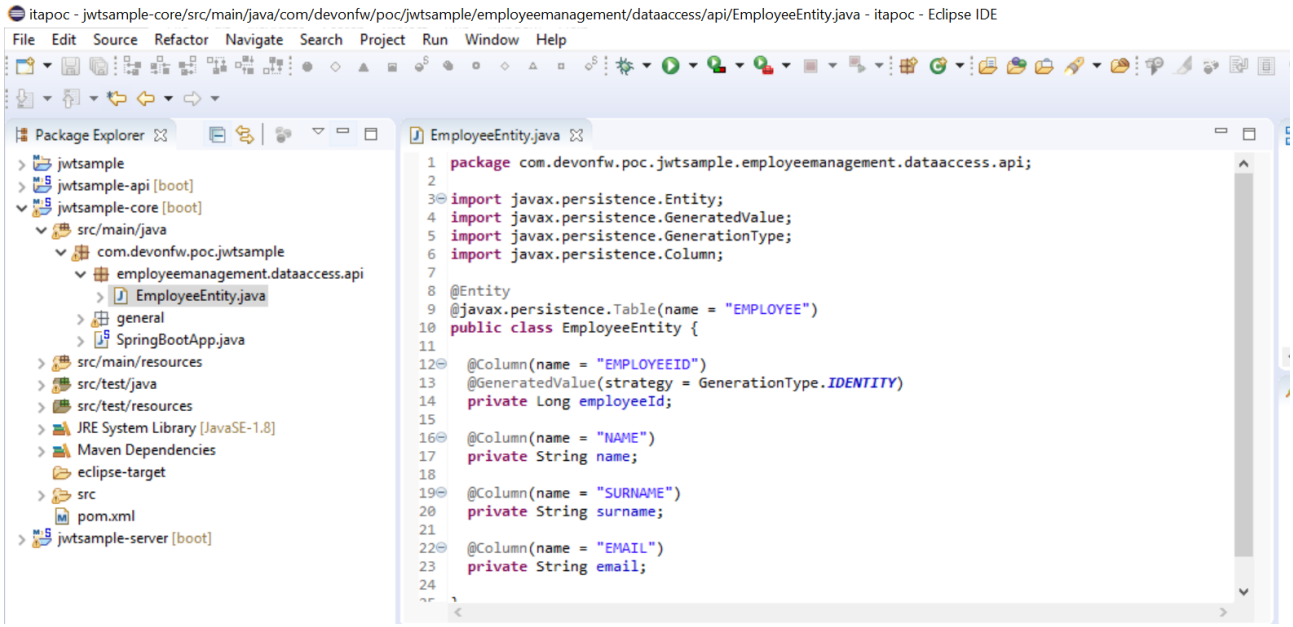


Now you can add:

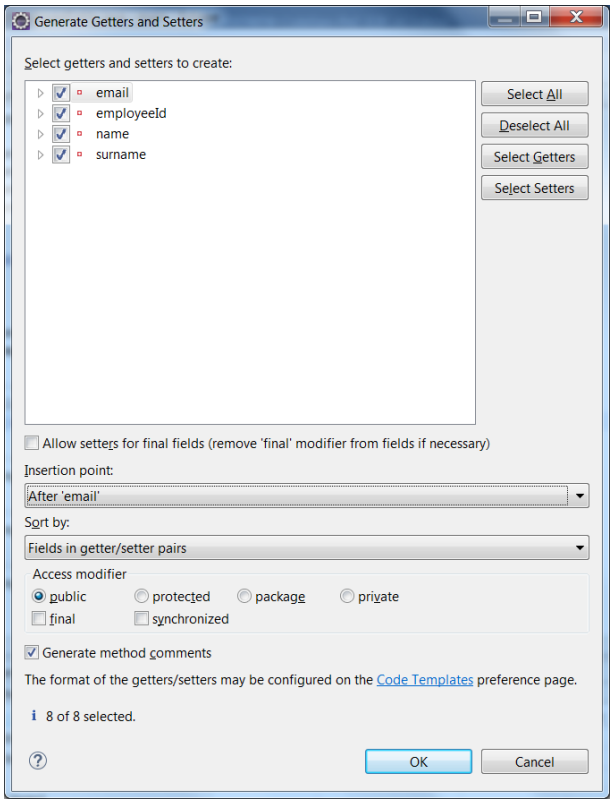
- the TABLE NAME
- the attributes (i.e: name,surname,email etc…)

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Column;

@Entity
@javax.persistence.Table(name = "EMPLOYEE")
public class EmployeeEntity {

  @Column(name = "EMPLOYEEID")
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long employeeId;

  @Column(name = "NAME")
  private String name;

  @Column(name = "SURNAME")
  private String surname;
```

```
  @Column(name = "EMAIL")
  private String email;

}
```

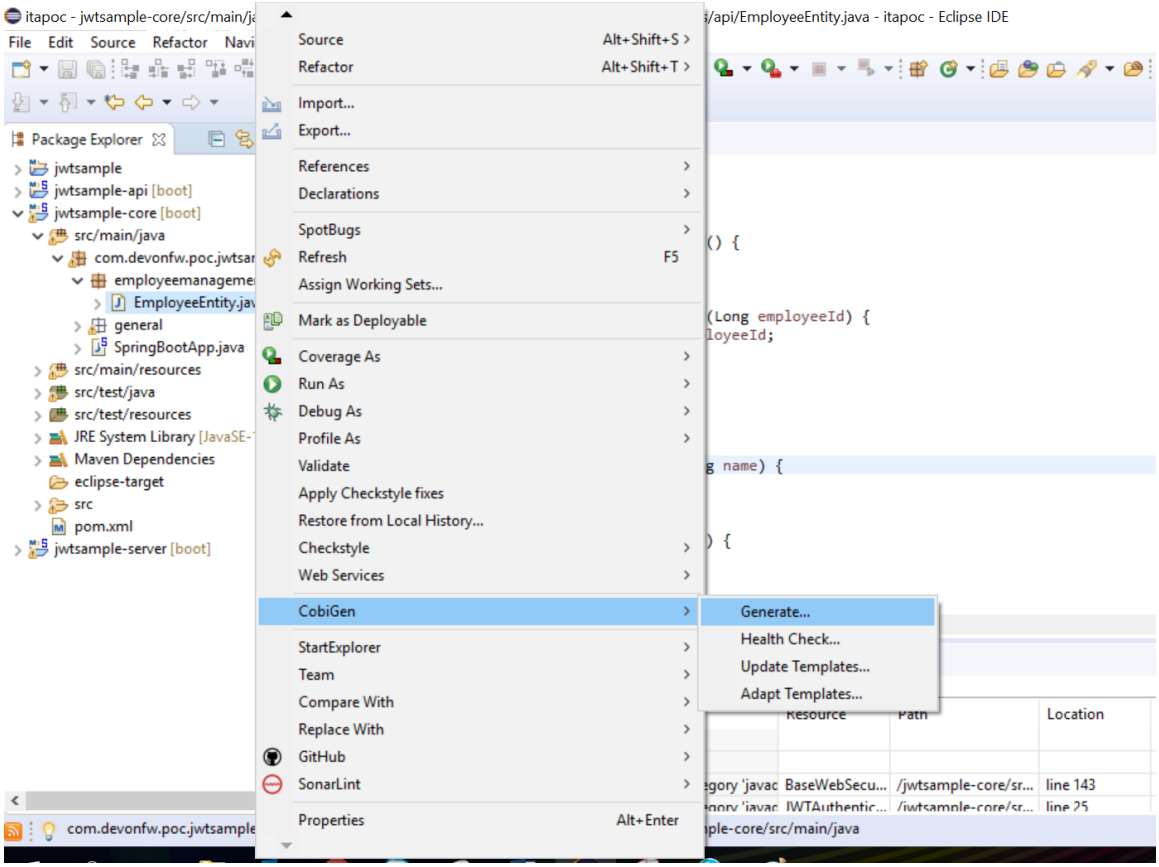


and then generate **getters** and **setters** for all attributes …

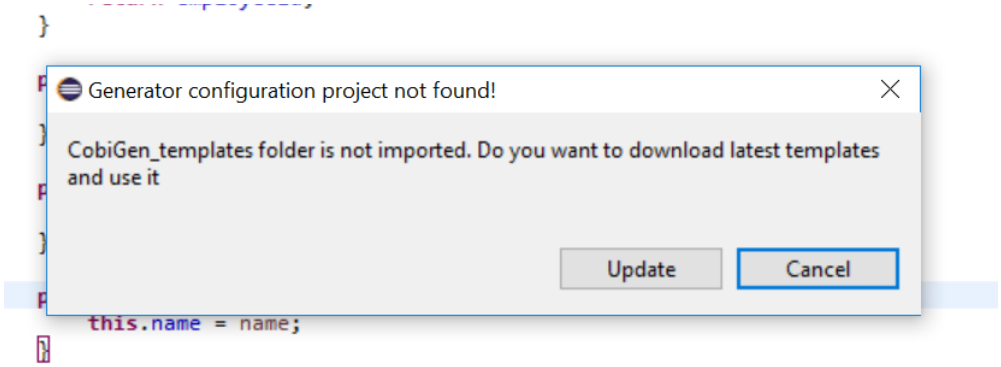


11. Once Getter and Setter are done, we can now **use CobiGen** (do it if u need to use code generator !)

Right click on Entity (**EmployeeEntity**.java file) and click on Generate Cobigen.

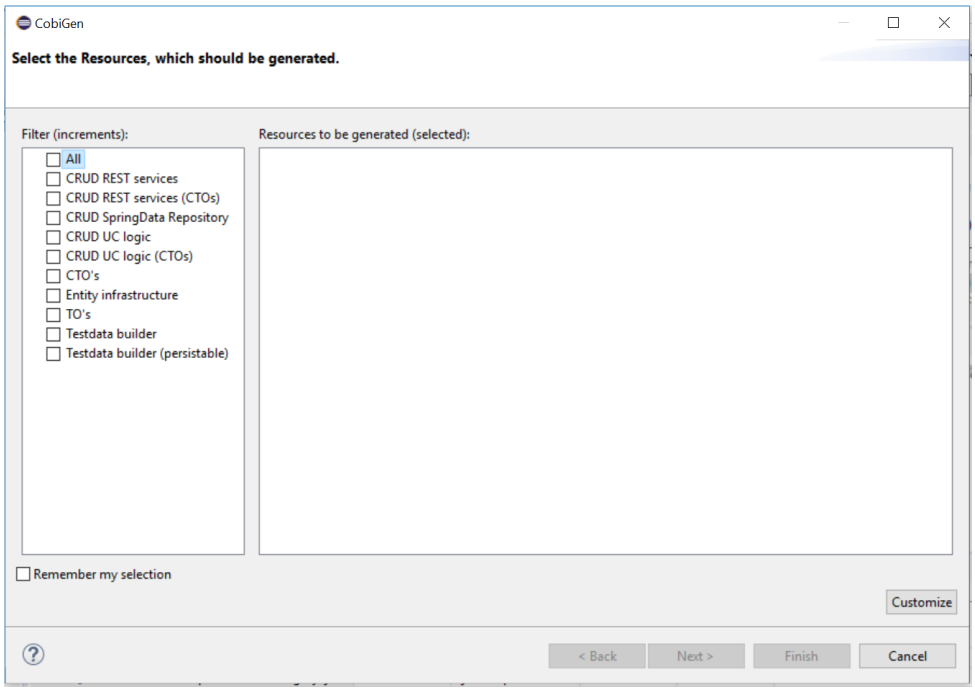


It will ask you to download the templates, click on *update*:

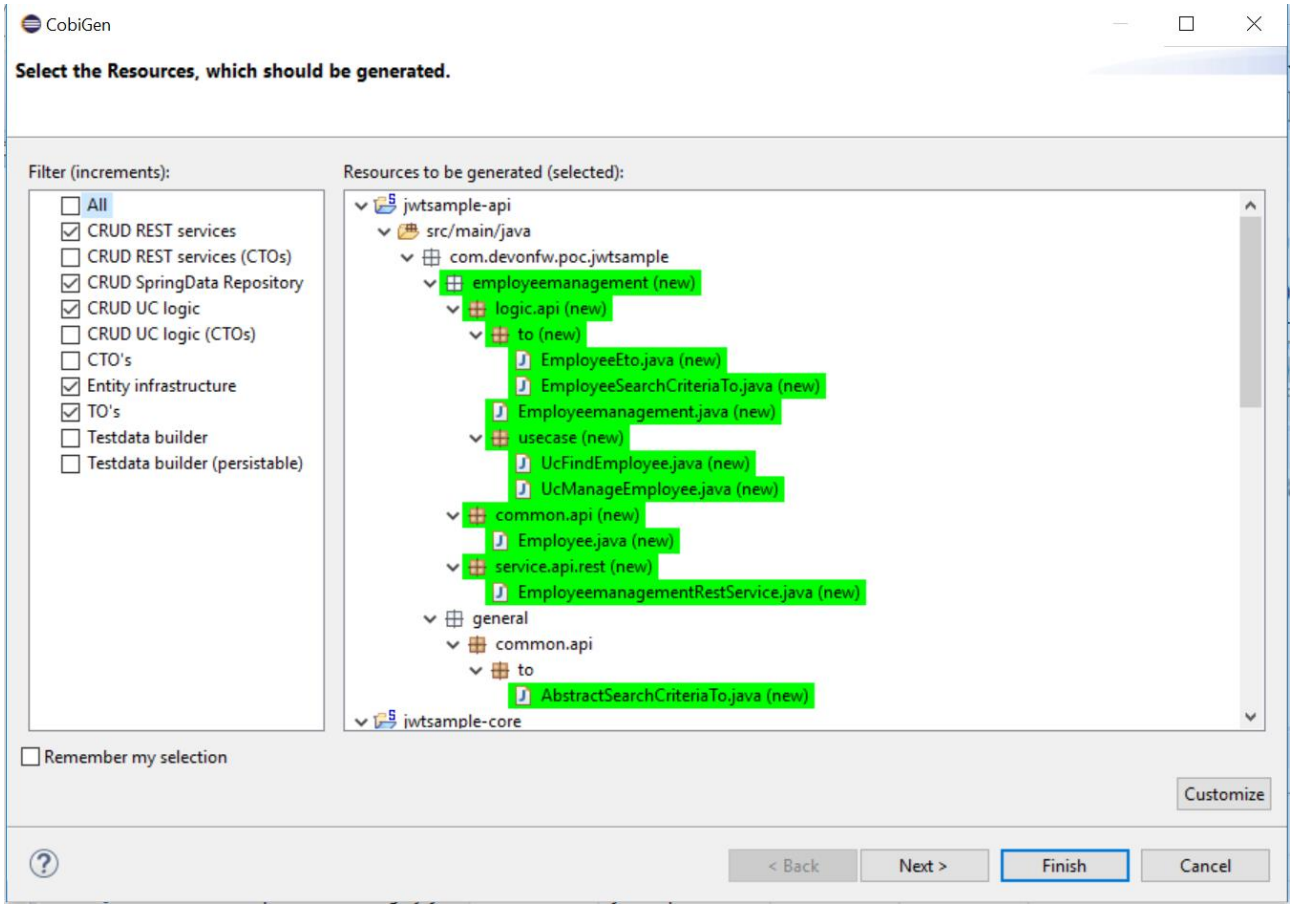


It will automatically download the latest version of *CobiGen_Templates*. After that, it will show you the next window:

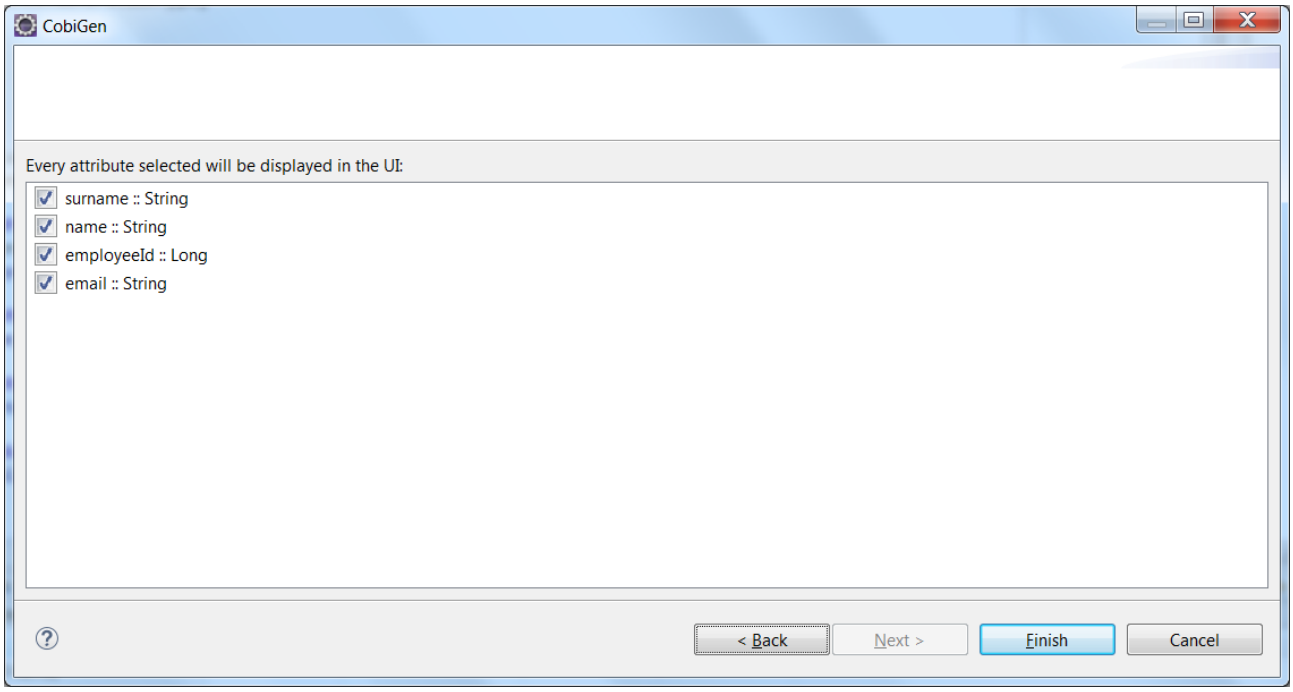
Attention: If you want to adapt the CobiGen_Templates, (normally this is not neccessary), you will find at the end of this document a tutorial on how to import them and adapt them!



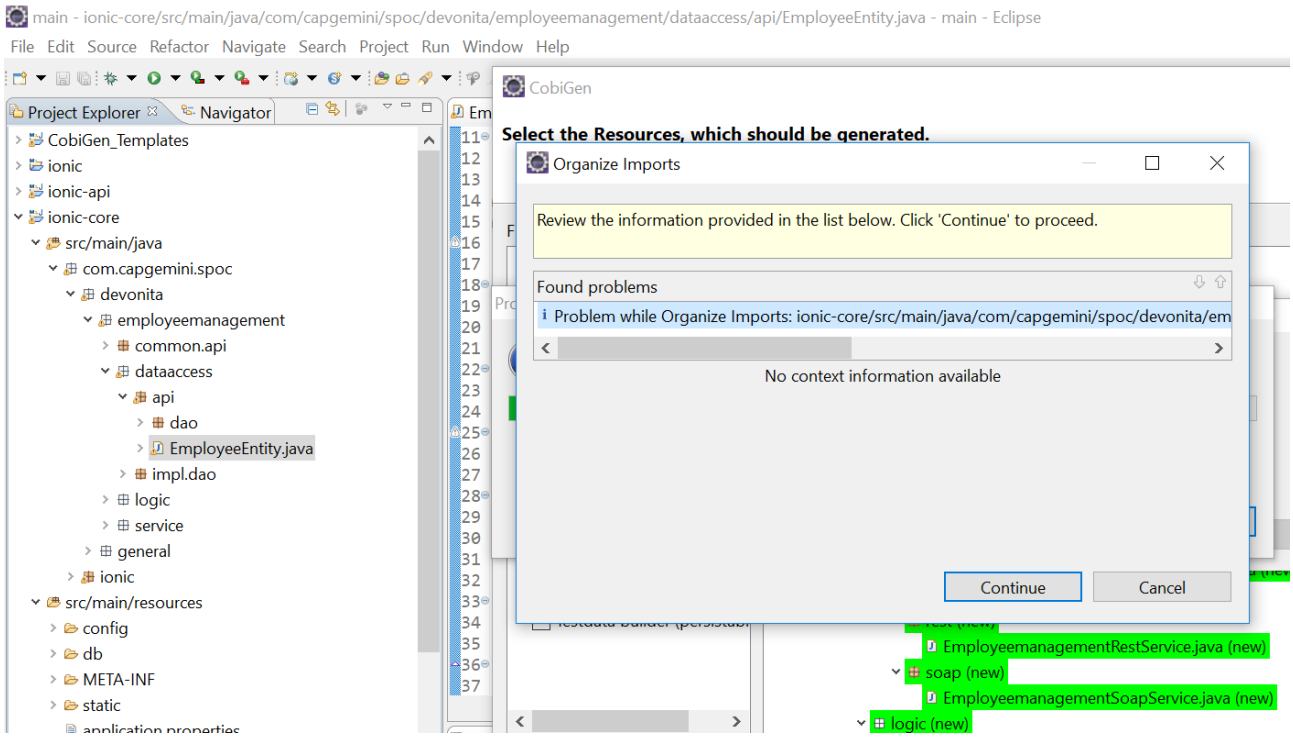
12. Click on all the option selected as below:



13. Click on Next

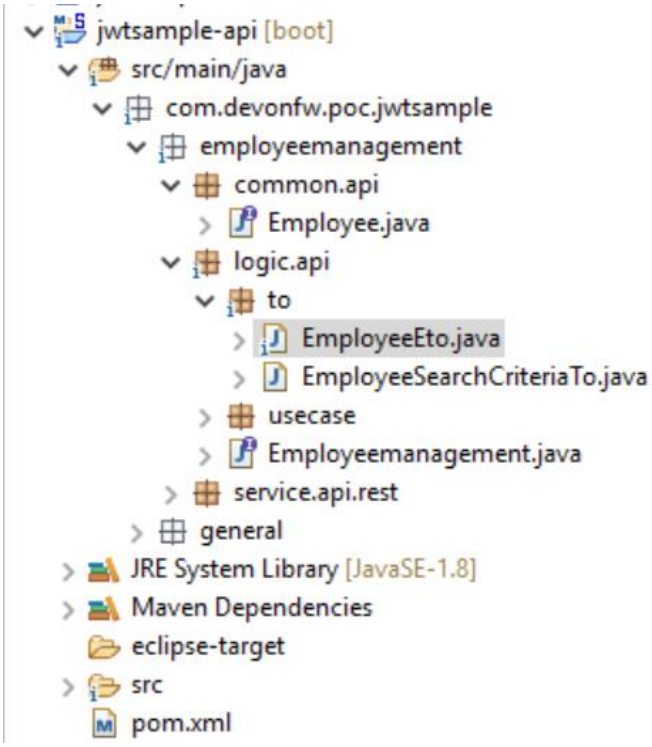


14. Click on finish. Below Screen would be seen. Click on continue

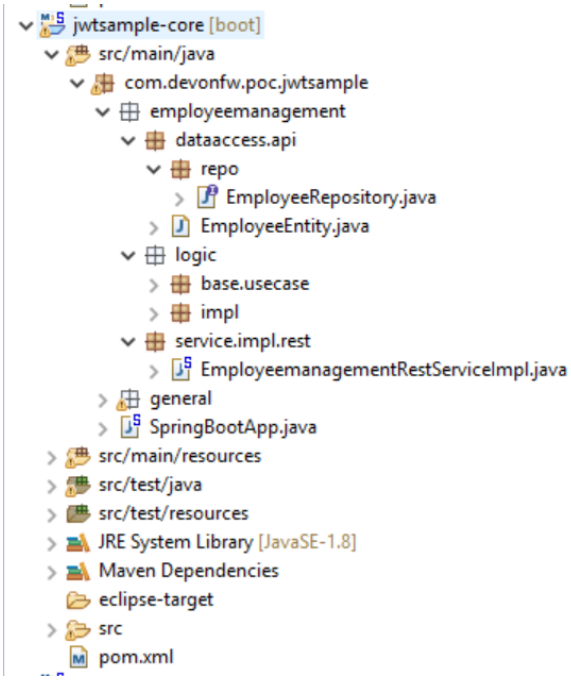


The entire <u>BE layer</u> structure having CRUD operation methods will be auto generated.

Some classes will be generated on the api part (*jwtsample-api)*, normally it will be interfaces, as shown below:



Some other classes will be generated on the core part (*jwtsample-core)*, normally it will be implementations as shown below:



AFTER CobiGen, you will see that the entity previously developed as POJO now has “changed” since it **inherit** from a Devonfw base class: **ApplicationPersistenceEntity**

From:

