

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.....	2
1.1	What's Devonfw	2
1.2	Devonfw Web site URL.....	3
1.3	What's CobiGen.....	4
1.4	HOW TO install Devonfw	5
2	Steps to create a Sample UI Angular4 Project through Cobigen.....	6
2.1	Back End (Services, DTO, DAO, DB)	7
2.2	Front End (Web App Angular + Ionic App)	24
2.2.1	Front End Web App Angular	24
2.2.2	Front End Mobile App.....	29
3	Adapt CobiGen_Templates:.....	38

1 Introduction

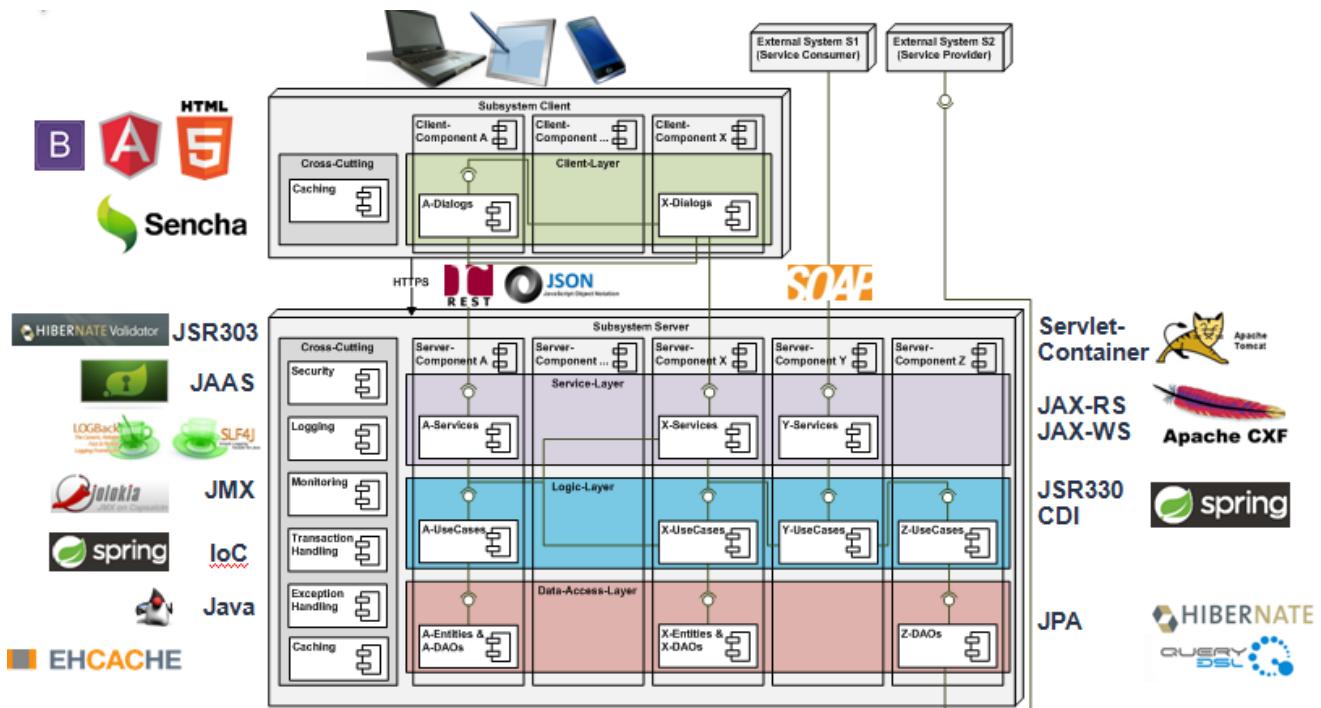
1.1 What's Devonfw

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 This platform is aimed to engagements where the client is flexible in the use of technology or uses outdated technology, It can offer a modern technology approach using our experience as a group.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.

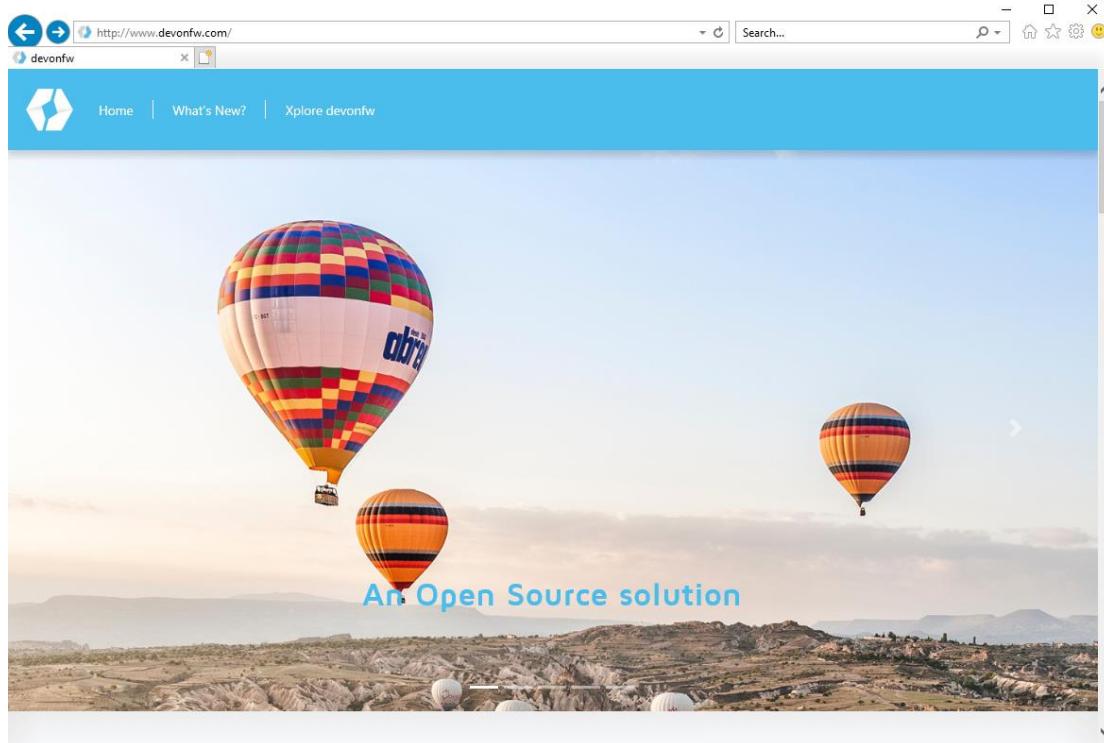
The current version of the platform is oriented to develop single-page web applications based on the Java EE programming model using the spring framework as the default implementation.

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.



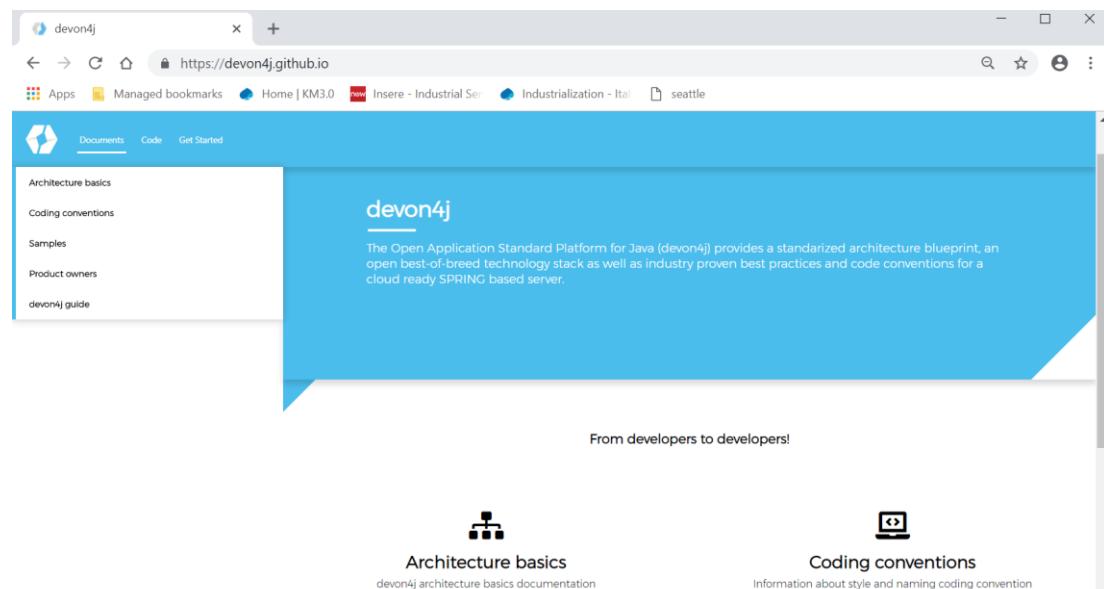
1.2 Devonfw Web site URL

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



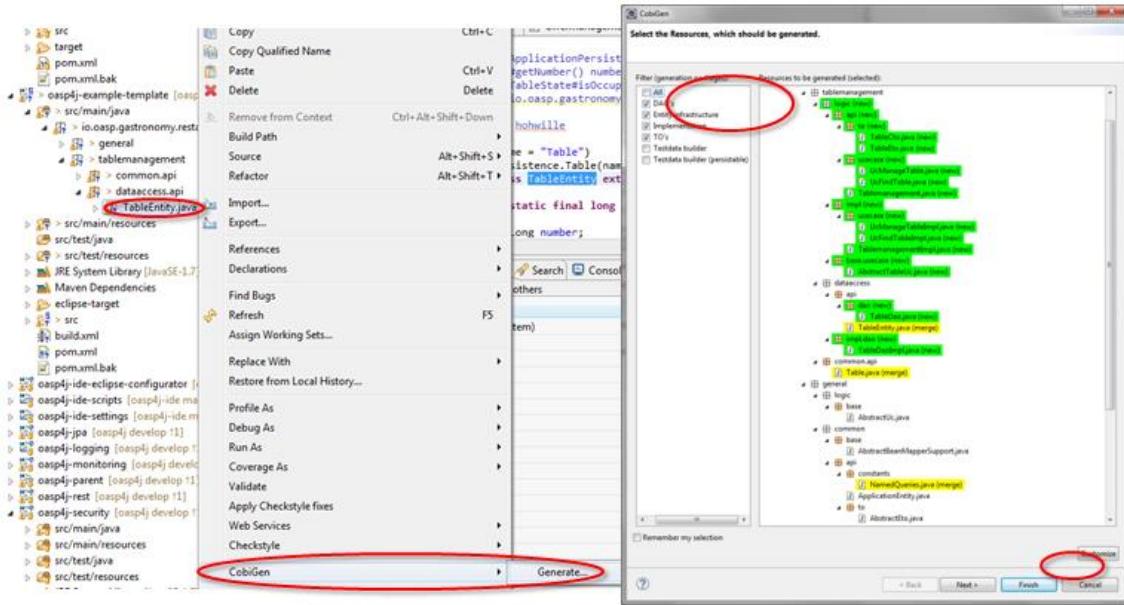
The Devonfw for JAVA is Devon4j and you can find everything here:

<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 devonfw 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:

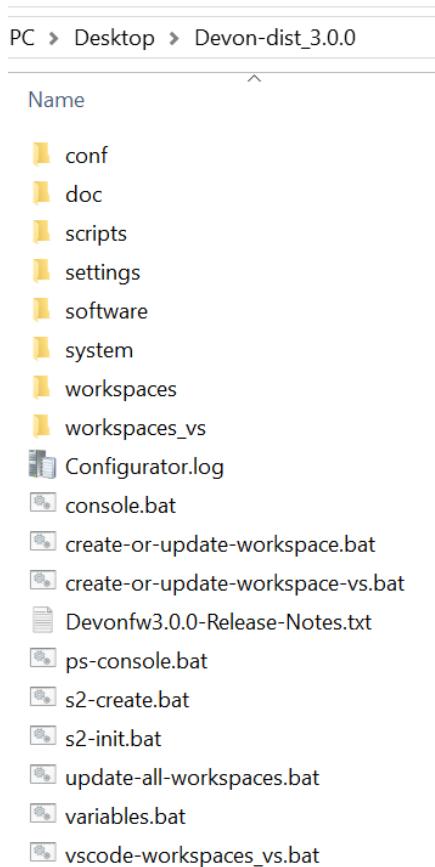
1. Can generate a whole **CRUD** application from a **single Entity class**.
2. You can save the effort for creating simple CRUD use case since CobiGen generates: **DAOs**, **DTOs**, Spring services and REST and SOAP services and even the client UI application (Angular and Ionic).
3. **Agility**: Boost 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:

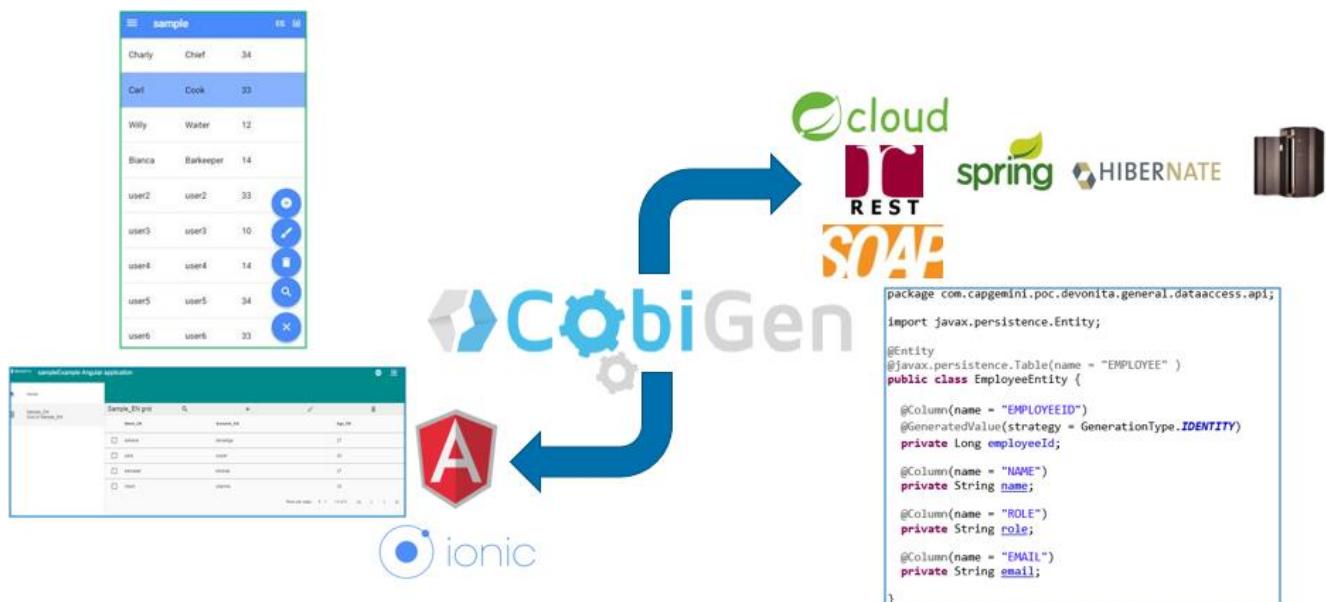
- Java
- Node and npm (<https://nodejs.org/dist/v10.7.0/node-v10.7.0-x64.msi>)
- Capacitor (<https://capacitor.ionicframework.com/docs/getting-started/>)
 - `npm install -g @capacitor/core @capacitor/cli`

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)

CobiGen Code generator



FE: Angular, IONIC ← DTO → BE: Hiberante, Spring Services

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)

```
C:\WINDOWS\system32\cmd.exe
IDE environment has been initialized.
Copied workspaces\main\development\settings\maven\settings.xml to conf\.m2\settings.xml
Copied workspaces\main\development\settings\version\settings.json to conf\settings.json
lug 16, 2018 4:12:14 PM io.oasp.ide.eclipse.configurator.core.Configurator logCall
INFO: io.oasp.ide.eclipse.configurator.core.Configurator -u
lug 16, 2018 4:12:14 PM io.oasp.ide.eclipse.configurator.core.Configurator main
INFO: Updating workspace
lug 16, 2018 4:12:14 PM io.oasp.ide.eclipse.configurator.core.Configurator collectWorkspaceFiles
INFO: Collected 78 configuration files.
lug 16, 2018 4:12:15 PM io.oasp.ide.eclipse.configurator.core.Configurator main
INFO: Completed
Eclipse preferences for workspace: "main" have been created/updated
Created eclipse-main.bat
Finished creating/updating workspace: "main"

Press any key to continue . . .
```

