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

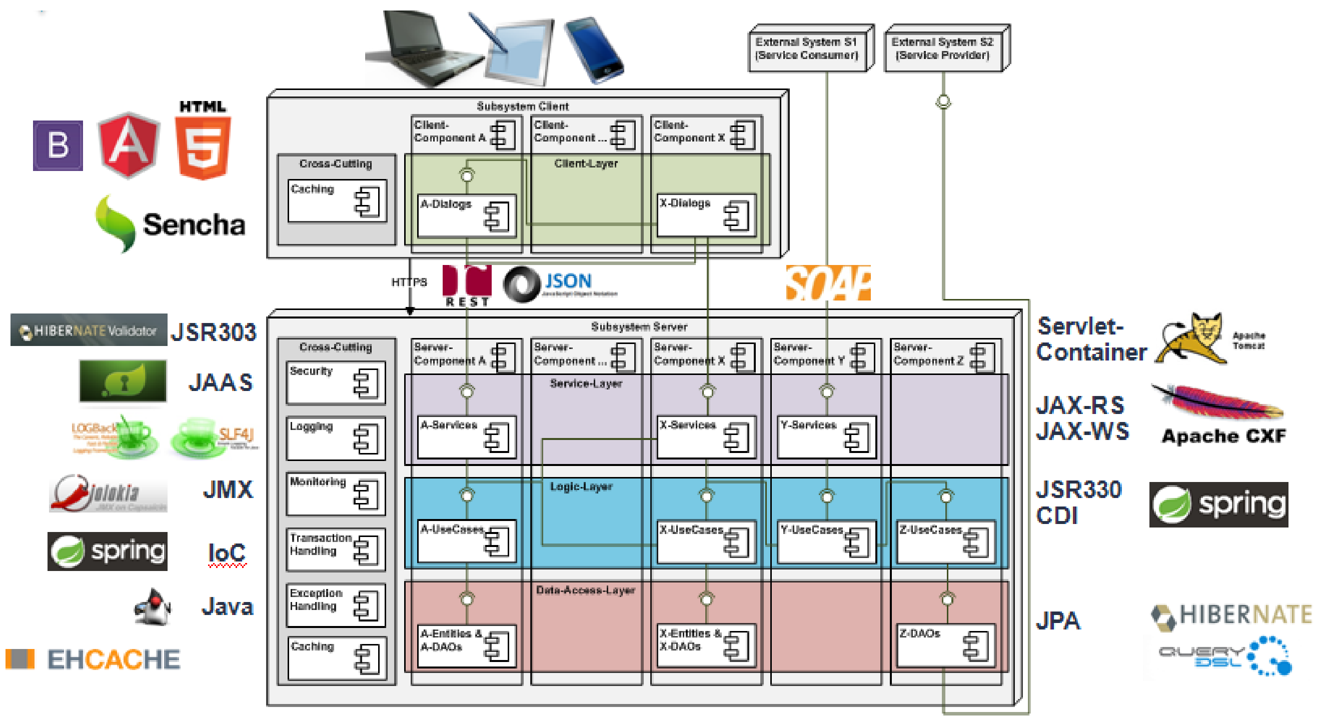
1.1 What's devonfw

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 This platform is aimed to engagements where the client is flexible in the use of technology or uses outdated technology,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.

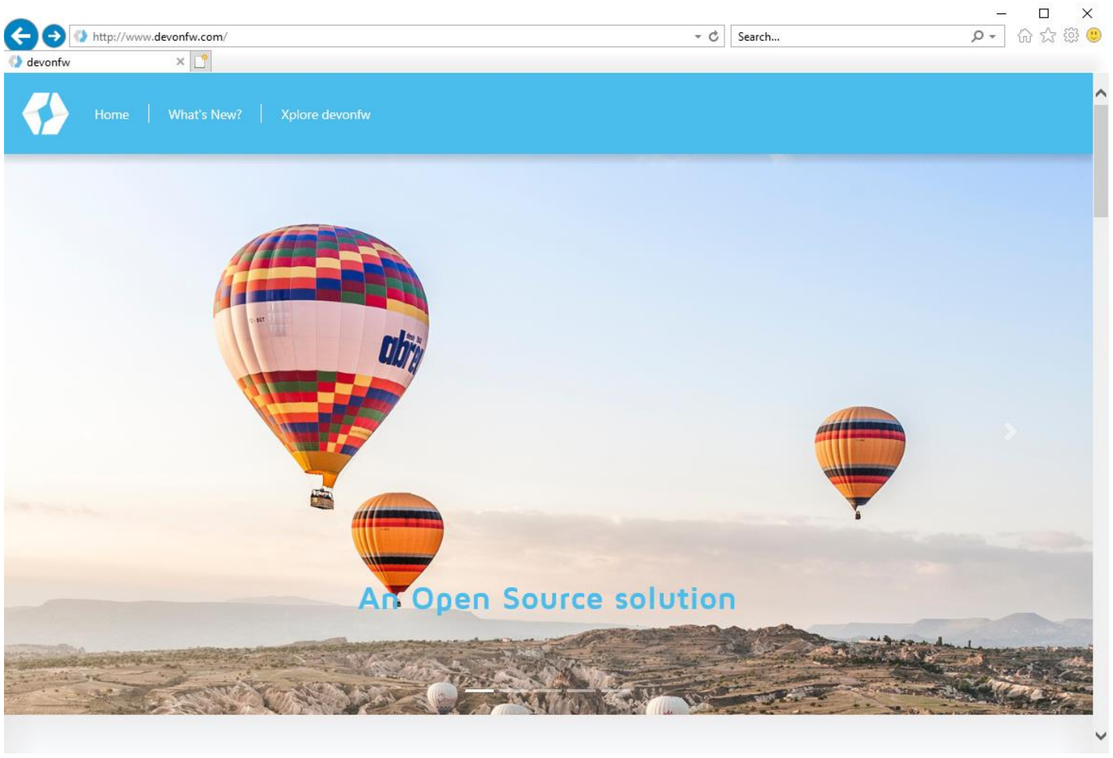
The current version of the platform is oriented to develop single-page web applications based on the Java EE programming model using the spring framework as the default implementation.

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.

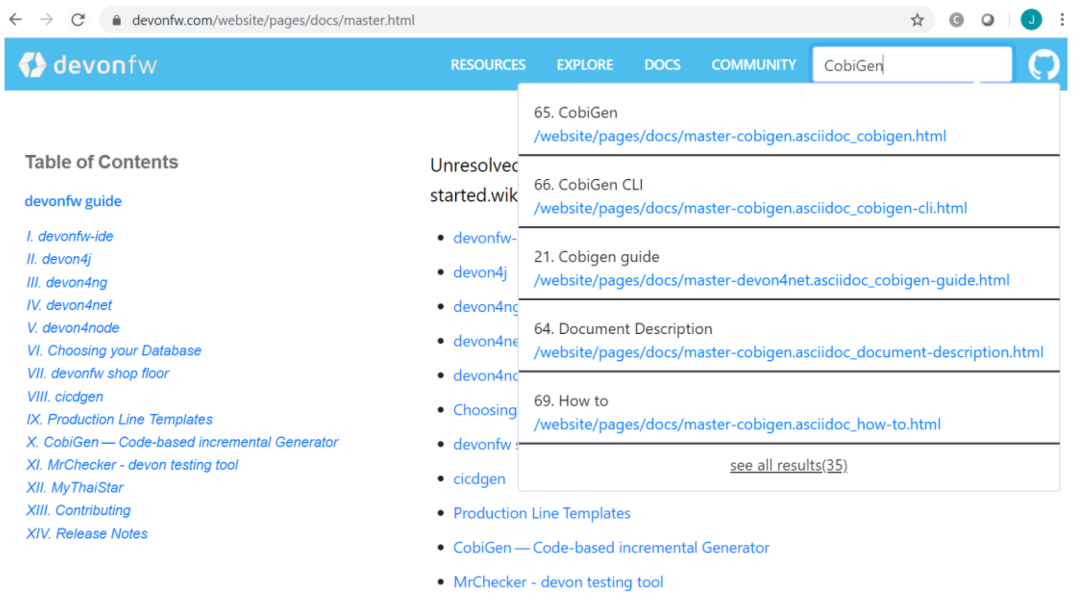


1.2 devonfw Web site URL

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



All the documentation about devonfw can be found under docs. There is a search bar which you can use to quickly find documentation:



65. CobiGen
/website/pages/docs/master-cobigen.asciidoc_cobigen.html

66. CobiGen CLI
/website/pages/docs/master-cobigen.asciidoc_cobigen-cli.html

21. Cobigen guide
/website/pages/docs/master-devon4net.asciidoc_cobigen-guide.html

64. Document Description
/website/pages/docs/master-cobigen.asciidoc_document-description.html

69. How to
/website/pages/docs/master-cobigen.asciidoc_how-to.html

see all results(35)

Table of Contents

devonfw guide

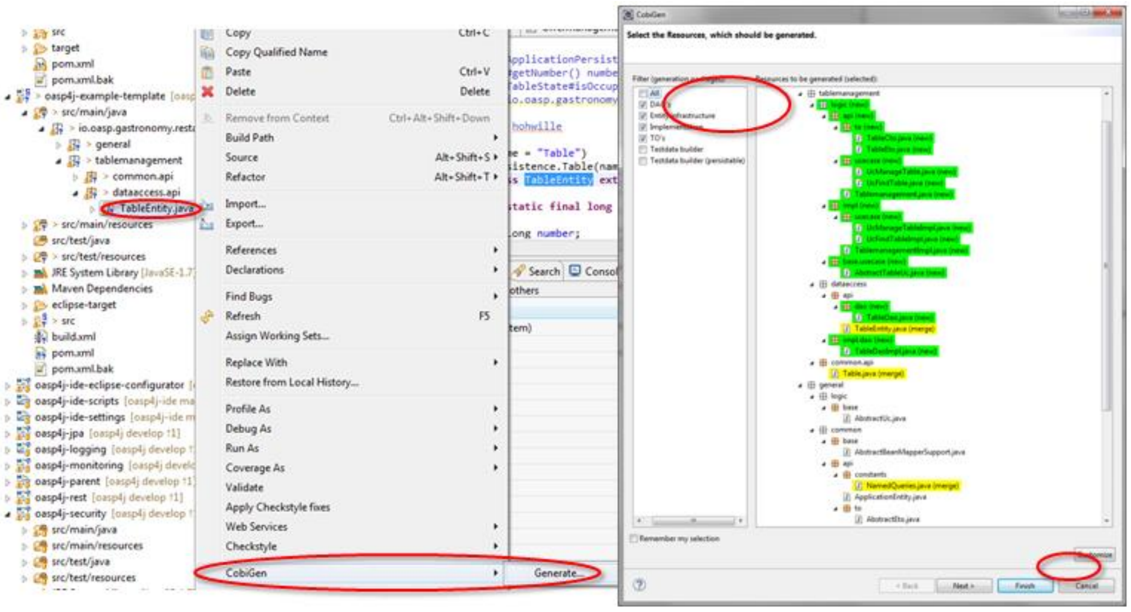
I. devonfw-ide
II. devon4j
III. devon4ng
IV. devon4net
V. devon4node
VI. Choosing your Database
VII. devonfw shop floor
VIII. cicdgen
IX. Production Line Templates
X. CobiGen — Code-based incremental Generator
XI. MrChecker - devon testing tool
XII. MyThaiStar
XIII. Contributing
XIV. Release Notes

Unresolvec
started.wik

- devonfw-
- devon4j
- devon4ng
- devon4ne
- devon4nc
- Choosing
- devonfw
- cicdgen
- Production Line Templates
- CobiGen — Code-based incremental Generator
- MrChecker - devon testing tool

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:

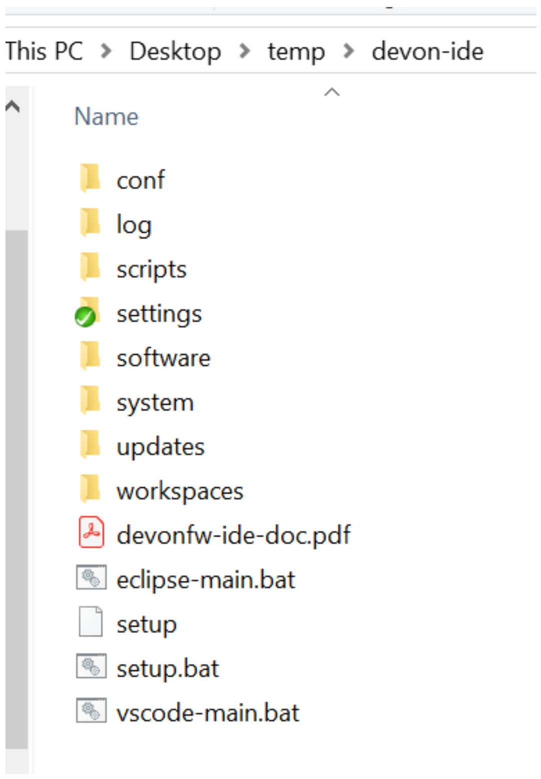
1. Can generate a whole **CRUD** application from a **single Entity class.**
2. You can save the effort for creating simple CRUD use case since CobiGen generates: **DAO**s, **DTOs**, Spring services and REST and SOAP services and even the client UI application (Angular and Ionic).
3. Agility: Boost development-speed, 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. First, we will install the devonfw-ide. It is a tool that will setup your IDE within minutes. Please follow the install guide here.

i.e: see the image below to see how the devonfw ide looks like (the folder name doesn't have to be the same):



This PC > Desktop > temp > devon-ide

Name

conf
log
scripts
settings
software
system
updates
workspaces
devonfw-ide-doc.pdf
eclipse-main.bat
setup
setup.bat
vscode-main.bat

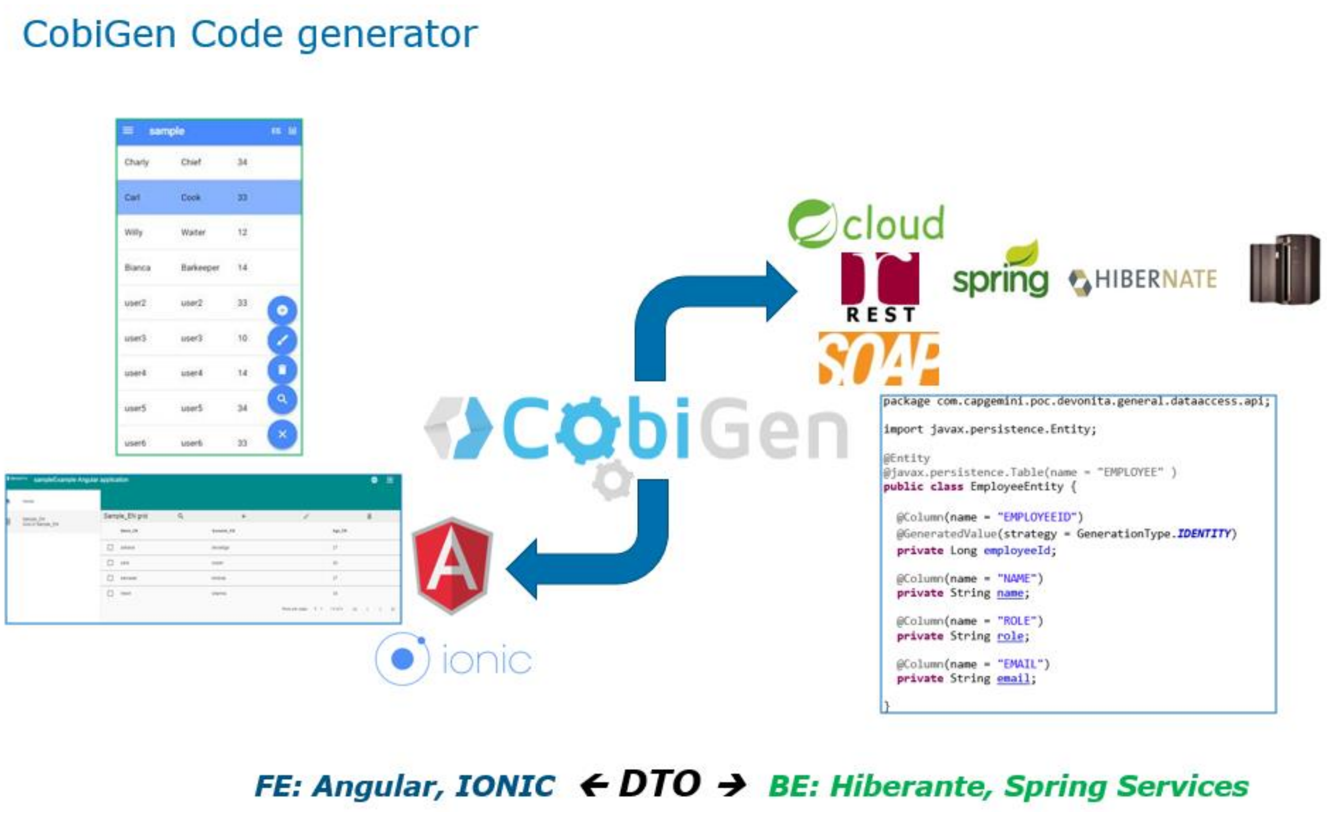
Since We're going to develop an E2E application (Mobile app + Web FE and BE services/DB) remember the following pre-conditions:

- Java (included in devonfw-ide)
- Node and npm (https://nodejs.org/dist/v10.7.0/node-v10.7.0-x64.msi) (included in devonfw-ide)
- 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)



```
package com.capgemini.poc.devonita.general.dataaccess.api;

import javax.persistence.Entity;

@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 = "ROLE")
  private String role;

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

}
```

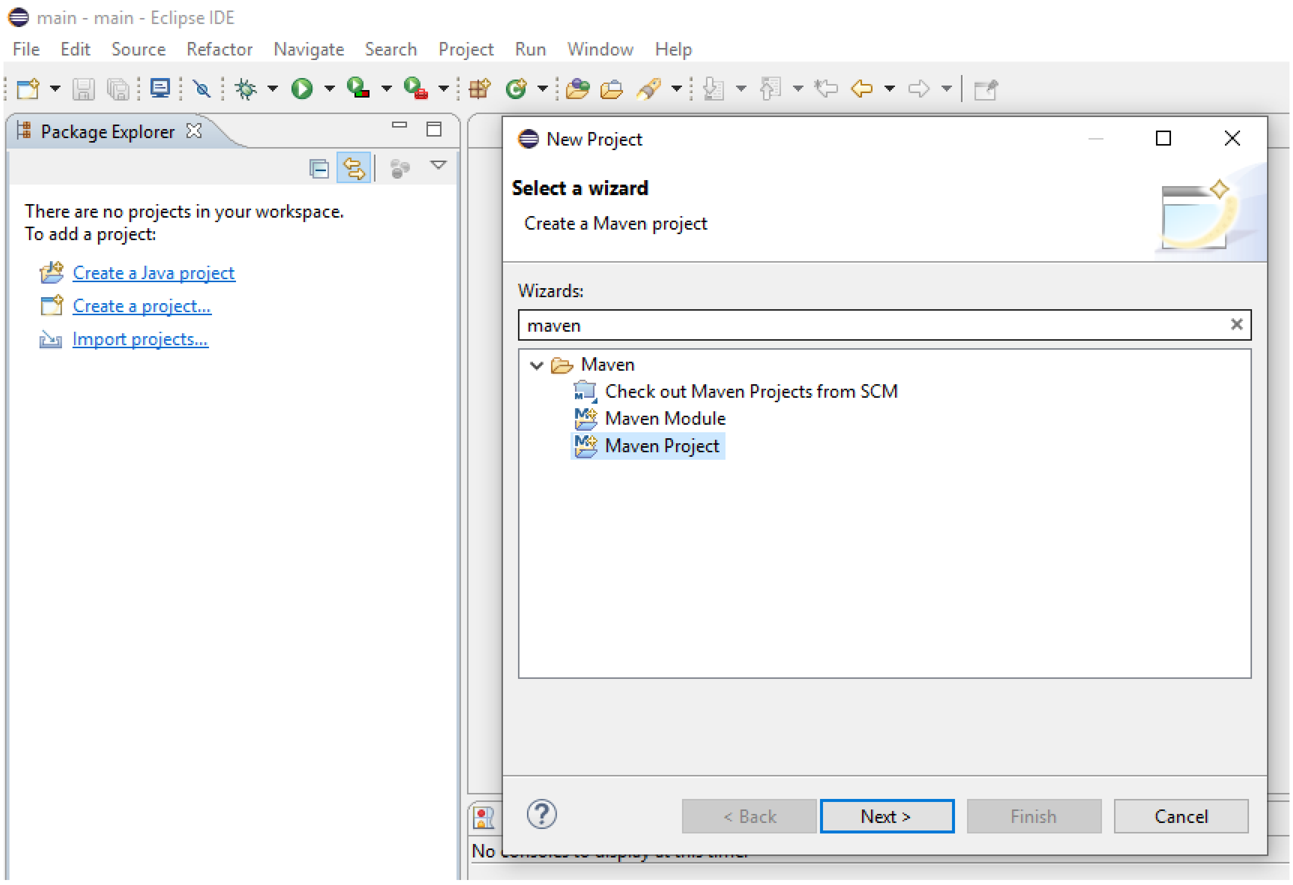
So, ready to go! We’re going to start from the BE part …

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

1. Run the **eclipse-main.bat** present within the devon-ide folder

   It will open eclipse for you

2. Now we are going to create a new devon4j project. Right click -> New -> Other and we search for Maven project.


main - main - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

There are no projects in your workspace.
To add a project:

Create a Java project
Create a project...
Import projects...