```
public class EmployeeEntity {
```

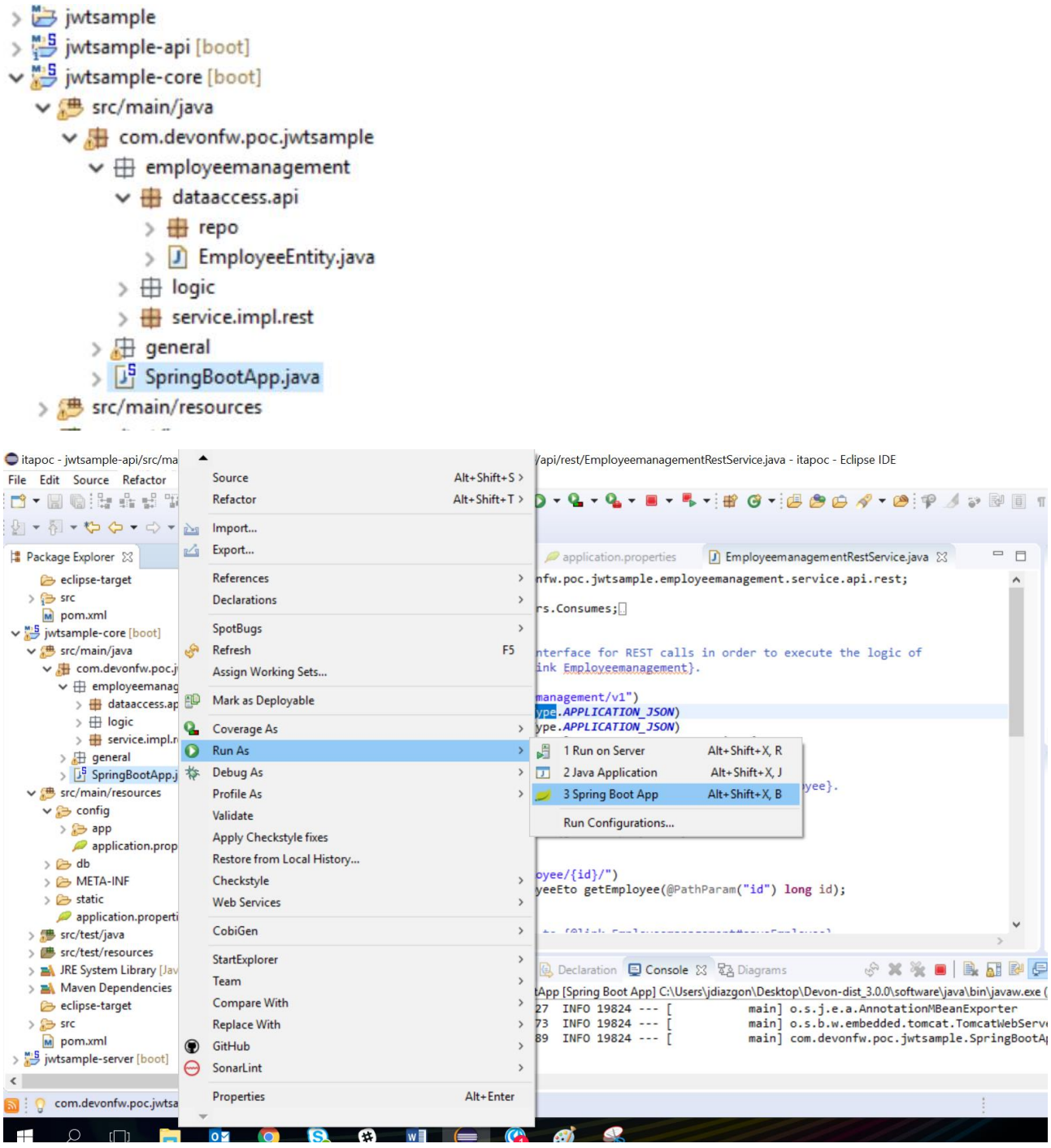
to:

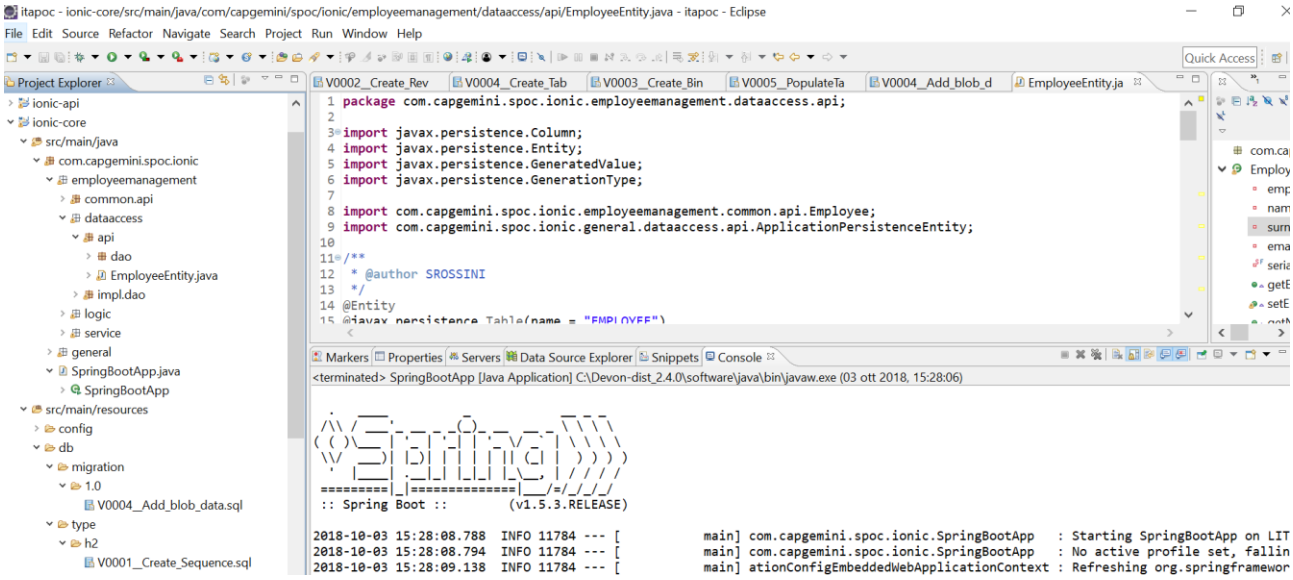
```
public class EmployeeEntity extends ApplicationPersistenceEntity implements Employee {
```

```
@Entity
@javax.persistence.Table(name = "EMPLOYEE")
public class EmployeeEntity extends ApplicationPersistenceEntity implements Employee {

  @Column(name = "EMPLOYEEID")
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long employeeId;

  @Column(name = "NAME")
  private String name;

  @Column(name = "SURNAME")
  private String surname;

  @Column(name = "EMAIL")
  private String email;

  private static final long serialVersionUID = 1L;