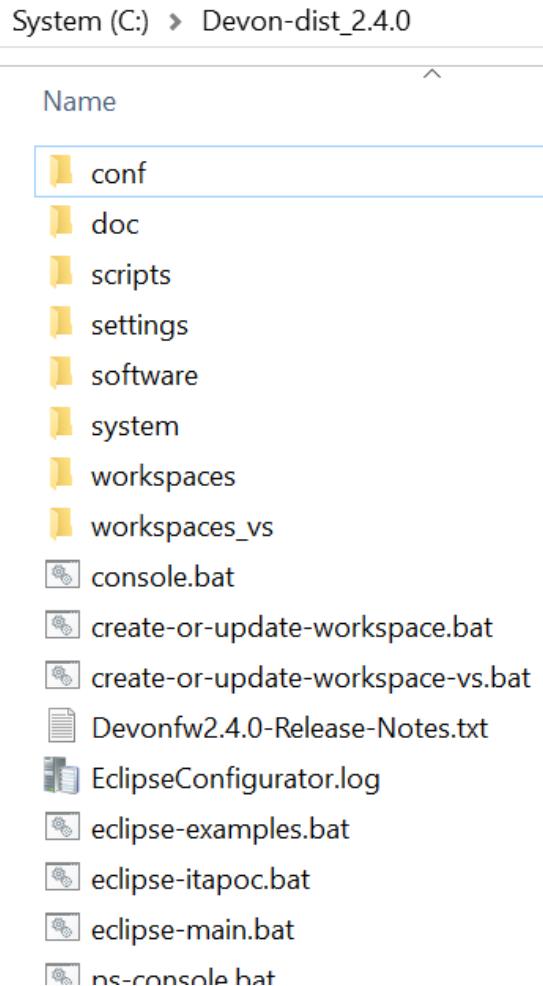
2. Run the **update-all-workspaces.bat**, again present within the Devon-dist_N.n.m folder of your path folder location (i.e: Devon-dist_2.4.0)

```
C:\WINDOWS\system32\cmd.exe
IDE environment has been initialized.
IDE environment has been initialized.
Copied workspaces\main\development\settings\version\settings.json to conf\settings.json
lug 16, 2018 4:12:47 PM io.oasp.ide.eclipse.configurator.core.Configurator logCall
INFO: io.oasp.ide.eclipse.configurator.core.Configurator -u
lug 16, 2018 4:12:47 PM io.oasp.ide.eclipse.configurator.core.Configurator main
INFO: Updating workspace
lug 16, 2018 4:12:47 PM io.oasp.ide.eclipse.configurator.core.Configurator collectWorkspaceFiles
INFO: Collected 78 configuration files.
lug 16, 2018 4:12:47 PM io.oasp.ide.eclipse.configurator.core.Configurator main
INFO: Completed
Eclipse preferences for workspace: "examples" have been created/updated
Created eclipse-examples.bat
Finished creating/updating workspace: "examples"

IDE environment has been initialized.
Copied workspaces\main\development\settings\version\settings.json to conf\settings.json
lug 16, 2018 4:12:48 PM io.oasp.ide.eclipse.configurator.core.Configurator logCall
INFO: io.oasp.ide.eclipse.configurator.core.Configurator -u
lug 16, 2018 4:12:48 PM io.oasp.ide.eclipse.configurator.core.Configurator main
INFO: Updating workspace
lug 16, 2018 4:12:48 PM io.oasp.ide.eclipse.configurator.core.Configurator collectWorkspaceFiles
INFO: Collected 78 configuration files.
lug 16, 2018 4:12:48 PM io.oasp.ide.eclipse.configurator.core.Configurator main
INFO: Completed
Eclipse preferences for workspace: "main" have been created/updated
Finished creating/updating workspace: "main"

Finished updating all workspaces
Press any key to continue . . .
```

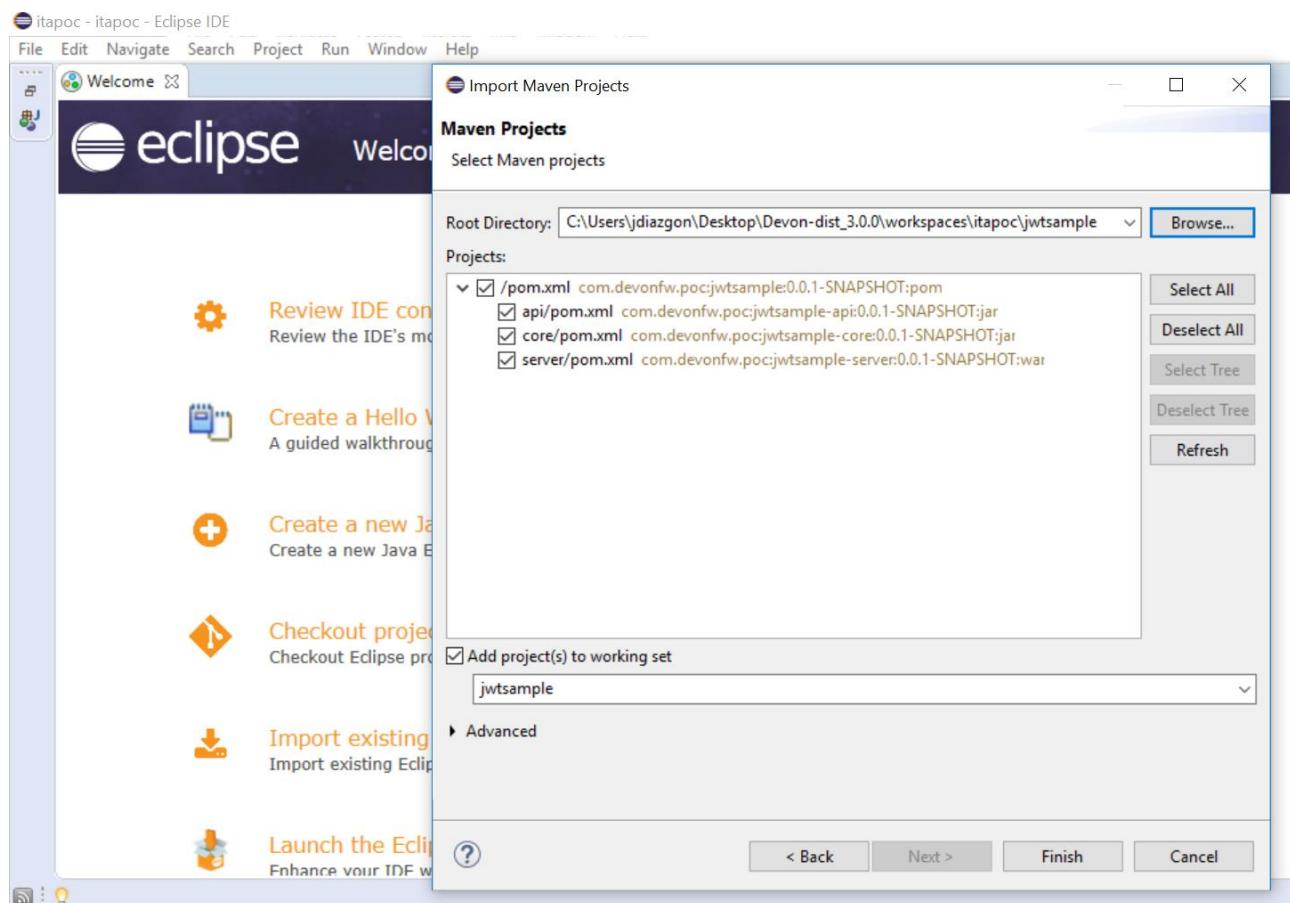
3. **Copy** the Devon-dist_3.0.0\workspaces\examples\jwtsample directory in a new dir (i.e: for thisPoC WE will use Devon-dist_3.0.0\workspaces\itapoc)
4. Now run **update-all-workspaces.bat**
5. Execute **eclipse-<<directory name>>.bat** from Devon-dist_N.m.m folder (i.e: **eclipse-itapoc.bat**)



It will open eclipse for you

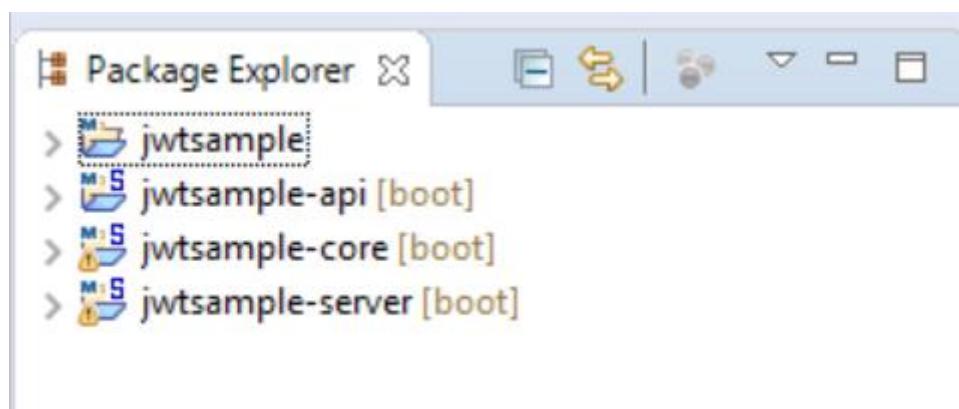


6. Now import the Projects pointing to the Devon-dist_3.0.0\workspaces\itapoc\jwtssample directory



Click **FINISH**

Now We have the following 4 projects.



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

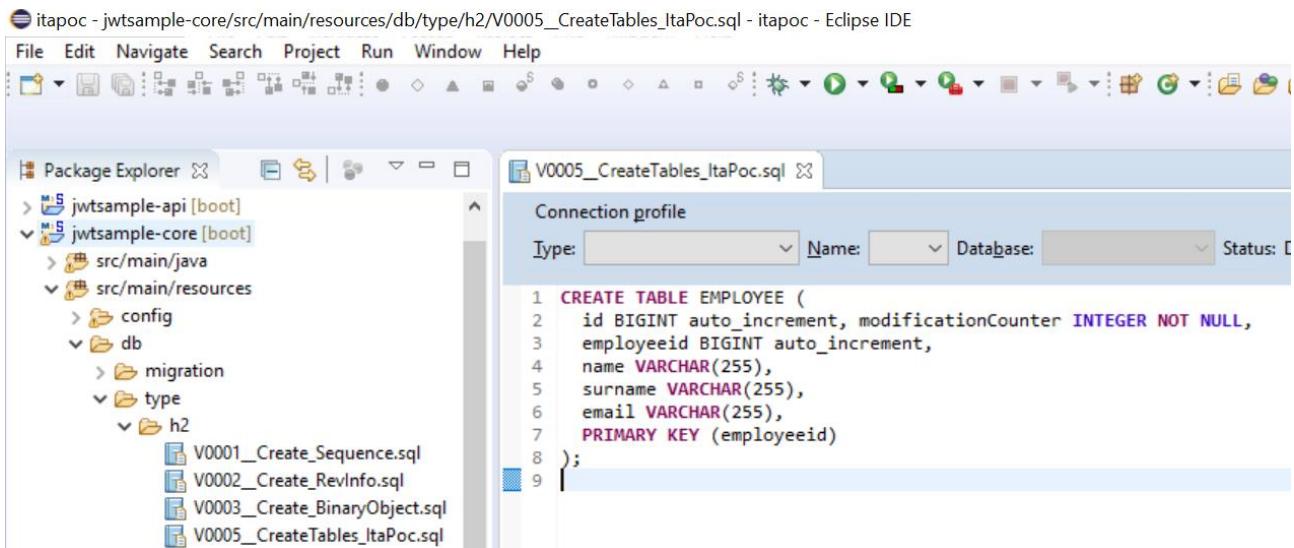
7. Create a new **SQL file** (i.e: V0005__CreateTables_ItaPoc.sql) inside *jwtssample-core* and insert the following script:

```

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 !



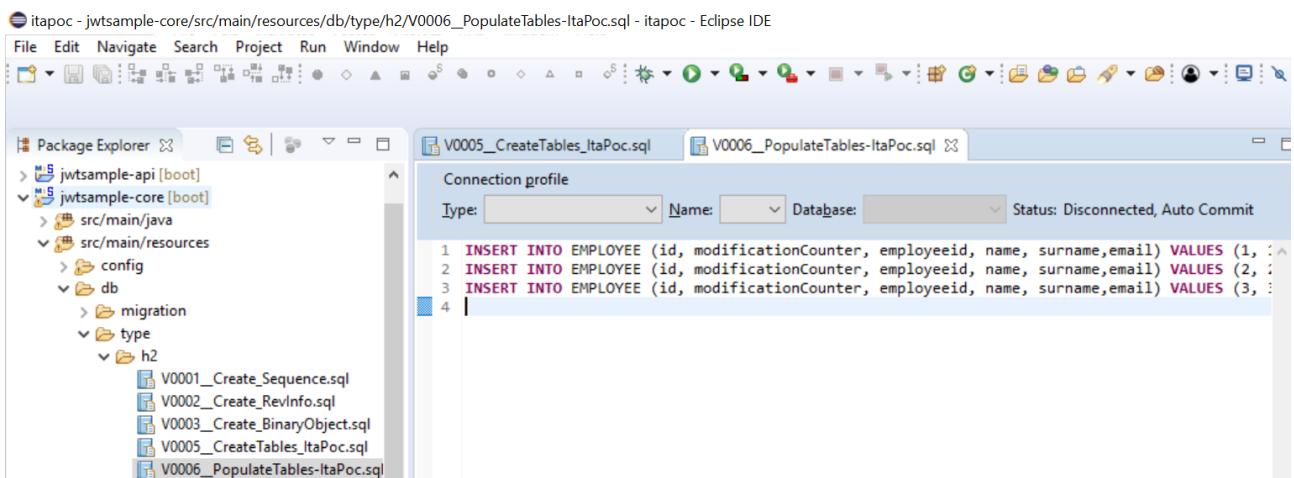
- Now create another SQL file (i.e: V0006__PopulateTables-ItaPoc.sql) and add following script about the INSERT in order to populate the table created before

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