New Project

Select a wizard

Create a Maven project

Wizards:

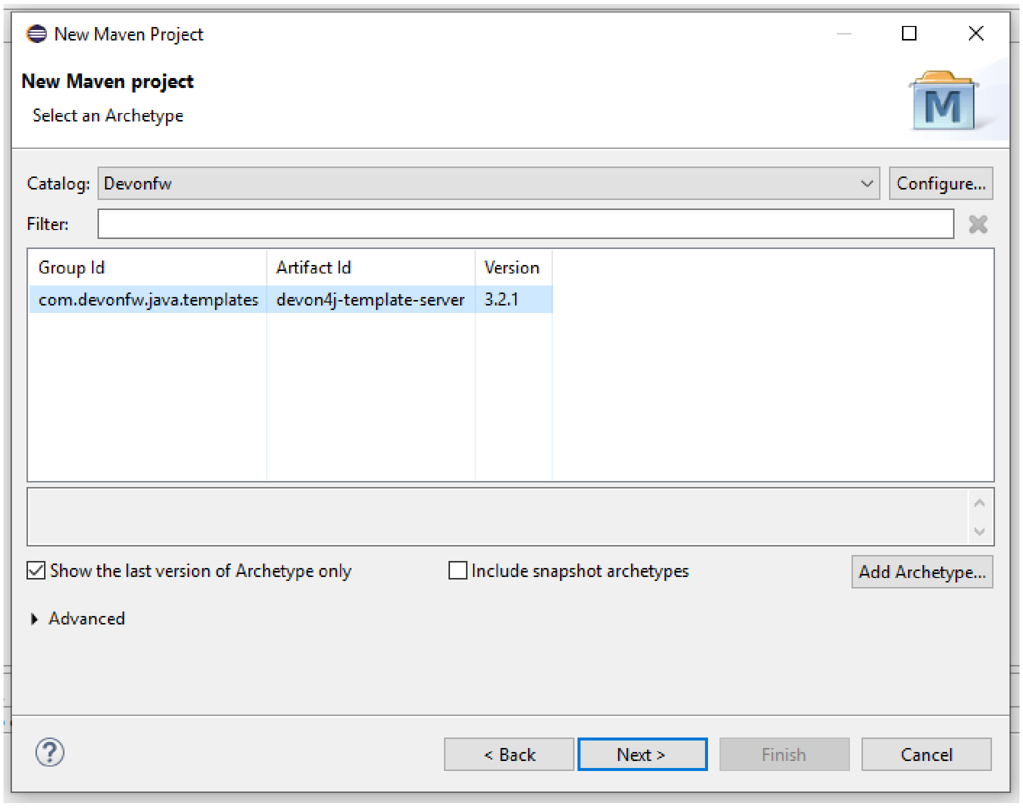
maven

Maven
- Check out Maven Projects from SCM
- Maven Module
- Maven Project

< Back | Next > | Finish | Cancel

No consoles to display at this time.

3. Click on Next and select "Devonfw" catalogue. We will pick devon4j-template-server from the list.


New Maven Project

New Maven project

Select an Archetype

Catalog: Devonfw

Filter:

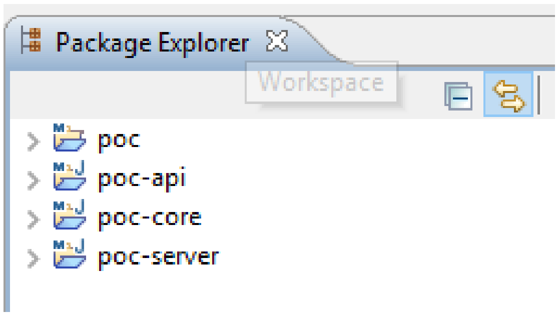
Group Id	Artifact Id	Version
com.devonfw.java.templates	devon4j-template-server	3.2.1

☑ Show the last version of Archetype only ☐ Include snapshot archetypes

4. Click Next. On the next screen you will define the groupId and artifactID. You can use any name for your project. In our case we will set `grouId=com.devonfw` and `artifactId=poc.` **Note:** H2 database is set by default, but it is possible to choose other options (hana, mysql, postgresql...).

Click **FINISH**

Now We have the following 4 projects.

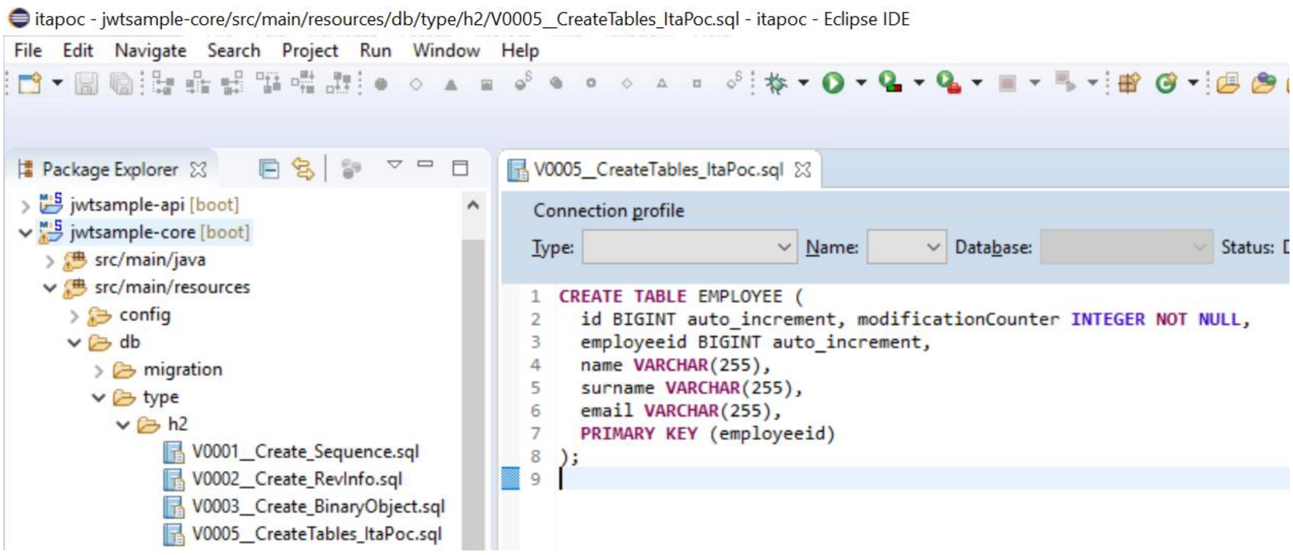


BEFORE to start to create an Entity class, remember to create the tables !

5. Create a new **SQL file** (i.e: V0005__CreateTables_ItaPoc.sql) inside *poc*-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 !



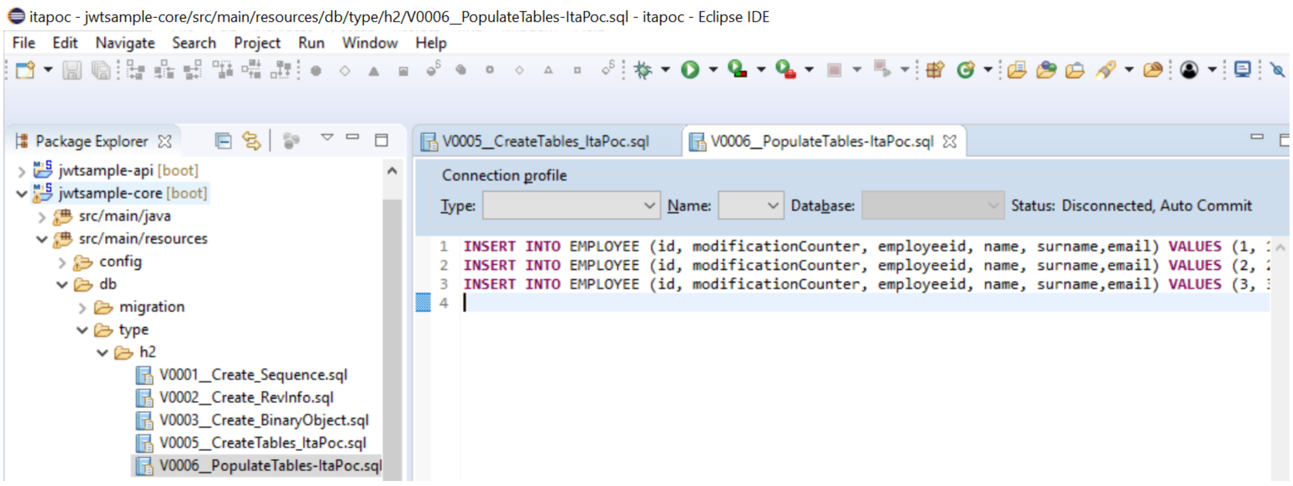
itapoc - jwtsample-core/src/main/resources/db/type/h2/V0005__CreateTables_ItaPoc.sql - itapoc - Eclipse IDE

```
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)
);
```

6. 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');
```



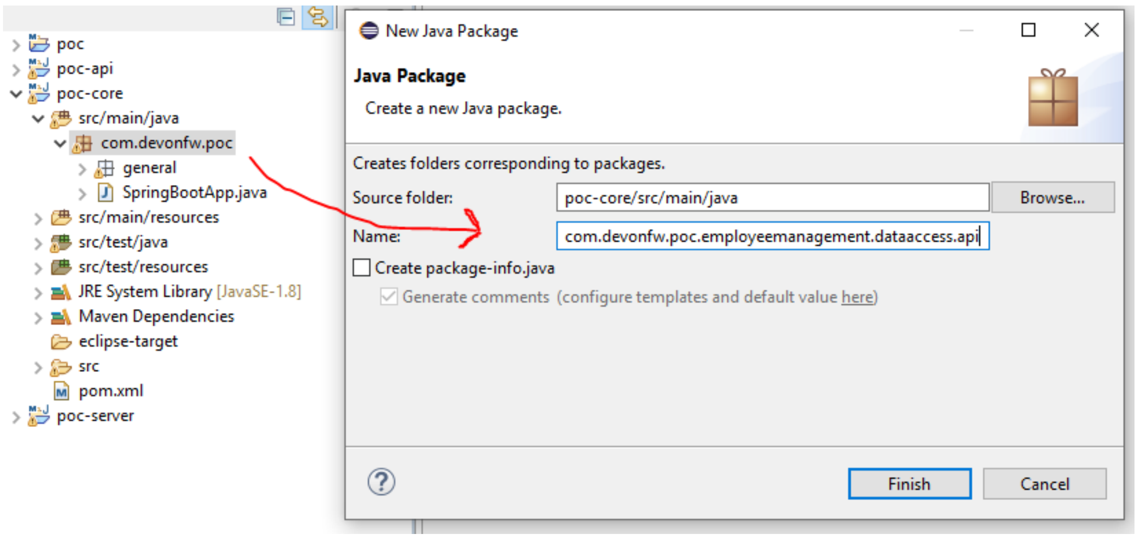
itapoc - jwtsample-core/src/main/resources/db/type/h2/V0006__PopulateTables-ItaPoc.sql - itapoc - Eclipse IDE

Status: Disconnected, Auto Commit

```
INSERT INTO EMPLOYEE (id, modificationCounter, employeeid, name, surname,email) VALUES (1,
INSERT INTO EMPLOYEE (id, modificationCounter, employeeid, name, surname,email) VALUES (2,
INSERT INTO EMPLOYEE (id, modificationCounter, employeeid, name, surname,email) VALUES (3,
```

NOW you can create the Entity class 😉

7. First of all create a package "employeemanagement.dataacess.api" under the folder "poc-core". **Note:** It is important to follow this naming convention for CobiGen to work properly.

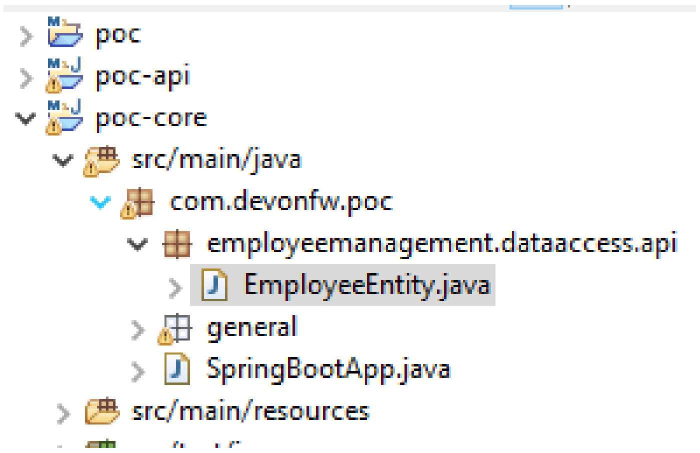


Creates folders corresponding to packages.

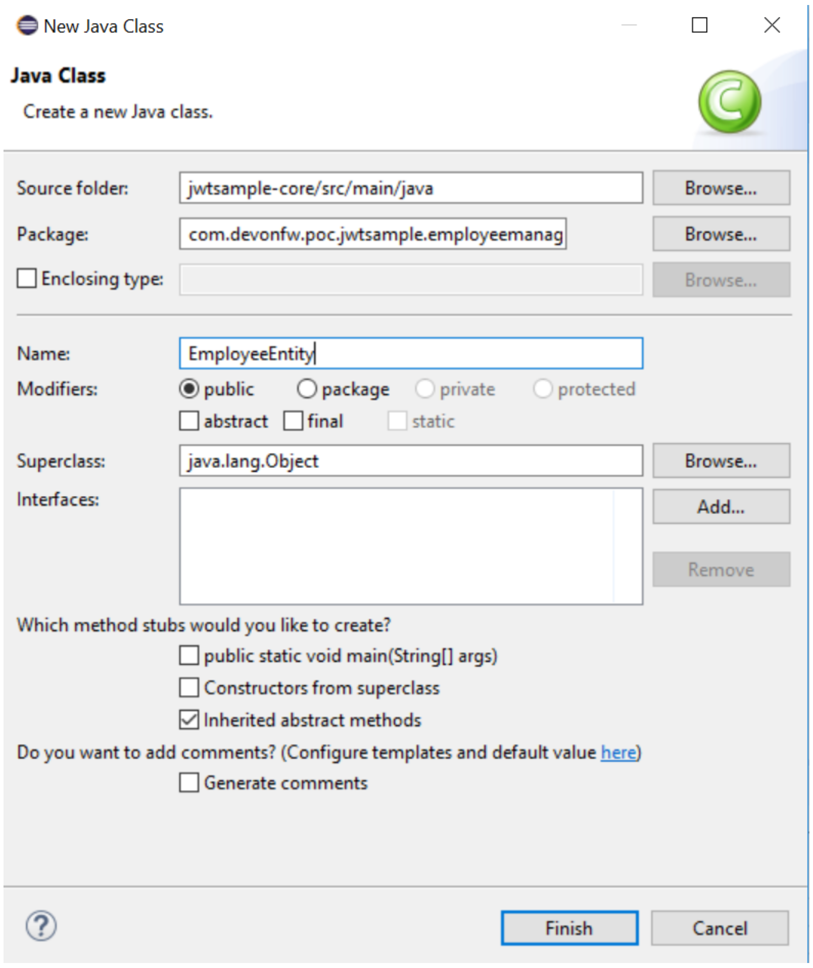
Source folder: poc-core/src/main/java

Name: com.devonfw.poc.employeemanagement.dataaccess.api

8. Now under this new package created



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

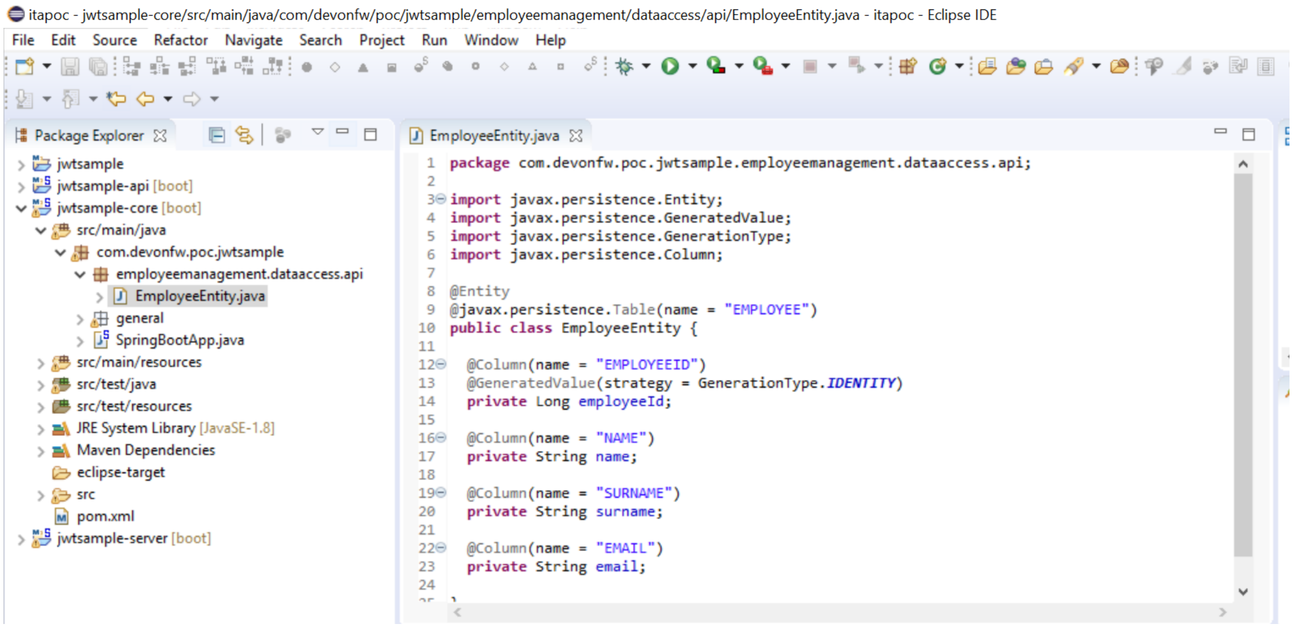


Source folder: jwtsample-core/src/main/java
Package: com.devonfw.poc.jwtsample.employeemanag
Enclosing type: [ ]
Name: EmployeeEntity
Modifiers: [✓ public] package private protected
abstract final static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
[ ] public static void main(String[] args)
[ ] Constructors from superclass
[✓] Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
[ ] Generate comments

Now you can add:

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

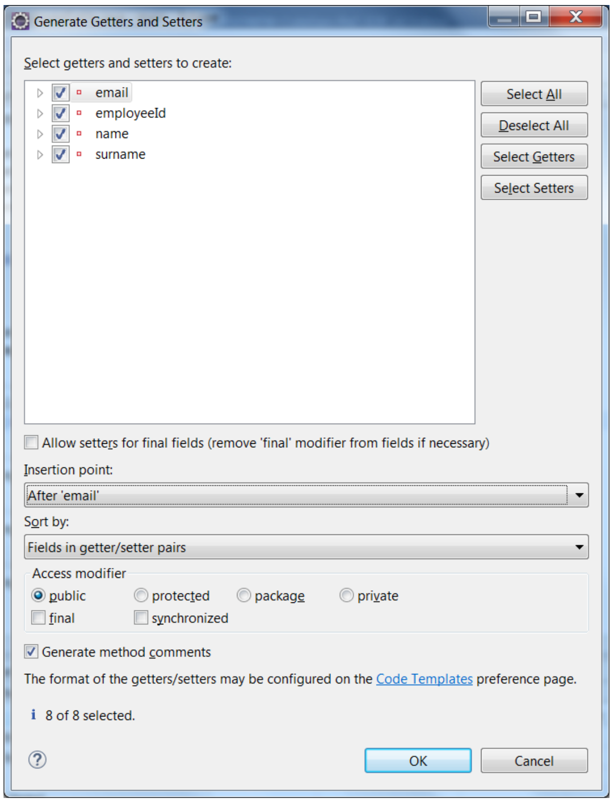
```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Column;

@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;

}
```



itapoc - jwtsample-core/src/main/java/com/devonfw/poc/jwtsample/employeemanagement/dataaccess/api/EmployeeEntity.java - itapoc - Eclipse IDE

```
package com.devonfw.poc.jwtsample.employeemanagement.dataaccess.api;

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 …



Generate Getters and Setters

Select getters and setters to create:

- [✓] email
- [✓] employeeId
- [✓] name
- [✓] surname

[ ] Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point: After 'email'

Sort by: Fields in getter/setter pairs

Access modifier: [✓ public] protected package private

[ ] final [ ] synchronized

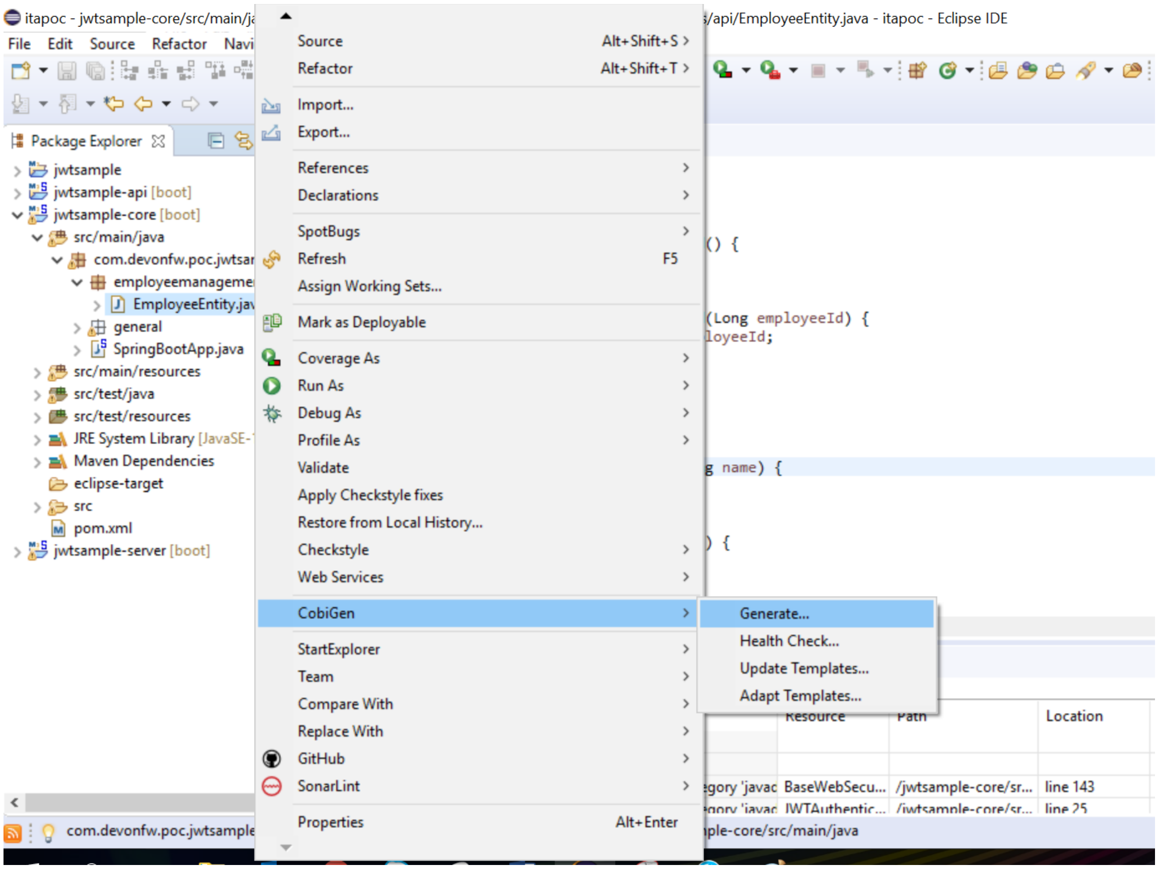
[✓] Generate method comments

The format of the getters/setters may be configured on the Code Templates preference page.