  /**
   * @return employeeId
   */
  public Long getEmployeeId() {

    return this.employeeId;
  }
```

BEFORE to generate the FE, please start the Tomcat server to check that BE Layer has been generated properly.

To start a server you just have to right click on "*SpringBootApp.java*" -> *run as* -> *Spring Boot app*

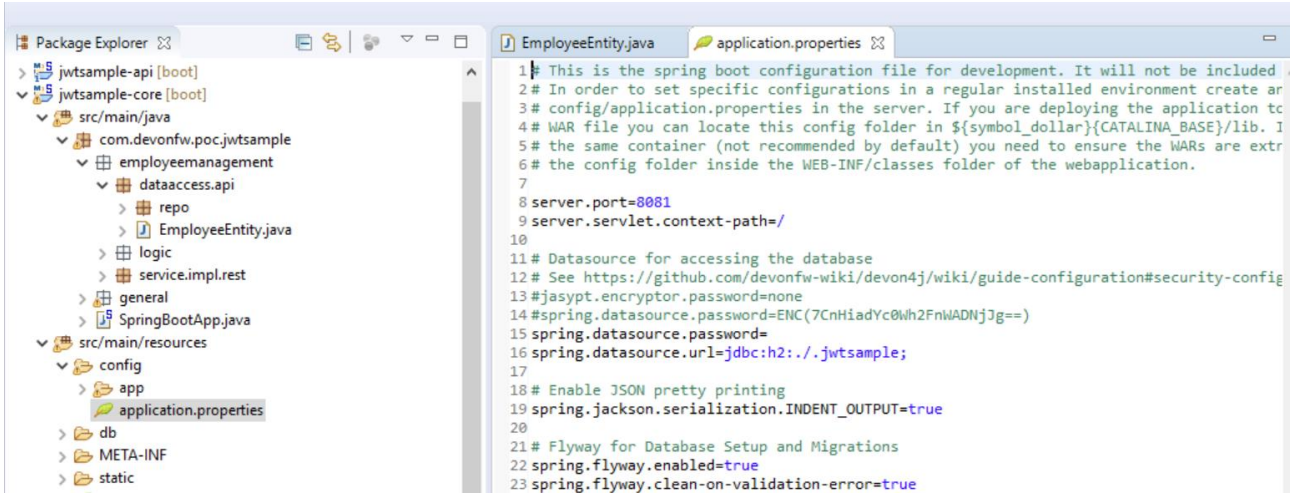

jwtsample
jwtsample-api [boot]
jwtsample-core [boot]
src/main/java
com.devonfw.poc.jwtsample
employeemanagement
dataaccess.api
repo
EmployeeEntity.java
logic
service.impl.rest
general
SpringBootApp.java
src/main/resources
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
References
Declarations
SpotBugs
Refresh F5
Assign Working Sets...
Mark as Deployable
Coverage As
Run As
Debug As
Profile As
Validate
Apply Checkstyle fixes
Restore from Local History...
Checkstyle
Web Services
CobiGen
StartExplorer
Team
Compare With
Replace With
GitHub
SonarLint
Properties Alt+Enter
1 Run on Server Alt+Shift+X, R
2 Java Application Alt+Shift+X, J
3 Spring Boot App Alt+Shift+X, B
Run Configurations...



BE DONE

Last but not least: We make a quick REST services test !

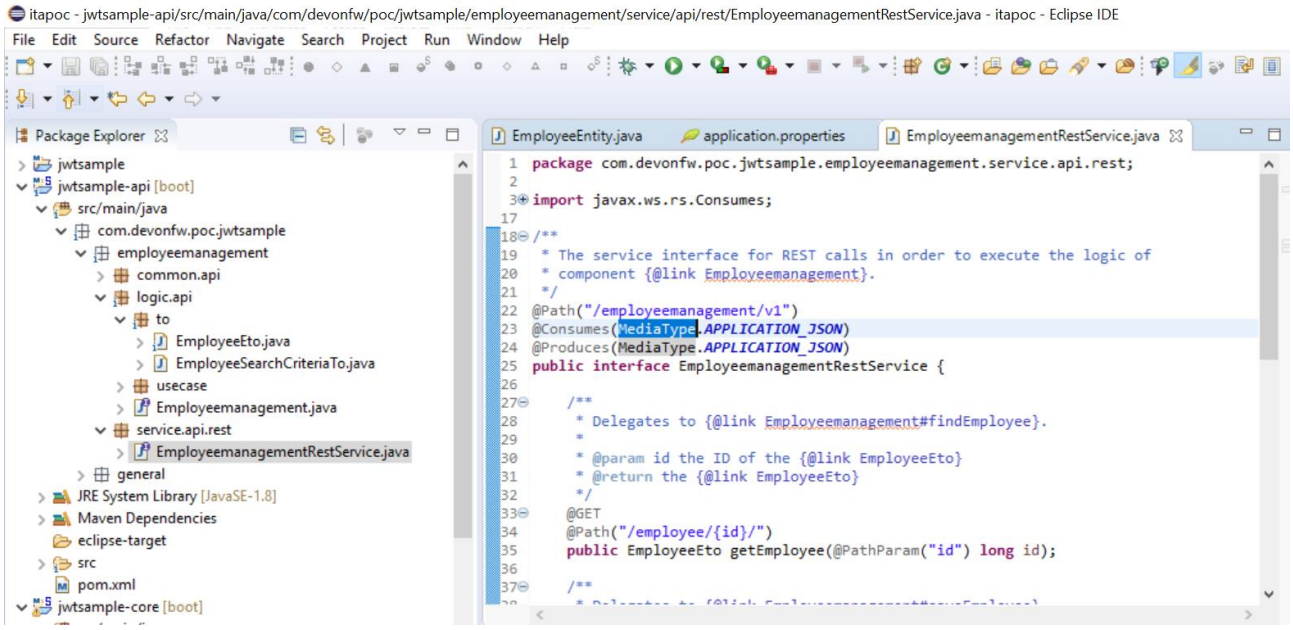
See in the application.properties the TCP Port and the PATH



Now compose the Rest service URL:

http://<server>/<app>/services/rest/<rest service class path>/<service method path>

- <server> refers to server with port no. (ie: localhost:8081)
- <app> is in the application.propeeties (empty in our case, see above)
- <rest service class path> refers to EmployeemanagementRestService: (i.e: /employeemanagement/v1)
- <service method path>/employee/{id} (i.e: for getEmployee method)

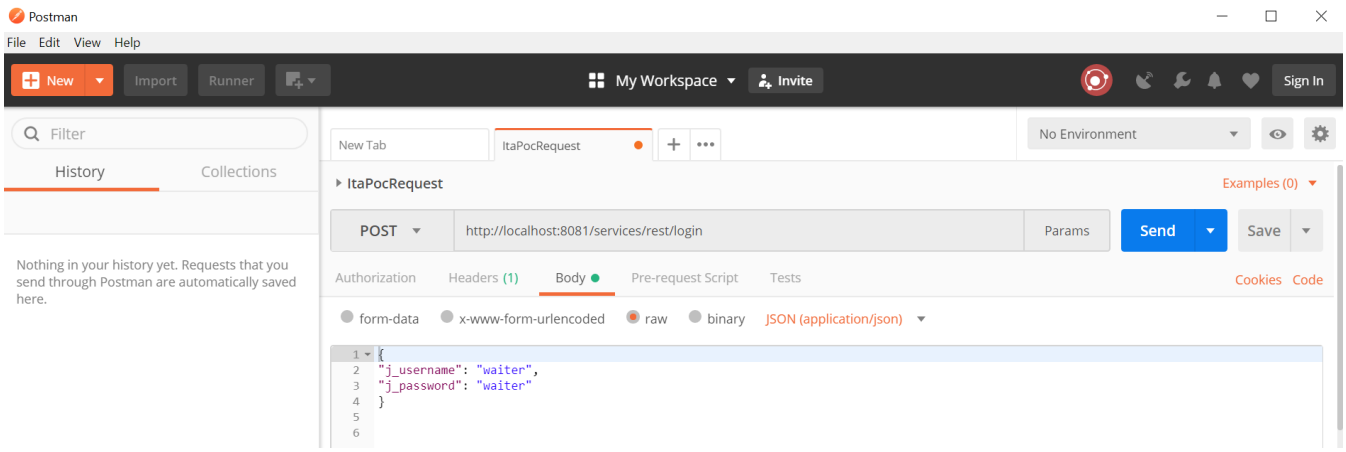


URL of getEmployee for this example is:

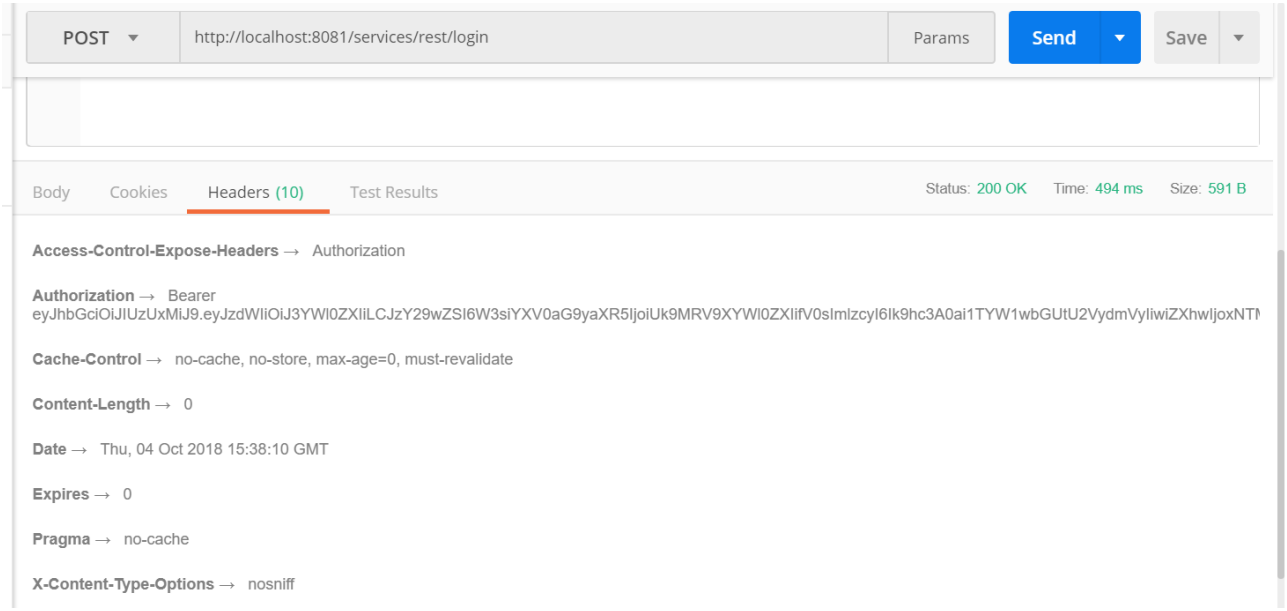
http://localhost:8081/services/rest/employeemanagement/v1/employee/search (for all employees)
http://localhost:8081/services/rest/employeemanagement/v1/employee/1 (for the specific employee)

Now download Postman tool [https://www.getpostman.com/apps]

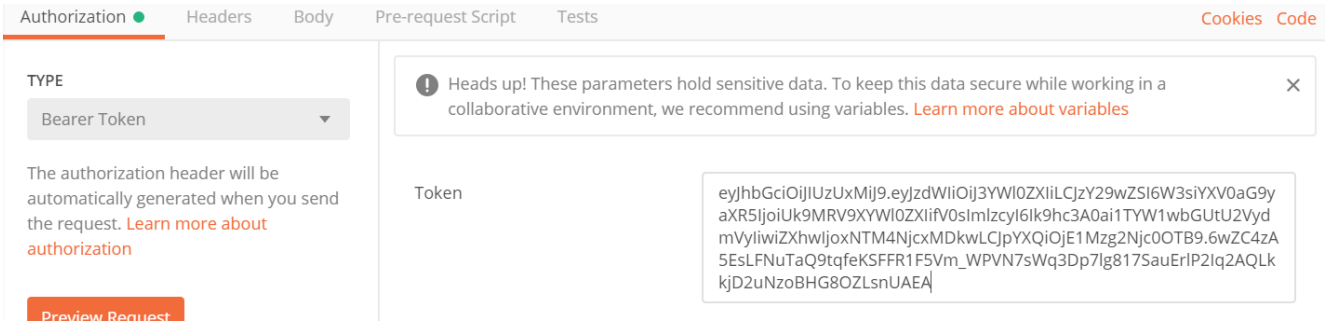
Once done, you have to create a POST Request for the LOGIN and insert in the body the JSON containing the username and password “waiter”