```

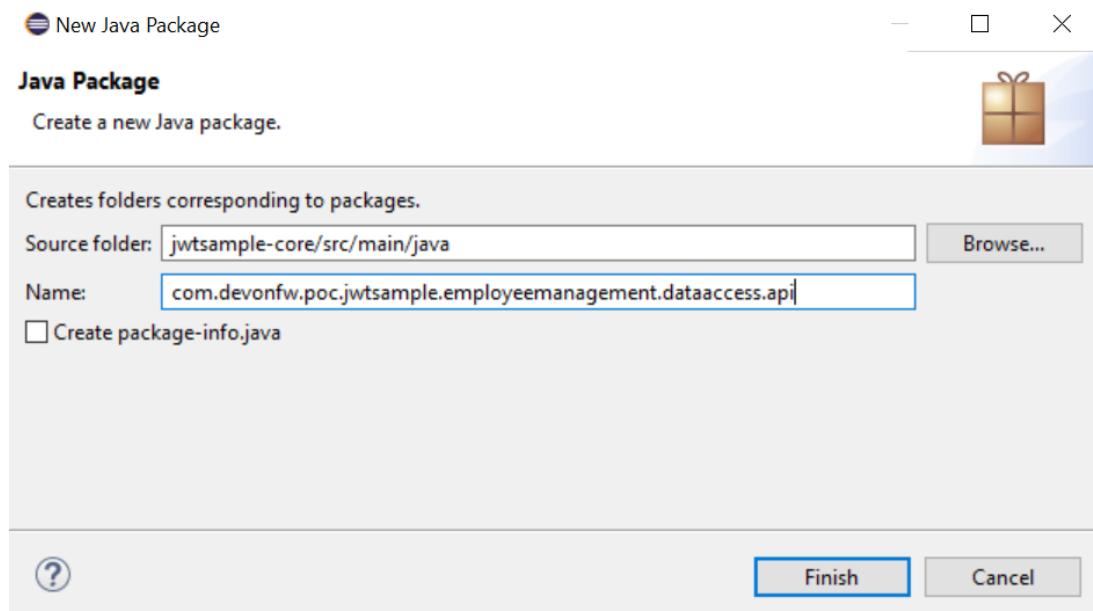
INSERT INTO EMPLOYEE (id, modificationCounter, employeeid, name, surname, email) VALUES
(1, 1, 1, 'Stefano', 'Rossini', 'stefano.rossini@capgemini.com');
INSERT INTO EMPLOYEE (id, modificationCounter, employeeid, name, surname, email) VALUES
(2, 2, 2, 'Angelo', 'Muresu', 'angelo.muresu@capgemini.com');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

```

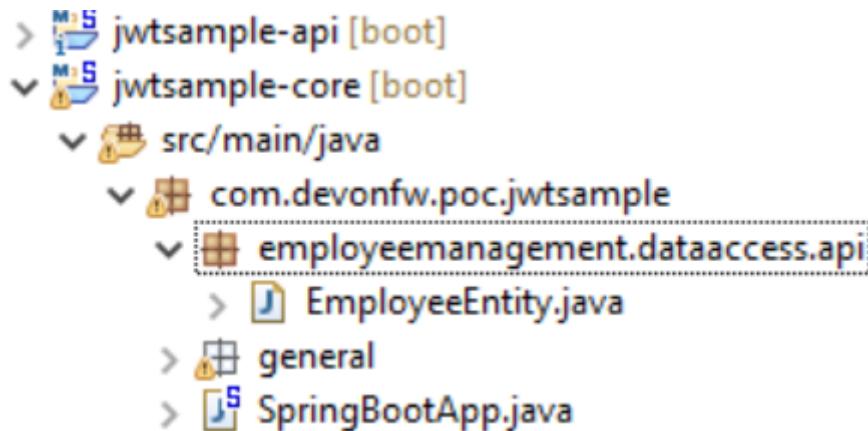


NOW you can create the Entity class 😊

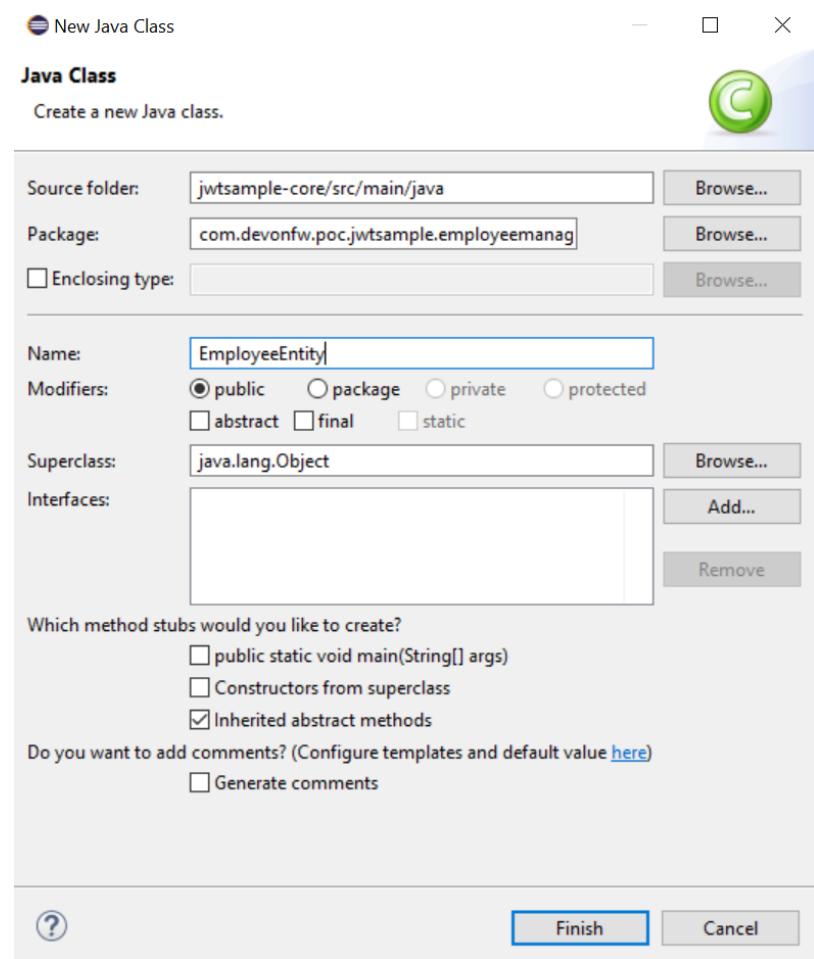
9. First of all create a package under the folder "jwtsample-core"



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;
}

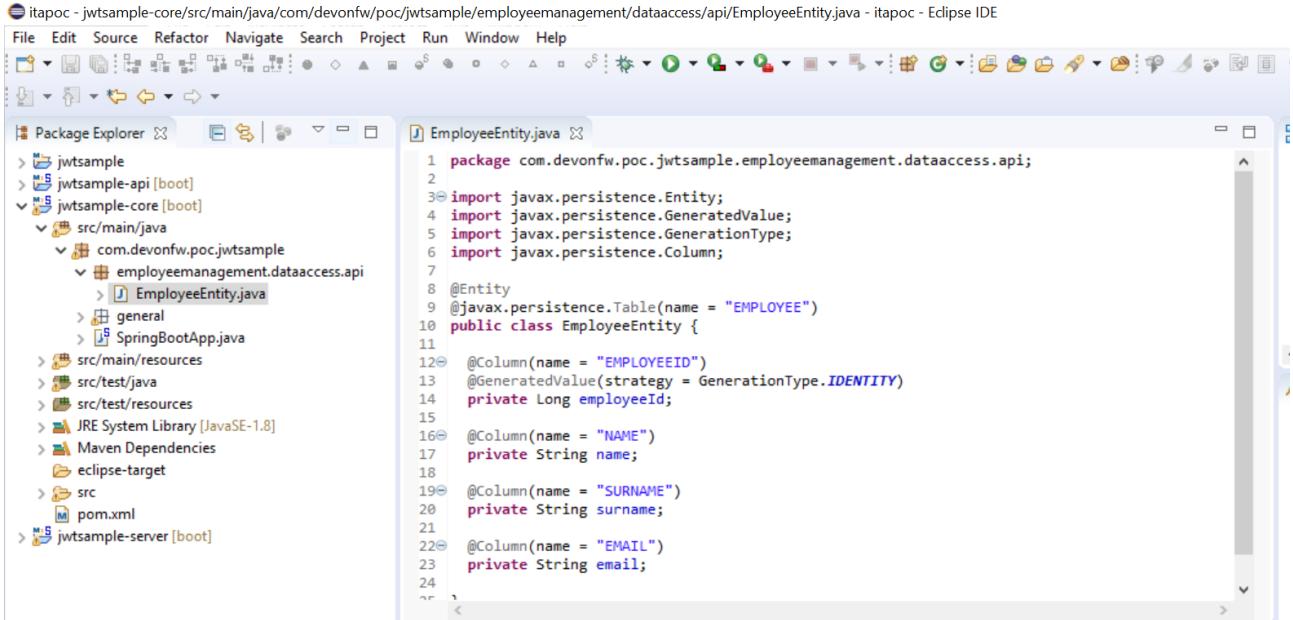
```

```

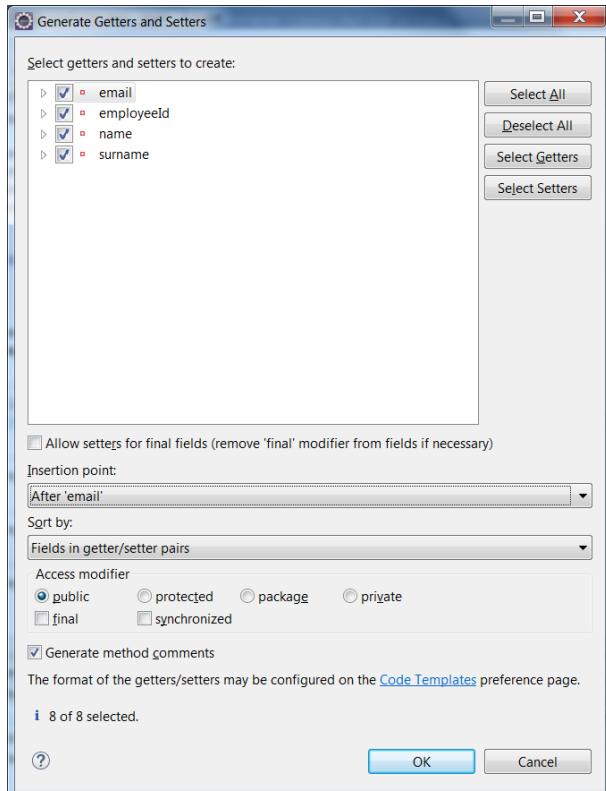
    @Column(name = "EMAIL")
    private String email;

}

```

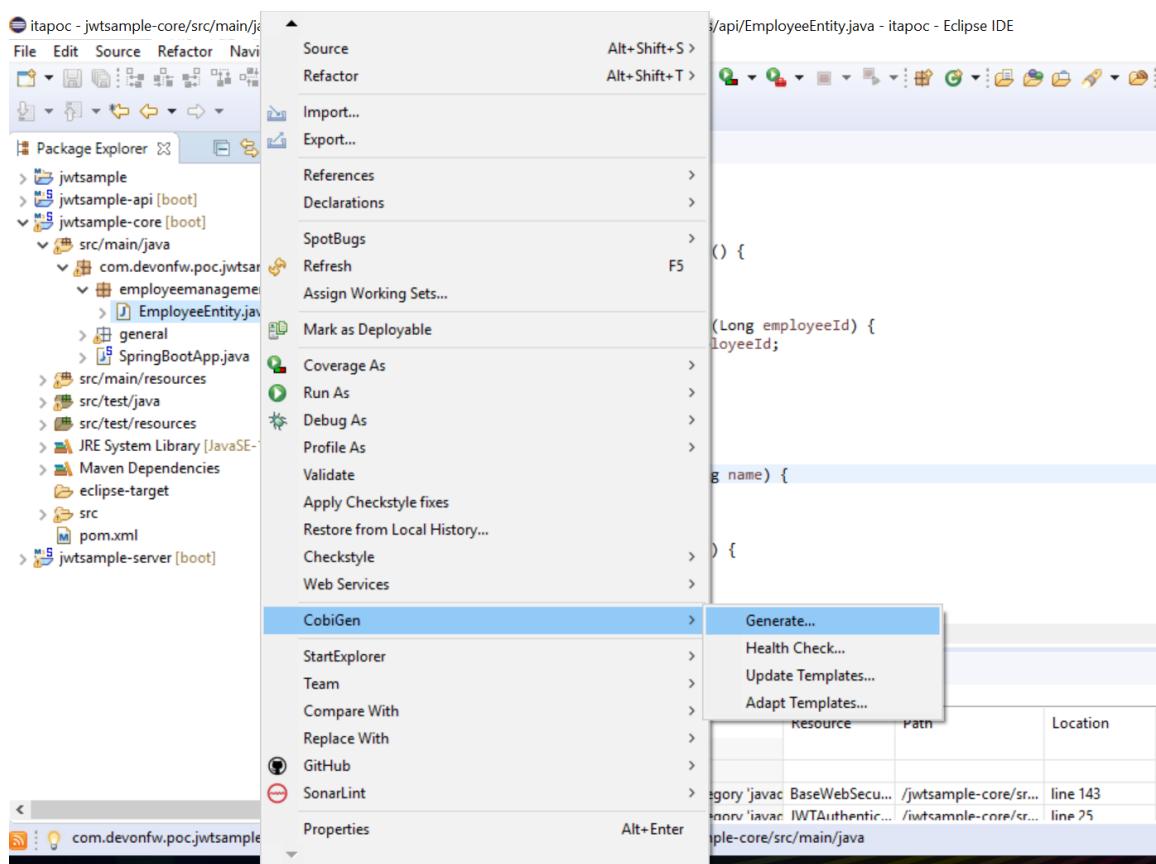


and then generate **getters** and **setters** for all attributes ...

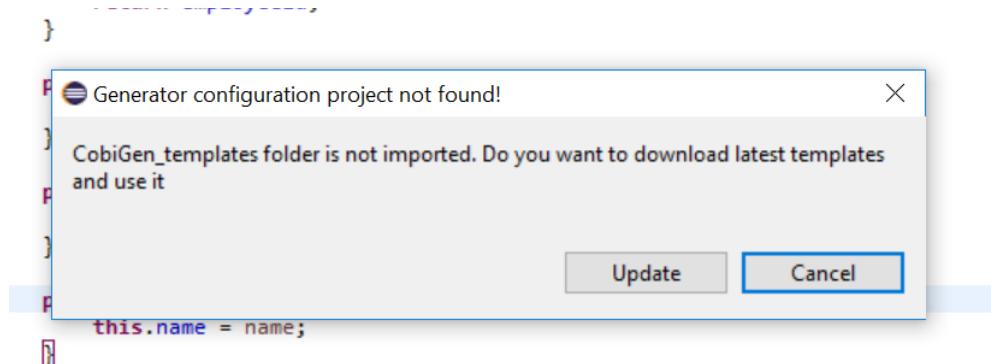


11. Once Getter and Setter are done, we can now **use CobiGen** (do it if u need to use code generator !)

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

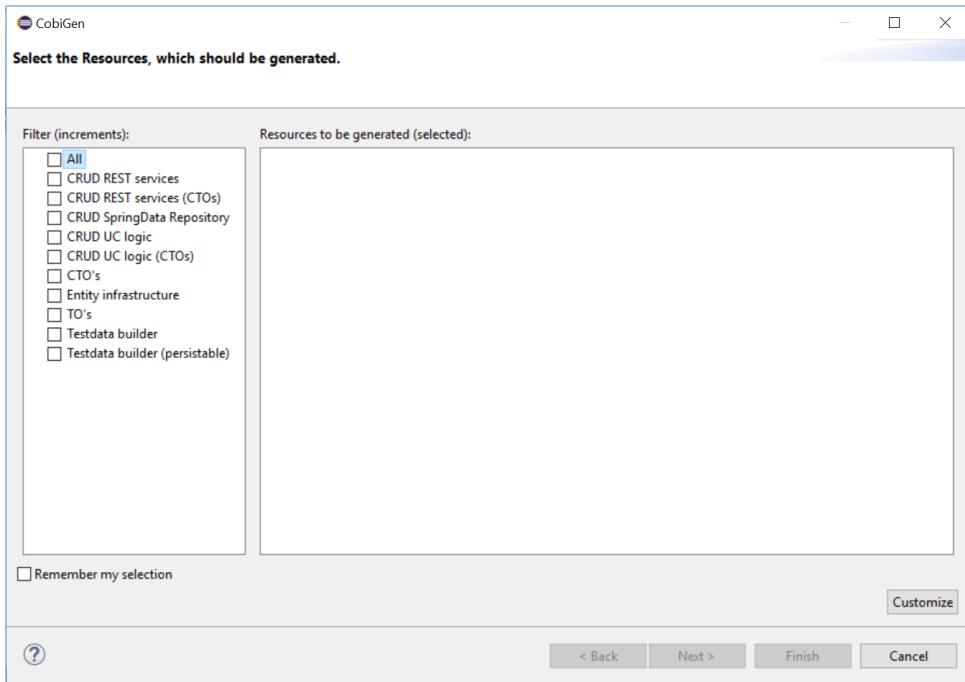


It will ask you to download the templates, click on *update*:

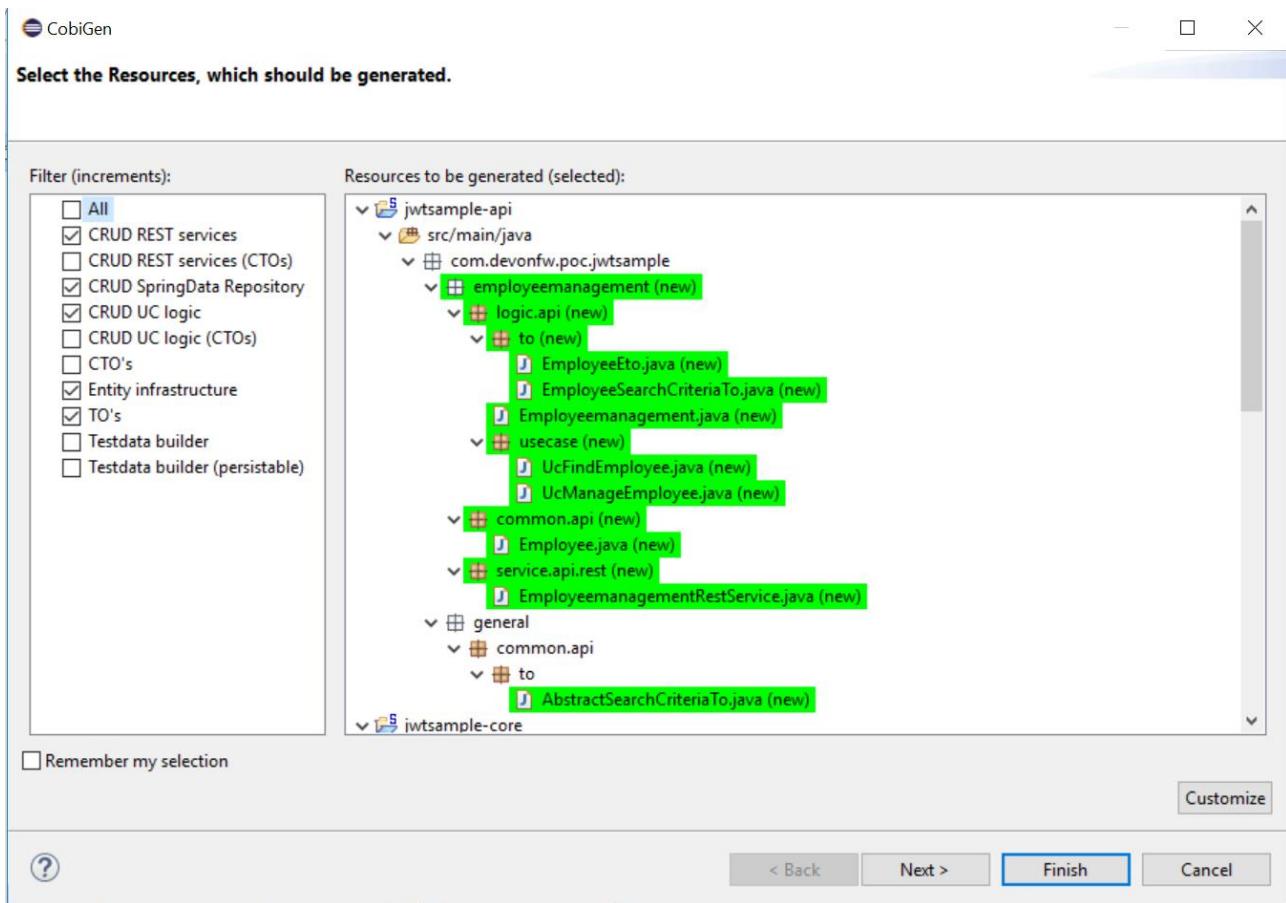


It will automatically download the latest version of *CobiGen_Templates*. After that, it will show you the next window:

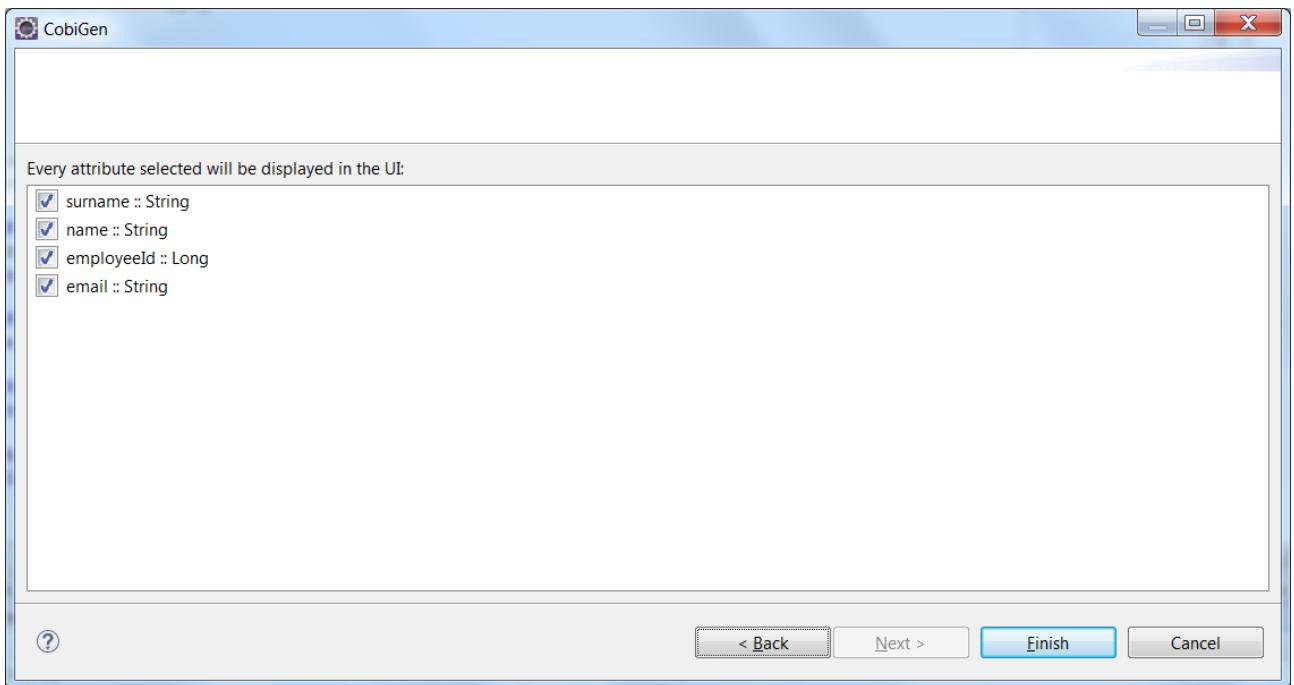
Attention: If you want to adapt the *CobiGen_Templates*, (normally this is not necessary), you will find at the end of this document a tutorial on how to import them and adapt them!