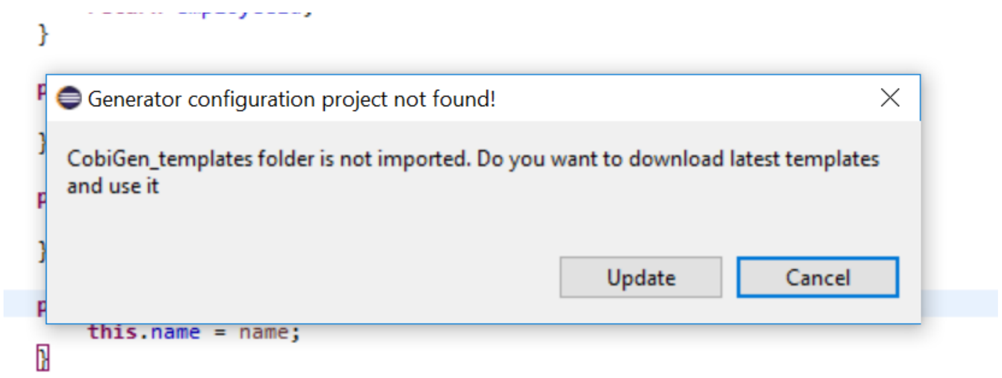
8 of 8 selected.

9. 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*:

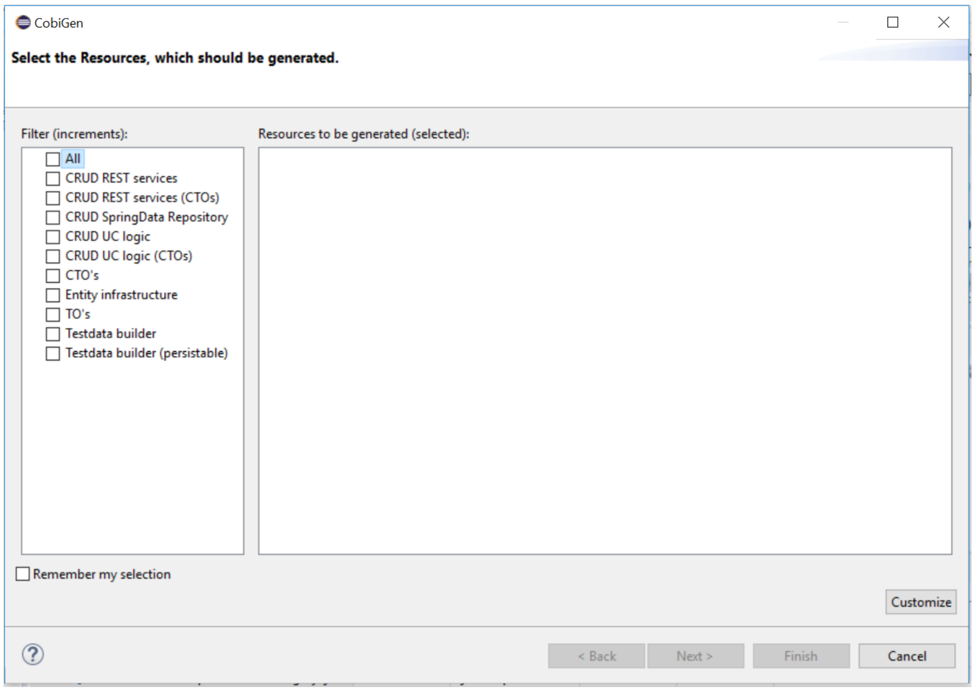


Generator configuration project not found!

CobiGen_templates folder is not imported. Do you want to download latest templates and use it

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!


CobiGen

Select the Resources, which should be generated.

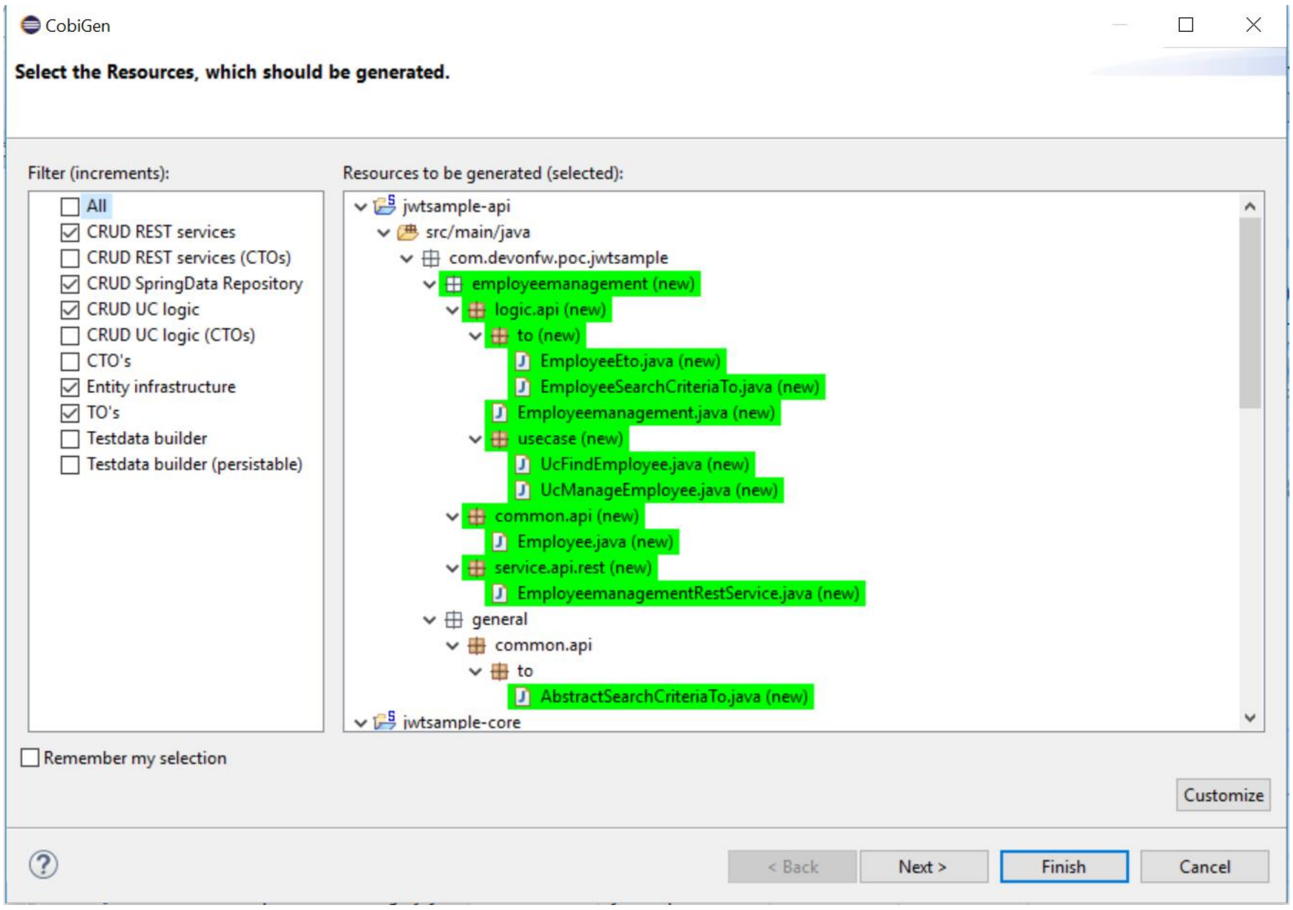
Filter (increments):

- [ ] All
- [ ] CRUD REST services
- [ ] CRUD REST services (CTOs)
- [ ] CRUD SpringData Repository
- [ ] CRUD UC logic
- [ ] CRUD UC logic (CTOs)
- [ ] CTO's
- [ ] Entity infrastructure
- [ ] TO's
- [ ] Testdata builder
- [ ] Testdata builder (persistable)

Resources to be generated (selected):

- [ ] Remember my selection

10. Click on all the option selected as below:


CobiGen

Select the Resources, which should be generated.

Filter (increments):

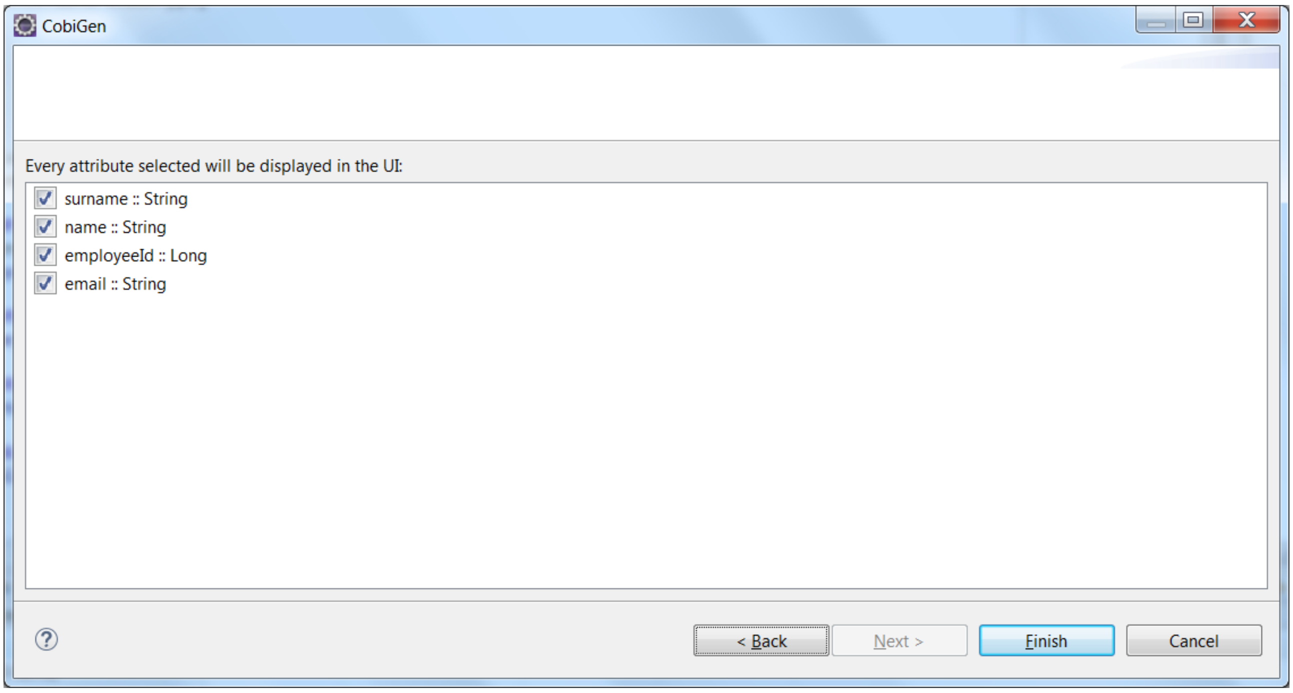
- [ ] All
- [x] CRUD REST services
- [ ] CRUD REST services (CTOs)
- [x] CRUD SpringData Repository
- [x] CRUD UC logic
- [ ] CRUD UC logic (CTOs)
- [ ] CTO's
- [x] Entity infrastructure
- [x] TO's
- [ ] Testdata builder
- [ ] Testdata builder (persistable)

Resources to be generated (selected):

```
jwtsample-api
  src/main/java
    com.devonfw.poc.jwtsample
      employeemanagement (new)
        logic.api (new)
          to (new)
            EmployeeEto.java (new)
            EmployeeSearchCriteriaTo.java (new)
          Employeemanagement.java (new)
          usecase (new)
            UcFindEmployee.java (new)
            UcManageEmployee.java (new)
        common.api (new)
          Employee.java (new)
        service.api.rest (new)
          EmployeemanagementRestService.java (new)
      general
        common.api
          to
            AbstractSearchCriteriaTo.java (new)
jwtsample-core
```

- [ ] Remember my selection

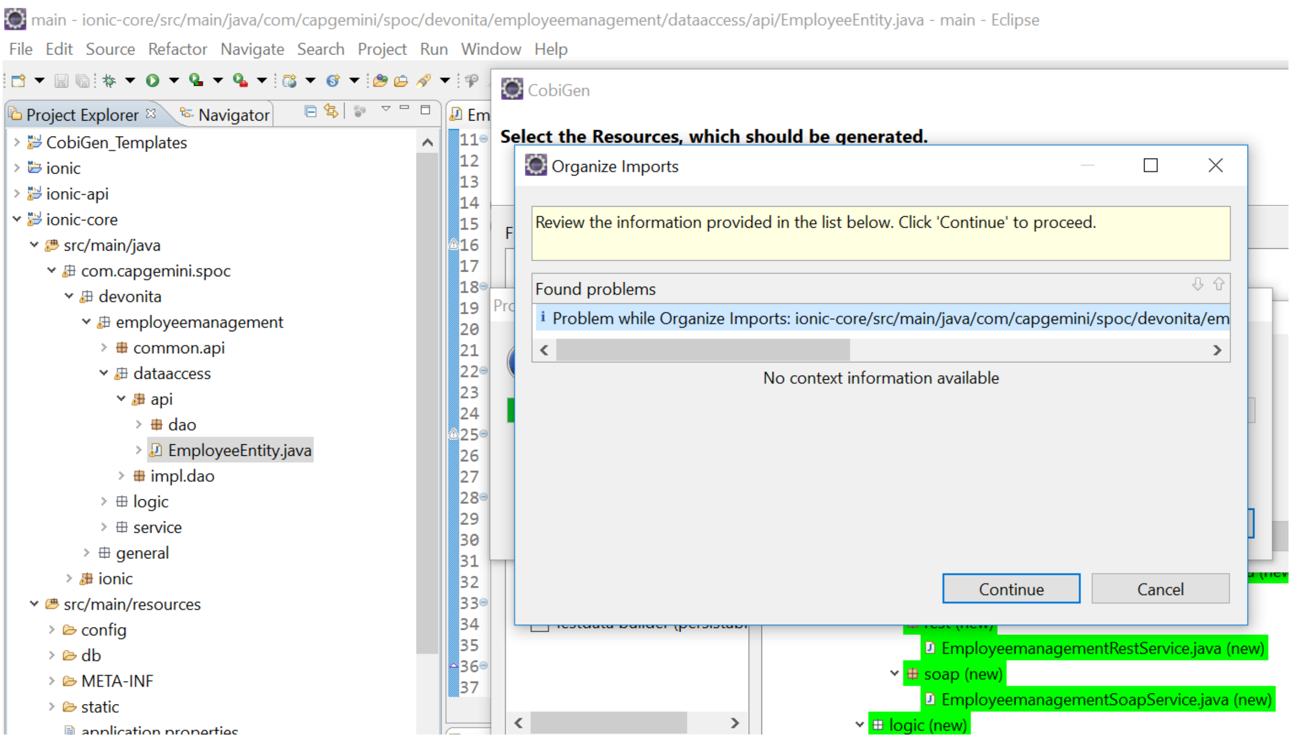
11. Click on Next



Every attribute selected will be displayed in the UI:

- [✓] surname :: String
- [✓] name :: String
- [✓] employeeId :: Long
- [✓] email :: String

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


main - ionic-core/src/main/java/com/capgemini/spoc/devonita/employeemanagement/dataaccess/api/EmployeeEntity.java - main - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Navigator
CobiGen_Templates
ionic
ionic-api
ionic-core
src/main/java
com.capgemini.spoc
devonita
employeemanagement
common.api
dataaccess
api
dao
EmployeeEntity.java
impl.dao
logic
service
general
ionic
src/main/resources
config
db
META-INF
static
CobiGen
Select the Resources, which should be generated.
Organize Imports
Continue
Cancel
EmployeemanagementRestService.java (new)
soap (new)
EmployeemanagementSoapService.java (new)
logic (new)

Review the information provided in the list below. Click 'Continue' to proceed.

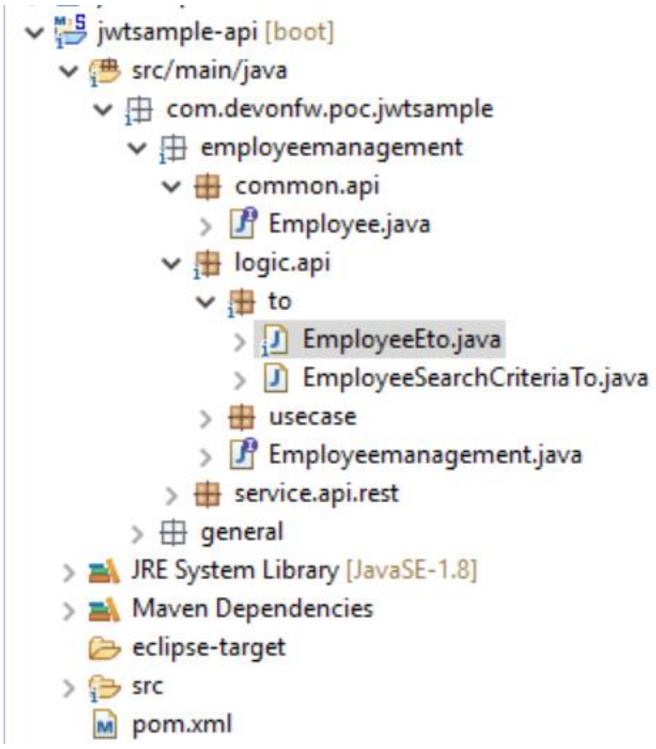
Found problems

Problem while Organize Imports: ionic-core/src/main/java/com/capgemini/spoc/devonita/em

No context information available

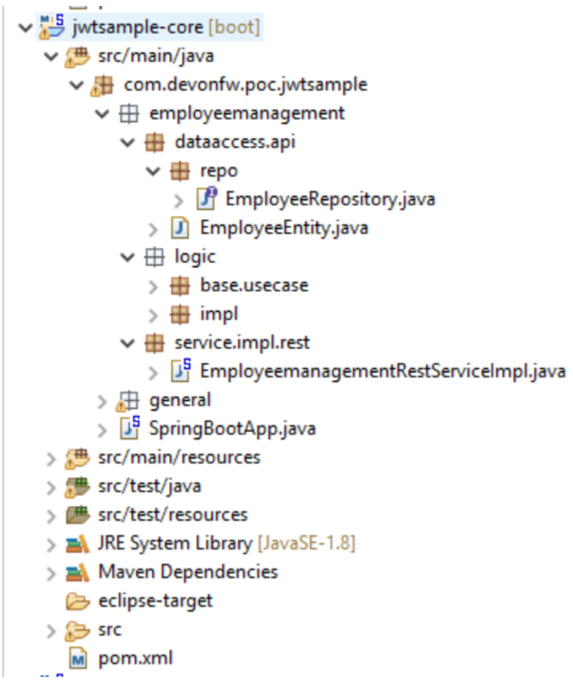
The entire <u>BE layer</u> structure having CRUD operation methods will be auto generated.