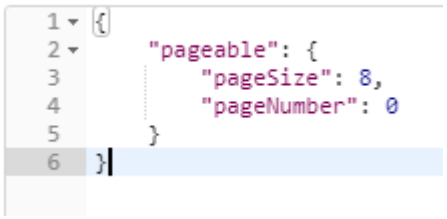
Once done with success (**Status: 200 OK**) …

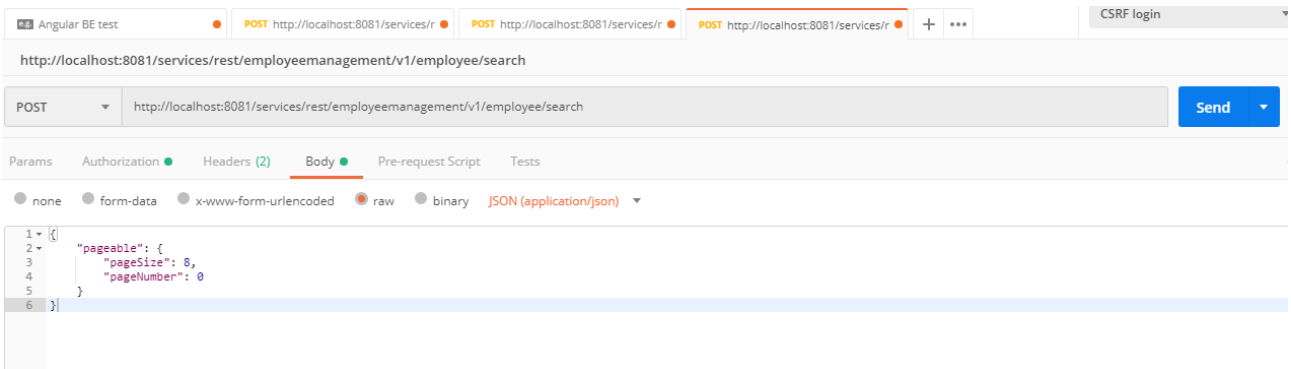


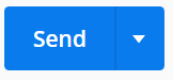
… We create a NEW POST Request and We copy the Authorization ➔ Bearer field (see above) and We paste it in the Token field (see below)



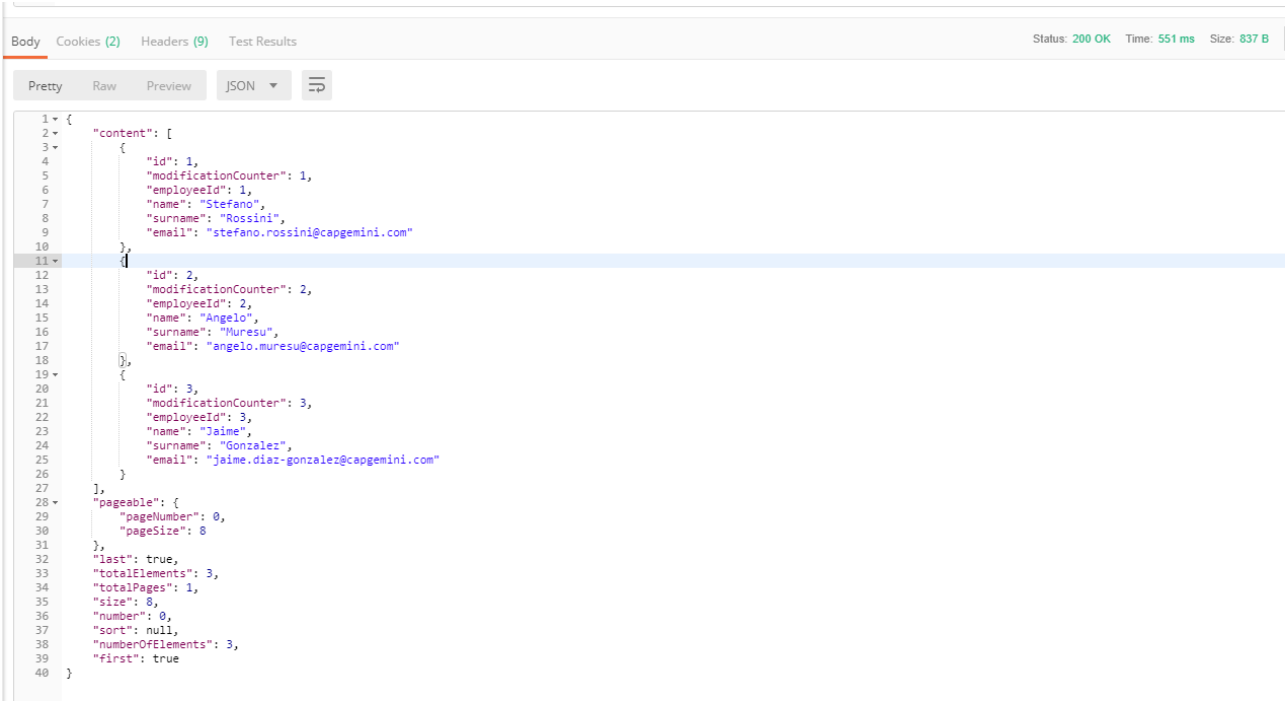
and specific the JSON parameters for the pagination of the Request that We're going to send:





Now you can click 

Now you ‘ve to check that response has got **Status: 200 OK** and to see the below list of Employee 😉



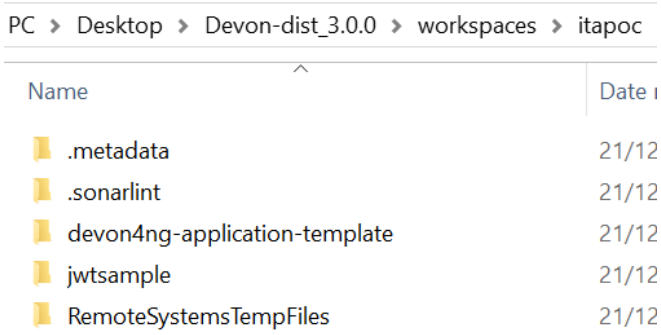
Now that We have successfully tested the BE is time to go to create the FE !

2.2 Front End (Web App Angular + Ionic App)

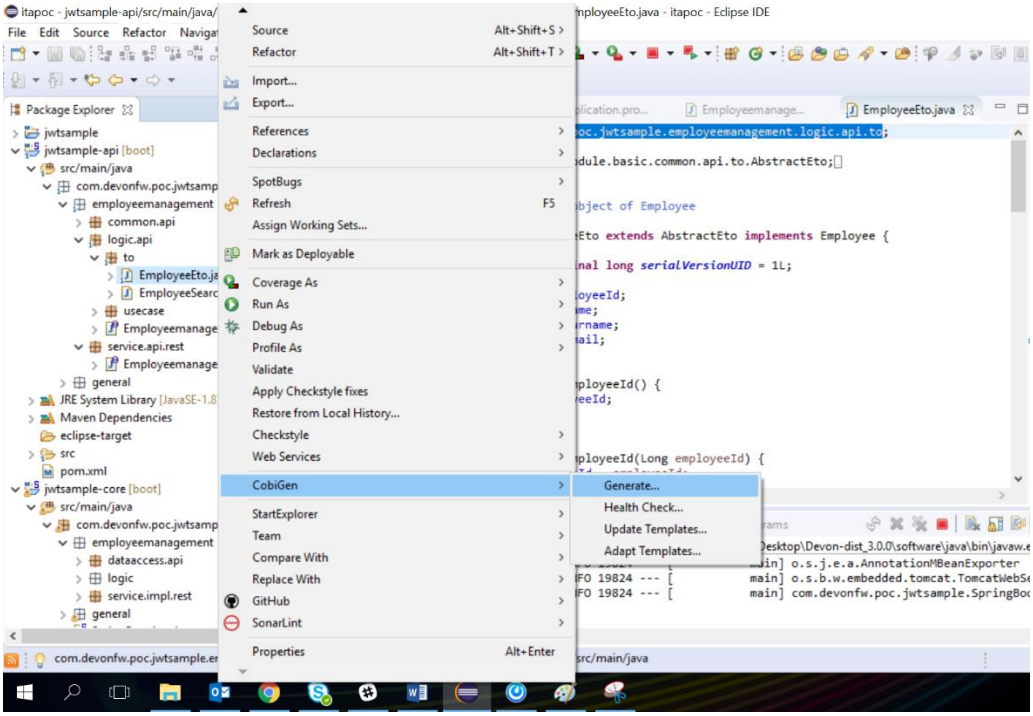
Let's start now with FE with angular Web and then Ionic app.

2.2.1 Front End Web App Angular

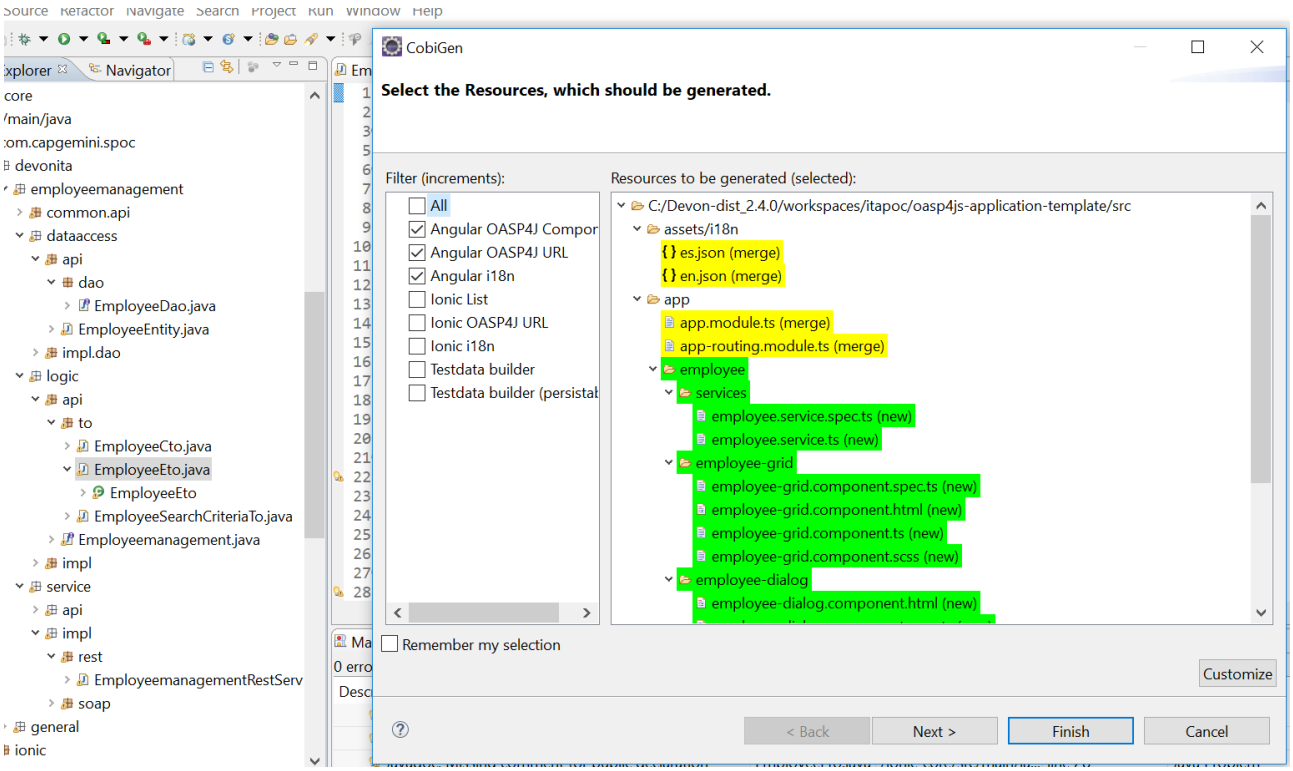
1. Now, for the Angular structure to be auto-generated, first copy the "**devon4ng-application-template**" project from the location
(Path location \Devon-dist_X.X.0\Devon-dist_X.X.0\workspaces**examples**) and paste it in the your project location (i.e: \Devon-dist_3.0.0\workspaces**itapoc**)



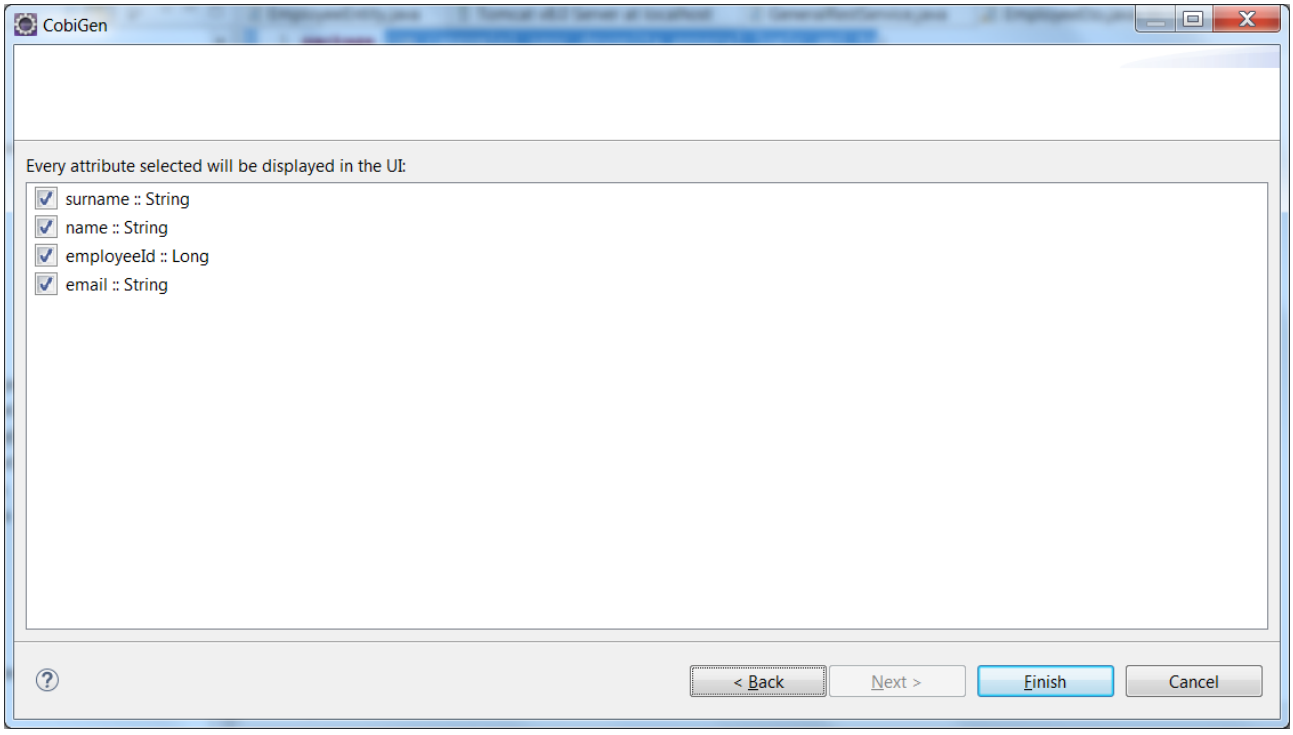
2. Once done, right click on the **EmployeeEto.java** (that is the Entity transport object) file present under the package "`com.devonfw.poc.jwtsample.employeemanagement.logic.api.to`"



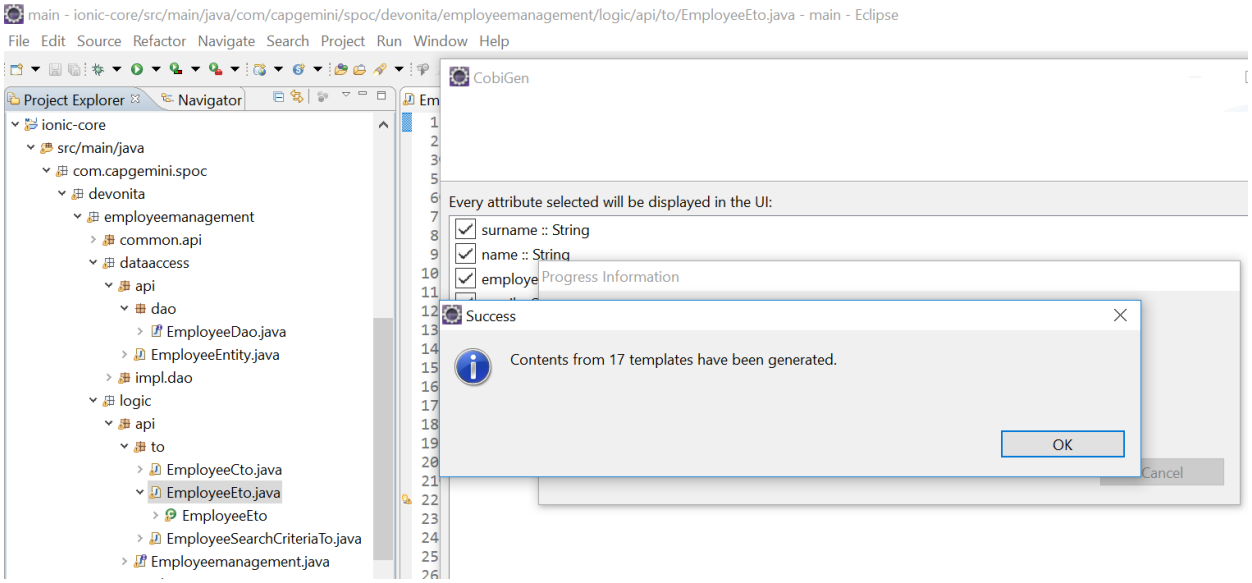
3. Click on the selected options as seen in the screenshot:



4. Click on Next

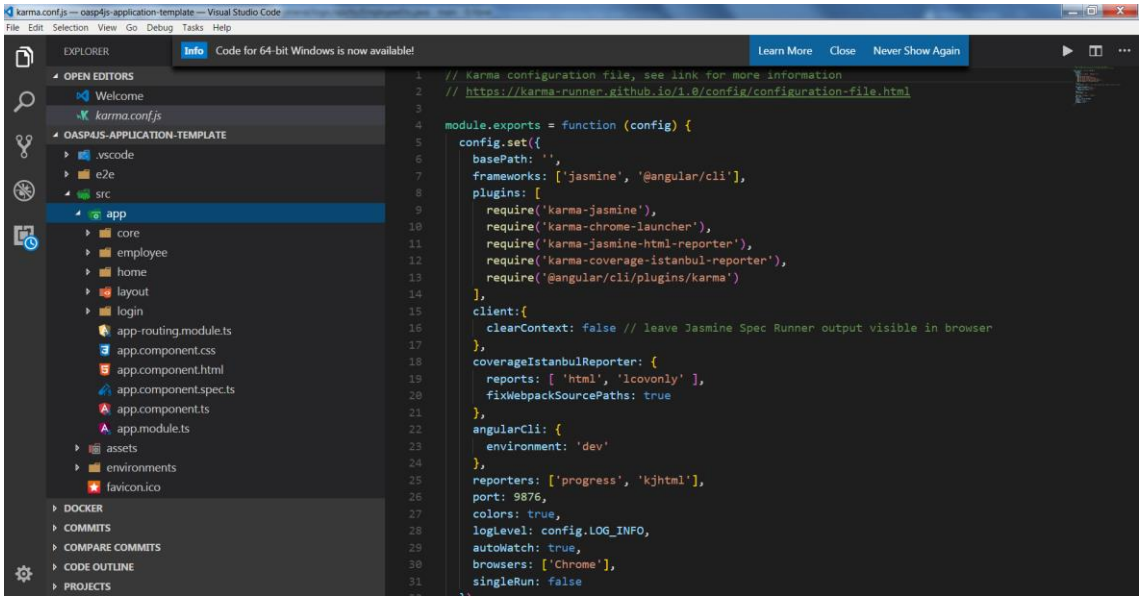


5. Click on Finish



6. The entire ANGULAR structure has been auto generated.

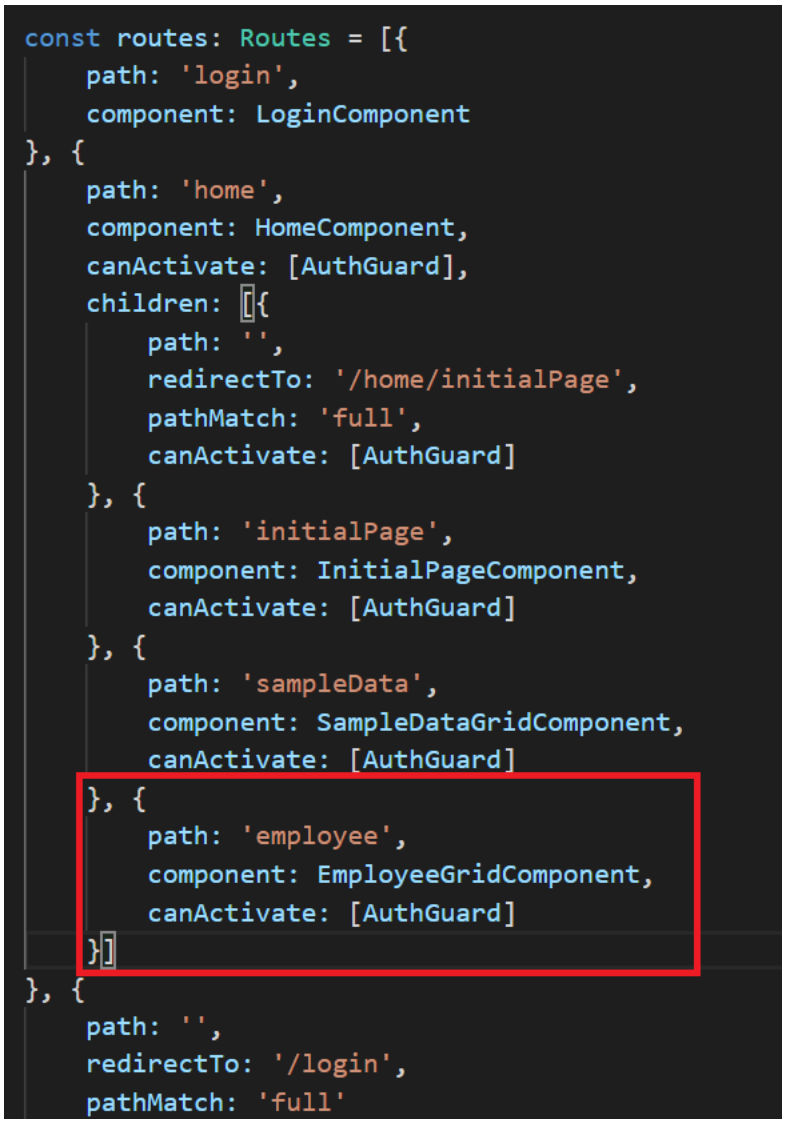
The entire <u>Angular FE layer</u> structure having CRUD operation methods will be auto generated.



7. IMPORTANT now you have to add in the ***app-routing.module.ts*** file the next content, as a child of HomeComponent, in order to enable the route of the new generated component

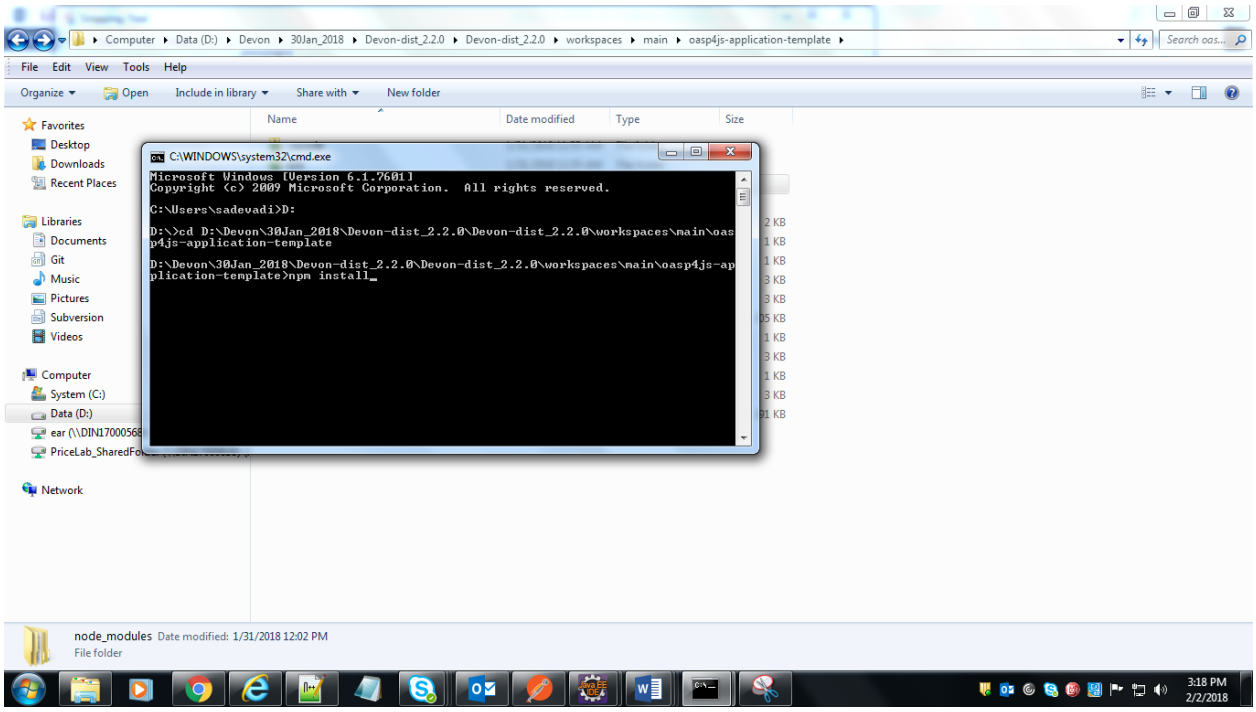
```
,{
    path: 'employee',
    component: EmployeeGridComponent,
    canActivate: [AuthGuard],
},
```

Following picture explain where to place the above content:

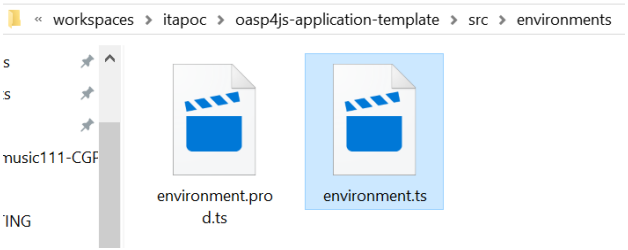