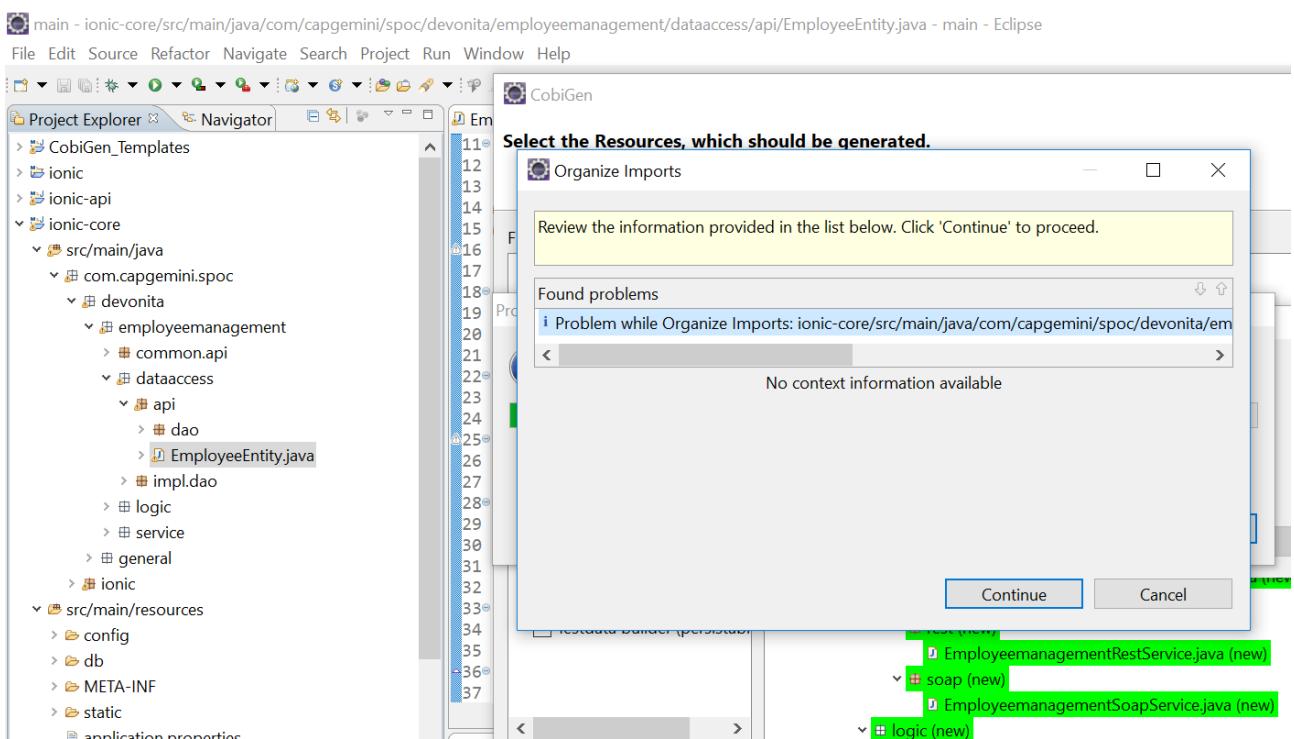
12. Click on all the option selected as below:



13. Click on Next

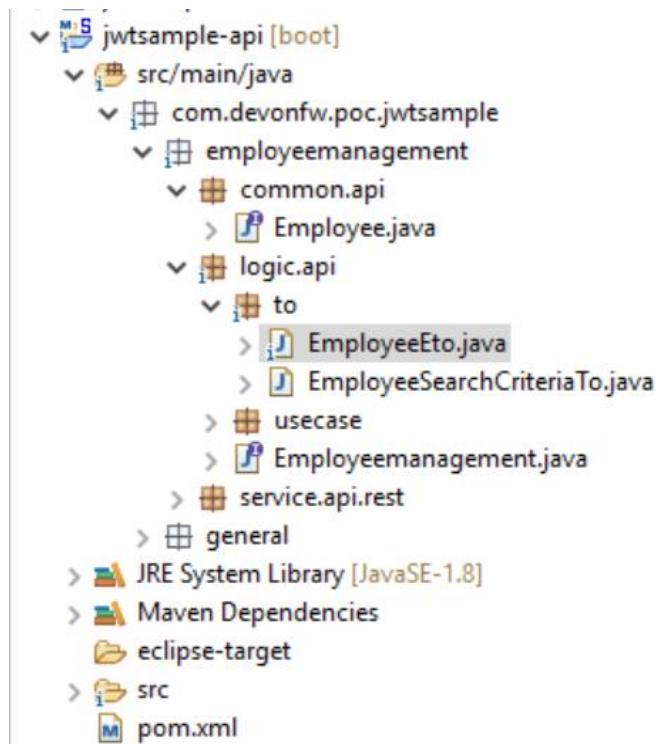


14. Click on finish. Below Screen would be seen. Click on continue

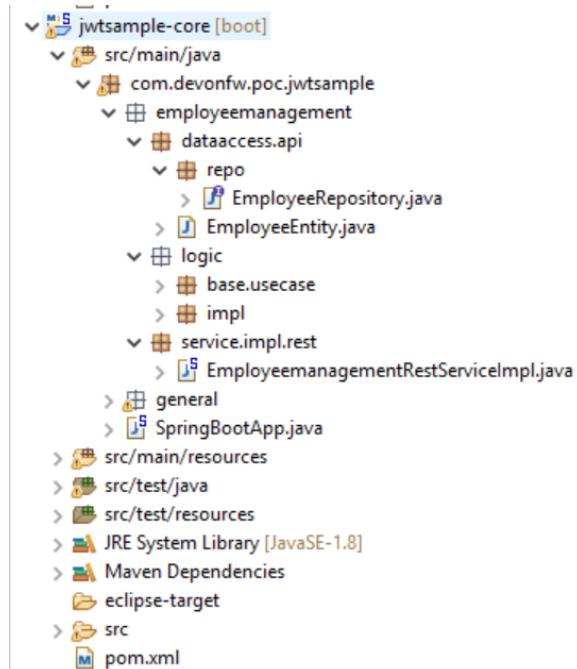


The entire BE layer structure having CRUD operation methods will be auto generated.

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 {
```

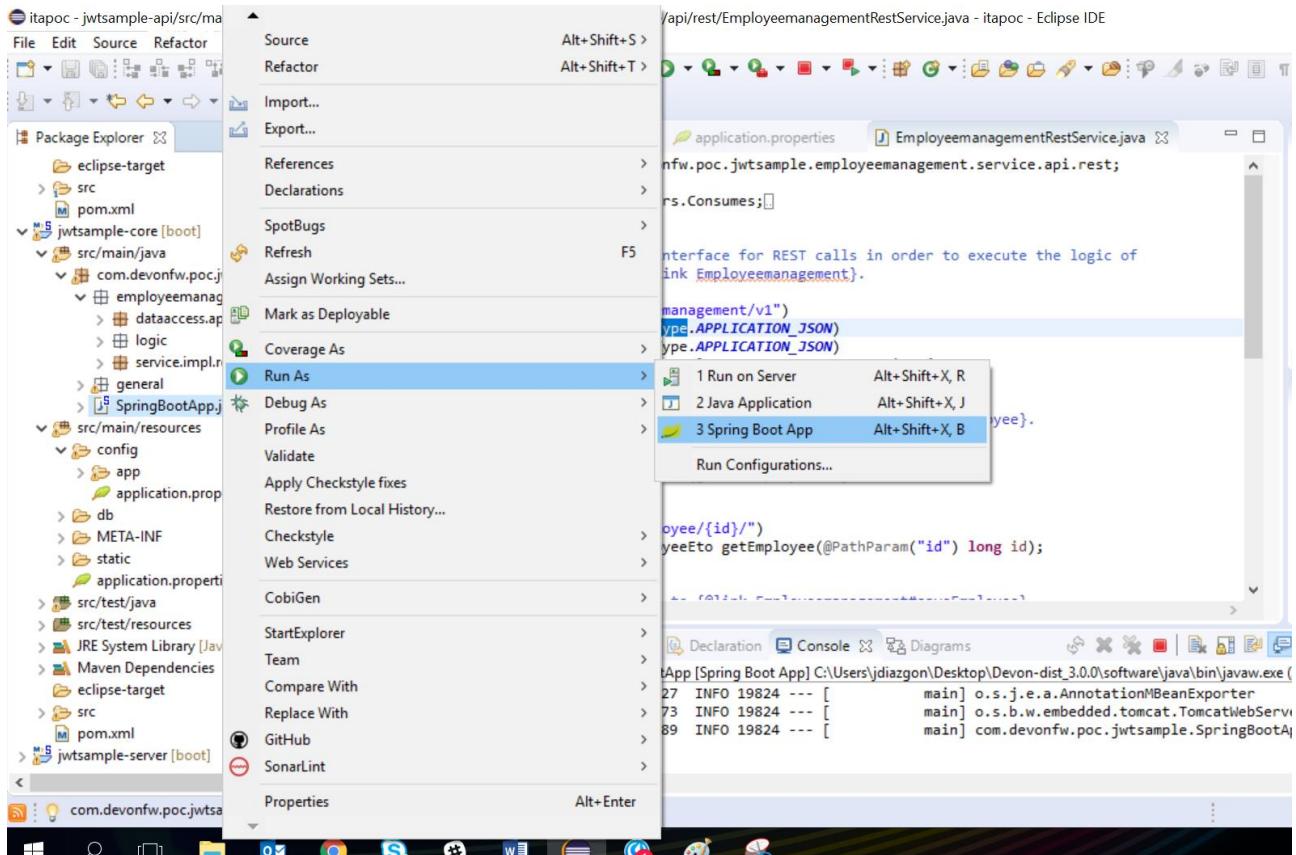
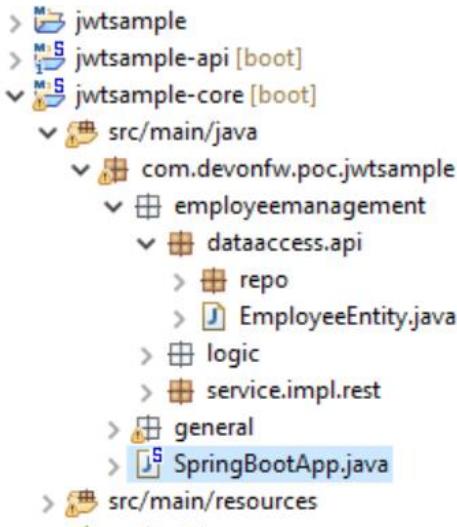
to:

```
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*



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "itapoc - ionic-core".
- Code Editor:** Displays the code for `EmployeeEntity.java`, which includes imports for javax.persistence.Column, Entity, GeneratedValue, GenerationType, Employee, and ApplicationPersistenceEntity, along with annotations like `@Entity` and `@Table(name = "EMPLOYEE")`.
- Console:** Shows the Spring Boot application log output:

```

2018-10-03 15:28:08.788 INFO 11784 --- [           main] com.capgemini.spc.ionic.SpringBootApp : Starting SpringBootApp on LIT
2018-10-03 15:28:08.794 INFO 11784 --- [           main] com.capgemini.spc.ionic.SpringBootApp : No active profile set, falling back to default profiles: default
2018-10-03 15:28:09.138 INFO 11784 --- [           main] o.s.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@6a1...

```

BE DONE 😊

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure for "jwtsample-api" and "jwtsample-core".
- Code Editor:** Displays the `application.properties` file content:

```

1# This is the spring boot configuration file for development. It will not be included.
2# In order to set specific configurations in a regular installed environment create an
3# config/application.properties in the server. If you are deploying the application to
4# WAR file you can locate this config folder in ${symbol_dollar}{CATALINA_BASE}/lib. I
5# the same container (not recommended by default) you need to ensure the WARs are extr
6# the config folder inside the WEB-INF/classes folder of the webapplication.
7
8server.port=8081
9server.servlet.context-path=
10
11# Datasource for accessing the database
12# See https://github.com/devonfw-wiki/devon4j/wiki/guide-configuration#security-config
13#jaspyt.encryptor.password=null
14#spring.datasource.password=ENC(7CnHiadYc0Wh2FnWADNjJg==)
15spring.datasource.password=
16spring.datasource.url=jdbc:h2:./.jwtsample;
17
18# Enable JSON pretty printing
19spring.jackson.serialization.INDENT_OUTPUT=true
20
21# Flyway for Database Setup and Migrations
22spring.flyway.enabled=true
23spring.flyway.clean-on-validation-error=true

```

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)

```

itapoc - jwtsample-api/src/main/java/com/devonfw/poc/jwtsample/employeemanagement/service/api/rest/EmployeemanagementRestService.java - itapoc - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer EmployeeEntity.java application.properties EmployeemanagementRestService.java
jwtsample jwtsample-api [boot]
src/main/java com.devonfw.poc.jwtsample employeemanagement common.api logic.api to EmployeeEto.java EmployeeSearchCriteriaTo.java usecase Employeemanagement.java service.api.rest EmployeemanagementRestService.java general JRE System Library [JavaSE-1.8] Maven Dependencies eclipse-target src pom.xml jwtsample-core [boot]
1 package com.devonfw.poc.jwtsample.employeemanagement.service.api.rest;
2
3 import javax.ws.rs.Consumes;
4
5 /**
6  * The service interface for REST calls in order to execute the logic of
7  * component {@link Employeemanagement}.
8  */
9 @Path("/employeemanagement/v1")
10 @Consumes(MediaType.APPLICATION_JSON)
11 @Produces(MediaType.APPLICATION_JSON)
12 public interface EmployeemanagementRestService {
13
14     /**
15      * Delegates to {@link Employeemanagement#findEmployee}.
16      *
17      * @param id the ID of the {@link EmployeeEto}
18      * @return the {@link EmployeeEto}
19      */
20     @GET
21     @Path("/employee/{id}")
22     public EmployeeEto getEmployee(@PathParam("id") long id);
23
24     /**
25      * ...
26      */
27 }

```

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”

Postman

New Import Runner +

My Workspace Invite

Examples (0) Examples (0)

History Collections

Nothing in your history yet. Requests that you send through Postman are automatically saved here.