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



```
jwtsample-api [boot]
  src/main/java
    com.devonfw.poc.jwtsample
      employeemanagement
        common.api
          Employee.java
        logic.api
          to
            EmployeeEto.java
            EmployeeSearchCriteriaTo.java
          usecase
          Employeemanagement.java
        service.api.rest
      general
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  eclipse-target
  src
  pom.xml
```

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



```
jwtsample-core [boot]
  src/main/java
    com.devonfw.poc.jwtsample
      employeemanagement
        dataaccess.api
          repo
            EmployeeRepository.java
          EmployeeEntity.java
        logic
          base.usecase
          impl
        service.impl.rest
          EmployeemanagementRestServiceImpl.java
      general
      SpringBootApp.java
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  eclipse-target
  src
  pom.xml
```

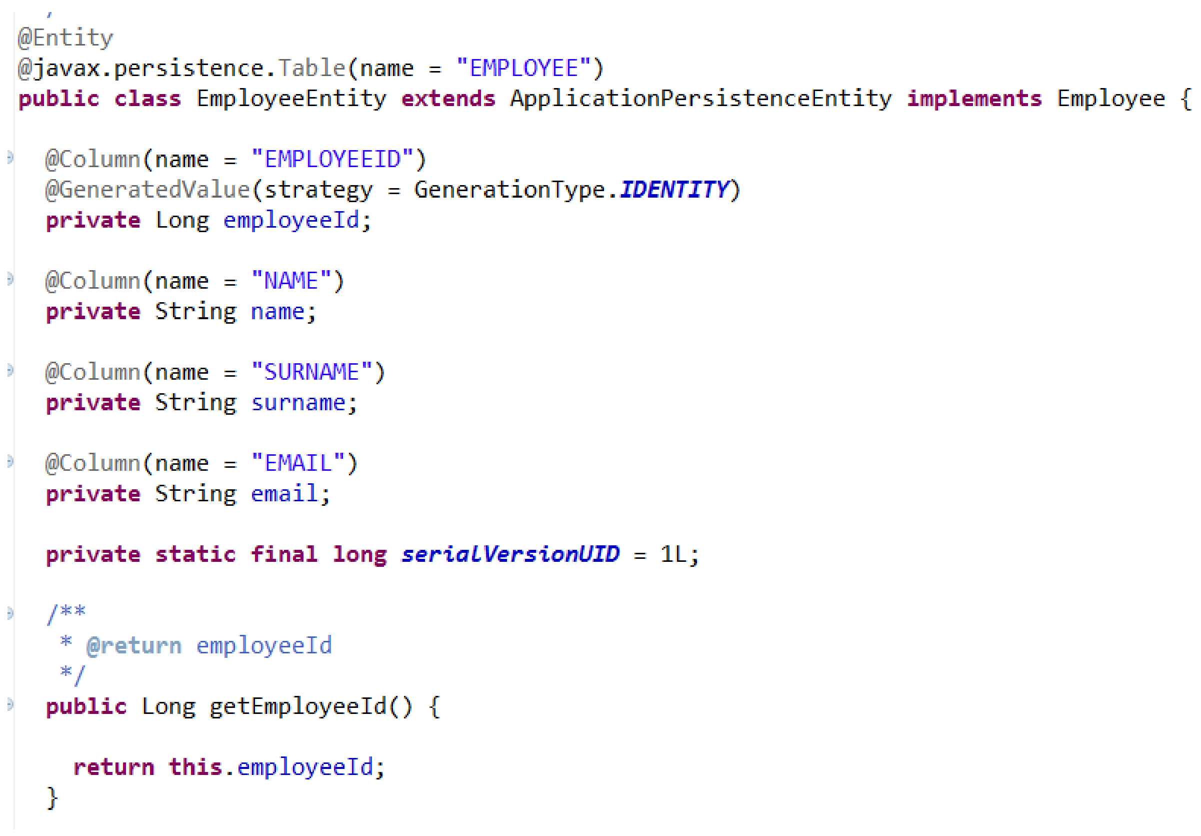
AFTER CobiGen, you will see that the entity previously developed as POJO now has "changed" since it **inherit** from a devonfw base class: **ApplicationPersistenceEntity**

From:

```
        public class EmployeeEntity {
```

to:

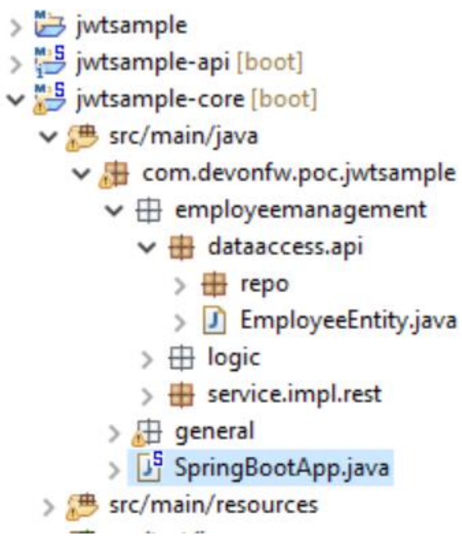
```
        public class EmployeeEntity extends ApplicationPersistenceEntity implements Employee {
```

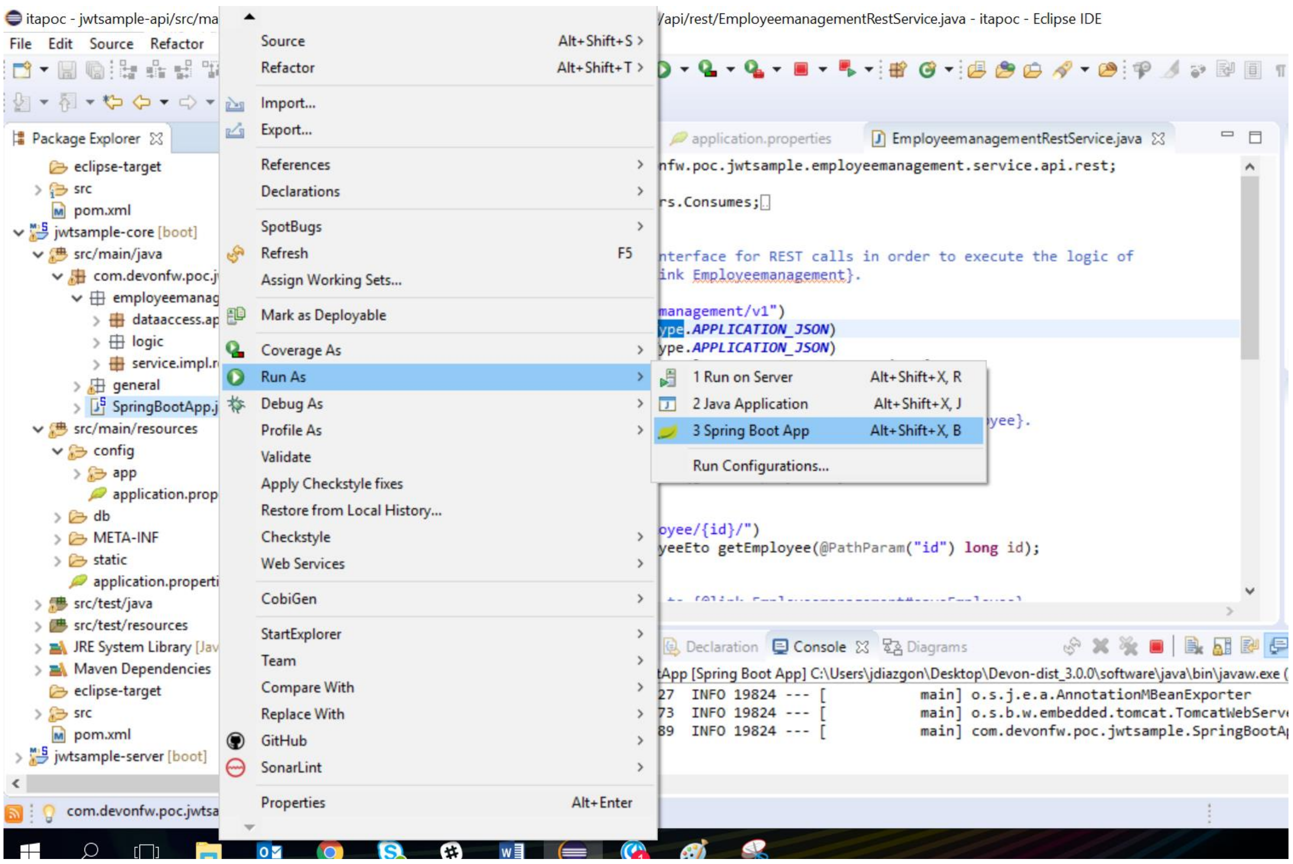


```
@Entity
@javax.persistence.Table(name = "EMPLOYEE")
public class EmployeeEntity extends ApplicationPersistenceEntity implements Employee {

  @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.

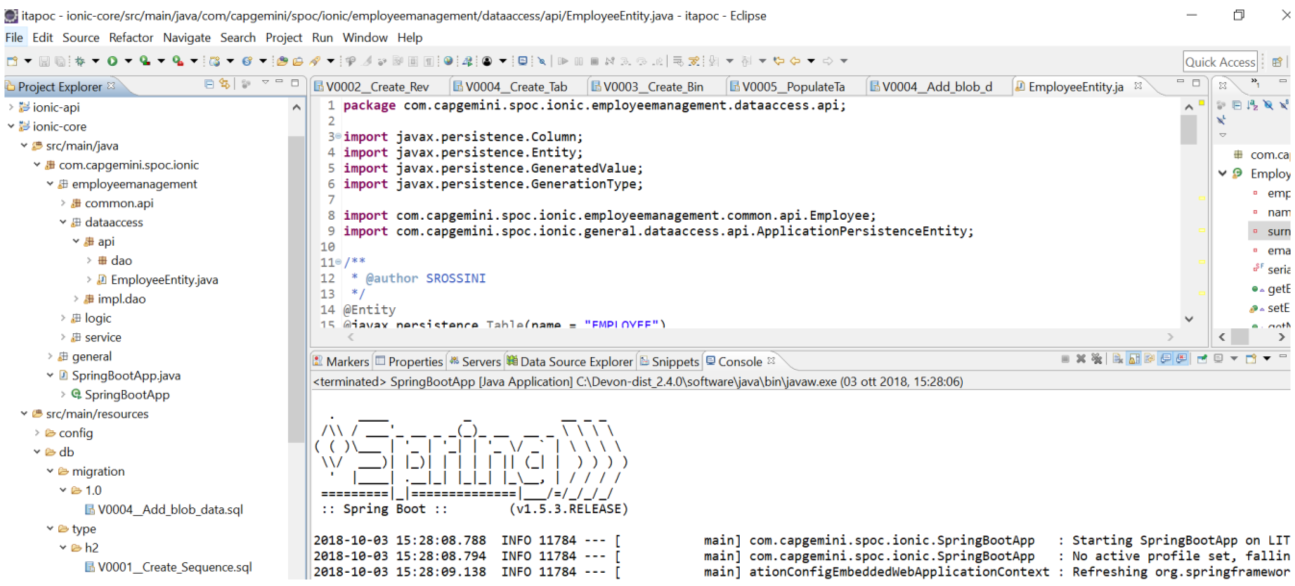
To start a server you just have to right click on “*SpringBootApp.java*” -> *run as* -> *Spring Boot app*





```
nfw.poc.jwtsample.employeemanagement.service.api.rest;

rs.Consumes;

nterface for REST calls in order to execute the logic of
ink Employeemanagement}.

management/v1")
ype.APPLICATION_JSON)
ype.APPLICATION_JSON)

yee}.

oyee/{id}/")
yeeEto getEmployee(@PathParam("id") long id);
```

```
App [Spring Boot App] C:\Users\jdiazgon\Desktop\Devon-dist_3.0.0\software\java\bin\javaw.exe (
27  INFO 19824 --- [           main] o.s.j.e.a.AnnotationMBeanExporter
73  INFO 19824 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServ
89  INFO 19824 --- [           main] com.devonfw.poc.jwtsample.SpringBootA
```



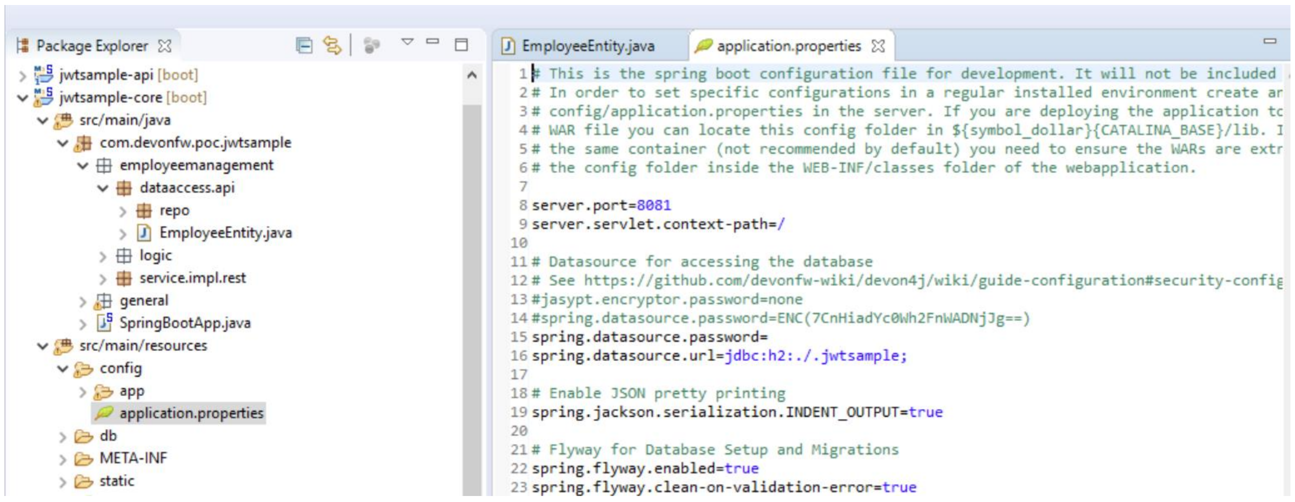
```
package com.capgemini.spoc.ionic.employeemanagement.dataaccess.api;

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

import com.capgemini.spoc.ionic.employeemanagement.common.api.Employee;
import com.capgemini.spoc.ionic.general.dataaccess.api.ApplicationPersistenceEntity;

/**
 * @author SROSSINI
 */
@Entity
@javax.persistence.Table(name = "EMPLOYEE")
```

```
<terminated> SpringBootApp [Java Application] C:\Devon-dist_2.4.0\software\java\bin\javaw.exe (03 ott 2018, 15:28:06)

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.3.RELEASE)

2018-10-03 15:28:08.788  INFO 11784 --- [           main] com.capgemini.spoc.ionic.SpringBootApp   : Starting SpringBootApp on LIT
2018-10-03 15:28:08.794  INFO 11784 --- [           main] com.capgemini.spoc.ionic.SpringBootApp   : No active profile set, fallin
2018-10-03 15:28:09.138  INFO 11784 --- [           main] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframewor
```

BE DONE 😊

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

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

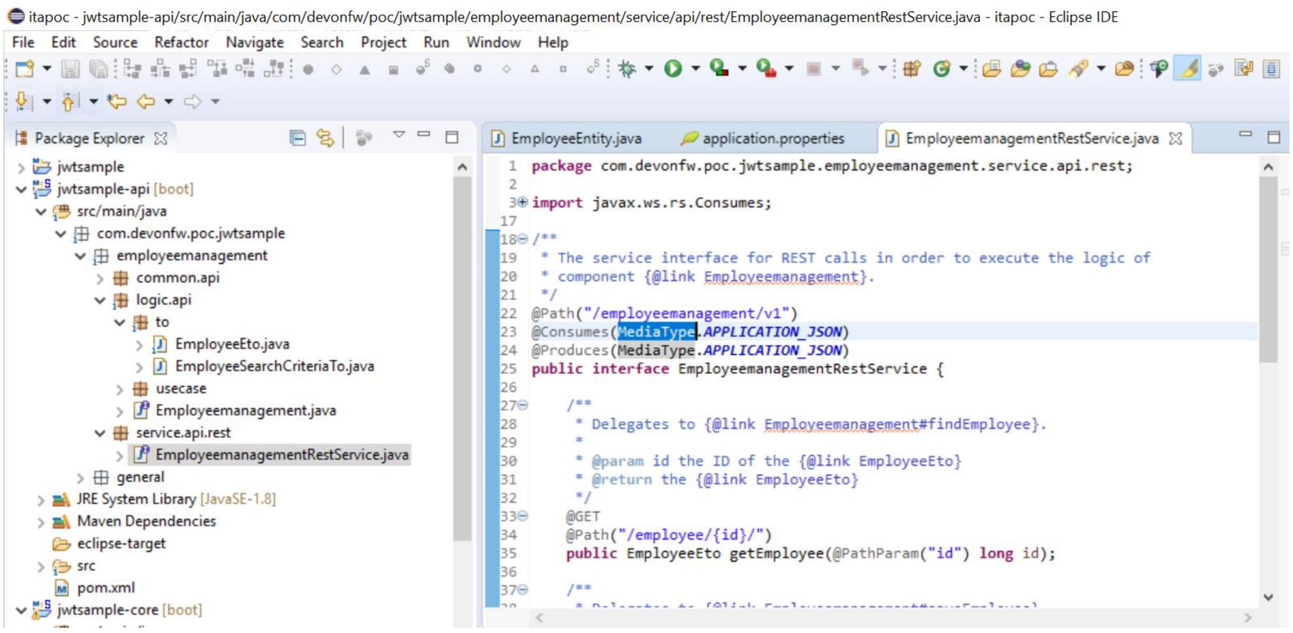


```
# This is the spring boot configuration file for development. It will not be included
# In order to set specific configurations in a regular installed environment create ar
# config/application.properties in the server. If you are deploying the application tc
# WAR file you can locate this config folder in ${symbol_dollar}{CATALINA_BASE}/lib. I
# the same container (not recommended by default) you need to ensure the WARs are extr
# the config folder inside the WEB-INF/classes folder of the webapplication.

server.port=8081
server.servlet.context-path=/

# Datasource for accessing the database
# See https://github.com/devonfw-wiki/devon4j/wiki/guide-configuration#security-config
#jasypt.encryptor.password=none
#spring.datasource.password=ENC(7CnHiadYc0Wh2FnWADNjJg==)
spring.datasource.password=
spring.datasource.url=jdbc:h2:./.jwtsample;

# Enable JSON pretty printing
spring.jackson.serialization.INDENT_OUTPUT=true

# Flyway for Database Setup and Migrations
spring.flyway.enabled=true
spring.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

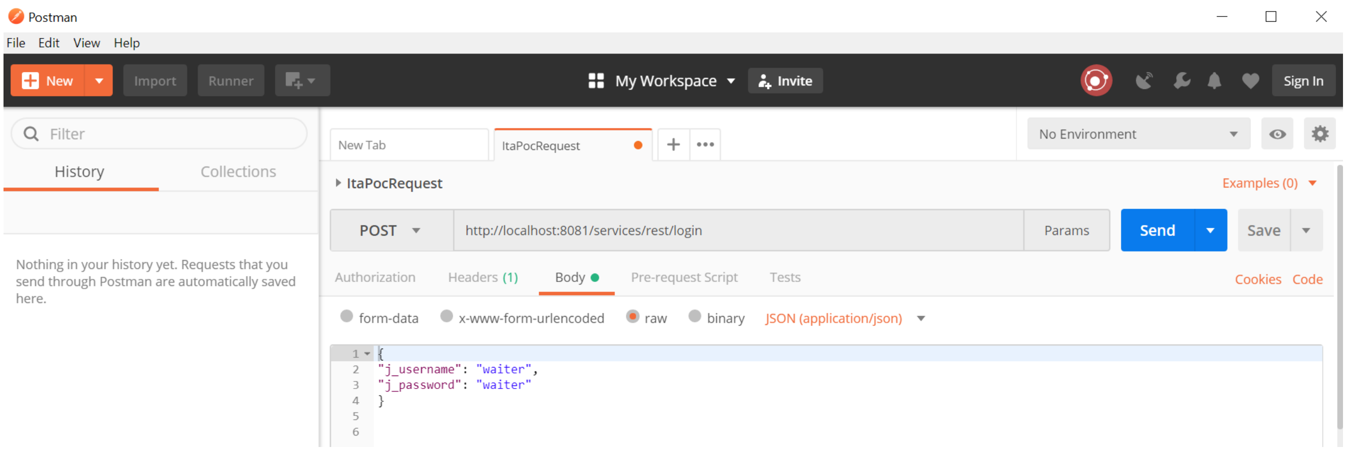
```
package com.devonfw.poc.jwtsample.employeemanagement.service.api.rest;

import javax.ws.rs.Consumes;

/**
 * The service interface for REST calls in order to execute the logic of
 * component {@link Employeemanagement}.
 */
@Path("/employeemanagement/v1")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public interface EmployeemanagementRestService {

    /**
     * Delegates to {@link Employeemanagement#findEmployee}.
     *
     * @param id the ID of the {@link EmployeeEto}
     * @return the {@link EmployeeEto}
     */
    @GET
    @Path("/employee/{id}/")
    public EmployeeEto getEmployee(@PathParam("id") long id);

    /**
```

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"



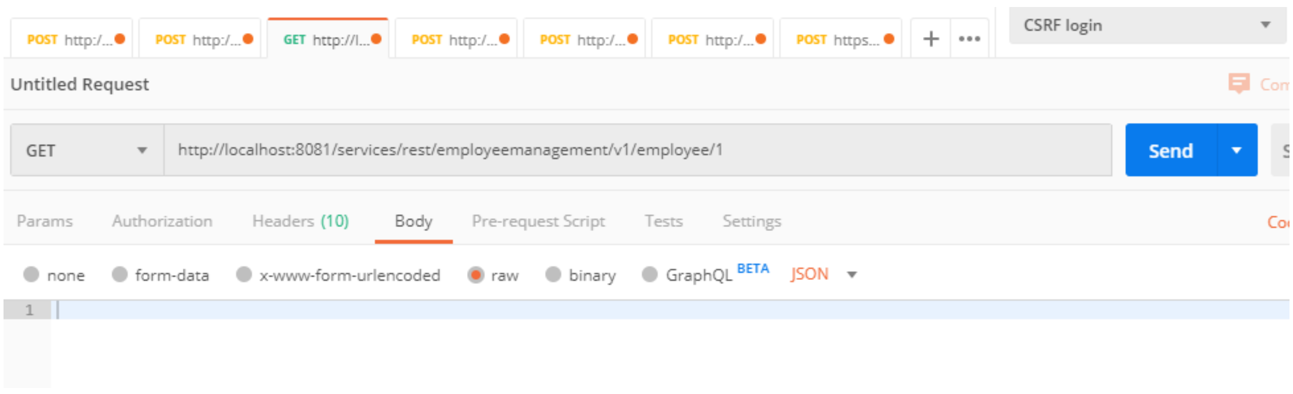
POST http://localhost:8081/services/rest/login

Body: raw, JSON (application/json)

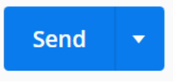
```
{
"j_username": "waiter",
"j_password": "waiter"
}
```

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

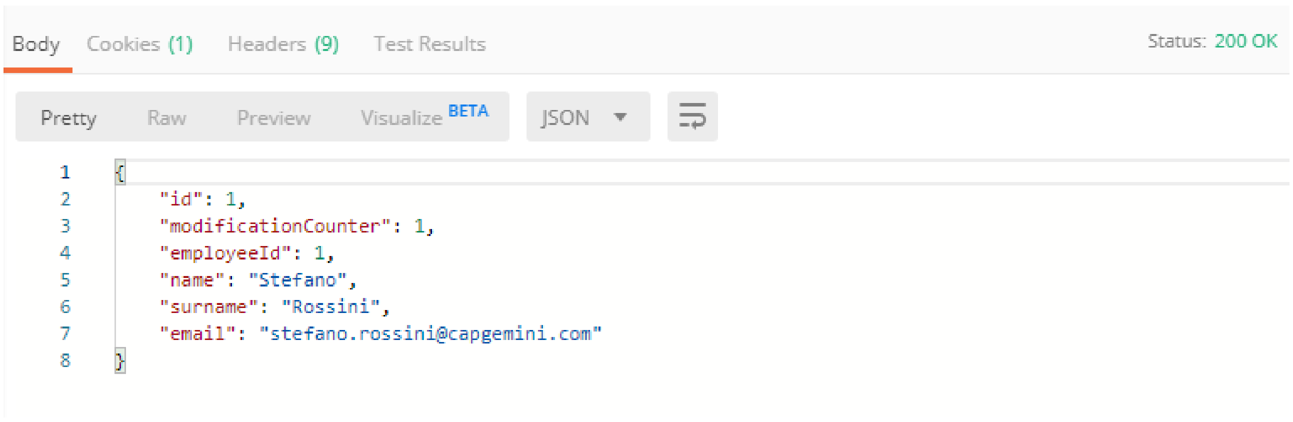
… We create a NEW GET Request in order to get one employee.



```
GET http://localhost:8081/services/rest/employeemanagement/v1/employee/1
```

Now you can click 

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



```
{
    "id": 1,
    "modificationCounter": 1,
    "employeeId": 1,
    "name": "Stefano",
    "surname": "Rossini",
    "email": "stefano.rossini@capgemini.com"
}
```

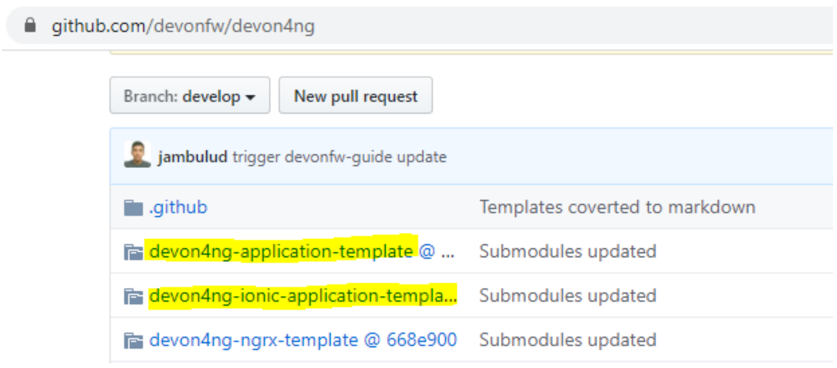
Now that We have successfully tested the BE is time to go to create the FE! We will also enable cors on the BE for the FEs to work (explained later).