```
const routes: Routes = [{
    path: 'login',
    component: LoginComponent
}, {
    path: 'home',
    component: HomeComponent,
    canActivate: [AuthGuard],
    children: [{
        path: '',
        redirectTo: '/home/initialPage',
        pathMatch: 'full',
        canActivate: [AuthGuard]
    }, {
        path: 'initialPage',
        component: InitialPageComponent,
        canActivate: [AuthGuard]
    }, {
        path: 'sampleData',
        component: SampleDataGridComponent,
        canActivate: [AuthGuard]
    }, {
        path: 'employee',
        component: EmployeeGridComponent,
        canActivate: [AuthGuard]
    }]
}, {
    path: '',
    redirectTo: '/login',
    pathMatch: 'full'
```

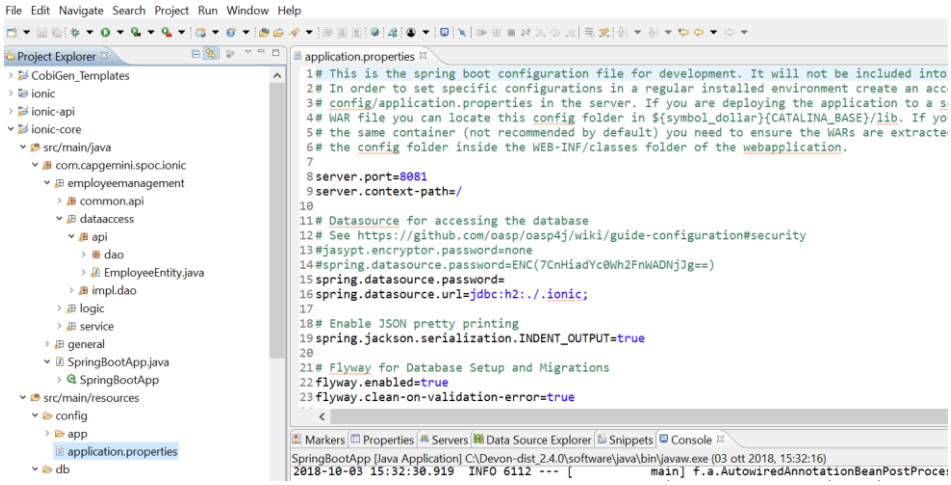
8. Now execute **console.bat** present on the path C:\Devon-dist_3.0.0\console.bat. It will open a console. Then type the path where your “devon4ng-application-template” project is present (i.e: C:\Devon-dist_3.0.0\workspaces\itapoc\devon4ng-application-template) and execute “npm install” command which would download all the required libraries and node modules folder would be generated.



9. Check the file **environment.ts** if the server path is correct. (for production you will have to change also the environment.prod.ts file)



In order to do that it's important to look at the application.properties to see the values as PATH, TCP port etc …



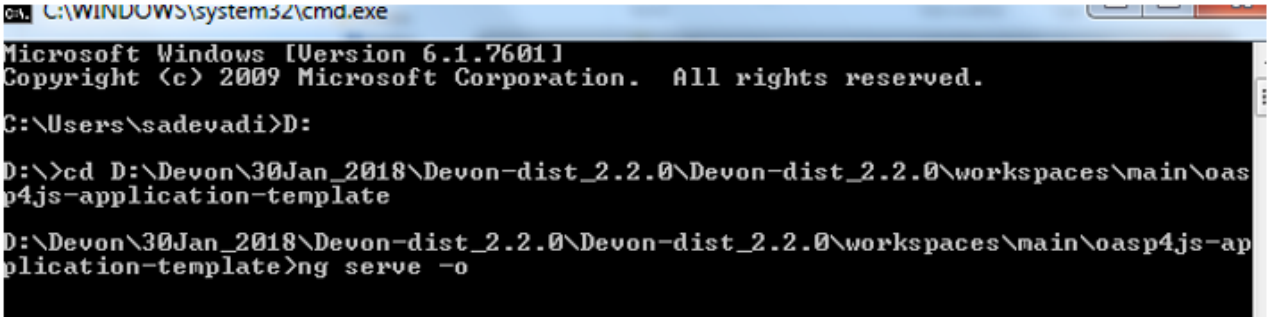
For example in this case the URL should be since the context path is empty the server URLS should be like:

```
export const environment = {
  production: false,
```

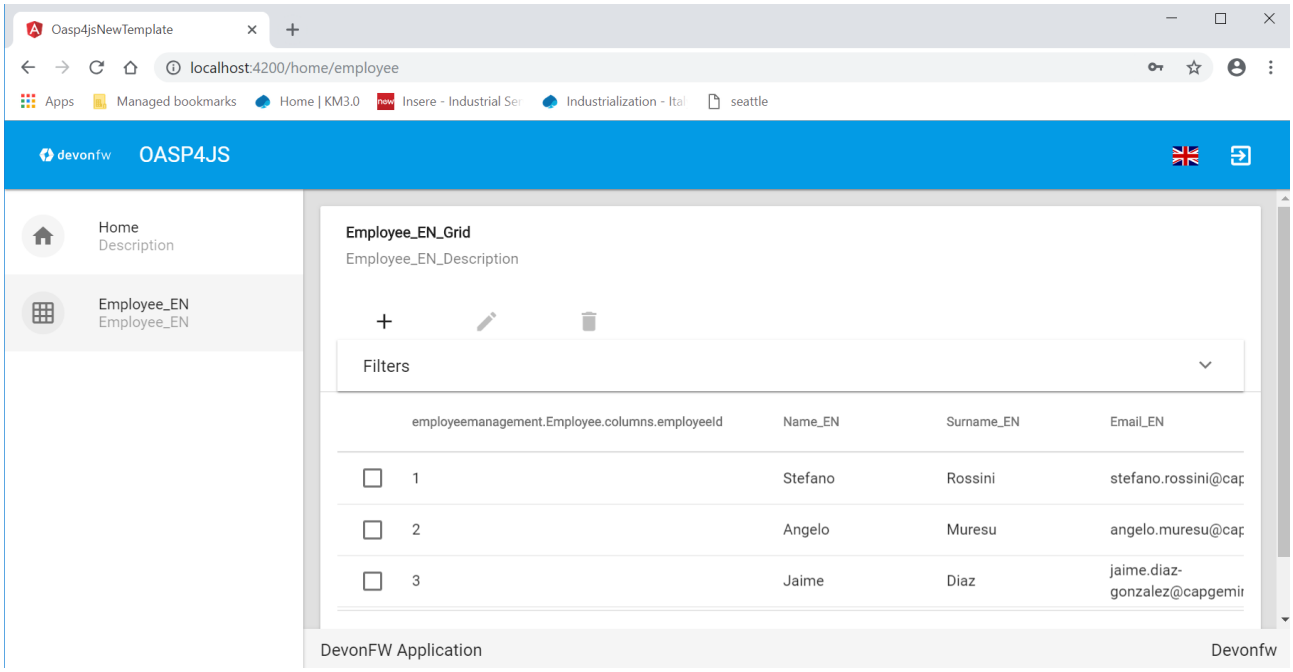
```
    restPathRoot: 'http://localhost:8081/',
    restServiceRoot: 'http://localhost:8081/services/rest/',
    security: 'jwt'
};
```

Warning: REMEMBER to set security filed to **jwt**

10. Now run the "**ng serve -o**" command to run the Angular Application.



11. If the command execution is **successful**, the below screen will **appear** and it would be automatically redirected to the url: http://localhost:4200/login

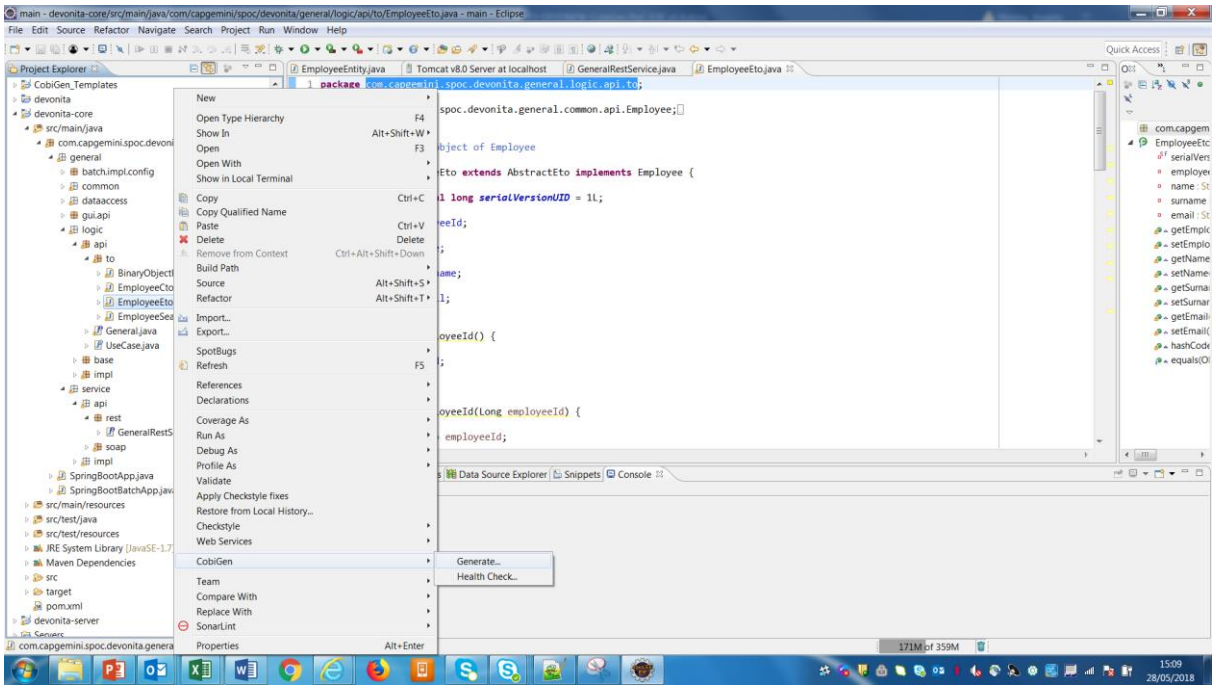


FE WebApp DONE

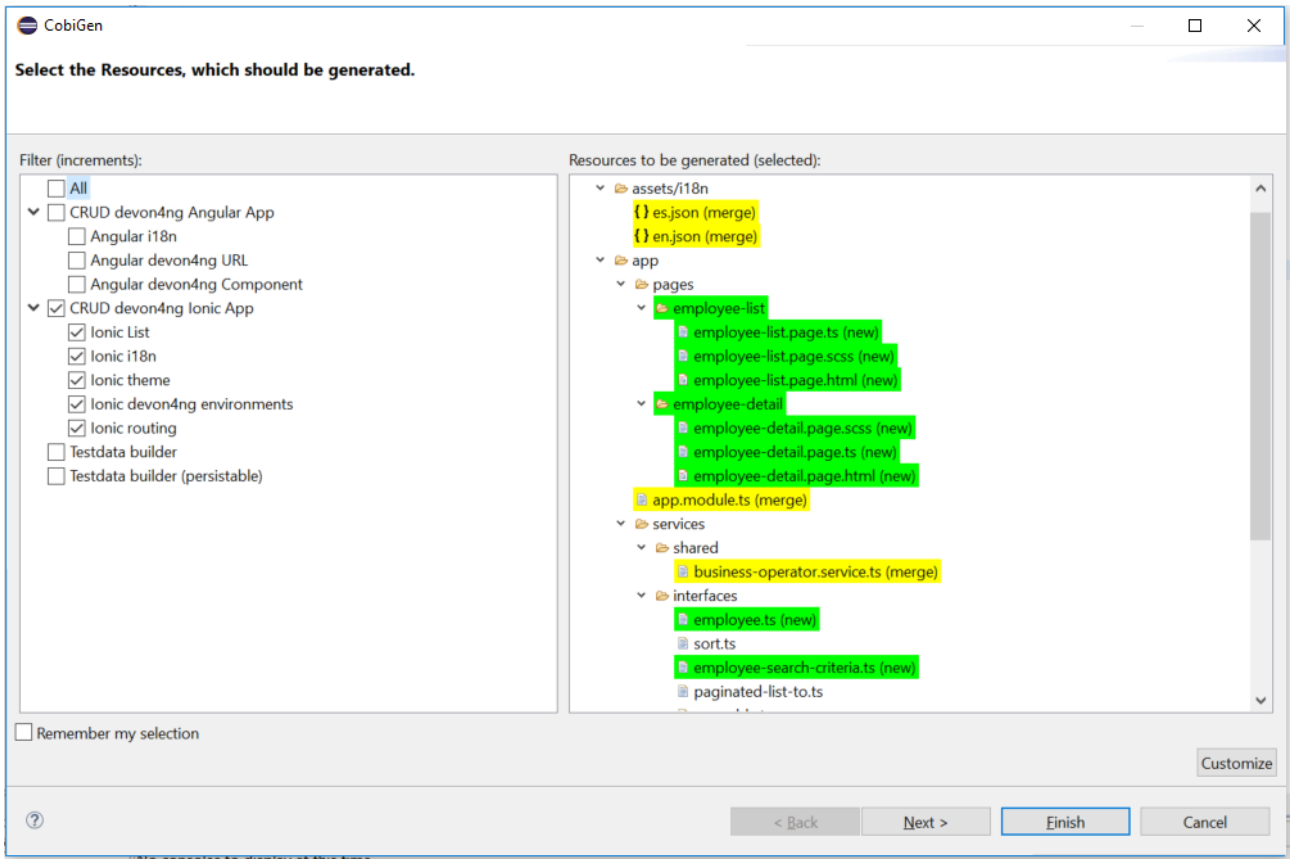
2.2.2 Front End Mobile App

1. Now, for the Ionic structure to be auto-generated, first copy the "**devon4ng-ionic-application-template**" project from the location (Path location \Devon-dist_X.X.0\Devon-dist_X.X.0\workspaces\examples) and paste it in your directory (i.e.: C:\Devon-dist_3.0.0\workspaces\itapoc).

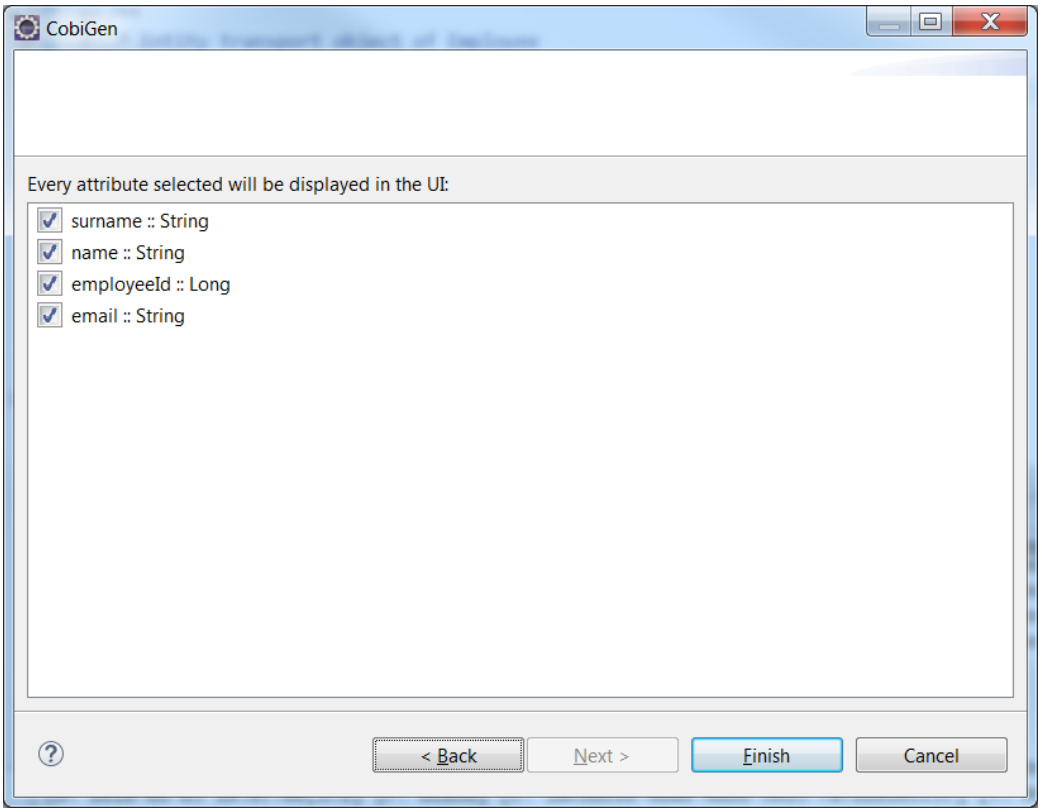