POST http://localhost:8081/services/rest/login

Authorization Headers (1) Body Pre-request Script Tests

Body (1)

form-data x-www-form-urlencoded raw binary JSON (application/json)

```

1 {
2   "j_username": "waiter",
3   "j_password": "waiter"
4 }
5
6

```

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

POST http://localhost:8081/services/rest/login

Headers (10)

Status: 200 OK Time: 494 ms Size: 591 B

Access-Control-Expose-Headers → Authorization

Authorization → Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJ3YWI0ZXliLCJzY29wZSI6W3siYXV0aG9yaXR5IjoiUk9MRV9XYWI0ZXifv0sImlzcyI6Ik9hc3A0ai1TYW1wbGUtU2VydmlvliwiZXhwIjoxNTI

Cache-Control → no-cache, no-store, max-age=0, must-revalidate

Content-Length → 0

Date → Thu, 04 Oct 2018 15:38:10 GMT

Expires → 0

Pragma → no-cache

X-Content-Type-Options → nosniff

... 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)

Authorization

Headers

Body

Pre-request Script

Tests

Cookies

Code

TYPE

Bearer Token

Token

eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJ3YWI0ZXliLCJzY29wZSI6W3siYXV0aG9yaXR5IjoiUk9MRV9XYWI0ZXifv0sImlzcyI6Ik9hc3A0ai1TYW1wbGUtU2VydmlvliwiZXhwIjoxNTM4NjcxMDkwLCJpYXQiOjE1Mzg2Njc0OTB9.6wZC4zA5EsLFNuTaQ9tqfeKSFR1F5Vm_WPVN7sWq3Dp7lg817sauErIP2lq2AQLkjD2uNzoBHg8OZLsnUAEA|

Preview Request

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

Angular BE test

POST http://localhost:8081/services/r

POST http://localhost:8081/services/r

POST http://localhost:8081/services/r

CSRF login

http://localhost:8081/services/rest/employeemanagement/v1/employee/search

POST http://localhost:8081/services/rest/employeemanagement/v1/employee/search

Send

Params

Authorization

Headers (2)

Body

Pre-request Script

Tests

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```

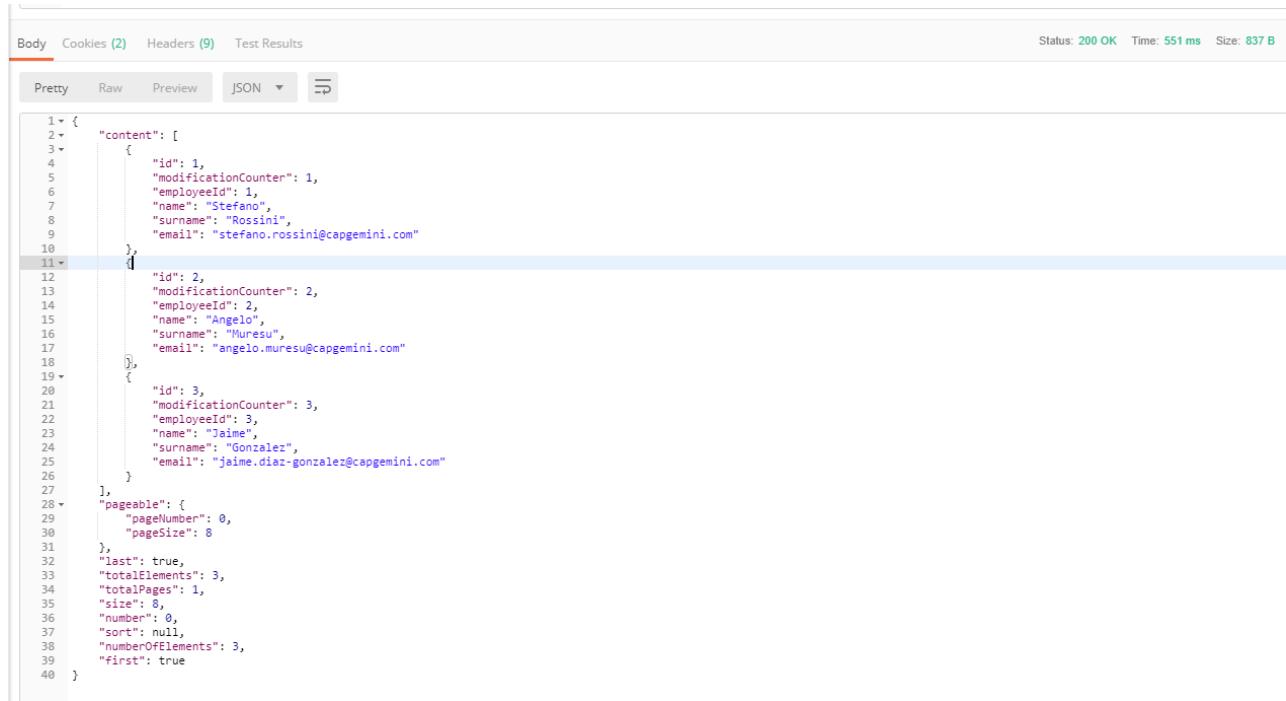
1 {
2   "pageable": {
3     "pageSize": 8,
4     "pageNumber": 0
5   }
6 }

```

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
😊



The screenshot shows a Postman request result. At the top, there are tabs for Body, Cookies (2), Headers (9), and Test Results. On the right, it displays Status: 200 OK, Time: 551 ms, and Size: 837 B. Below these, there are buttons for Pretty, Raw, Preview, and JSON, with JSON currently selected. The JSON response is displayed in a code editor-like interface with line numbers from 1 to 40. The response content is as follows:

```
1 {  
2   "content": [  
3     {  
4       "id": 1,  
5       "modificationCounter": 1,  
6       "employeeId": 1,  
7       "name": "Stefano",  
8       "surname": "Rossini",  
9       "email": "stefano.rossini@capgemini.com"  
10    },  
11    {  
12      "id": 2,  
13      "modificationCounter": 2,  
14      "employeeId": 2,  
15      "name": "Angelo",  
16      "surname": "Muresu",  
17      "email": "angelo.muresu@capgemini.com"  
18    },  
19    {  
20      "id": 3,  
21      "modificationCounter": 3,  
22      "employeeId": 3,  
23      "name": "Jaime",  
24      "surname": "Gonzalez",  
25      "email": "jaime.diaz-gonzalez@capgemini.com"  
26    },  
27  ],  
28  "pageable": {  
29    "pageNumber": 0,  
30    "pageSize": 8  
31  },  
32  "last": true,  
33  "totalElements": 3,  
34  "totalPages": 1,  
35  "size": 8,  
36  "number": 0,  
37  "sort": null,  
38  "numberOfElements": 3,  
39  "first": true  
40 }
```

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)

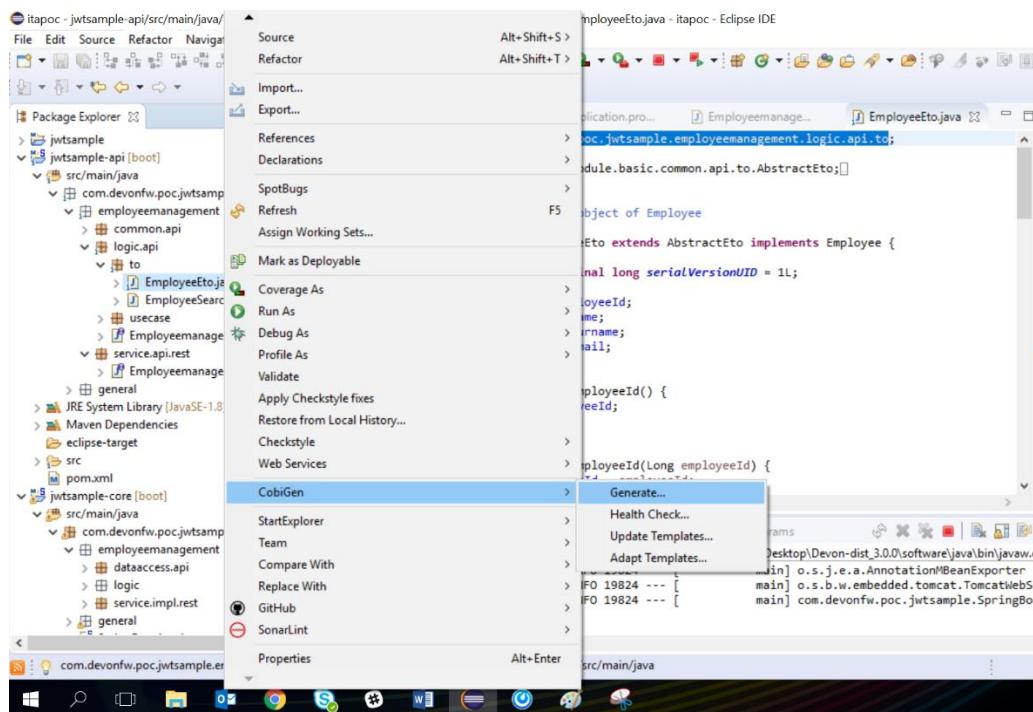
Let's start now with FE with angular Web and then Ionic app.

2.2.1 Front End Web App Angular

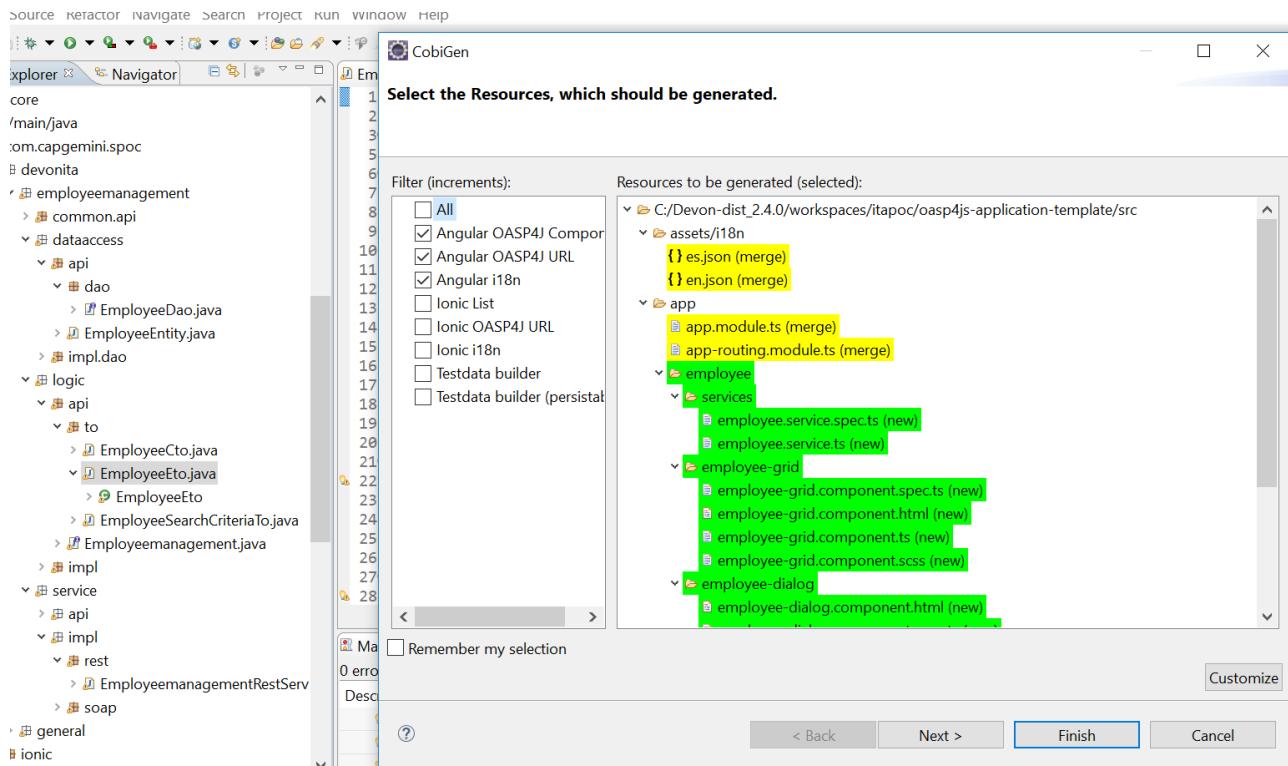
- Now, for the Angular structure to be auto-generated, first copy the “**devon4ng-application-template**” project from the location
(Path location \Devon-dist_X.X.0\Devon-dist_X.X.0\workspaces\examples) and paste it in the your project location (i.e: \Devon-dist_3.0.0\workspaces\itapoc)

Name	Date
.metadata	21/12
.sonarlint	21/12
devon4ng-application-template	21/12
jwtsample	21/12
RemoteSystemsTempFiles	21/12

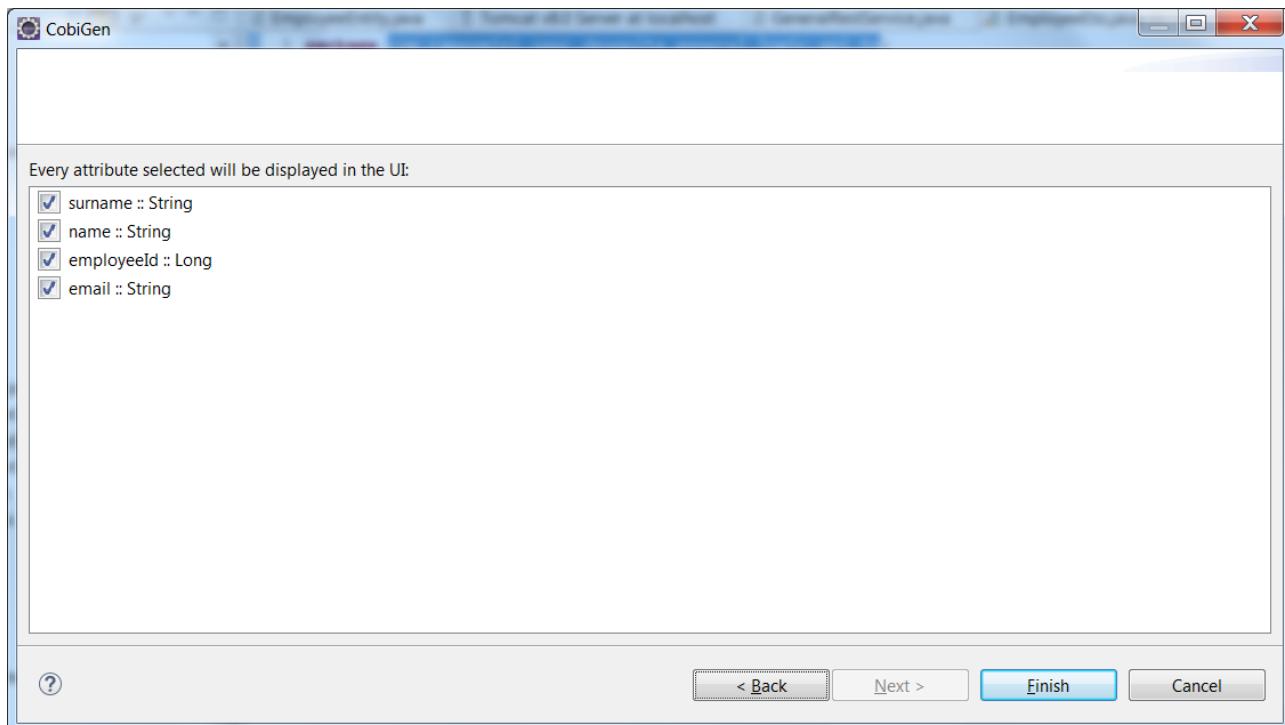
- Once done, right click on the **EmployeeEto.java** (that is the Entity transport object) file present under the package “com.devonfw.poc.jwtsample.employeemanagement.logic.api.to”



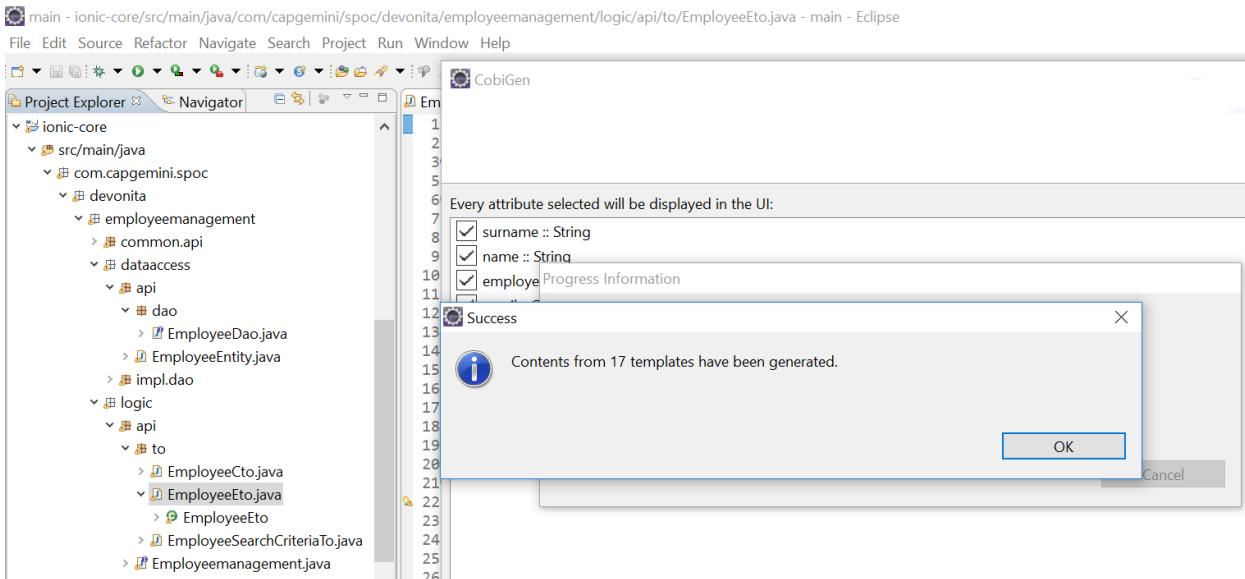
- Click on the selected options as seen in the screenshot:



4. Click on Next

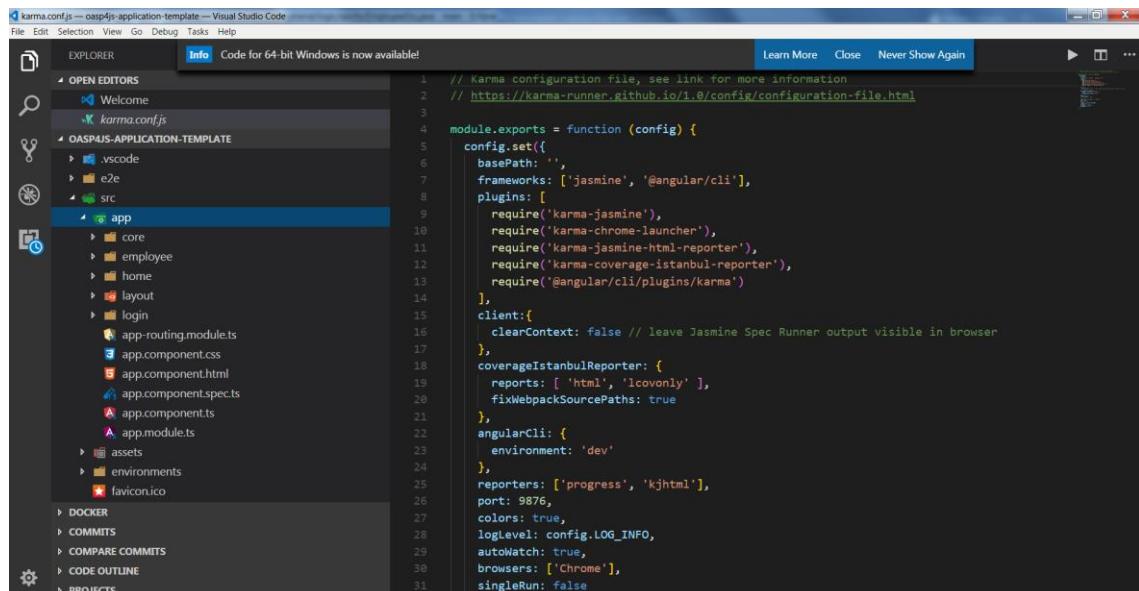


5. Click on Finish



- The entire ANGULAR structure has been auto generated.

The entire Angular FE layer structure having CRUD operation methods will be auto generated.



- 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