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 we need a base application where to generate our code. Go to https://github.com/devonfw/devon4ng and you will see three different git submodules: `devon4ng-application-template`, `devon4ng-ionic-application-template` and `devon4ng-ngrx-template`. We will not use in this tutorial ngrx (state management), but CobiGen is also compatible with it, feel free to test in another time.

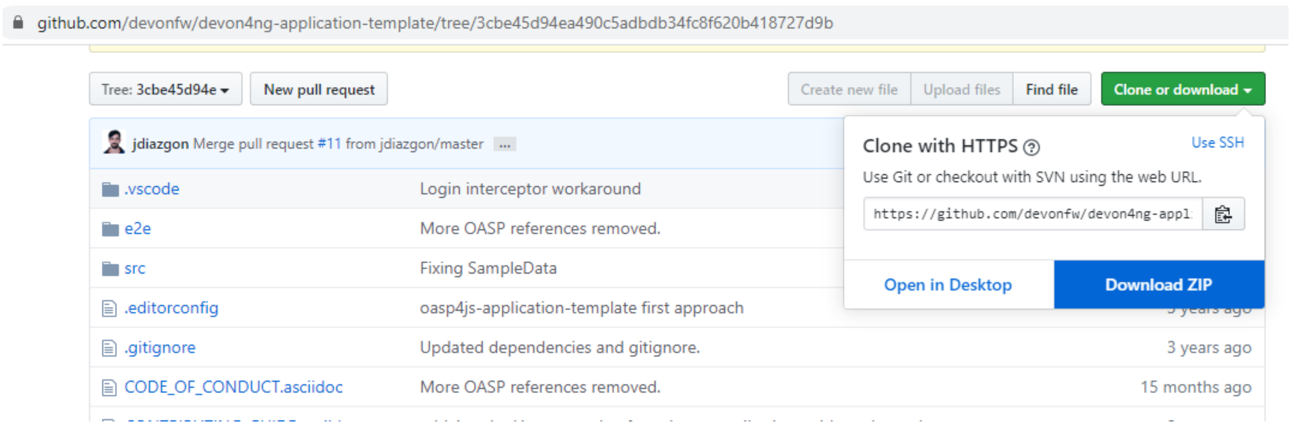


github.com/devonfw/devon4ng

jambulud trigger devonfw-guide update

.github	Templates coverted to markdown
devon4ng-application-template @ ...	Submodules updated
devon4ng-ionic-application-templa...	Submodules updated
devon4ng-ngrx-template @ 668e900	Submodules updated

2. Click on `devon4ng-application-template` and download as a zip the repository:



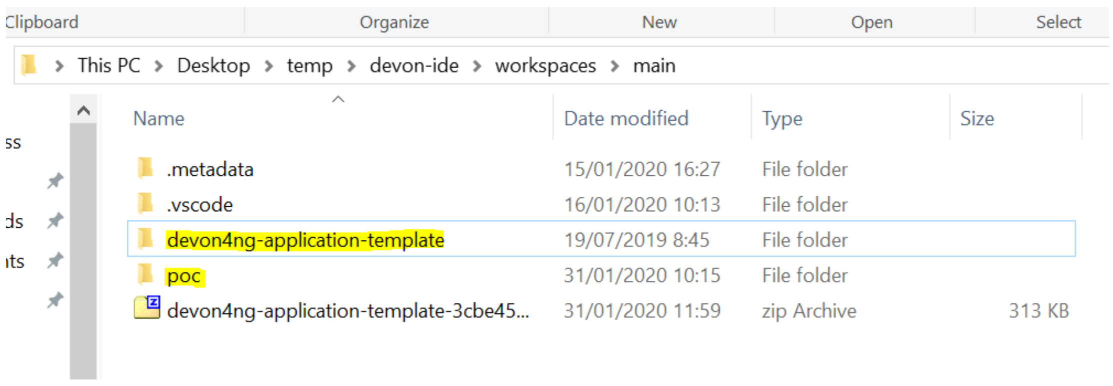
github.com/devonfw/devon4ng-application-template/tree/3cbe45d94ea490c5adbdb34fc8f620b418727d9b

jdiazgon Merge pull request #11 from jdiazgon/master ...

.vscode	Login interceptor workaround	
e2e	More OASP references removed.	
src	Fixing SampleData	
.editorconfig	oasp4js-application-template first approach	3 years ago
.gitignore	Updated dependencies and gitignore.	3 years ago
CODE_OF_CONDUCT.asciidoc	More OASP references removed.	15 months ago

Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/devonfw/devon4ng-appl

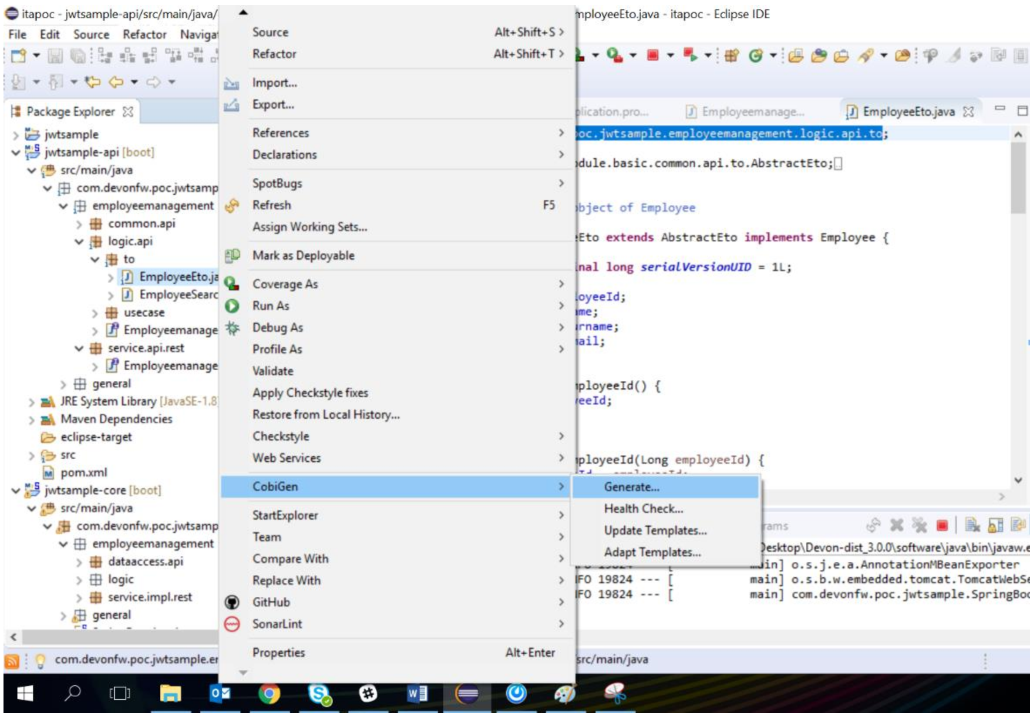
3. Extract the "**devon4ng-application-template-….**" zip file contents to the location (Path location \Devon-ide\workspaces\main) and rename it to "**devon4ng-application-template".**
Note: It is very important to use the same name for CobiGen to know where to generate the code.



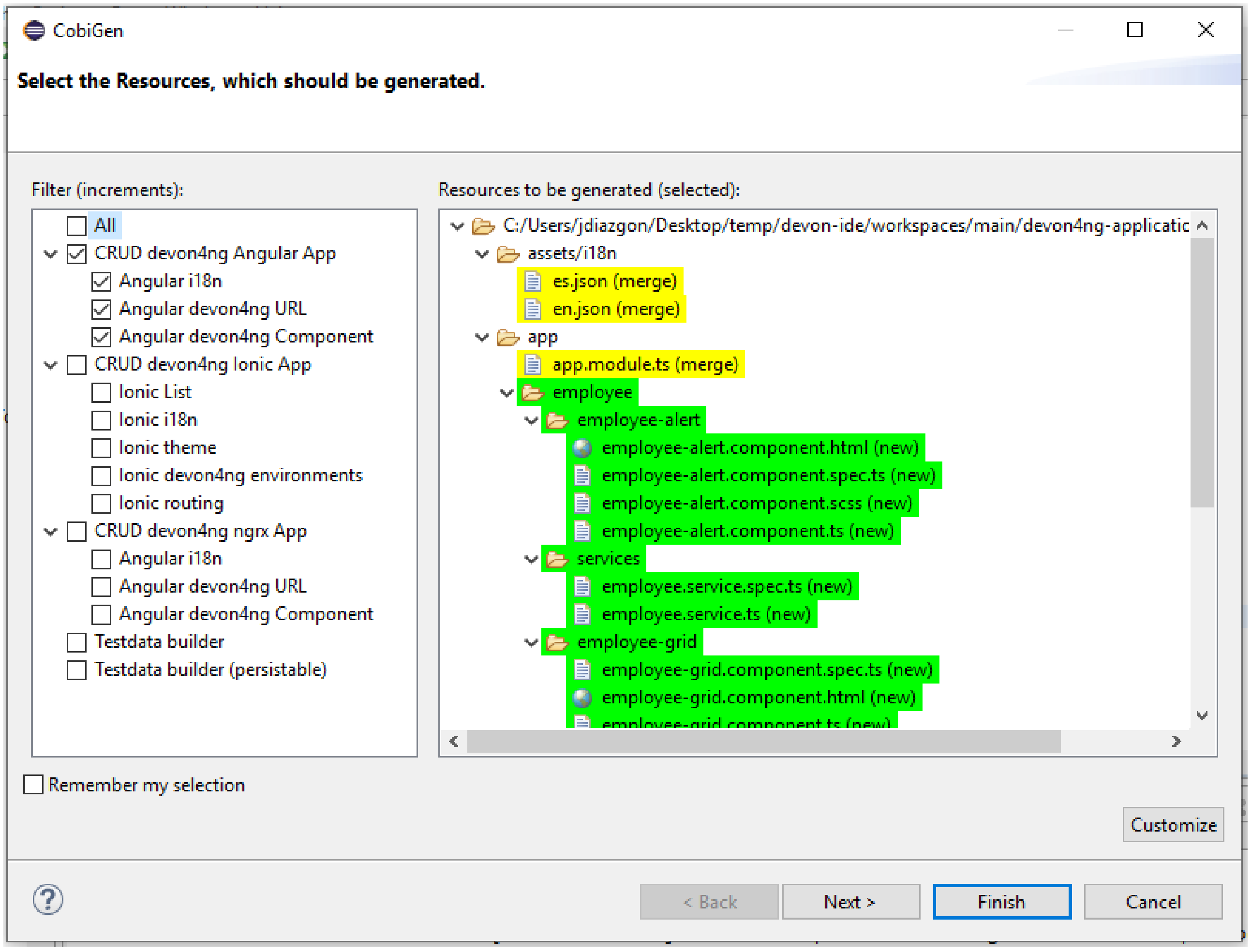
Name	Date modified	Type	Size
.metadata	15/01/2020 16:27	File folder	
.vscode	16/01/2020 10:13	File folder	
devon4ng-application-template	19/07/2019 8:45	File folder	
poc	31/01/2020 10:15	File folder	
devon4ng-application-template-3cbe45...	31/01/2020 11:59	zip Archive	313 KB

Note: As you can see, the front-end is next to our devon4j project (poc).

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



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


CobiGen

Select the Resources, which should be generated.

Filter (increments):

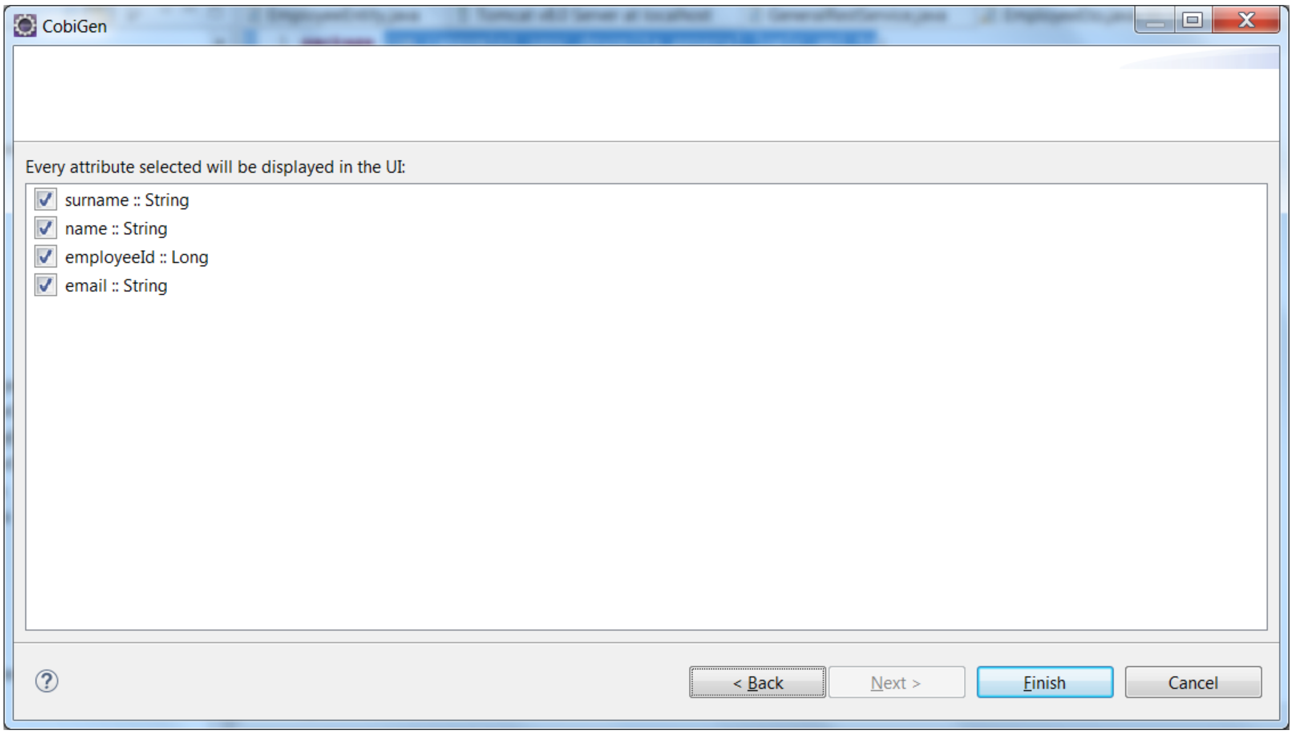
- [ ] All
- [x] CRUD devon4ng Angular App
  - [x] Angular i18n
  - [x] Angular devon4ng URL
  - [x] Angular devon4ng Component
- [ ] CRUD devon4ng Ionic App
  - [ ] Ionic List
  - [ ] Ionic i18n
  - [ ] Ionic theme
  - [ ] Ionic devon4ng environments
  - [ ] Ionic routing
- [ ] CRUD devon4ng ngrx App
  - [ ] Angular i18n
  - [ ] Angular devon4ng URL
  - [ ] Angular devon4ng Component
- [ ] Testdata builder
- [ ] Testdata builder (persistable)

Resources to be generated (selected):

```
C:/Users/jdiazgon/Desktop/temp/devon-ide/workspaces/main/devon4ng-applicatic
  assets/i18n
    es.json (merge)
    en.json (merge)
  app
    app.module.ts (merge)
    employee
      employee-alert
        employee-alert.component.html (new)
        employee-alert.component.spec.ts (new)
        employee-alert.component.scss (new)
        employee-alert.component.ts (new)
      services
        employee.service.spec.ts (new)
        employee.service.ts (new)
      employee-grid
        employee-grid.component.spec.ts (new)
        employee-grid.component.html (new)
        employee-grid.component.ts (new)
```

- [ ] Remember my selection

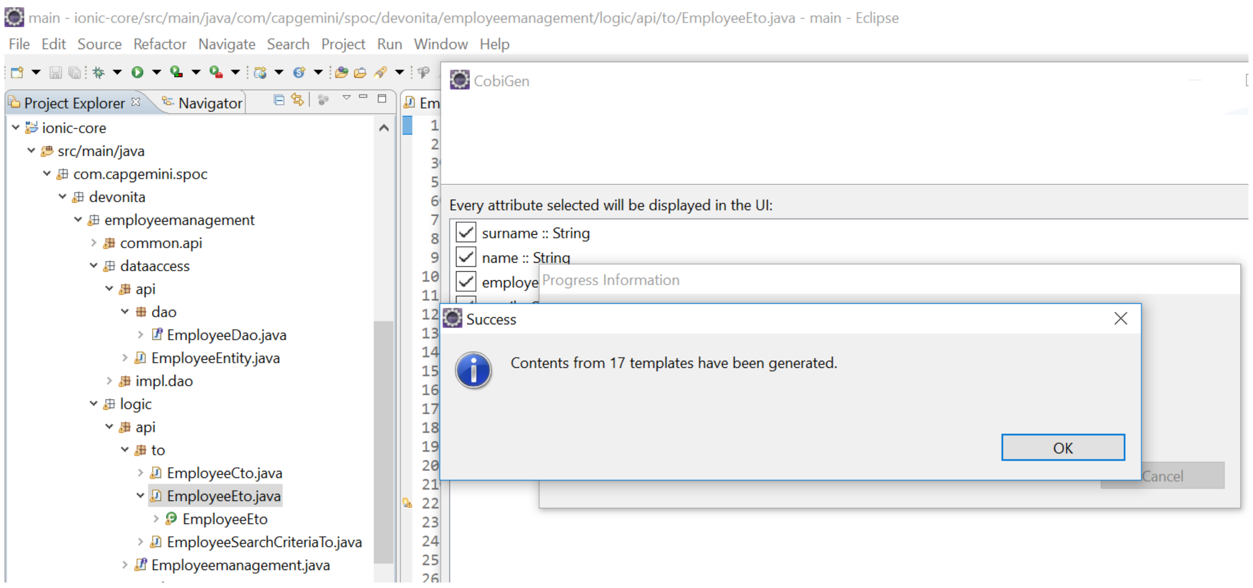
6. Click on Next


CobiGen

Every attribute selected will be displayed in the UI:

- [x] surname :: String
- [x] name :: String
- [x] employeeId :: Long
- [x] email :: String

7. Click on Finish



main - ionic-core/src/main/java/com/capgemini/spoc/devonita/employeemanagement/logic/api/to/EmployeeEto.java - main - Eclipse

Every attribute selected will be displayed in the UI:

- [✓] surname :: String
- [✓] name :: String
- [✓] employe

Progress Information

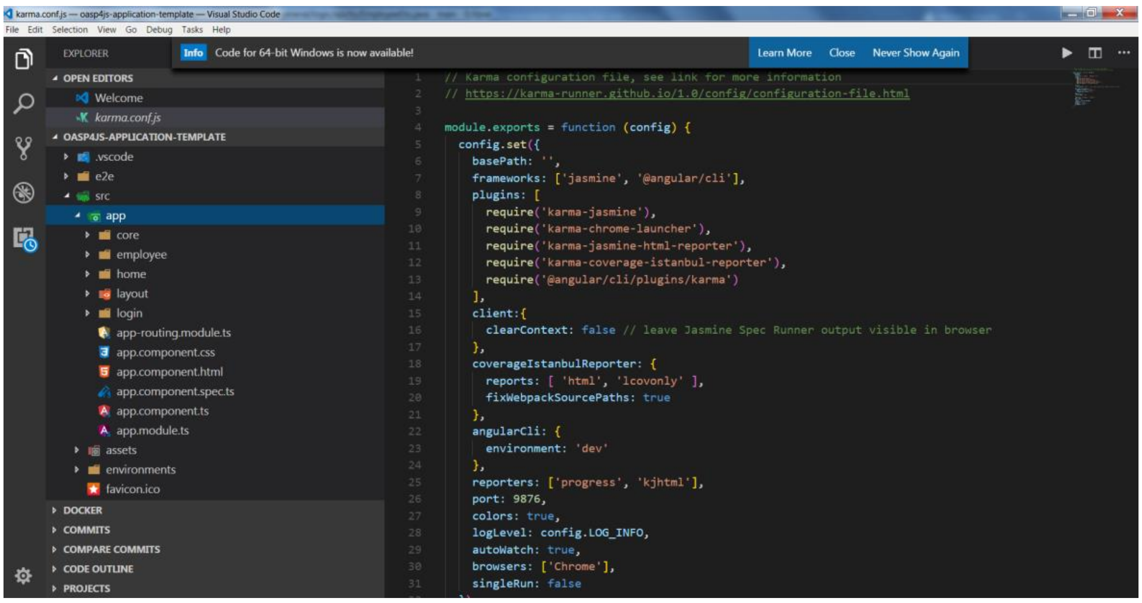
Success

Contents from 17 templates have been generated.