2. Once done, Right click on the **EmployeeEto.java** as you already did before in order to use CobiGen.



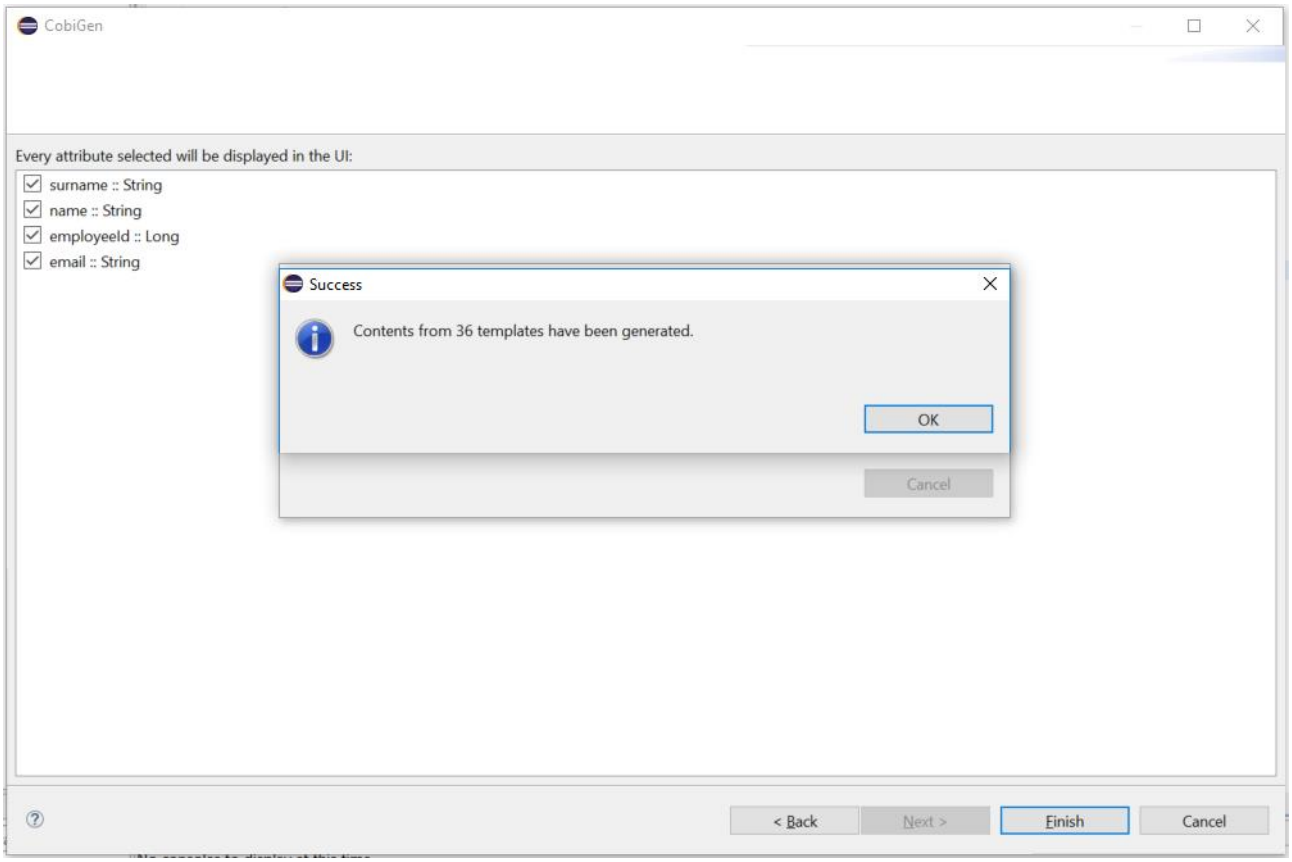
3. Click on the selected options as seen in the screenshot:



4. Click on Next.

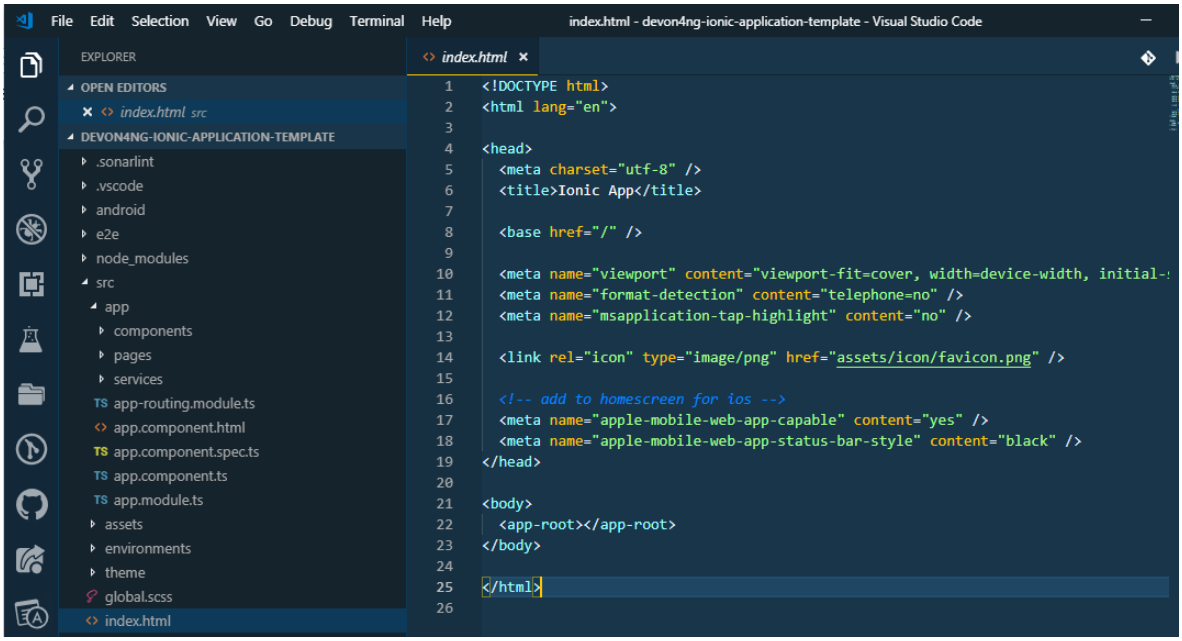


5. Click on Finish

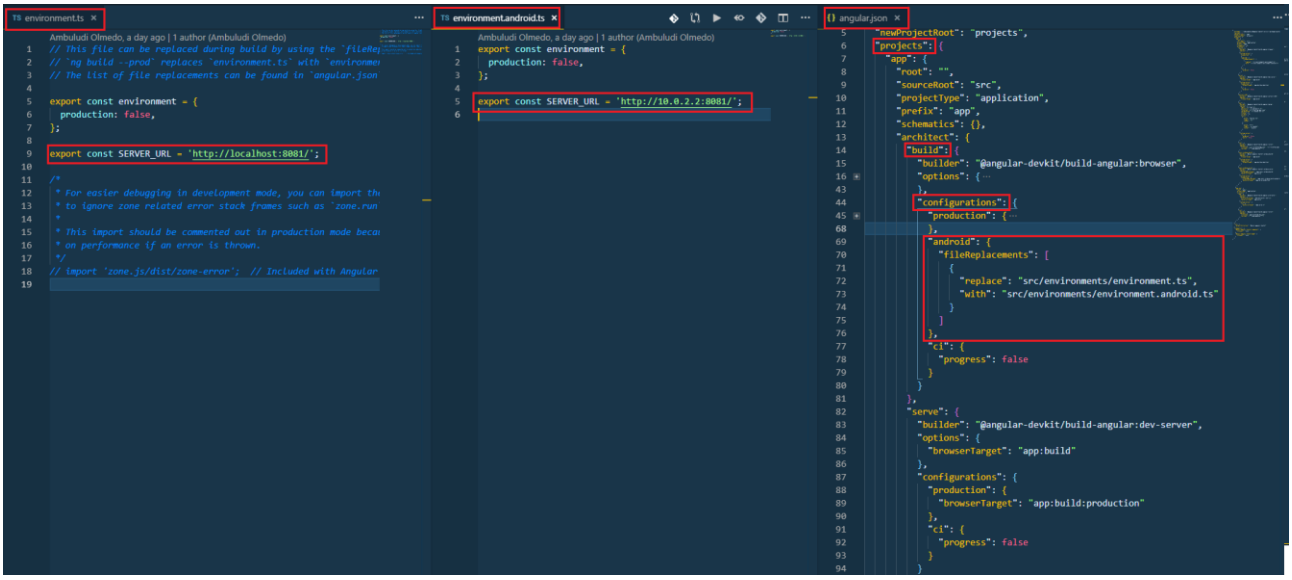


6. The entire ionic structure will be auto generated.

The entire Ionic APP structure would be auto generated having CRUD operation methods.



7. Change the server url (with correct serve url) in environment.ts, environment.prod.ts and environment.android.ts files (i.e: itapoc\devon4ng-ionic-application-template\src\environments\). The angular.json file inside the project has already a build configuration for android.

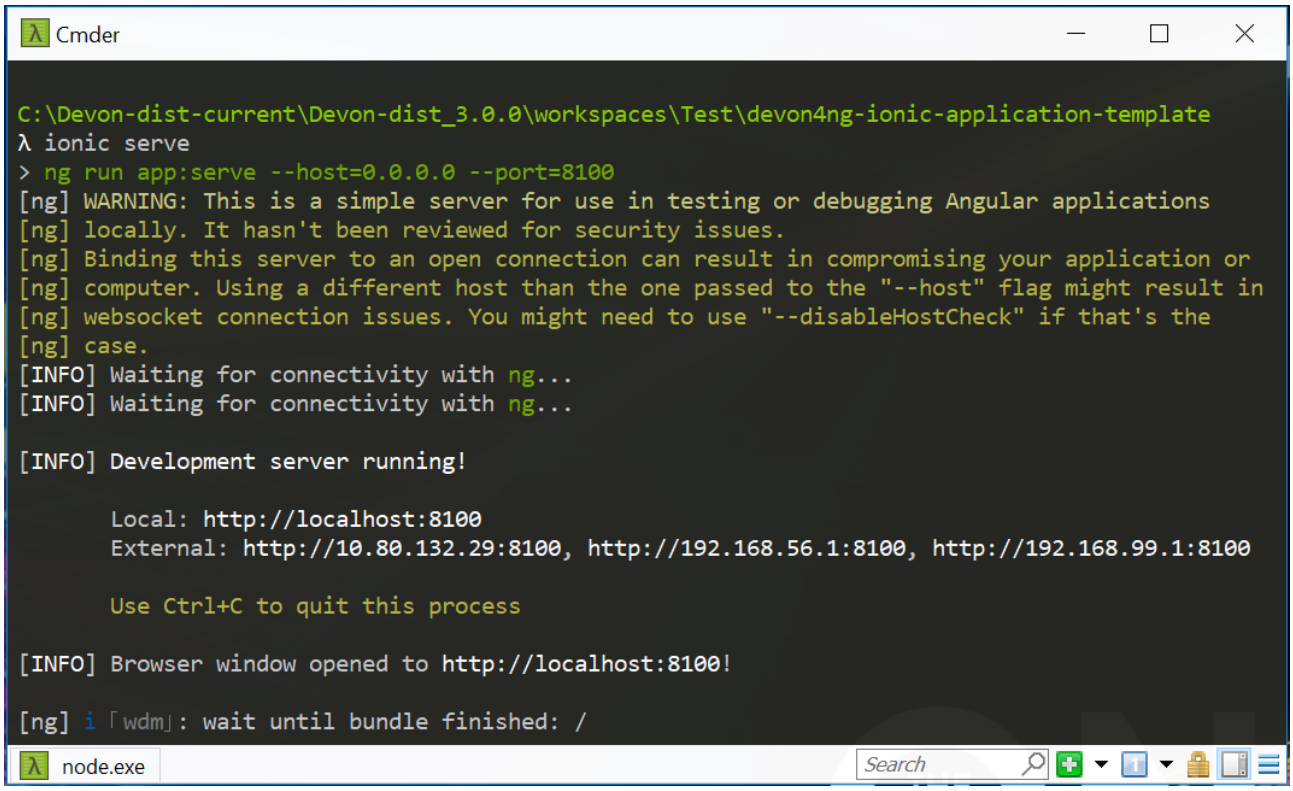


8. Now, open **console.bat** present on the path C:\Devon-dist_X.X.0\console.bat and type the path where your “devon4ng-ionic-application-template” project is present.

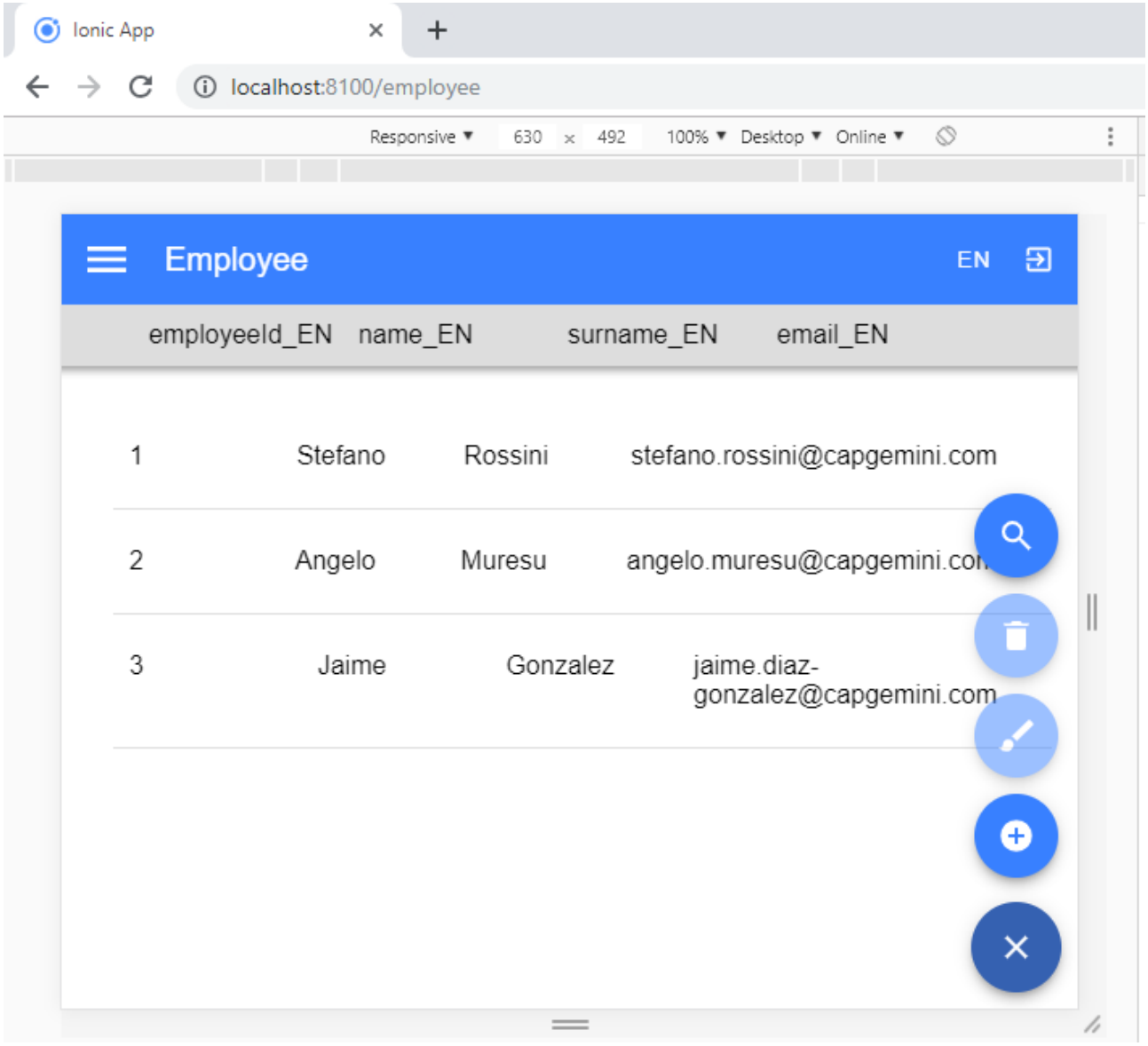
9. If you have Windows10