```

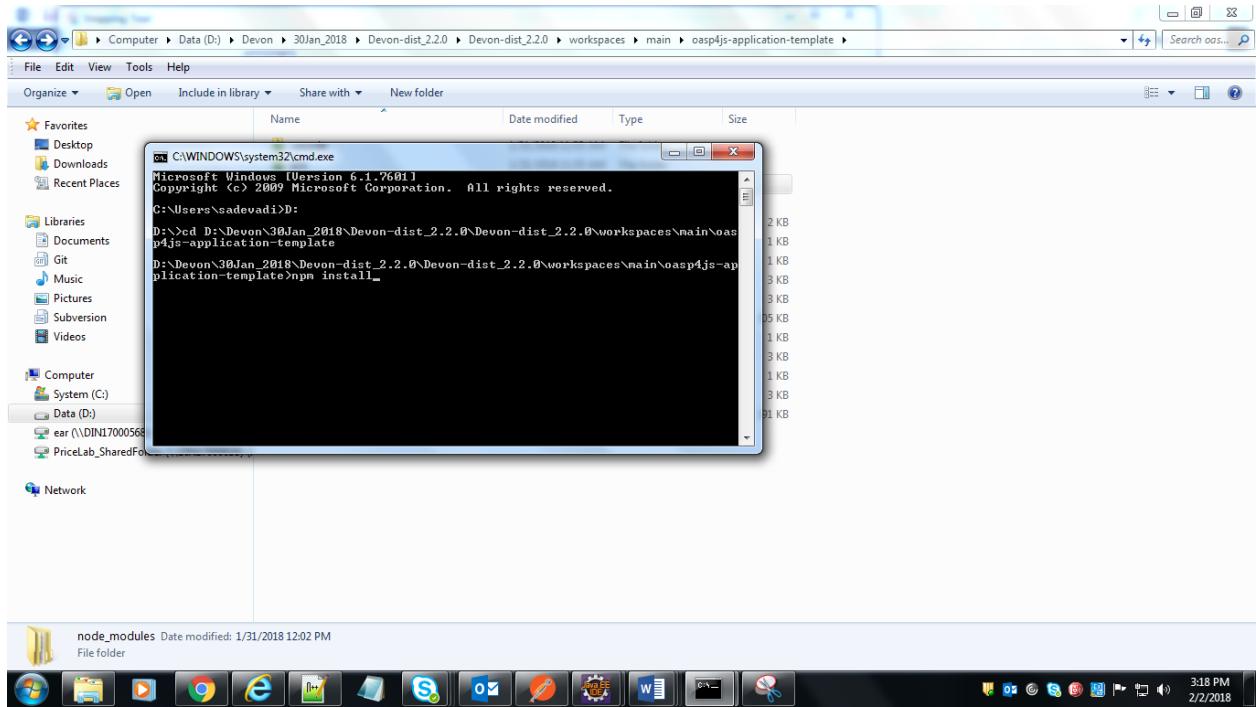
, {
  path: 'employee',
  component: EmployeeGridComponent,
  canActivate: [AuthGuard],
},

```

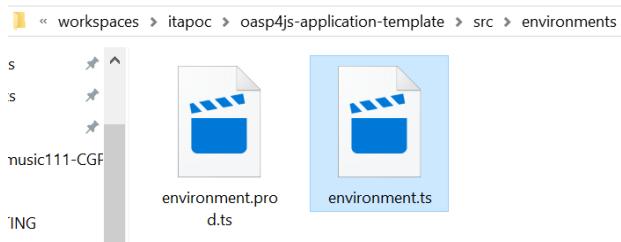
Following picture explain where to place the above content:

```
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'
  }
]
```

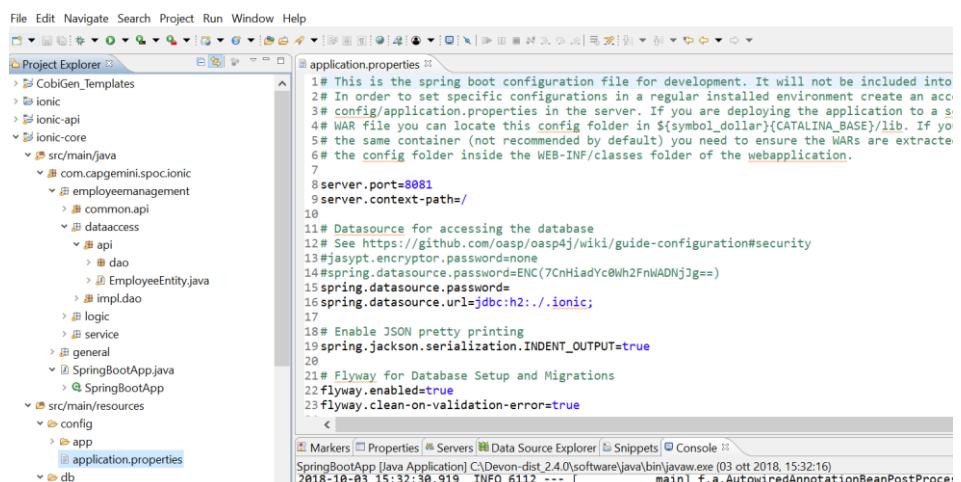
8. Now execute **console.bat** present on the path C:\Devon-dist_3.0.0\console.bat. It will open a console. Then type the path where your “devon4ng-application-template” project is present (i.e: C:\Devon-dist_3.0.0\workspaces\itapoc\devon4ng-application-template) and execute “npm install” command which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



- 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,
```

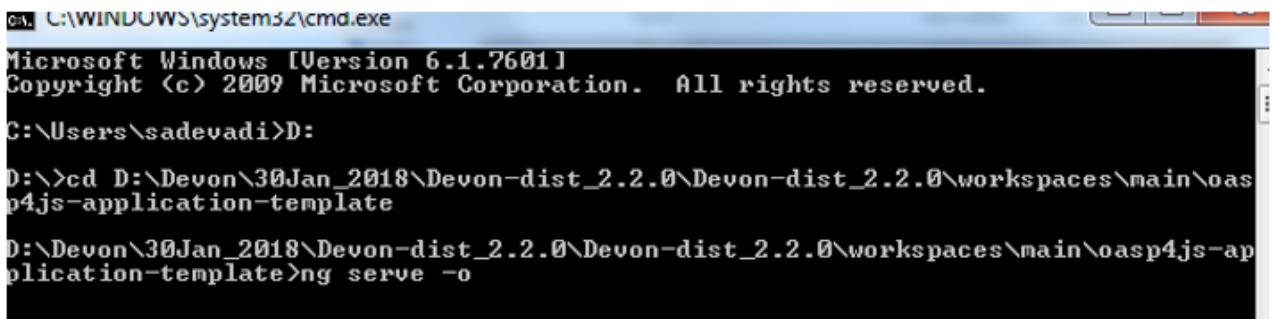
```

    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.



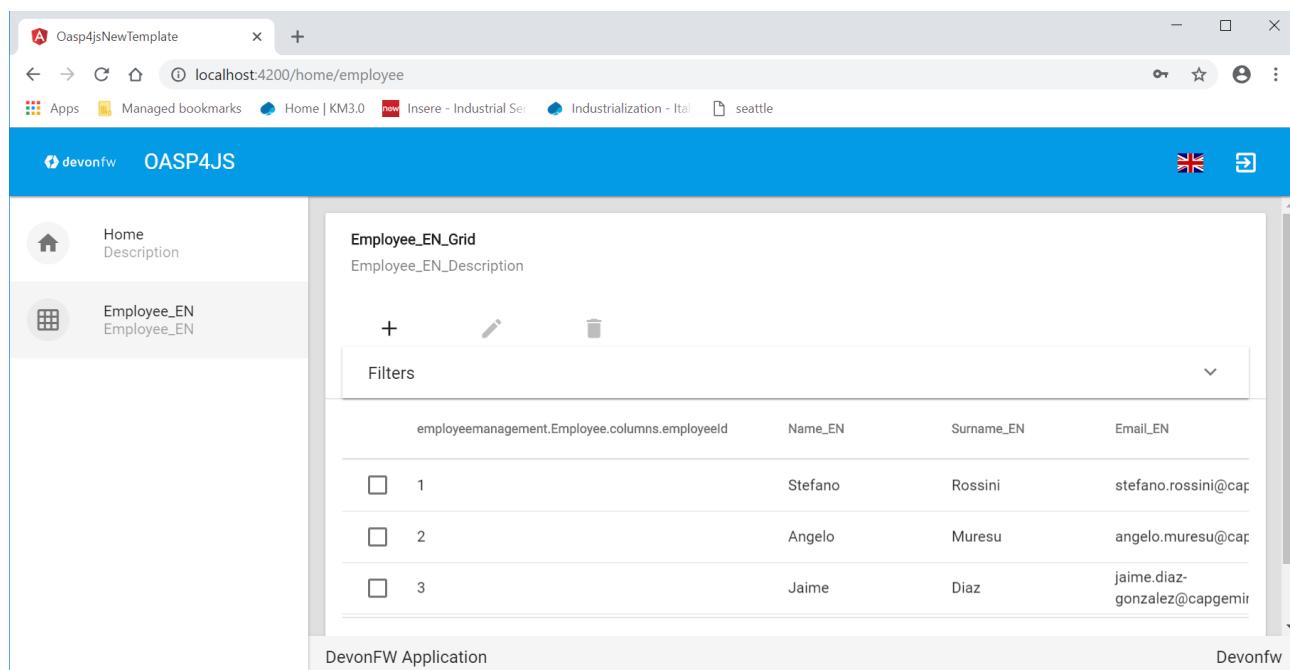
```

C:\ C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\sadevadi>D:
D:\>cd D:\Devon\30Jan_2018\Devon-dist_2.2.0\Devon-dist_2.2.0\workspaces\main\oasp4js-application-template
D:\Devon\30Jan_2018\Devon-dist_2.2.0\Devon-dist_2.2.0\workspaces\main\oasp4js-application-template>ng serve -o

```

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>



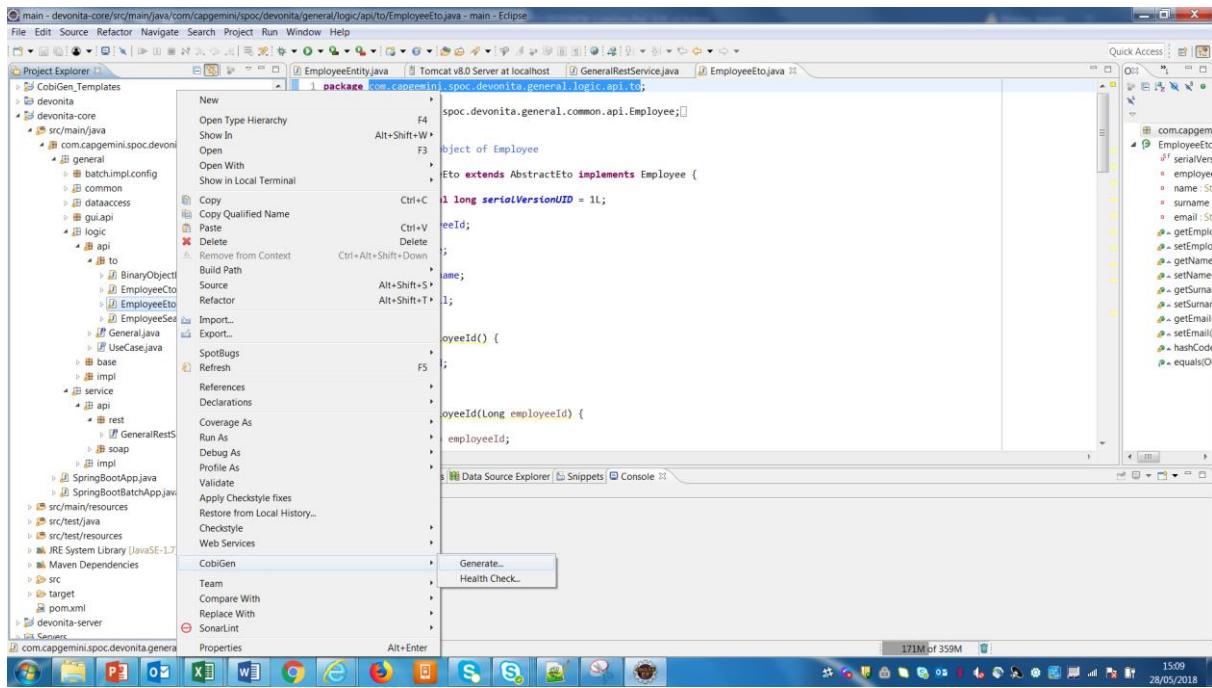
employeemanagement.Employee.columns.employeedId	Name_EN	Surname_EN	Email_EN
1	Stefano	Rossini	stefano.rossini@capgemini.com
2	Angelo	Muresu	angelo.muresu@capgemini.com
3	Jaime	Diaz	jaime.diaz-gonzalez@capgemini.com

FE WebApp DONE 😊

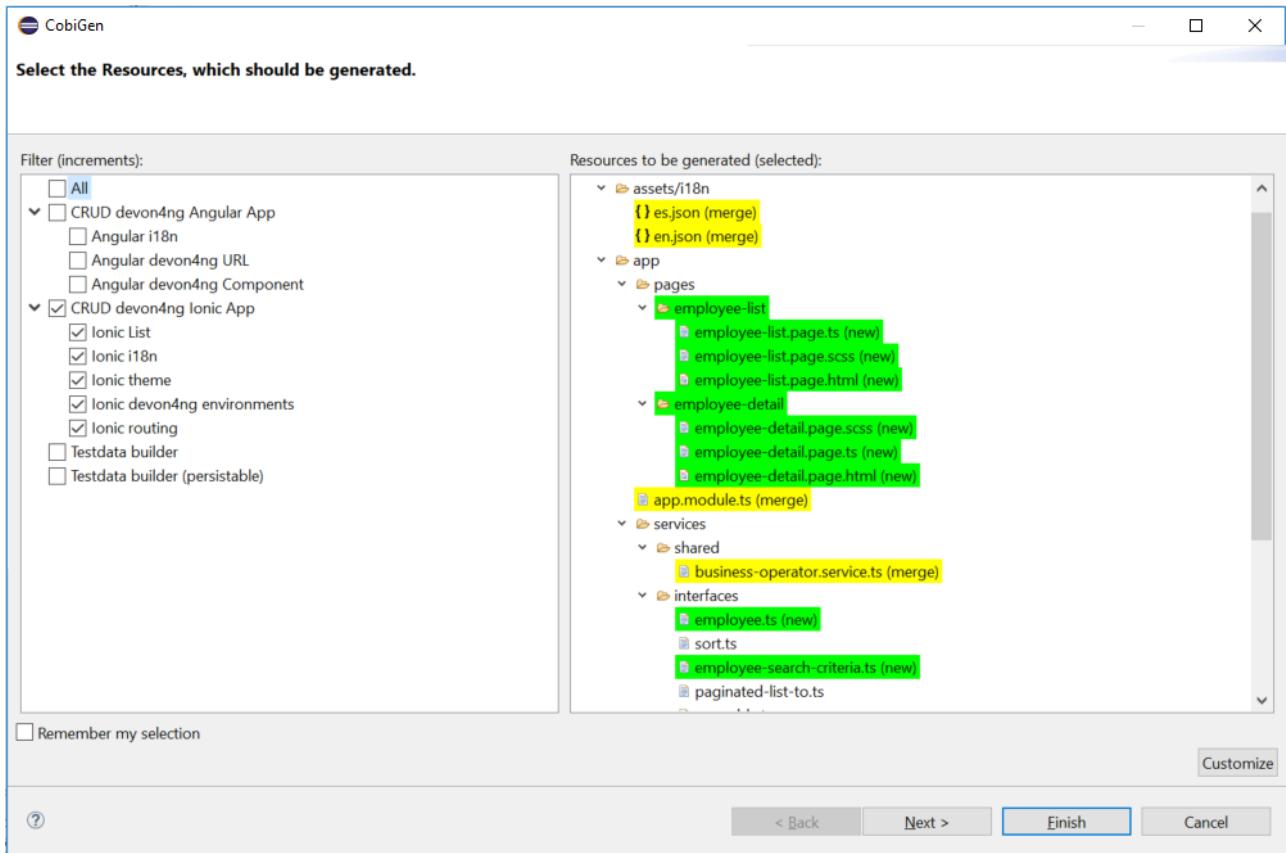
2.2.2 Front End Mobile App

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).

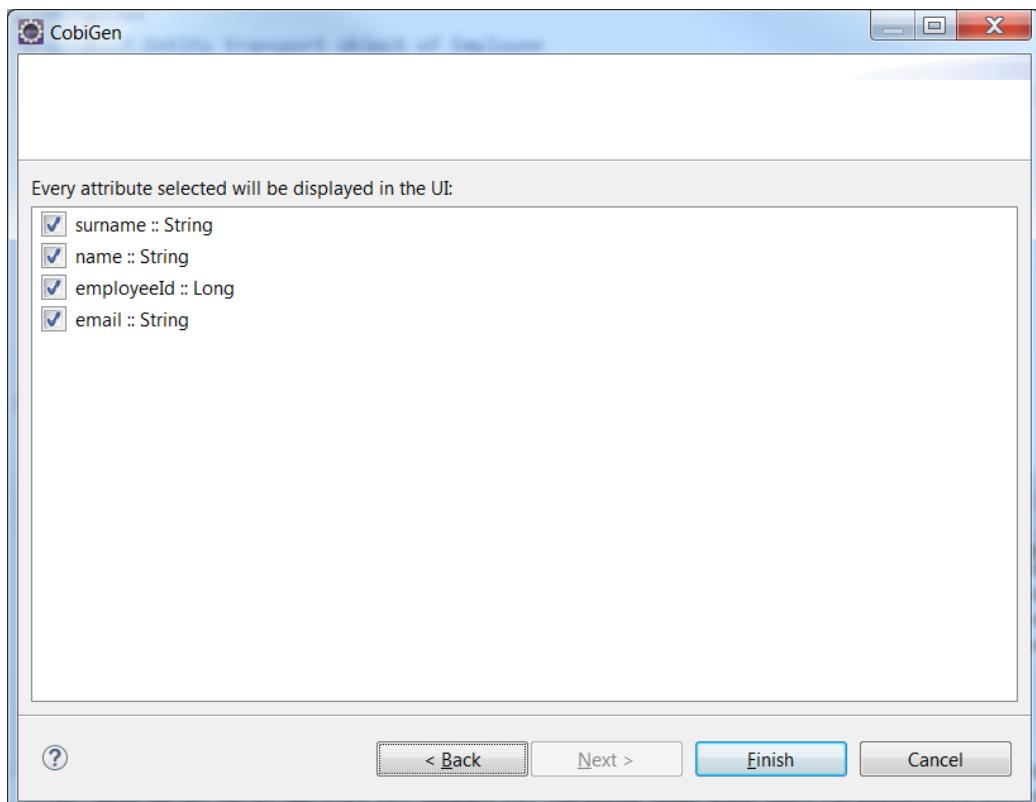
2. Once done, Right click on the **EmployeeEto.java** as you already did before in order to use CobiGen.



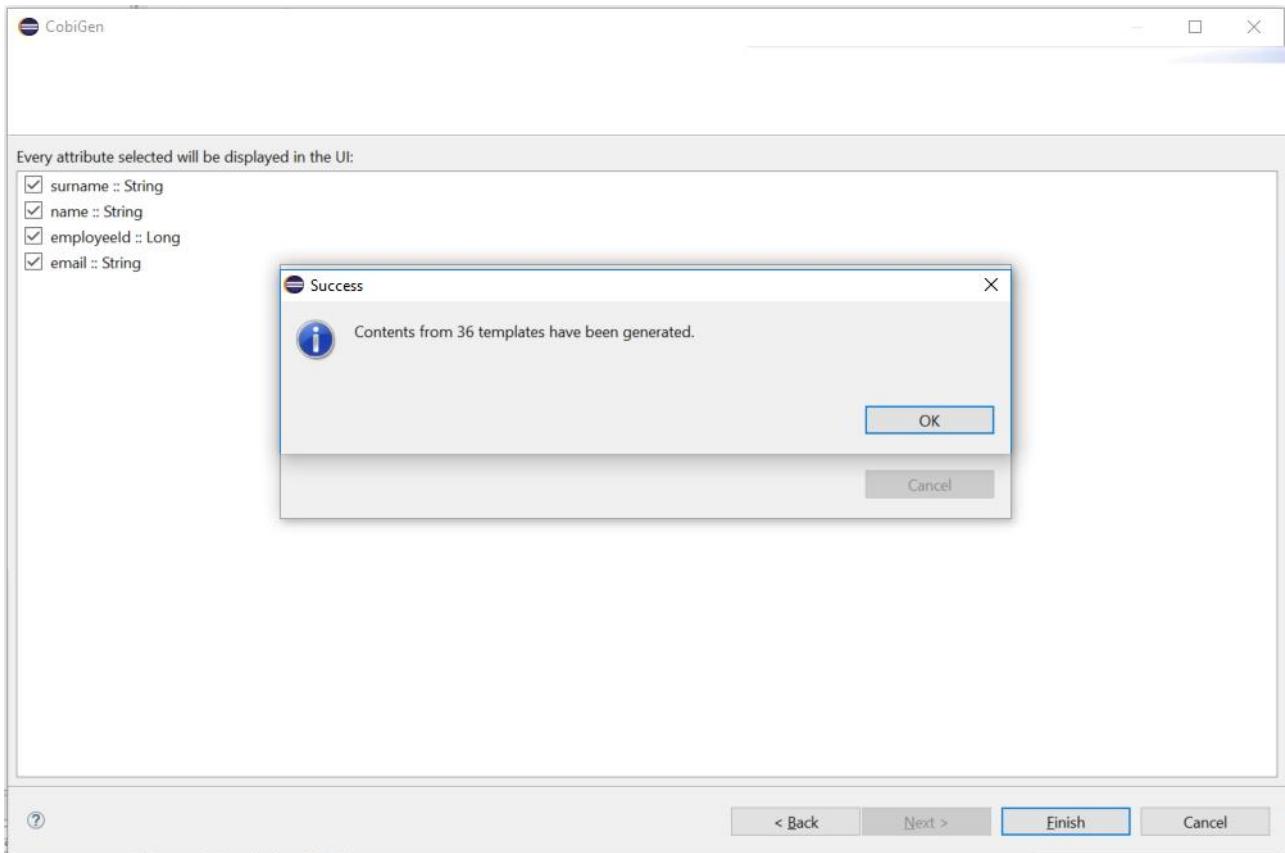
3. Click on the selected options as seen in the screenshot:



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.

The screenshot shows the Visual Studio Code interface with the file 'index.html' open. The code editor displays the following HTML content:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Ionic App</title>
  <base href="/" />
  <meta name="viewport" content="viewport-fit=cover, width=device-width, initial-scale=1, maximum-scale=1" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="msapplication-tap-highlight" content="no" />
  <link rel="icon" type="image/png" href="assets/icon/favicon.png" />
  <!-- add to homescreen for ios -->
  <meta name="apple-mobile-web-app-capable" content="yes" />
  <meta name="apple-mobile-web-app-status-bar-style" content="black" />
</head>
<body>
  <app-root></app-root>
</body>
</html>