OK

8. The entire ANGULAR structure has been auto generated.

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



karma.conf.js — oasp4js-application-template — Visual Studio Code

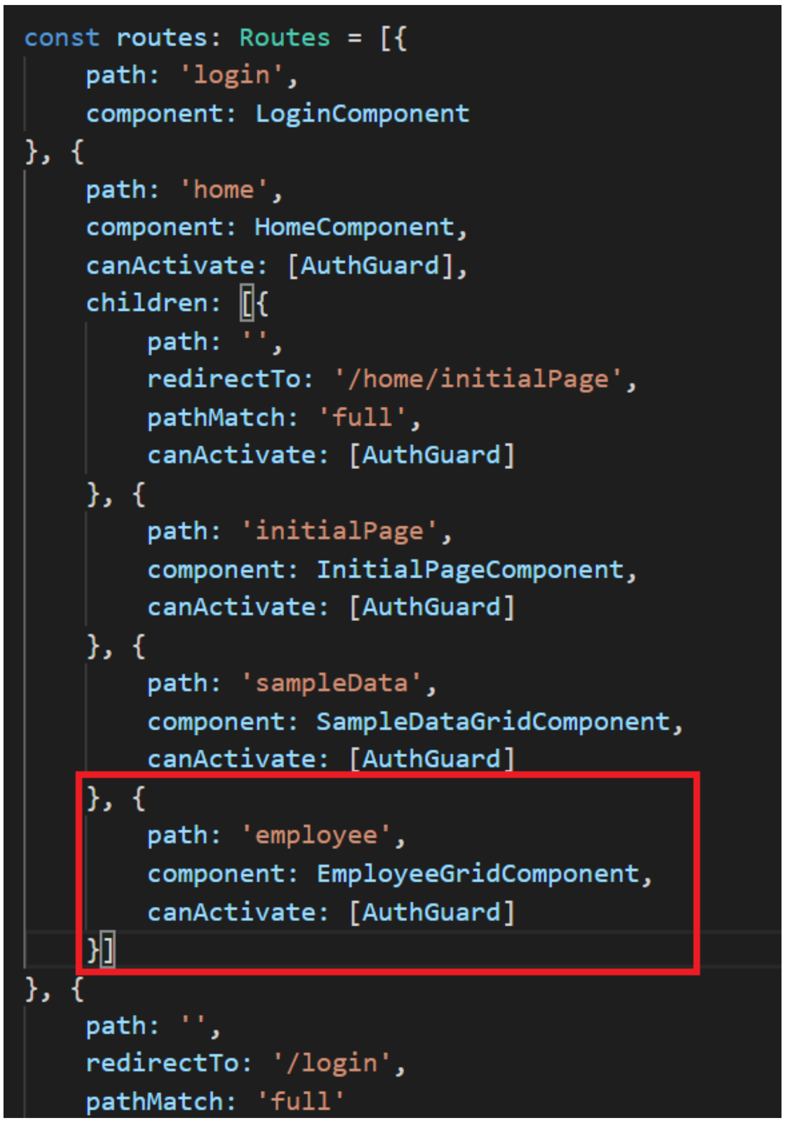
Info Code for 64-bit Windows is now available!

```
// Karma configuration file, see link for more information
// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine', '@angular/cli'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      require('karma-jasmine-html-reporter'),
      require('karma-coverage-istanbul-reporter'),
      require('@angular/cli/plugins/karma')
    ],
    client:{
      clearContext: false // leave Jasmine Spec Runner output visible in browser
    },
    coverageIstanbulReporter: {
      reports: [ 'html', 'lcovonly' ],
      fixWebpackSourcePaths: true
    },
    angularCli: {
      environment: 'dev'
    },
    reporters: ['progress', 'kjhtml'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false
```

9. 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

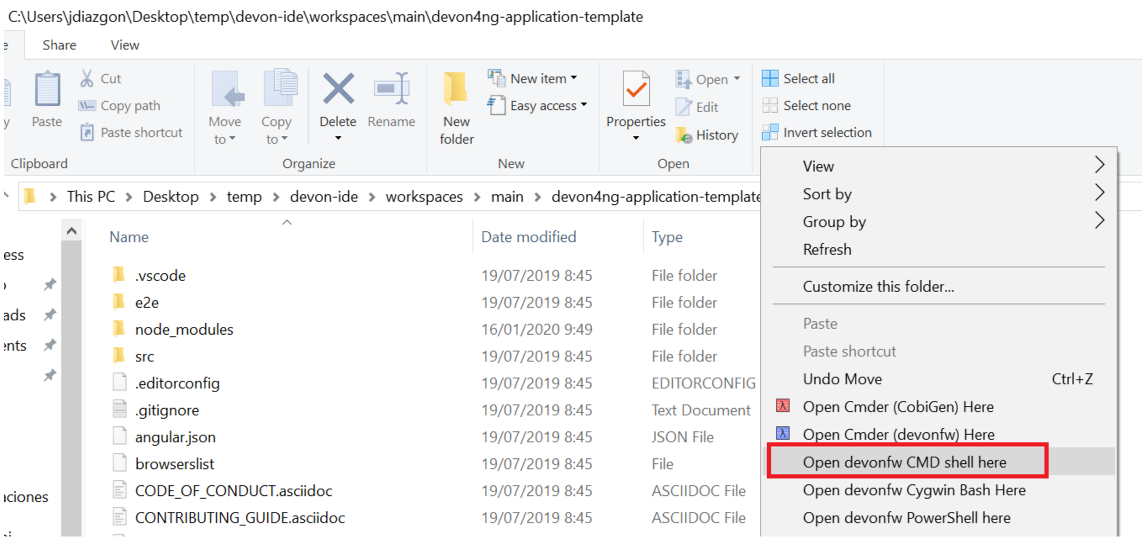
```
  ,{
      path: 'employee',
      component: EmployeeGridComponent,
      canActivate: [AuthGuard],
  },
```

Following picture explain where to place the above content:



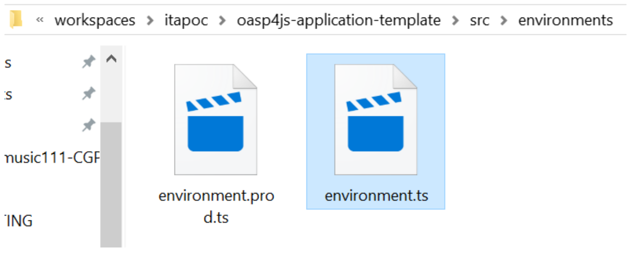
```
const routes: Routes = [{
    path: 'login',
    component: LoginComponent
}, {
    path: 'home',
    component: HomeComponent,
    canActivate: [AuthGuard],
    children: [{
        path: '',
        redirectTo: '/home/initialPage',
        pathMatch: 'full',
        canActivate: [AuthGuard]
  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

10. Now go to “workspaces\main\devon4ng-application-template” and right click, select “Open devonfw CMD shell here”. It will open a console. Execute “npm install” command which would download all the required libraries and node modules folder would be generated.

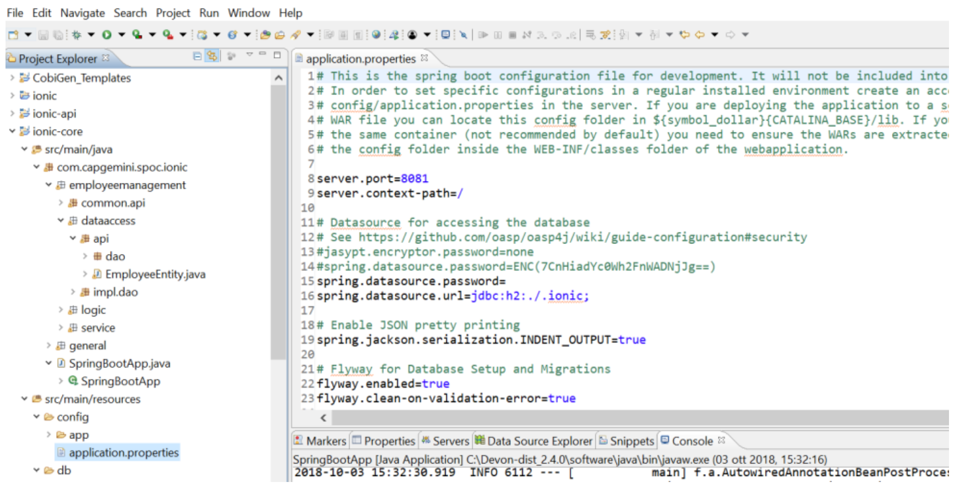

C:\Users\jdiazgon\Desktop\temp\devon-ide\workspaces\main\devon4ng-application-template
Share
View
Cut
Copy path
Paste shortcut
Paste
Move to
Copy to
Delete
Rename
New folder
New item
Easy access
Properties
Open
Edit
History
Select all
Select none
Invert selection
Clipboard
Organize
New
Open
This PC › Desktop › temp › devon-ide › workspaces › main › devon4ng-application-template
View
Sort by
Group by
Refresh
Customize this folder...
Paste
Paste shortcut
Undo Move Ctrl+Z
Open Cmder (CobiGen) Here
Open Cmder (devonfw) Here
Open devonfw CMD shell here
Open devonfw Cygwin Bash Here
Open devonfw PowerShell here

Name	Date modified	Type
.vscode	19/07/2019 8:45	File folder
e2e	19/07/2019 8:45	File folder
node_modules	16/01/2020 9:49	File folder
src	19/07/2019 8:45	File folder
.editorconfig	19/07/2019 8:45	EDITORCONFIG
.gitignore	19/07/2019 8:45	Text Document
angular.json	19/07/2019 8:45	JSON File
browserslist	19/07/2019 8:45	File
CODE_OF_CONDUCT.asciidoc	19/07/2019 8:45	ASCIIDOC File
CONTRIBUTING_GUIDE.asciidoc	19/07/2019 8:45	ASCIIDOC File

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


« workspaces › itapoc › oasp4js-application-template › src › environments
environment.prod.ts
environment.ts

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



```
# This is the spring boot configuration file for development. It will not be included into
# In order to set specific configurations in a regular installed environment create an acc
# config/application.properties in the server. If you are deploying the application to a s
# WAR file you can locate this config folder in ${symbol_dollar}{CATALINA_BASE}/lib. If yo
# the same container (not recommended by default) you need to ensure the WARs are extracte
# the config folder inside the WEB-INF/classes folder of the webapplication.

server.port=8081
server.context-path=/

# Datasource for accessing the database
# See https://github.com/oasp/oasp4j/wiki/guide-configuration#security
#jasypt.encryptor.password=none
#spring.datasource.password=ENC(7CnHiadYc0Wh2FnWADNjJg==)
spring.datasource.password=
spring.datasource.url=jdbc:h2:./.ionic;

# Enable JSON pretty printing
spring.jackson.serialization.INDENT_OUTPUT=true

# Flyway for Database Setup and Migrations
flyway.enabled=true
flyway.clean-on-validation-error=true
```

```
SpringBootApp [Java Application] C:\Devon-dist_2.4.0\software\java\bin\javaw.exe (03 ott 2018, 15:32:16)
2018-10-03 15:32:30.919  INFO 6112 --- [           main] f.a.AutowiredAnnotationBeanPostProce
```

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

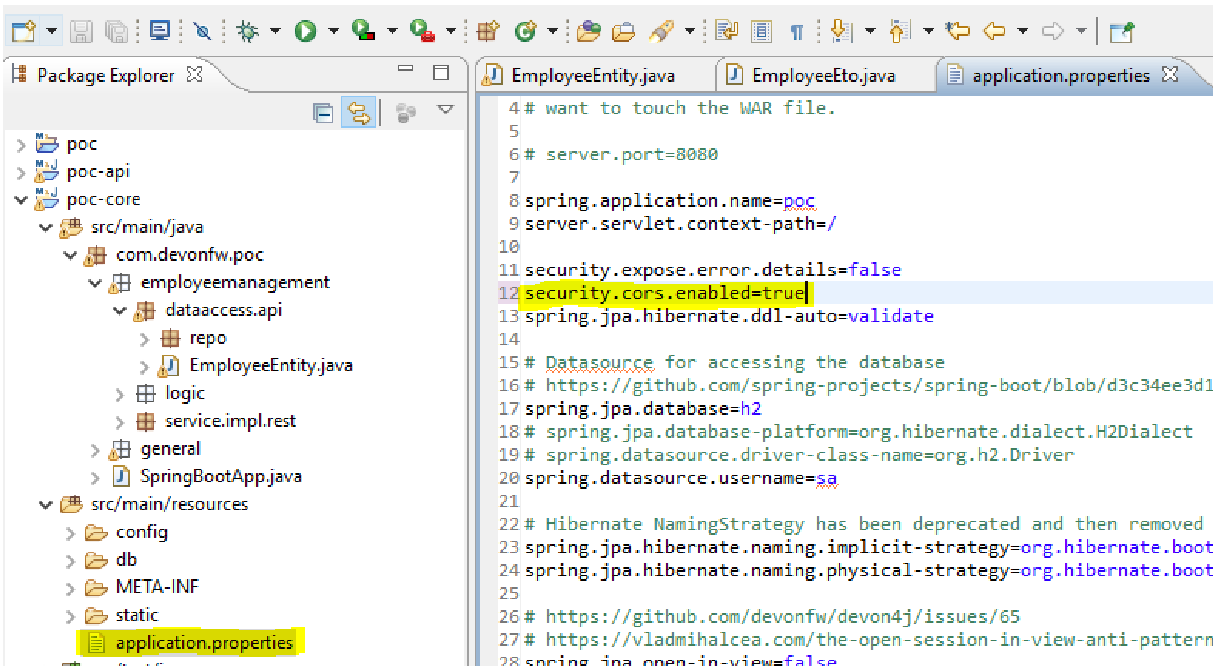
export const environment = {
  production: false,
  restPathRoot: 'http://localhost:8081/',
  restServiceRoot: 'http://localhost:8081/services/rest/',

security: **'csrf'**

};

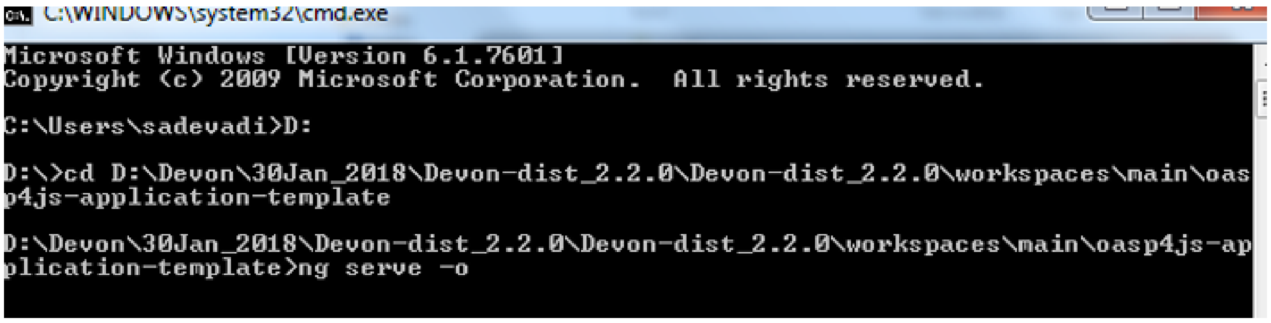
Warning: REMEMBER to set security filed to **csrf**

12. Now we need to enable cors on the back-end, in order to allow some cross-origin requests. Go to poc-core/src/main/resources/application.properties and set `security.cors.enabled=true`



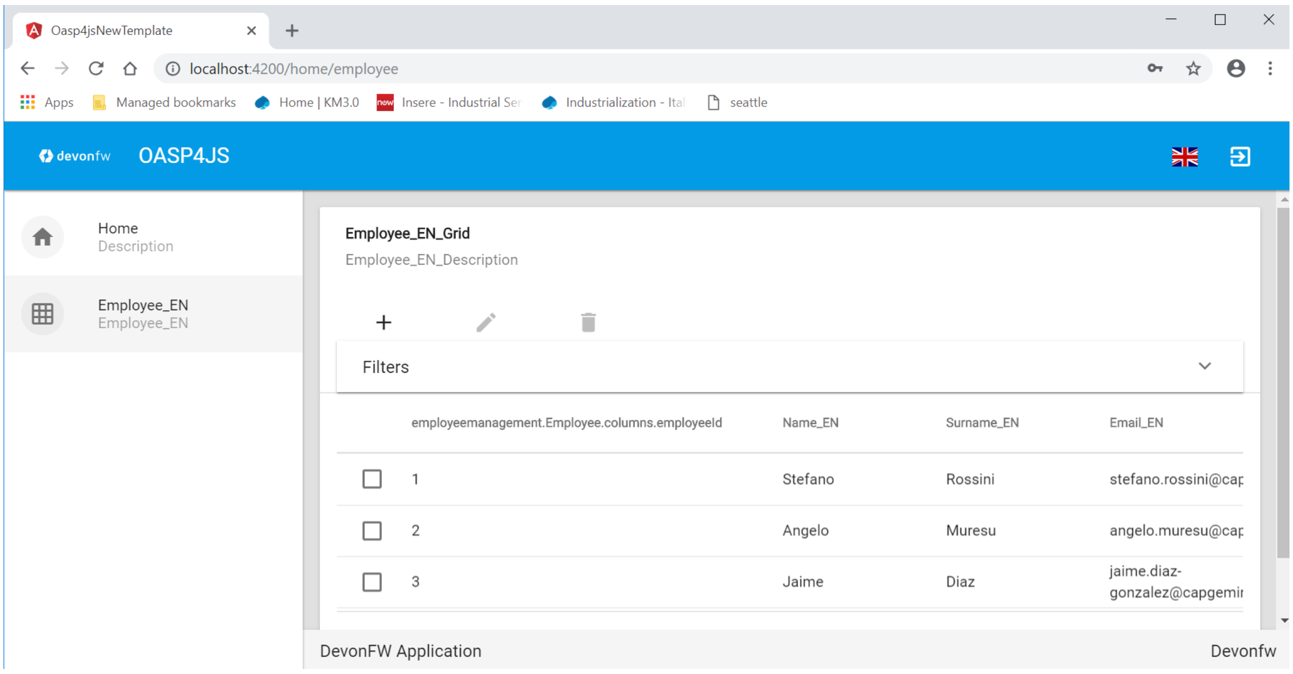
```
# want to touch the WAR file.