```
npm install @ionic/app-scripts@latest --save-dev
```

10. Run “**npm install**”.
11. Execute “**ionic serve**”.



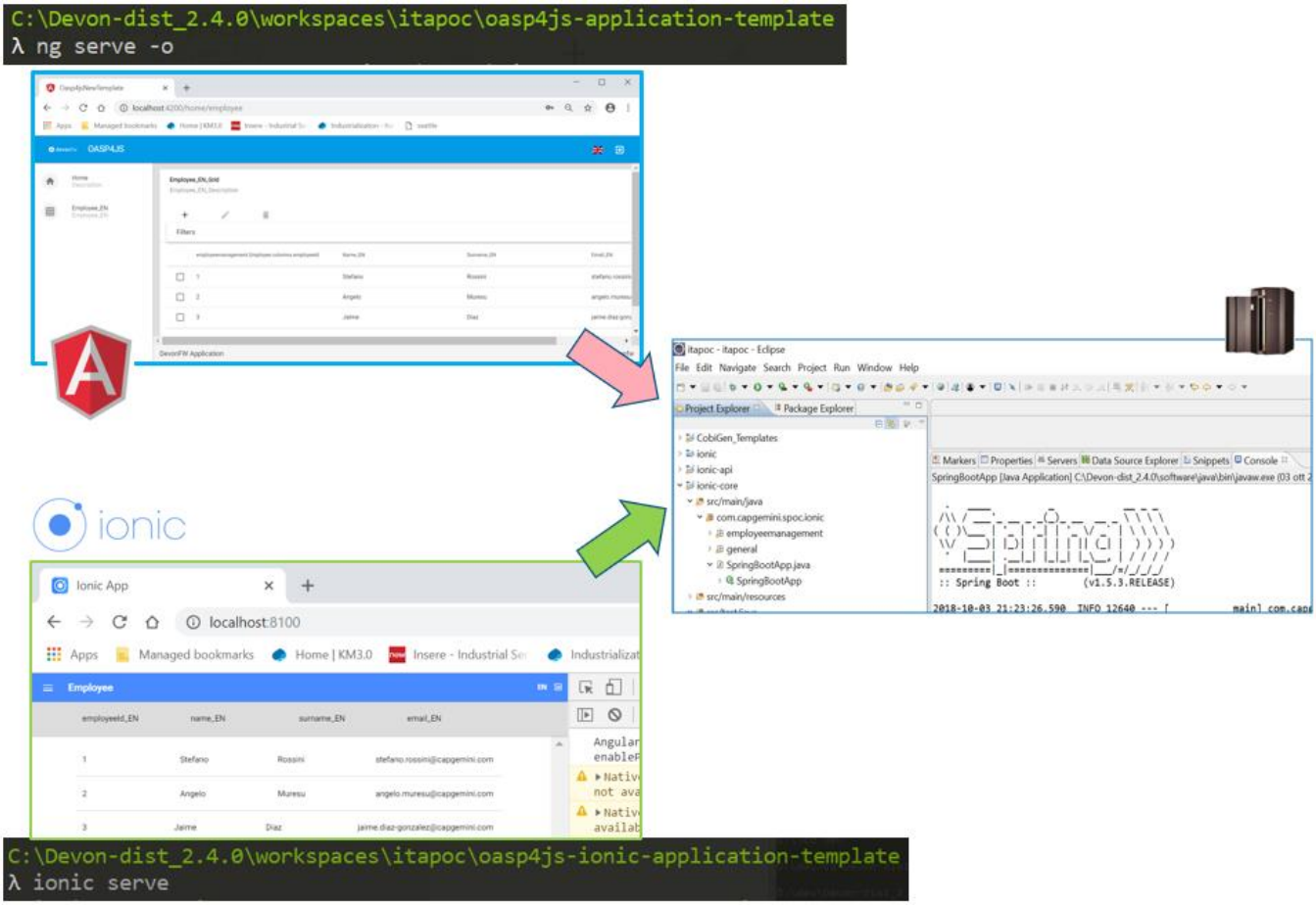
12. Once the execution is successful



FE Mobile App DONE ☺

So: well done ☺!

Starting from an Entity class you've successfully generated the Back-End layer (REST, SOAP, DTO, Spring services, Hibernate DAO), the Angular Web App and the Ionic mobile App!

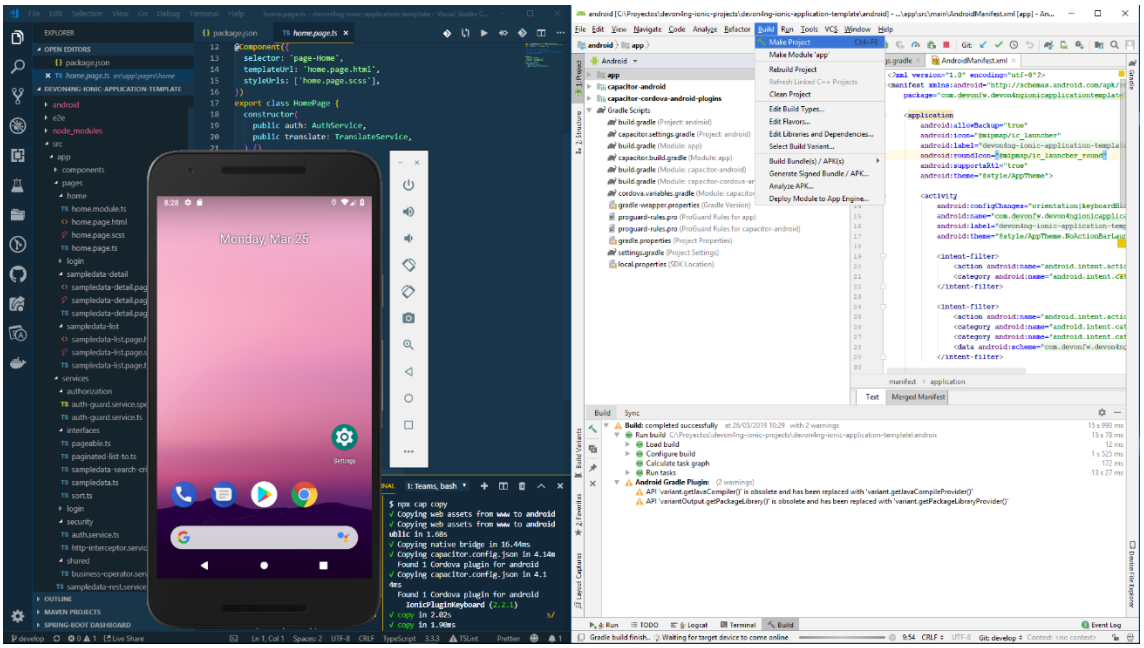


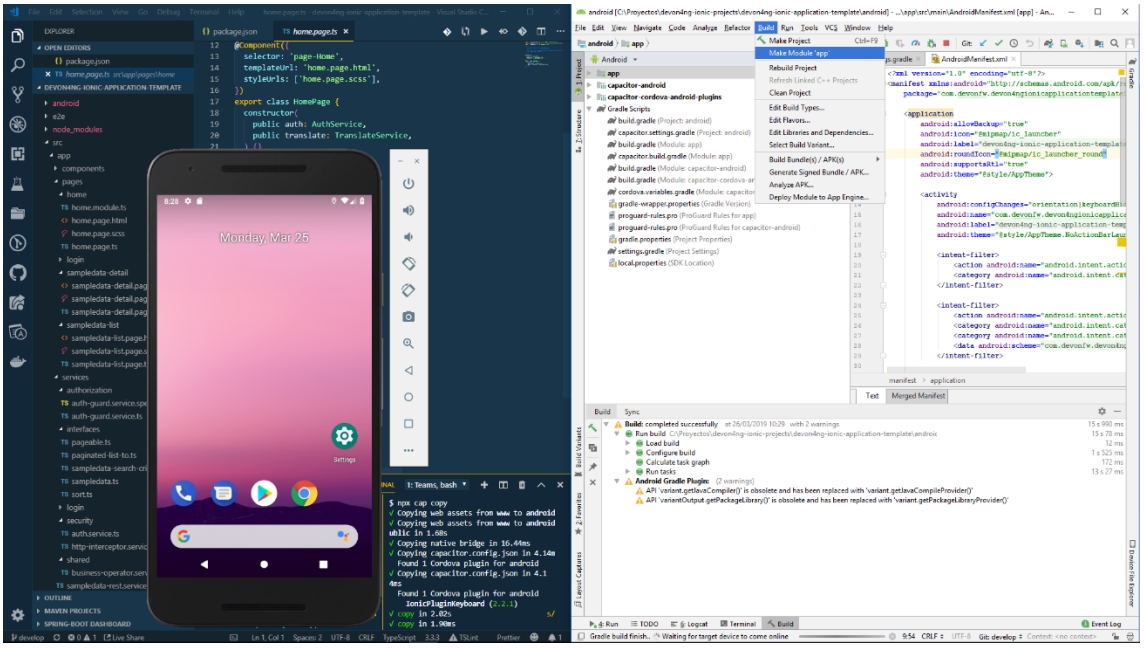
2.2.2.1 Generate APK

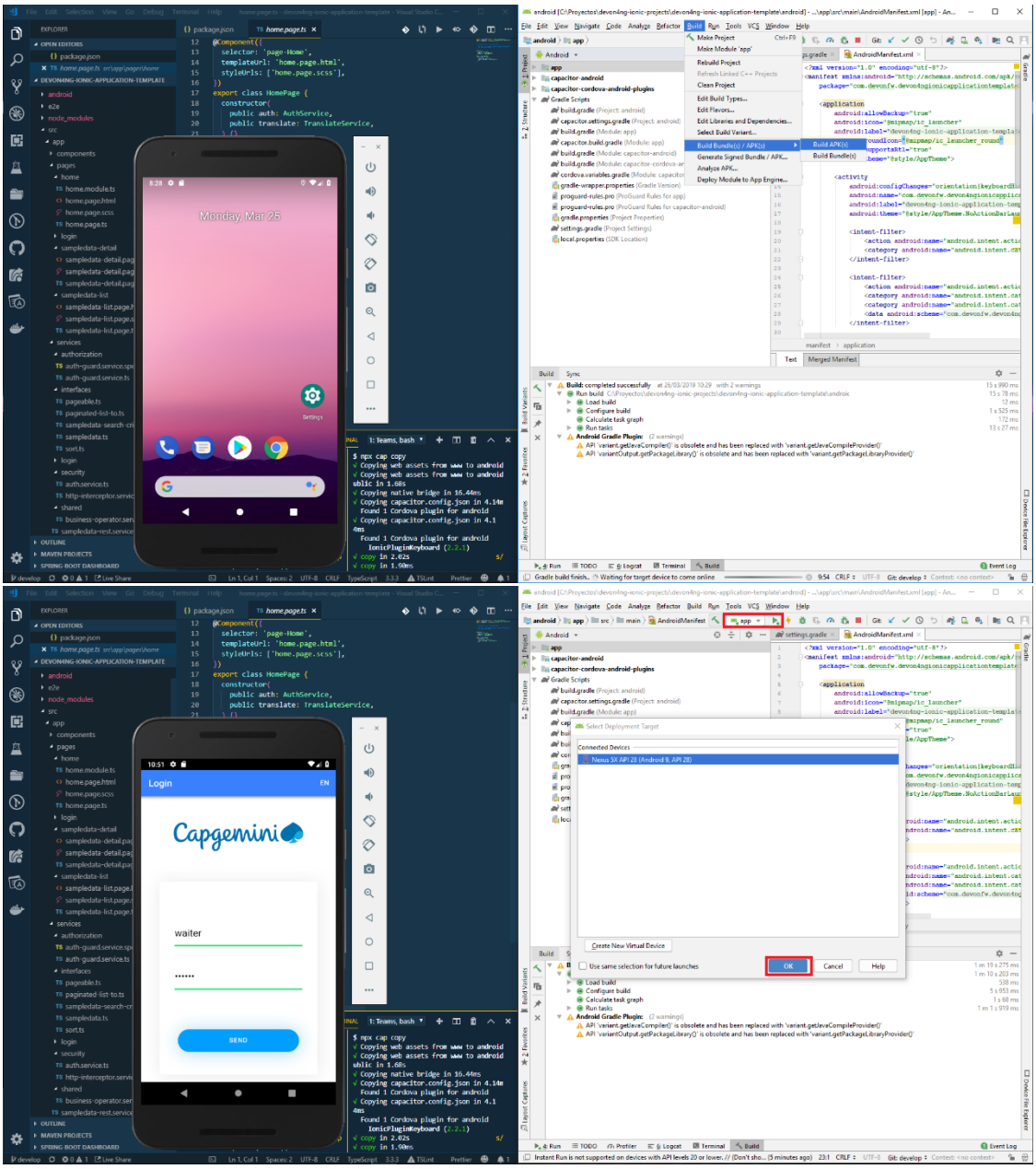
Since We're going to create apk remember the following pre-conditions:

- Gradle (https://gradle.org/install/)
- Android Studio (https://developer.android.com/studio)
- Android sdk (https://developer.android.com/studio/#command-tools)
- Capacitor (https://capacitor.ionicframework.com/docs/getting-started/)

1. Now, open cmd and type the path where your "devon4ng-ionic-application-template" project is present.
2. Run the following commands:
    a. npx cap init
    b. ionic build --configuration=android
    c. npx cap add android
    d. npx cap copy
    e. npx cap open android
3. Build the APK using Android studio.







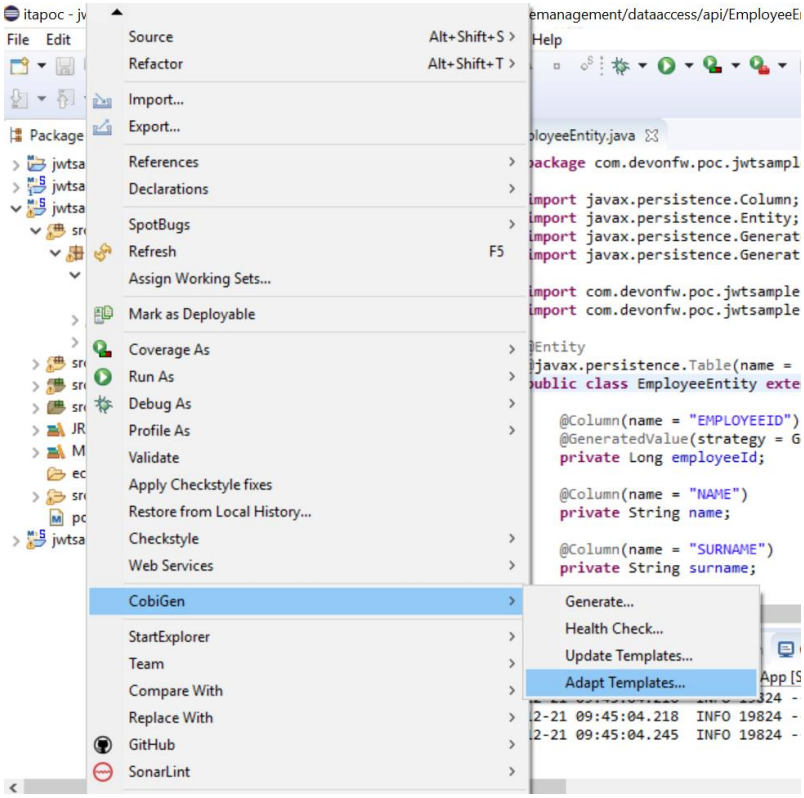
You can find your apk file in

/devon4ng-ionic-application-template/android/app/build/outputs/apk/debug

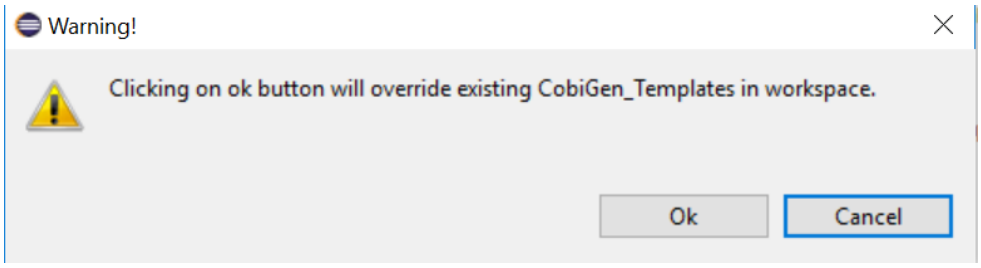
3 Adapt CobiGen_Templates:

After following this tutorial, you will have the CobiGen_Templates downloaded on your local machine. To import these templates you need to do the following:

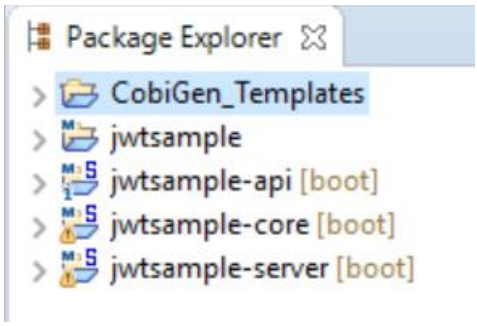
Right click in any part of the package explorer, then click on CobiGen -> Adapt templates

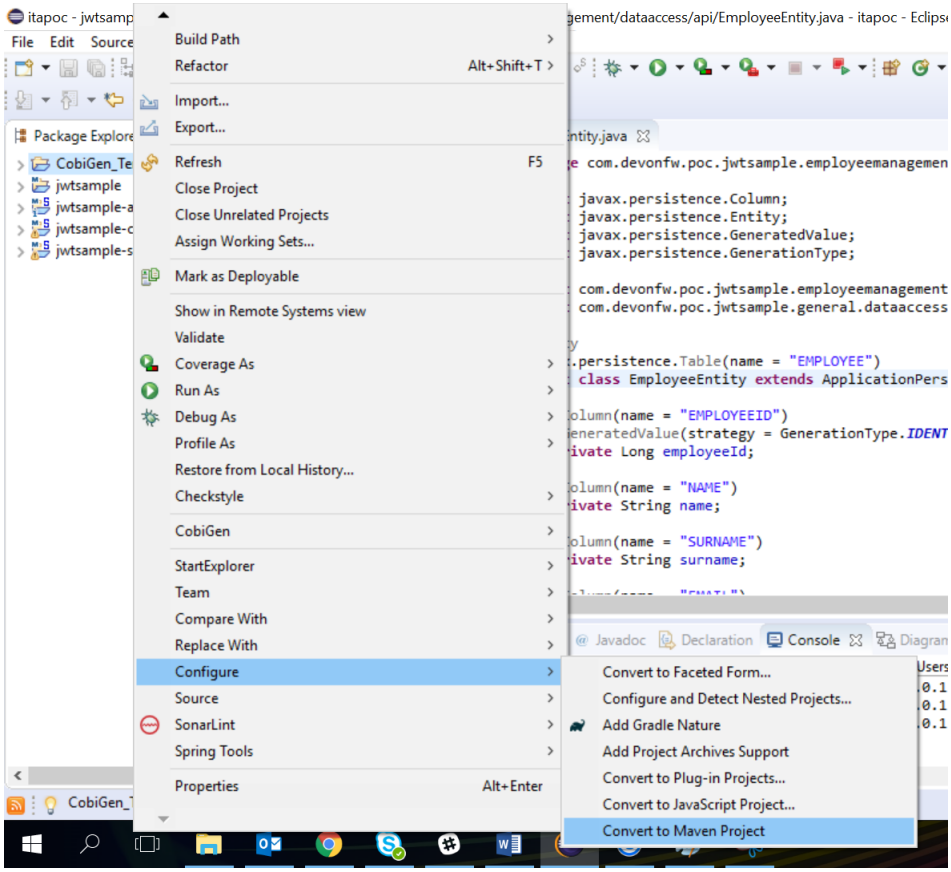


Click *Ok*:

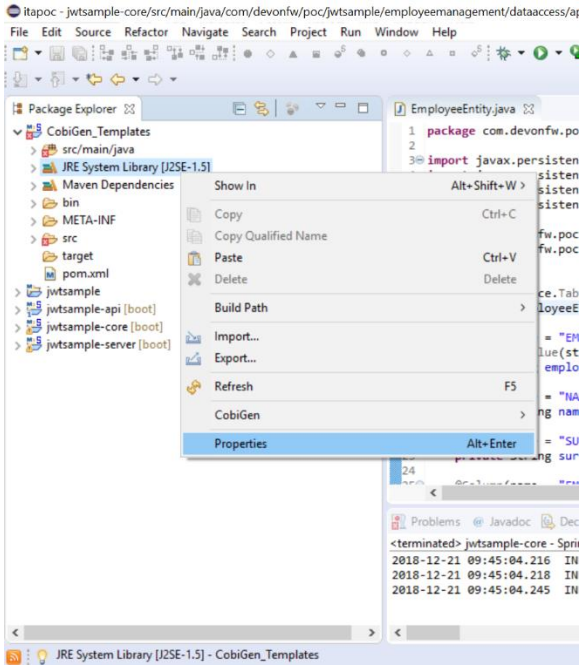


Now the CobiGen_Templates project will be automatically imported into your workspace, as shown on the image below:

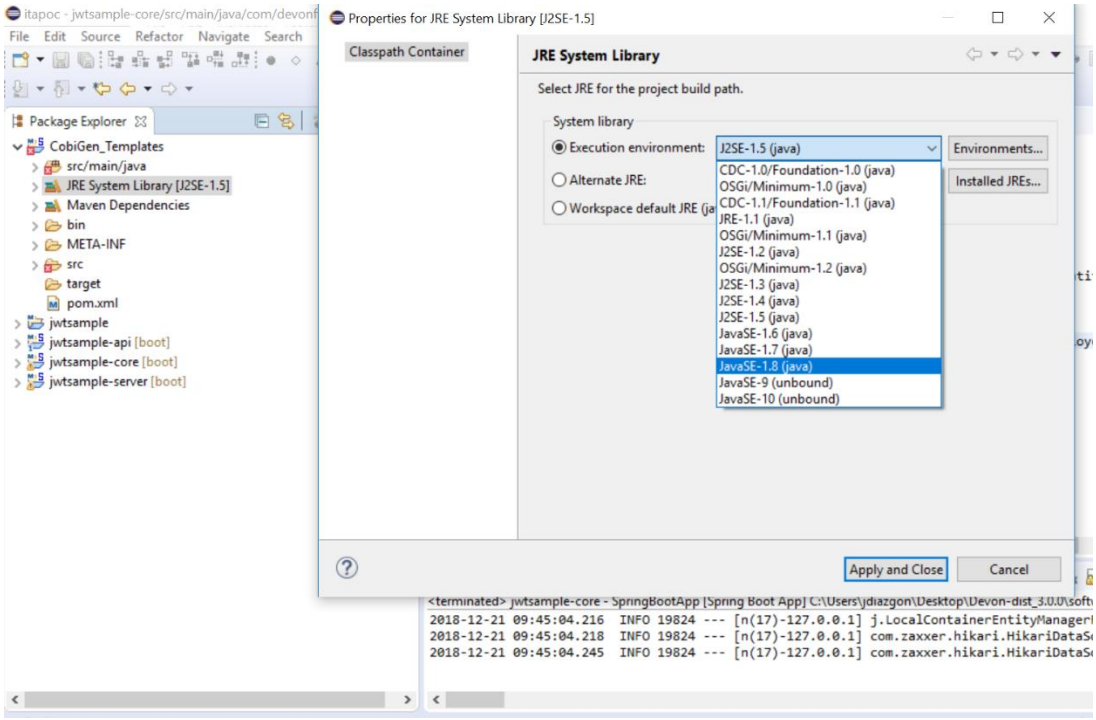




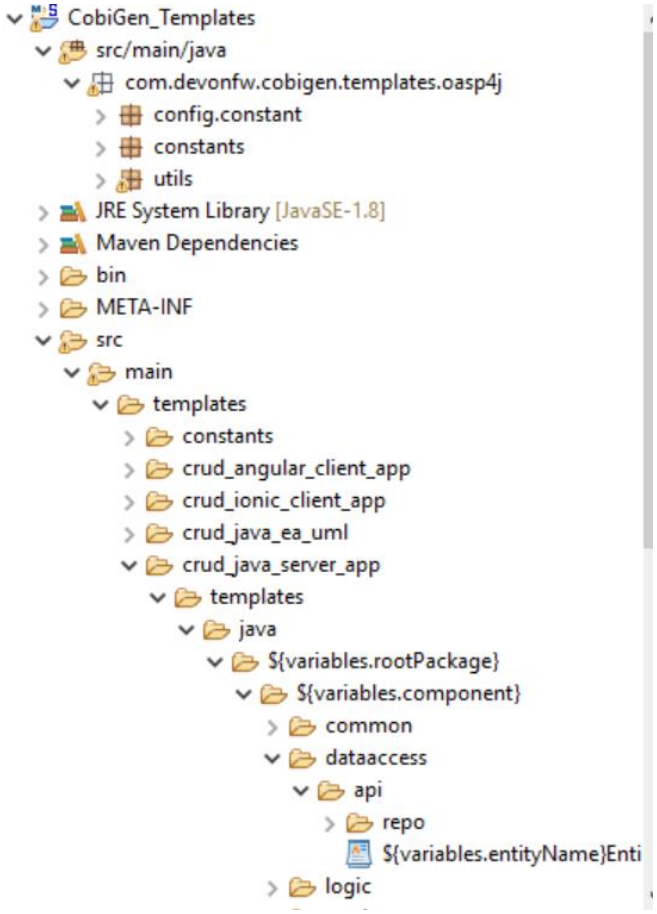
Now you just need to change the Java version of the project to JRE 1.8. Right click on the JRE system library, and then on *Properties:*



Now change the version to Java 1.8



Now you have successfully imported the CobiGen templates. If you want to edit them, you will find them in the folder *src/main/templates.* For instance, the Java templates are located here:



Now you can adapt the templates as much as you want. Documentation about this can be found on:

https://github.com/devonfw/tools-cobigen/wiki/Guide-to-the-Reader