```

The left sidebar shows the project structure under 'OPEN EDITORS' and 'DEVON4NG-IONIC-APPLICATION-TEMPLATE'.

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.

```

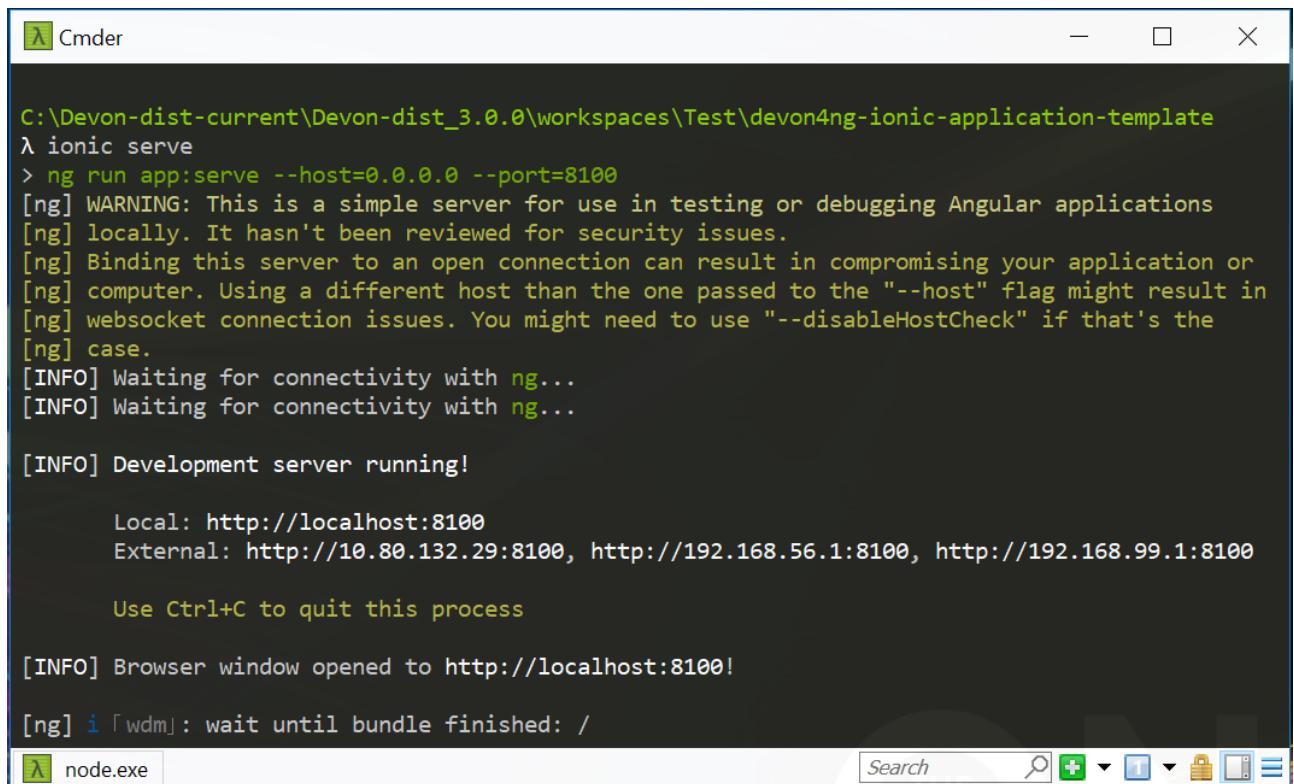
environment.ts
Ambuludi Olmedo, a day ago | 1 author (Ambuludi Olmedo)
1 // This file can be replaced during build by using the "fileReplacements" field in "angular.json"
2 // "ng build --prod" replaces 'environment.ts' with 'environment.prod.ts'
3 // The list of file replacements can be found in "angular.json"
4
5 export const environment = {
6   production: false,
7 };
8
9 export const SERVER_URL = 'http://localhost:8081/';
10
11 /*
12 * For easier debugging in development mode, you can import the
13 * to ignore zone related error stack frames such as "zone.run"
14 *
15 * This import should be commented out in production mode because
16 * on performance if an error is thrown.
17 */
18 // import 'zone.js/dist/zone-error'; // Included with Angular
19

environment.android.ts
Ambuludi Olmedo, a day ago | 1 author (Ambuludi Olmedo)
1 export const environment = {
2   production: false,
3 };
4
5 export const SERVER_URL = 'http://10.0.2.2:8081/';

angular.json
{
  "newProjectRoot": "projects",
  "projects": {
    "app": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "architect": "app",
      "schematics": {},
      "architects": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {},
          "configurations": {
            "production": {}
          }
        },
        "android": {
          "fileReplacements": [
            {
              "replace": "src/environments/environment.ts",
              "with": "src/environments/environment.android.ts"
            }
          ]
        },
        "cli": {
          "progress": false
        }
      }
    },
    "serve": {
      "builder": "@angular-devkit/build-angular:dev-server",
      "options": {
        "browserTarget": "app:build"
      },
      "configuration": {
        "production": {
          "browserTarget": "app:build:production"
        },
        "cli": {
          "progress": false
        }
      }
    }
  }
}

```

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**”.



C:\Devon-dist-current\Devon-dist_3.0.0\workspaces\Test\devon4ng-ionic-application-template
λ ionic serve
> ng run app:serve --host=0.0.0.0 --port=8100
[ng] WARNING: This is a simple server for use in testing or debugging Angular applications
[ng] locally. It hasn't been reviewed for security issues.
[ng] Binding this server to an open connection can result in compromising your application or
[ng] computer. Using a different host than the one passed to the "--host" flag might result in
[ng] websocket connection issues. You might need to use "--disableHostCheck" if that's the
[ng] case.
[INFO] Waiting for connectivity with ng...
[INFO] Waiting for connectivity with ng...

[INFO] Development server running!

Local: http://localhost:8100
External: http://10.80.132.29:8100, http://192.168.56.1:8100, http://192.168.99.1:8100

Use Ctrl+C to quit this process

[INFO] Browser window opened to http://localhost:8100!

[ng] i 「wdm」: wait until bundle finished: /

12. Once the execution is successful

The screenshot shows a mobile application interface titled "Employee". At the top, there is a blue header bar with the title "Employee" and a language switcher "EN". Below the header is a table with four columns: "employeed_EN", "name_EN", "surname_EN", and "email_EN". The table contains three rows of data:

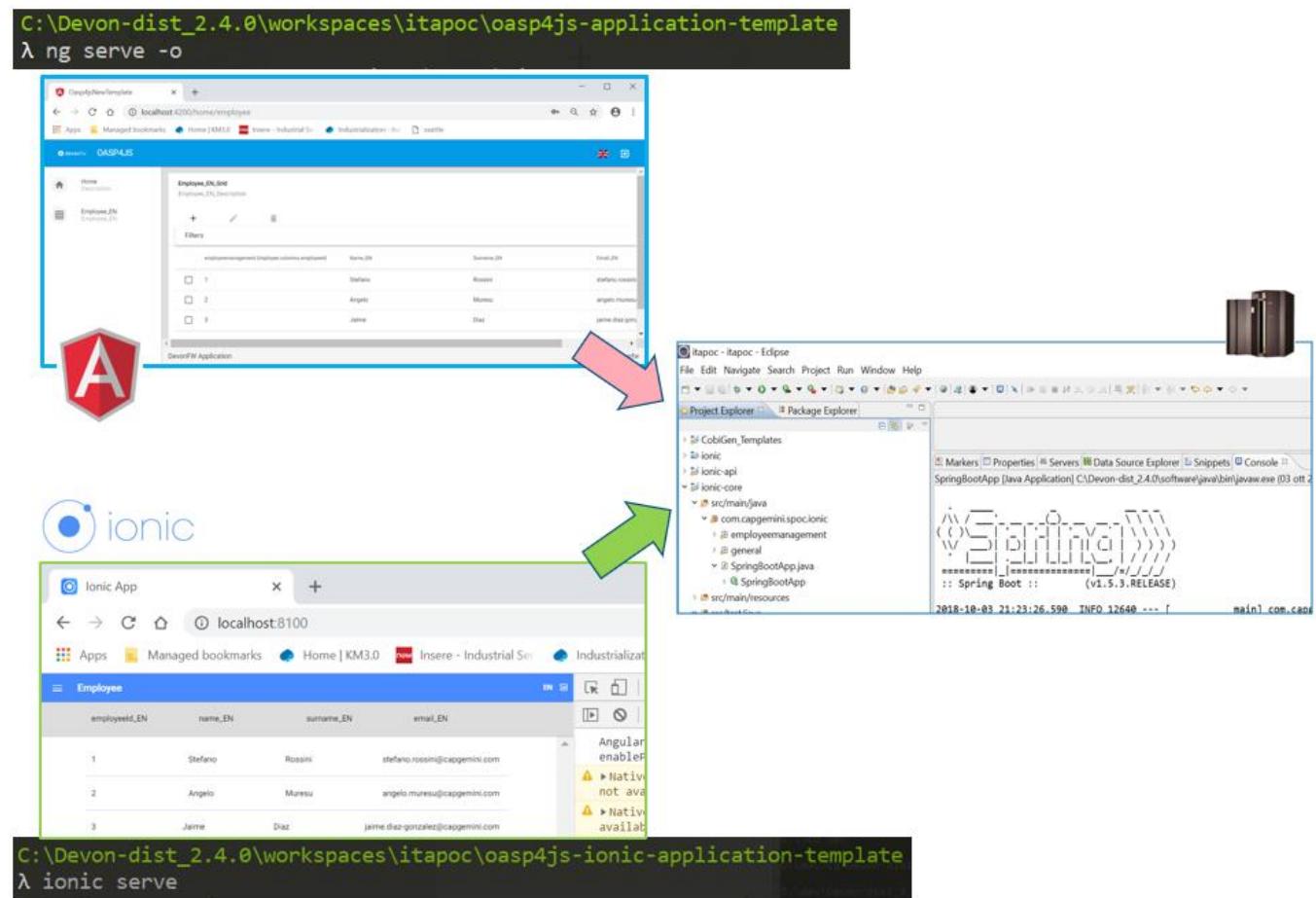
employeed_EN	name_EN	surname_EN	email_EN
1	Stefano	Rossini	stefano.rossini@capgemini.com
2	Angelo	Muresu	angelo.muresu@capgemini.com
3	Jaime	Gonzalez	jaime.diaz-gonzalez@capgemini.com

To the right of each row, there is a vertical stack of five circular icons with white symbols: a magnifying glass, a trash can, a pencil, a plus sign, and a cross.

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!