# server.port=8080

spring.application.name=poc
server.servlet.context-path=/

security.expose.error.details=false
security.cors.enabled=true
spring.jpa.hibernate.ddl-auto=validate

# Datasource for accessing the database
# https://github.com/spring-projects/spring-boot/blob/d3c34ee3d1
spring.jpa.database=h2
# spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
# spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa

# Hibernate NamingStrategy has been deprecated and then removed
spring.jpa.hibernate.naming.implicit-strategy=org.hibernate.boot
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot

# https://github.com/devonfw/devon4j/issues/65
# https://vladmihalcea.com/the-open-session-in-view-anti-pattern
spring.jpa.open-in-view=false
```

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



```
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\oas
p4js-application-template

D:\Devon\30Jan_2018\Devon-dist_2.2.0\Devon-dist_2.2.0\workspaces\main\oasp4js-ap
plication-template>ng serve -o
```

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



Employee_EN_Grid

Employee_EN_Description

Filters

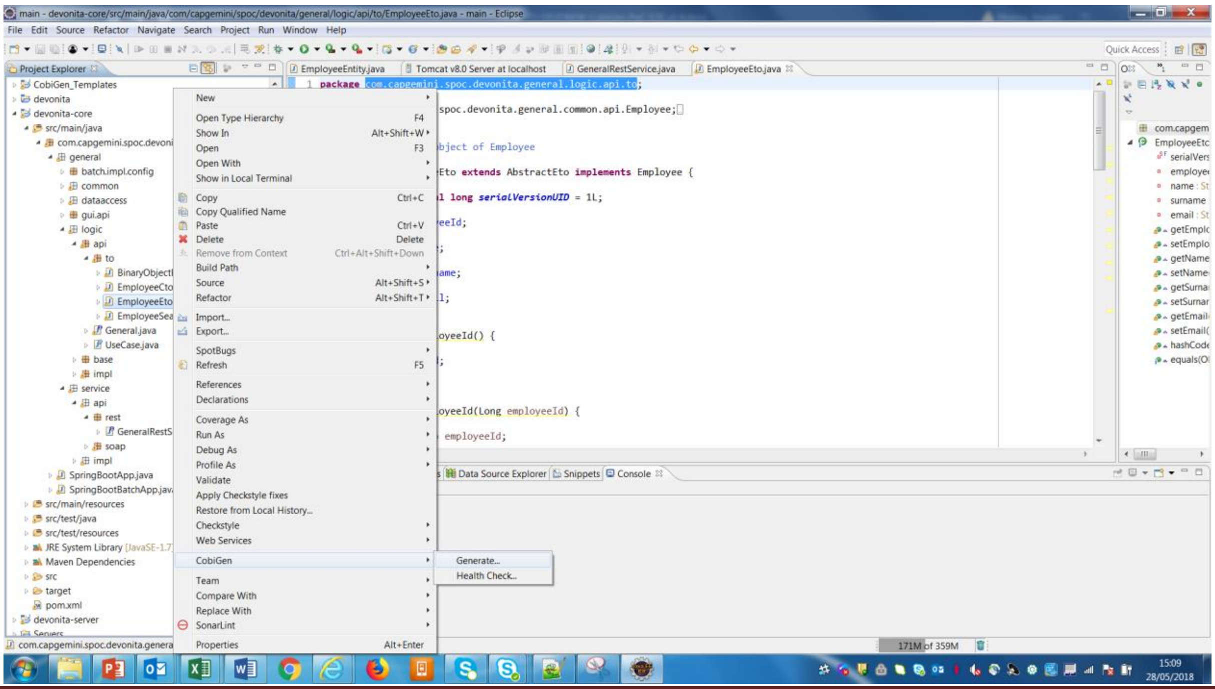
	employeemanagement.Employee.columns.employeeId	Name_EN	Surname_EN	Email_EN
☐	1	Stefano	Rossini	stefano.rossini@cap
☐	2	Angelo	Muresu	angelo.muresu@cap
☐	3	Jaime	Diaz	jaime.diaz-gonzalez@capgemi

DevonFW Application Devonfw

FE WebApp DONE 😊

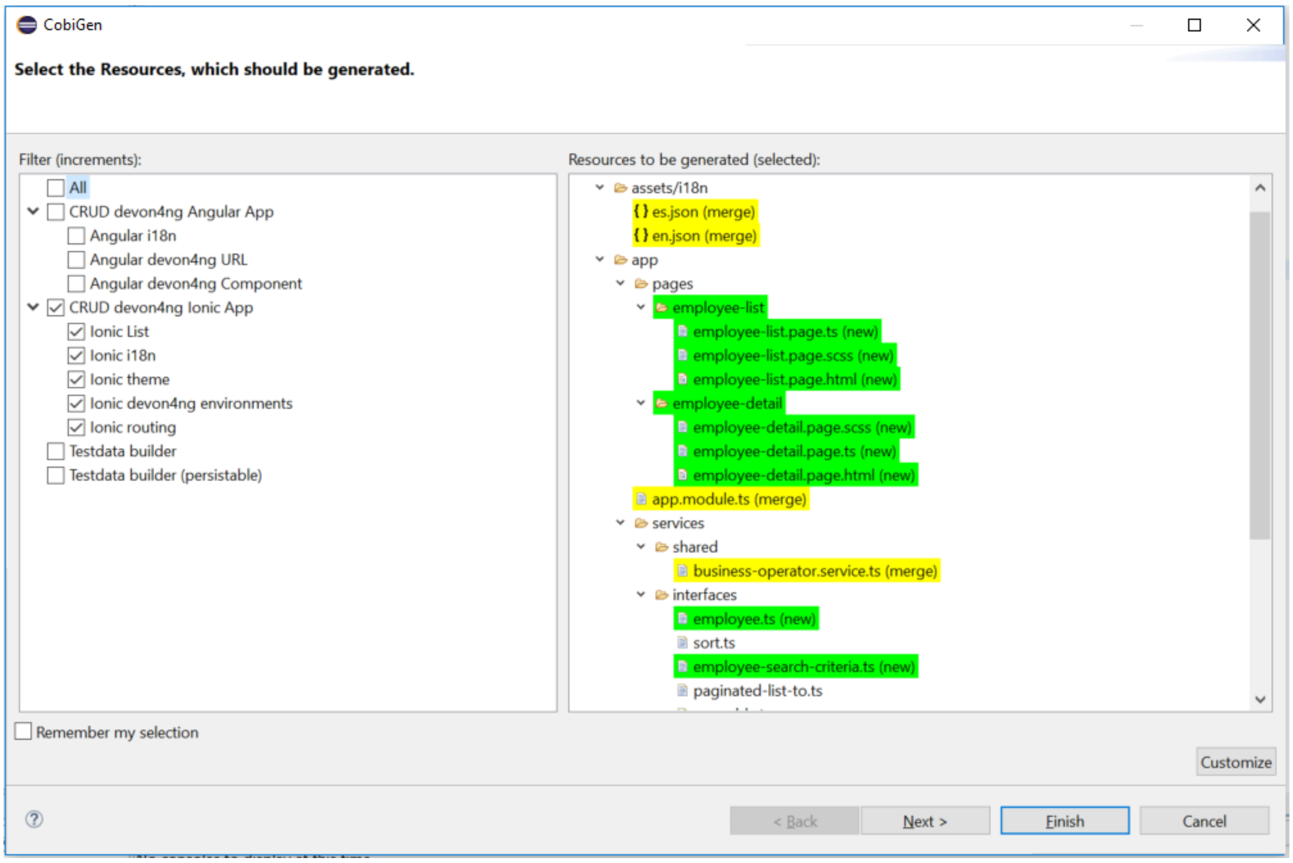
2.2.2 Front End Mobile App

1. Now, for the Ionic structure to be auto-generated, go to https://github.com/devonfw/devon4ng. Click "**devon4ng-ionic-application-template**" project and download it as a zip (just follow the same process as we did previously with Angular). **Remember**: the folder name should be "devon4ng-ionic-application-template" and it should be located next to "poc".
2. Once done, Right click on the **EmployeeEto.java** as you already did before in order to use CobiGen.



```
package com.capgemini.spoc.devonita.general.logic.api.to;
spoc.devonita.general.common.api.Employee;
bject of Employee
Eto extends AbstractEto implements Employee {
l long serialVersionUID = 1L;
```

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



CobiGen

Select the Resources, which should be generated.

Filter (increments):

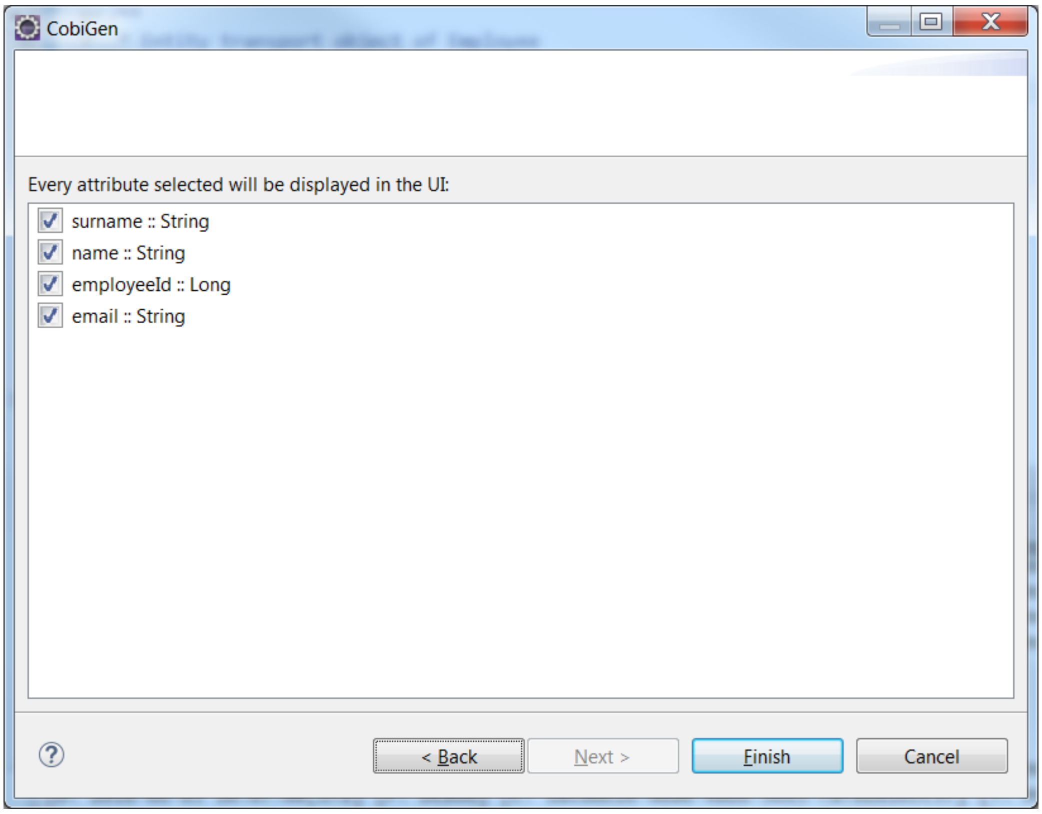
- [ ] All
- [ ] CRUD devon4ng Angular App
  - [ ] Angular i18n
  - [ ] Angular devon4ng URL
  - [ ] Angular devon4ng Component
- [x] CRUD devon4ng Ionic App
  - [x] Ionic List
  - [x] Ionic i18n
  - [x] Ionic theme
  - [x] Ionic devon4ng environments
  - [x] Ionic routing
- [ ] Testdata builder
- [ ] Testdata builder (persistable)

Resources to be generated (selected):

```
assets/i18n
    es.json (merge)
    en.json (merge)
app
    pages
        employee-list
            employee-list.page.ts (new)
            employee-list.page.scss (new)
            employee-list.page.html (new)
        employee-detail
            employee-detail.page.scss (new)
            employee-detail.page.ts (new)
            employee-detail.page.html (new)
    app.module.ts (merge)
    services
        shared
            business-operator.service.ts (merge)
        interfaces
            employee.ts (new)
            sort.ts
            employee-search-criteria.ts (new)
            paginated-list-to.ts
```

- [ ] Remember my selection

4. Click on Next.

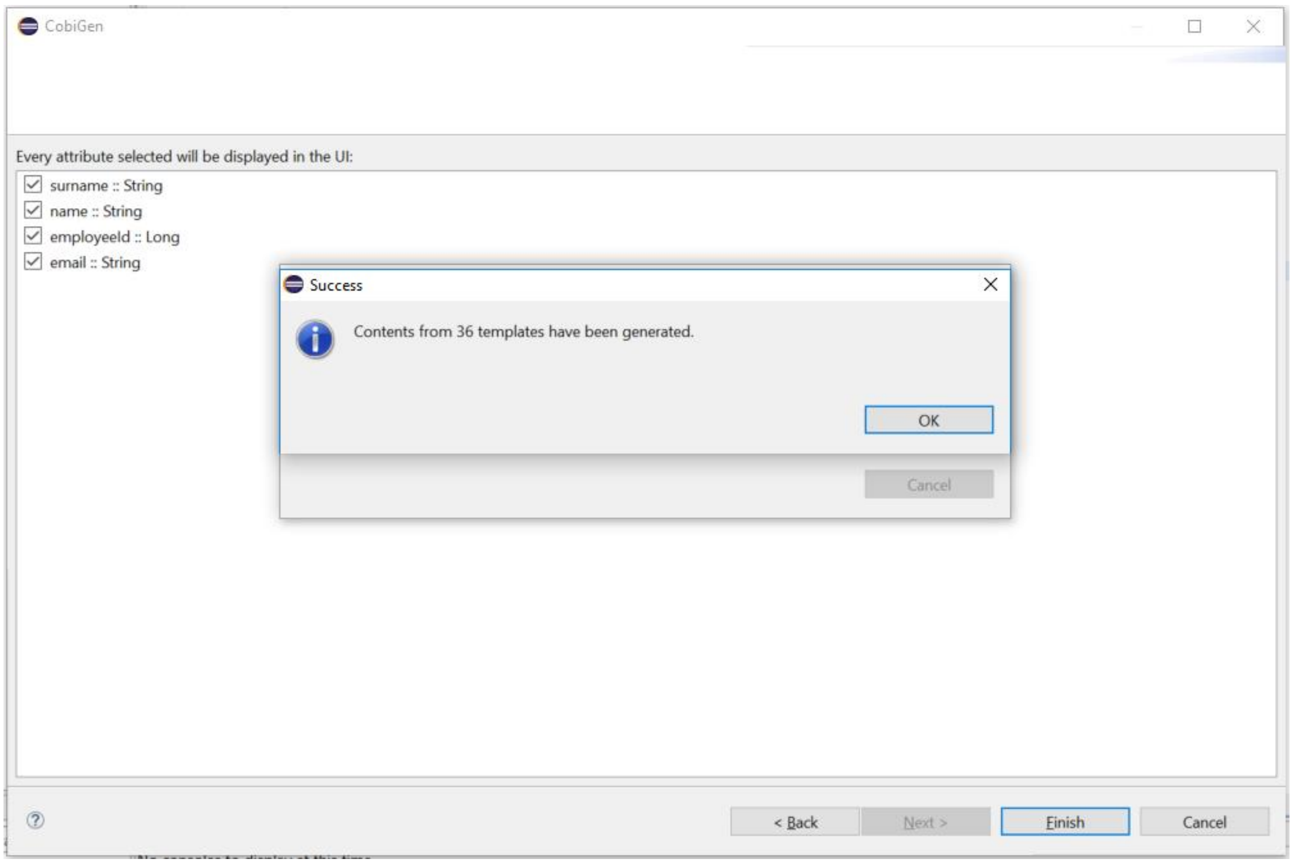


CobiGen

Every attribute selected will be displayed in the UI:

- [x] surname :: String
- [x] name :: String
- [x] employeeId :: Long
- [x] email :: String

5. Click on Finish



Every attribute selected will be displayed in the UI:

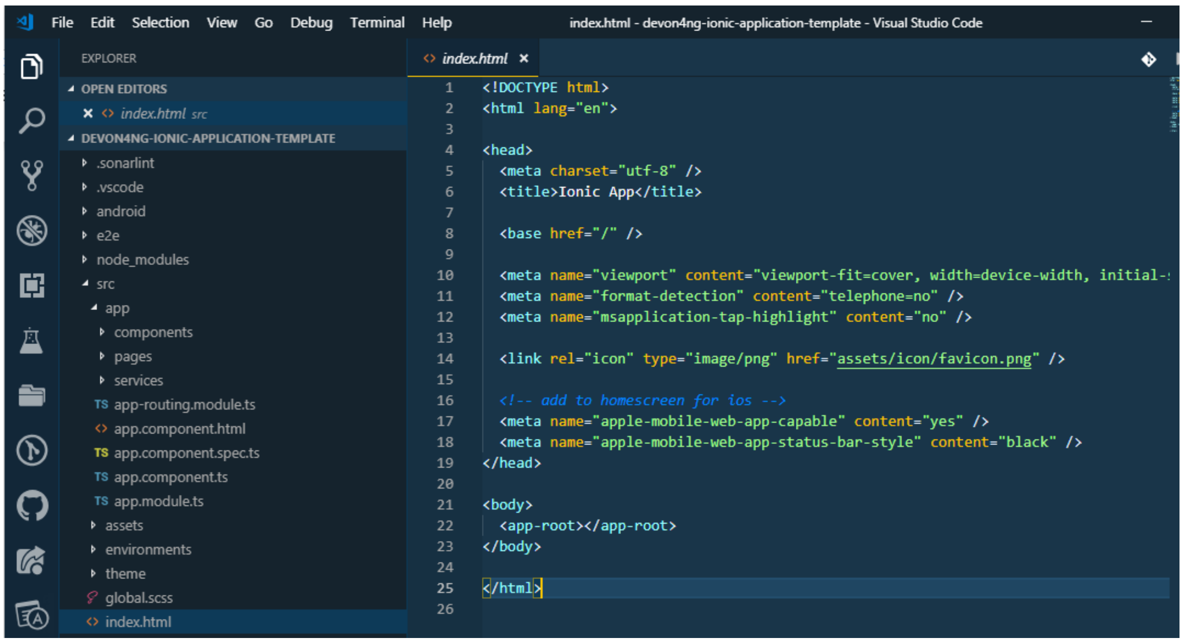
- [✓] surname :: String
- [✓] name :: String
- [✓] employeeId :: Long
- [✓] email :: String

Success

Contents from 36 templates have been generated.

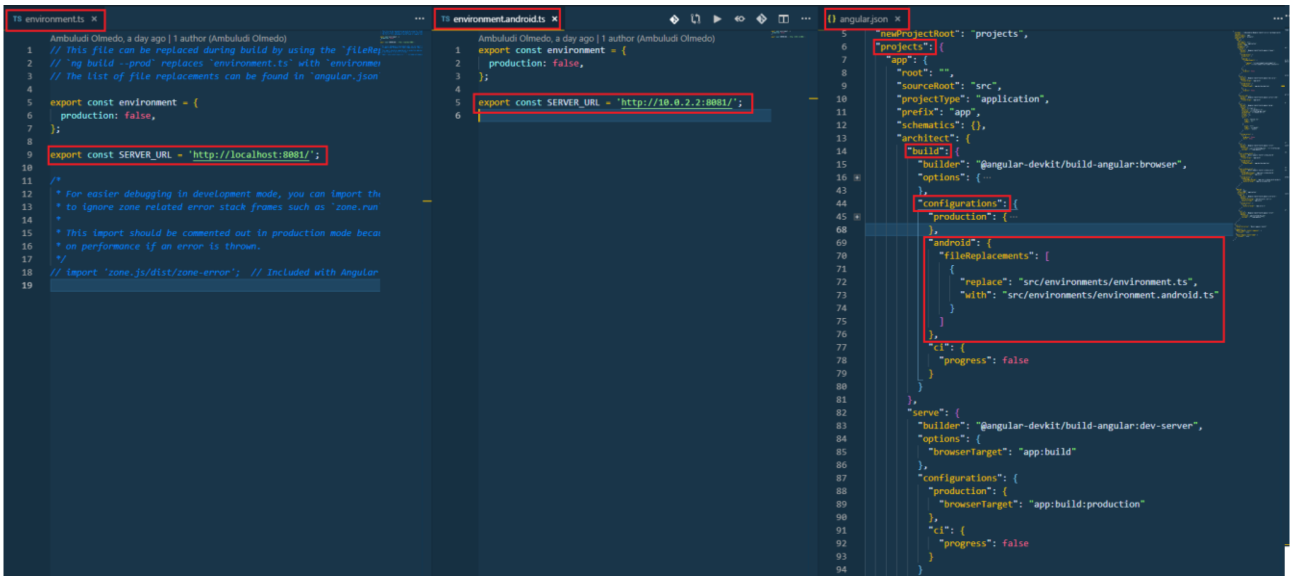
6. The entire ionic structure will be auto generated.

The entire <u>Ionic APP</u> structure would be auto generated having CRUD operation methods.



```html
<!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-
  <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>
```

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



```
// This file can be replaced during build by using the `fileRe
// `ng build --prod` replaces `environment.ts` with `environme
// The list of file replacements can be found in `angular.json

export const environment = {
  production: false,
};

export const SERVER_URL = 'http://localhost:8081/';

/*
 * For easier debugging in development mode, you can import th
 * to ignore zone related error stack frames such as `zone.run
 *
 * This import should be commented out in production mode beca
 * on performance if an error is thrown.
 */
// import 'zone.js/dist/zone-error';  // Included with Angular
```

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

export const SERVER_URL = 'http://10.0.2.2:8081/';
```