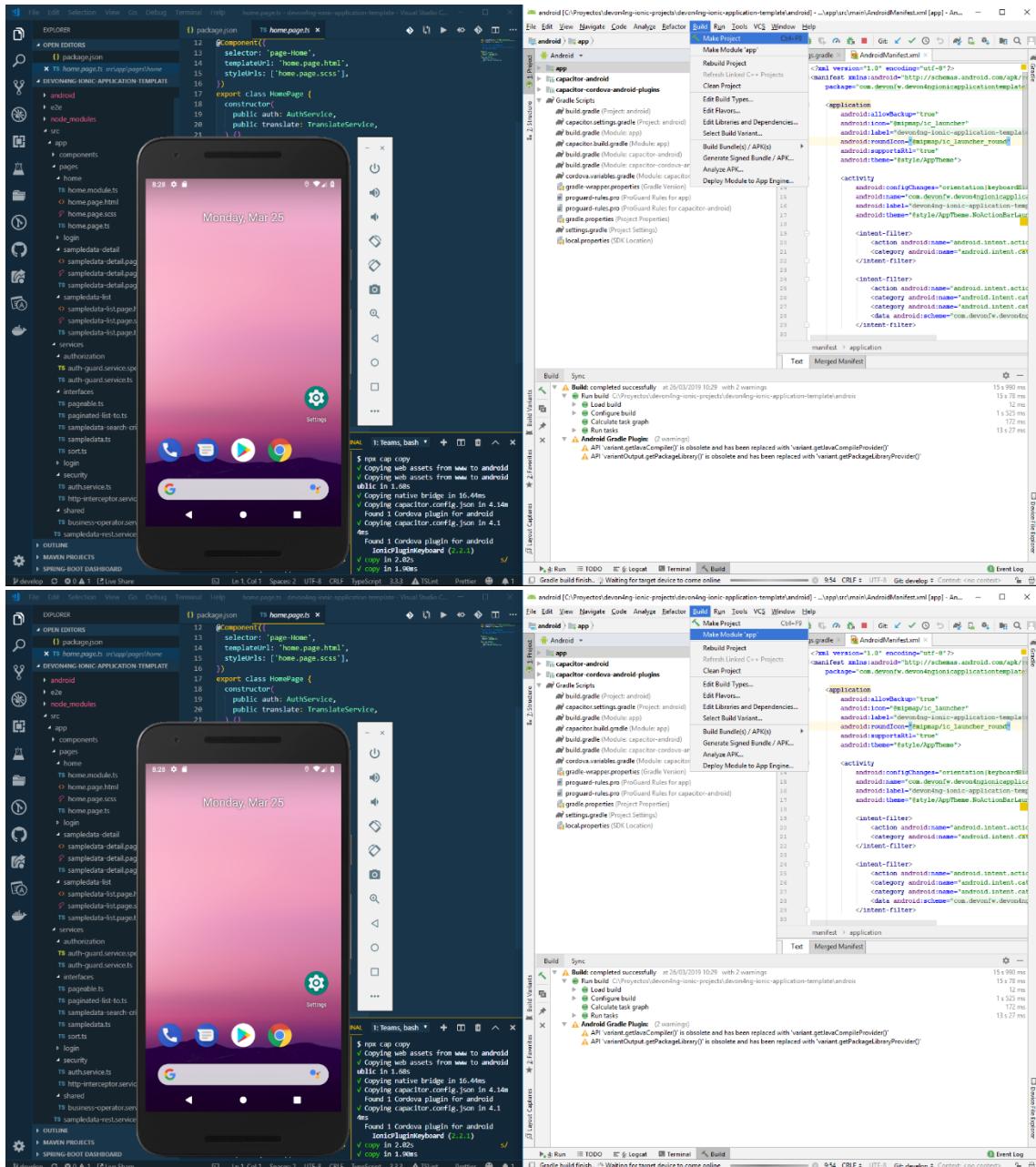


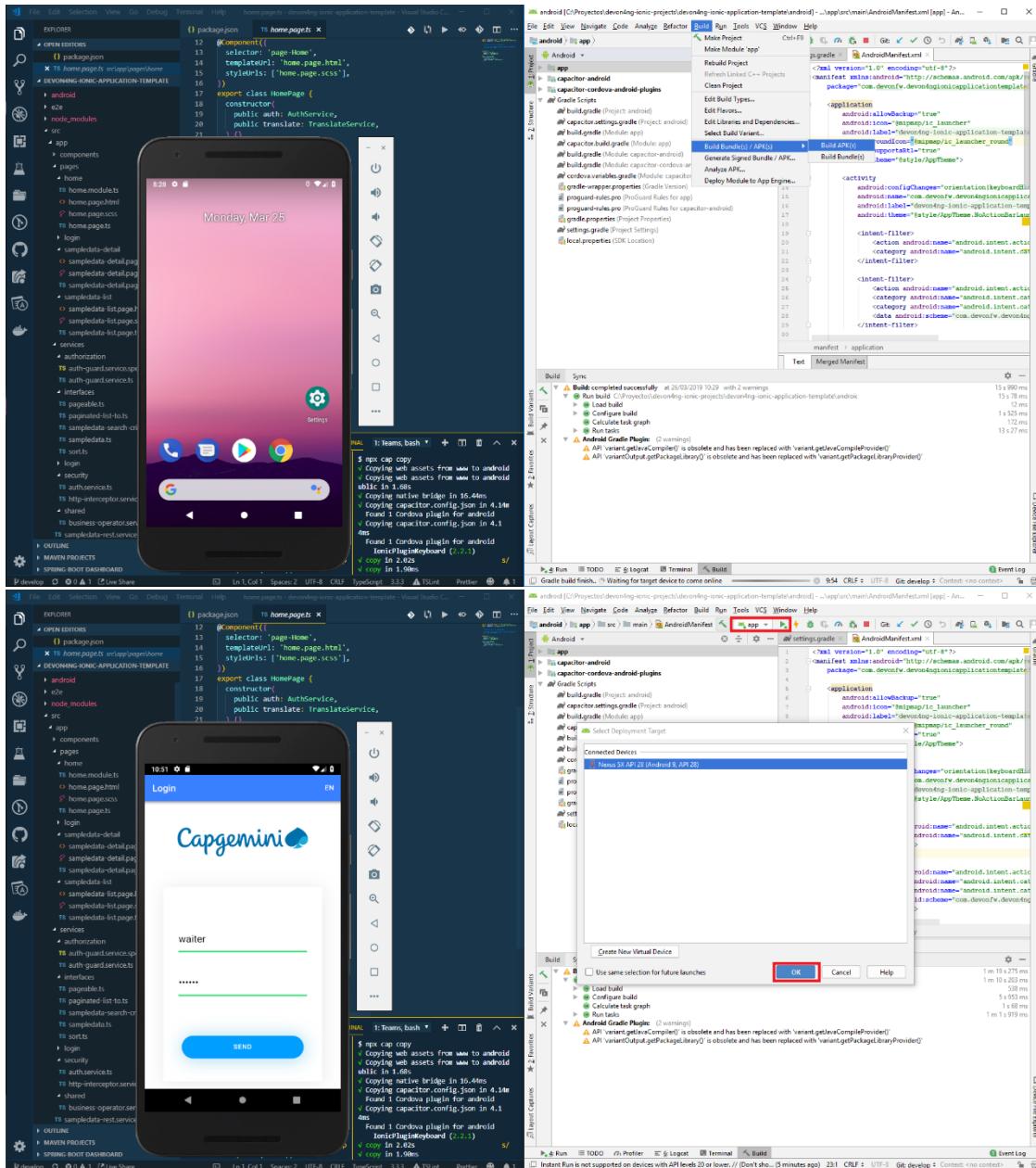
2.2.2.1 Generate APK

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.





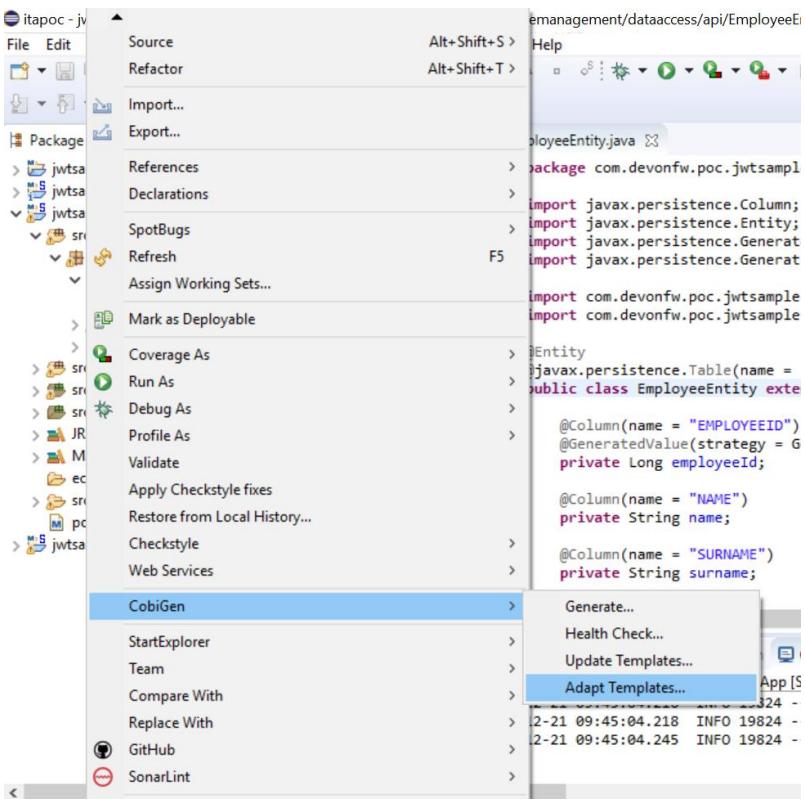
You can find your apk file in

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

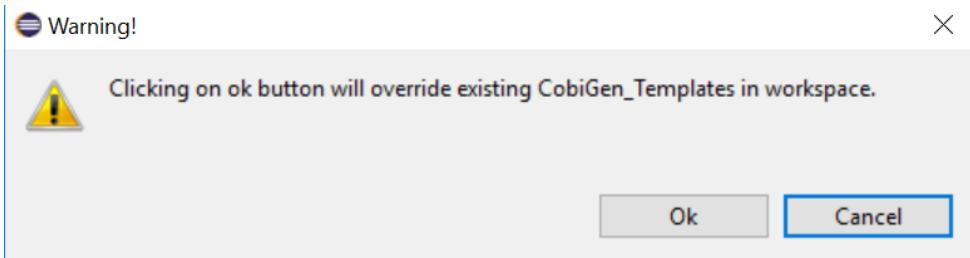
3 Adapt CobiGen_Templates:

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:

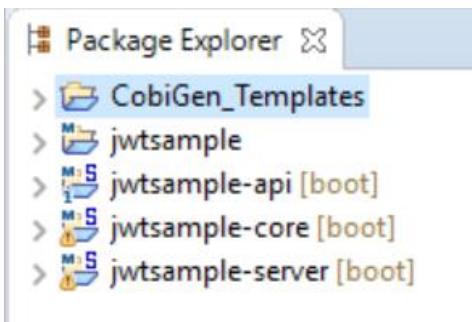
Right click in any part of the package explorer, then click on CobiGen -> Adapt templates

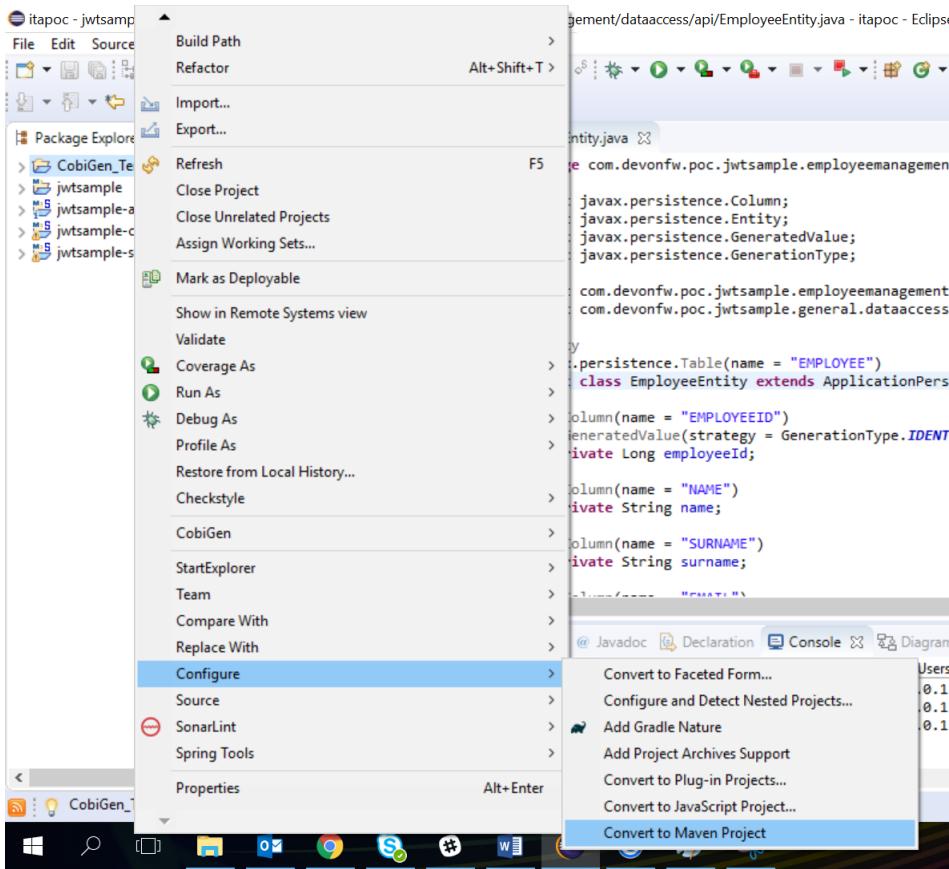


Click Ok:

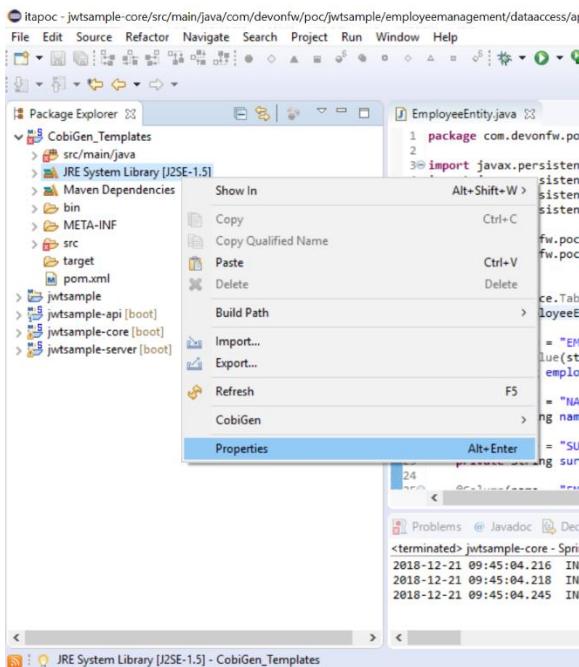


Now the CobiGen_Templates project will be automatically imported into your workspace, as shown on the image below:

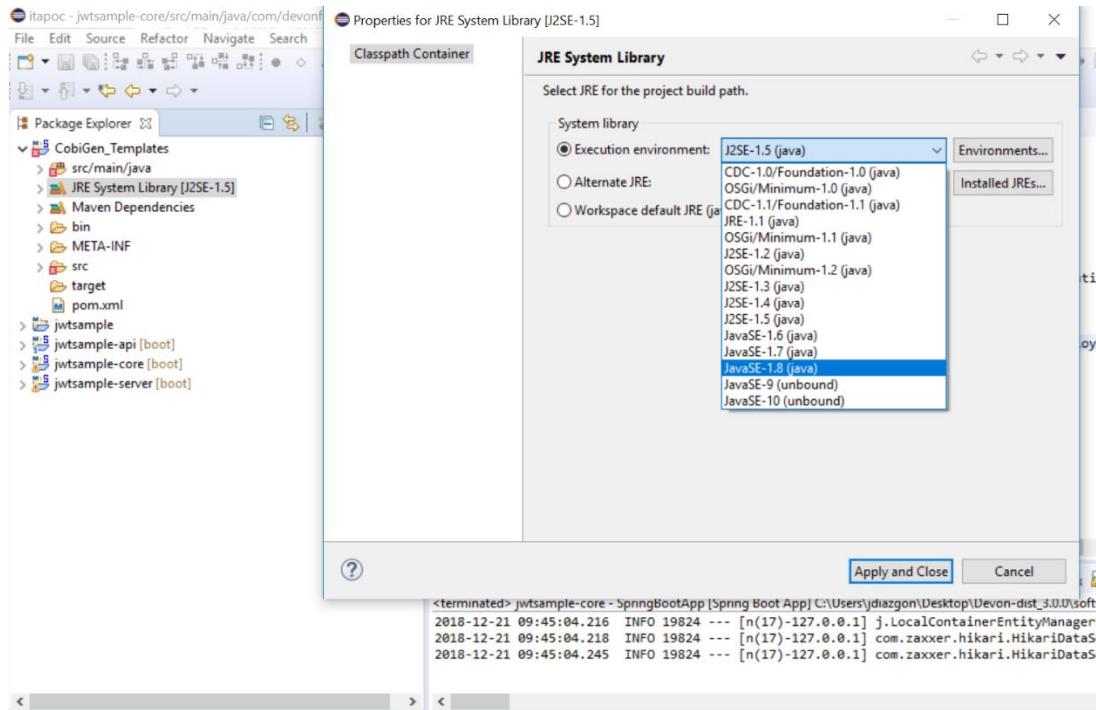




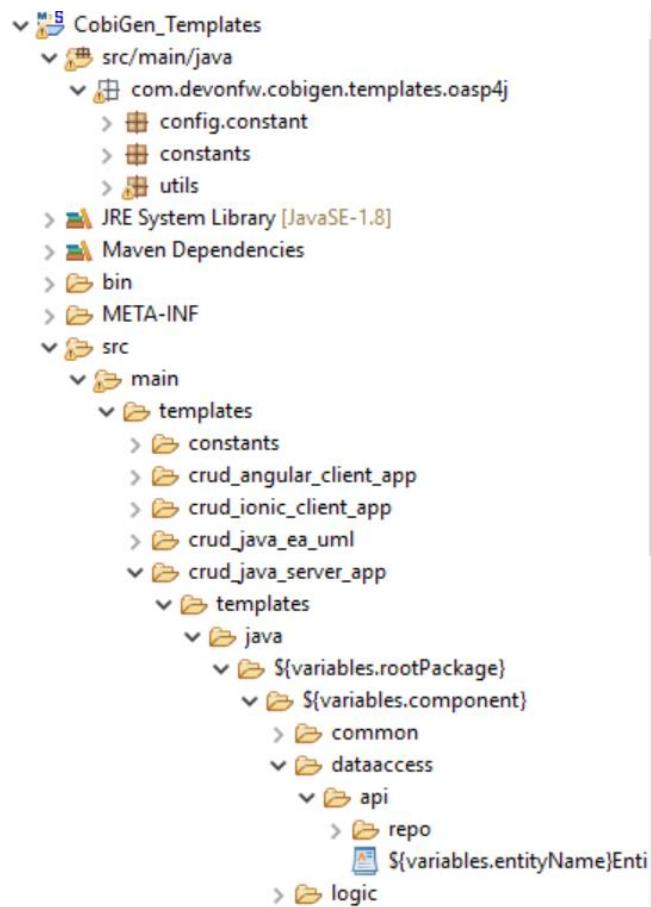
Now you just need to change the Java version of the project to JRE 1.8. Right click on the JRE system library, and then on *Properties*:



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>