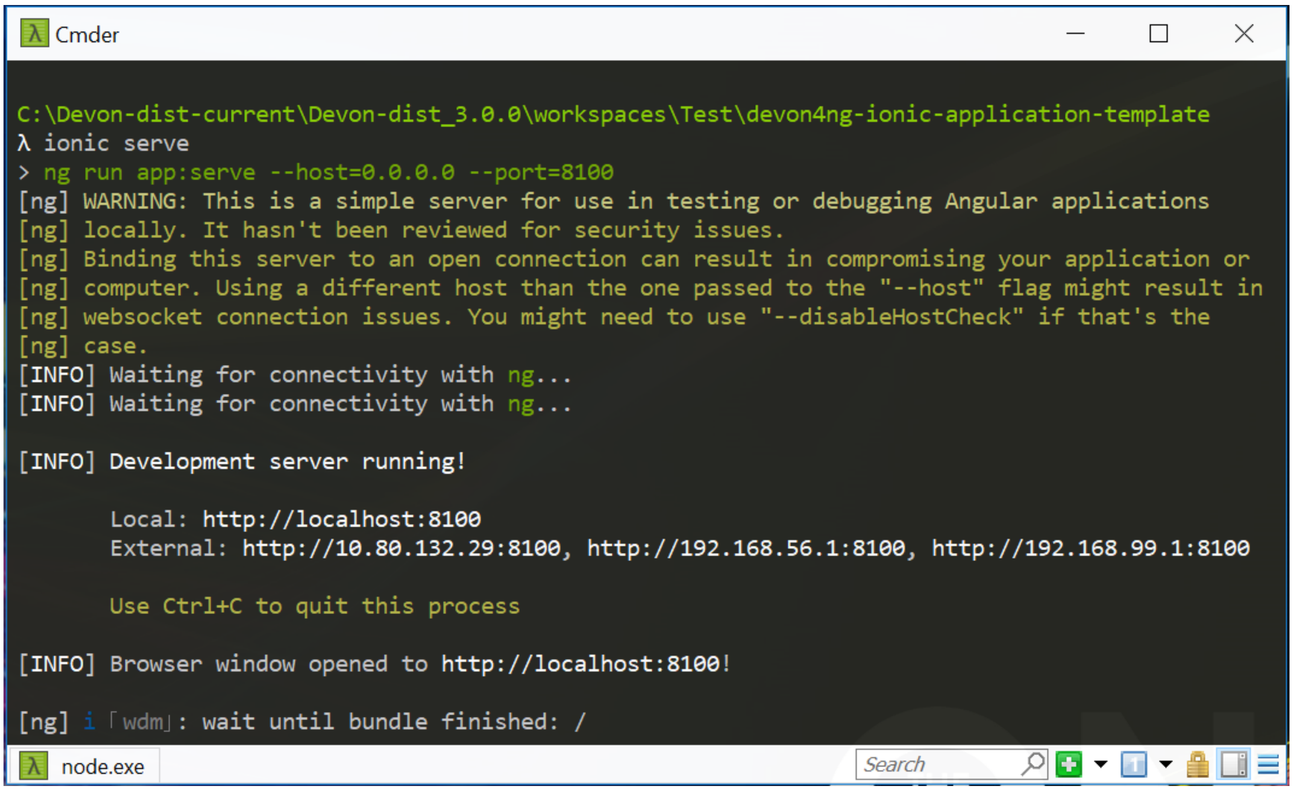
```
  "newProjectRoot": "projects",
  "projects": {
    "app": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "prefix": "app",
      "schematics": {},
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
     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

8. Now go to “workspaces\main\devon4ng-ionic-application-template” and right click, select “Open devonfw CMD shell here”. It will open a console.

9. If you have Windows10

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

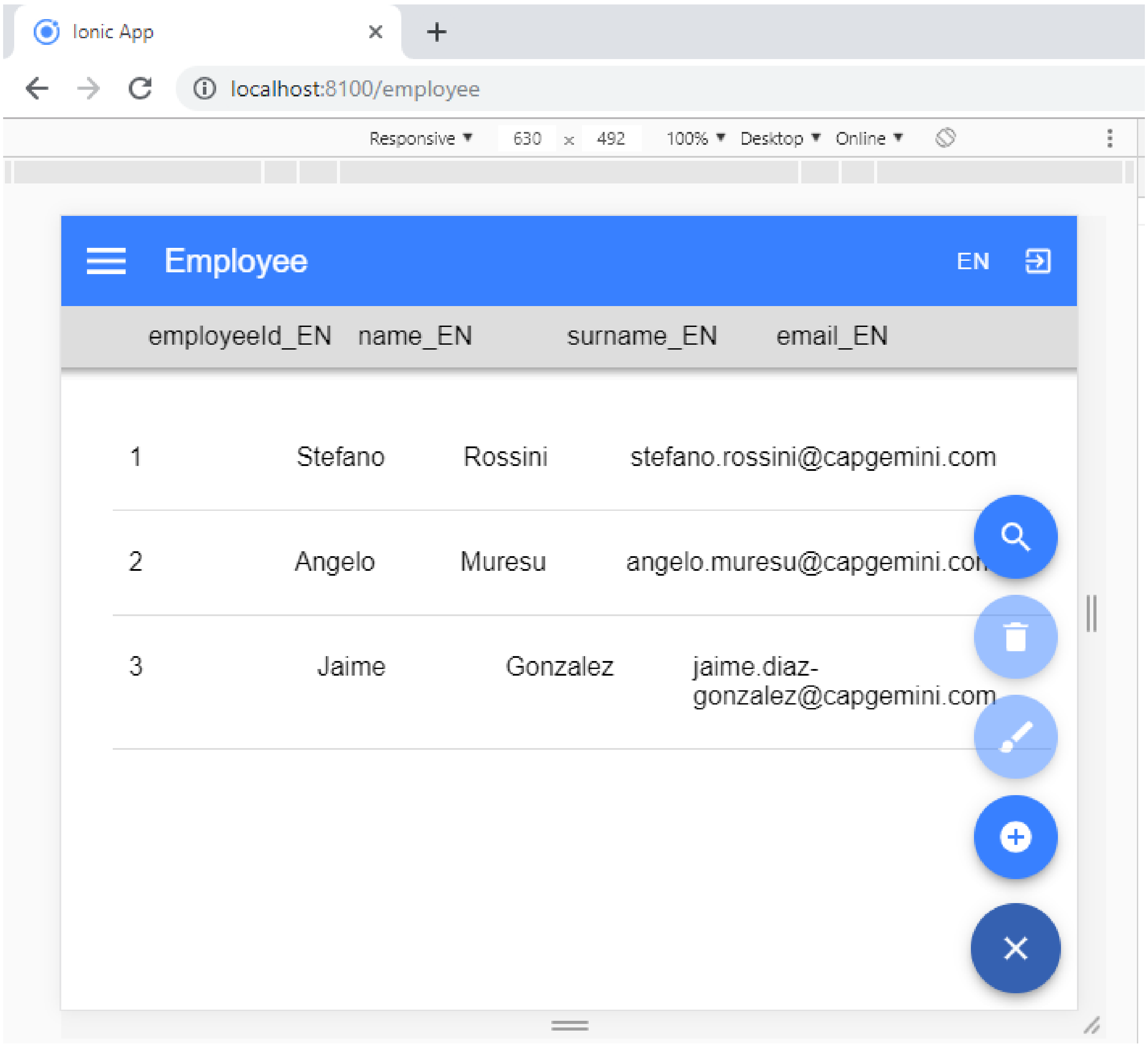
10. Run “**npm install**”.
11. Execute “**ionic serve**”.


Cmder

```
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: /
```

node.exe

12. Once the execution is successful

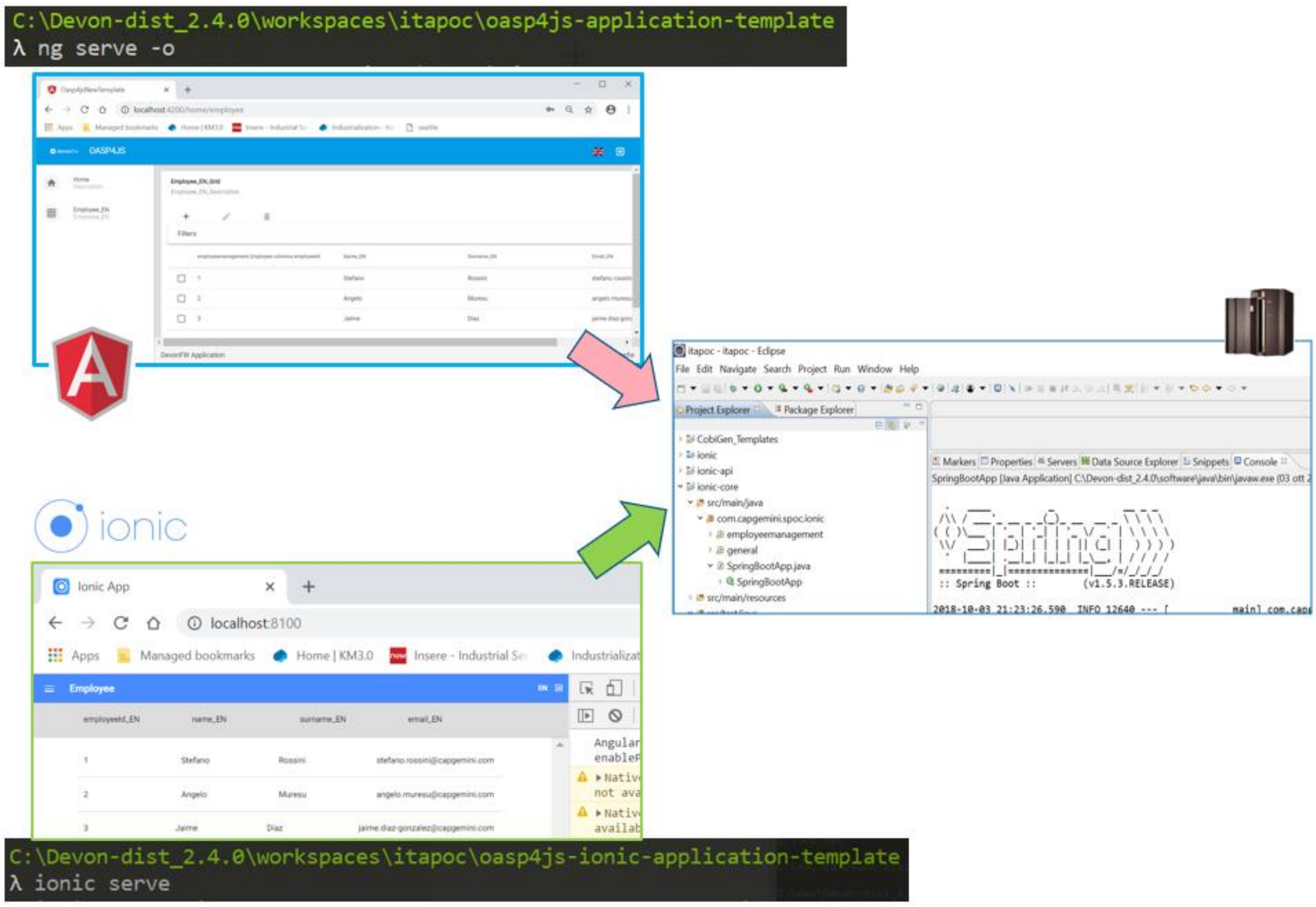


employeeId_EN	name_EN	surname_EN	email_EN
1	Stefano	Rossini	stefano.rossini@capgemini.com
2	Angelo	Muresu	angelo.muresu@capgemini.com
3	Jaime	Gonzalez	jaime.diaz-gonzalez@capgemini.com

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!



```
C:\Devon-dist_2.4.0\workspaces\itapoc\oasp4js-application-template
λ ng serve -o
```

```
:: Spring Boot ::        (v1.5.3.RELEASE)
```

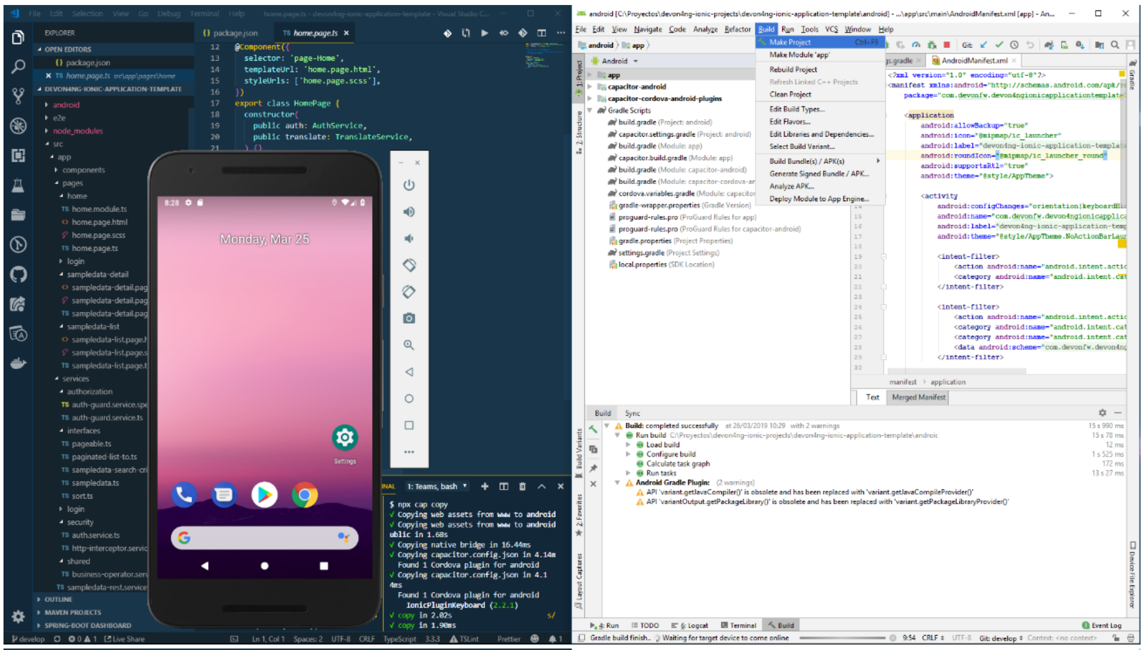
```
C:\Devon-dist_2.4.0\workspaces\itapoc\oasp4js-ionic-application-template
λ ionic serve
```

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.

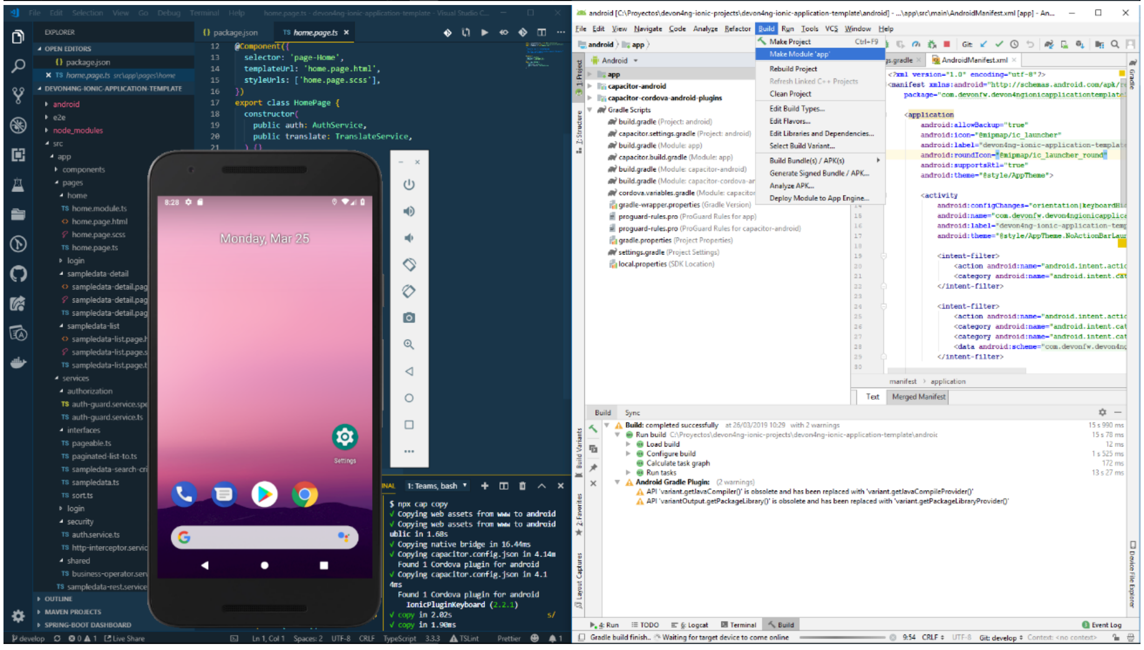


```
@Component({
  selector: 'page-Home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {
  constructor(
    public auth: AuthService,
    public translate: TranslateService,
  ) {}
```

```
$ npx cap copy
√ Copying web assets from www to android
√ Copying web assets from www to android
ublic in 1.68s
√ Copying native bridge in 16.44ms
√ Copying capacitor.config.json in 4.14m
  Found 1 Cordova plugin for android
√ Copying capacitor.config.json in 4.1
4ms
  Found 1 Cordova plugin for android
    IonicPluginKeyboard (2.2.1)
√ copy in 2.02s
√ copy in 1.90ms
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/
    package="com.devonfw.devon4ngionicapplicationtemplate

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="devon4ng-ionic-application-templa
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:configChanges="orientation|keyboardHi
            android:name="com.devonfw.devon4ngionicapplica
            android:label="devon4ng-ionic-application-temp
            android:theme="@style/AppTheme.NoActionBarLaun

            <intent-filter>
                <action android:name="android.intent.actic
                <category android:name="android.intent.cat
            </intent-filter>

            <intent-filter>
                <action android:name="android.intent.actic
                <category android:name="android.intent.cat
                <category android:name="android.intent.cat
                <data android:scheme="com.devonfw.devon4ng
            </intent-filter>
```

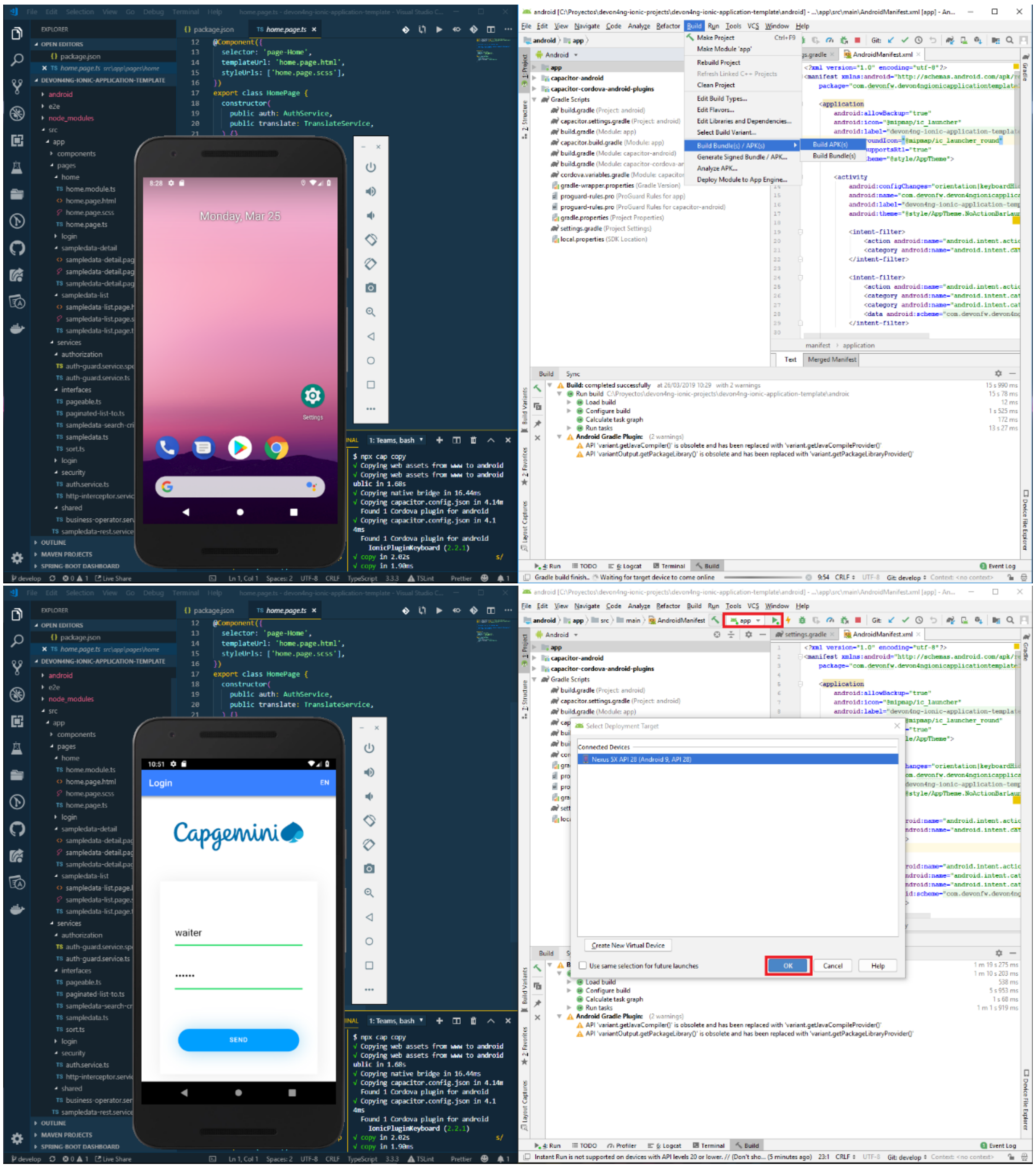
```
Build: completed successfully  at 26/03/2019 10:29  with 2 warnings      15 s 990 ms
  Run build C:\Proyectos\devon4ng-ionic-projects\devon4ng-ionic-application-template\androic   15 s 78 ms
    Load build                                                              12 ms
    Configure build                                                         1 s 525 ms
    Calculate task graph                                                    172 ms
    Run tasks                                                               13 s 27 ms
  Android Gradle Plugin:  (2 warnings)
    API 'variant.getJavaCompiler()' is obsolete and has been replaced with 'variant.getJavaCompileProvider()'
    API 'variantOutput.getPackageLibrary()' is obsolete and has been replaced with 'variant.getPackageLibraryProvider()'
```



```
@Component({
  selector: 'page-Home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {
  constructor(
    public auth: AuthService,
    public translate: TranslateService,
  ) {}
```

```
$ npx cap copy
√ Copying web assets from www to android
√ Copying web assets from www to android
ublic in 1.68s
√ Copying native bridge in 16.44ms
√ Copying capacitor.config.json in 4.14m
  Found 1 Cordova plugin for android
√ Copying capacitor.config.json in 4.1
4ms
  Found 1 Cordova plugin for android
    IonicPluginKeyboard (2.2.1)
√ copy in 2.02s
√ copy in 1.90ms
```

```
Build: completed successfully  at 26/03/2019 10:29  with 2 warnings      15 s 990 ms
  Run build C:\Proyectos\devon4ng-ionic-projects\devon4ng-ionic-application-template\androic   15 s 78 ms
    Load build                                                              12 ms
    Configure build                                                         1 s 525 ms
    Calculate task graph                                                    172 ms
    Run tasks                                                               13 s 27 ms
  Android Gradle Plugin:  (2 warnings)
    API 'variant.getJavaCompiler()' is obsolete and has been replaced with 'variant.getJavaCompileProvider()'
    API 'variantOutput.getPackageLibrary()' is obsolete and has been replaced with 'variant.getPackageLibraryProvider()'
```



```
@Component({
  selector: 'page-Home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {
  constructor(
    public auth: AuthService,
    public translate: TranslateService,
  ) {}
```

```
$ npx cap copy
√ Copying web assets from www to android
√ Copying web assets from www to android
ublic in 1.68s
√ Copying native bridge in 16.44ms
√ Copying capacitor.config.json in 4.14m
  Found 1 Cordova plugin for android
√ Copying capacitor.config.json in 4.1
4ms
  Found 1 Cordova plugin for android
    IonicPluginKeyboard (2.2.1)
√ copy in 2.02s
√ copy in 1.90ms
```

```
Build: completed successfully   at 26/03/2019 10:29   with 2 warnings     15 s 990 ms
  Run build C:\Proyectos\devon4ng-ionic-projects\devon4ng-ionic-application-template\android   15 s 78 ms
    Load build                                                           12 ms
    Configure build                                                      1 s 525 ms
    Calculate task graph                                                 172 ms
    Run tasks                                                            13 s 27 ms
  Android Gradle Plugin:   (2 warnings)
    API 'variant.getJavaCompiler()' is obsolete and has been replaced with 'variant.getJavaCompileProvider()'
    API 'variantOutput.getPackageLibrary()' is obsolete and has been replaced with 'variant.getPackageLibraryProvider()'
```

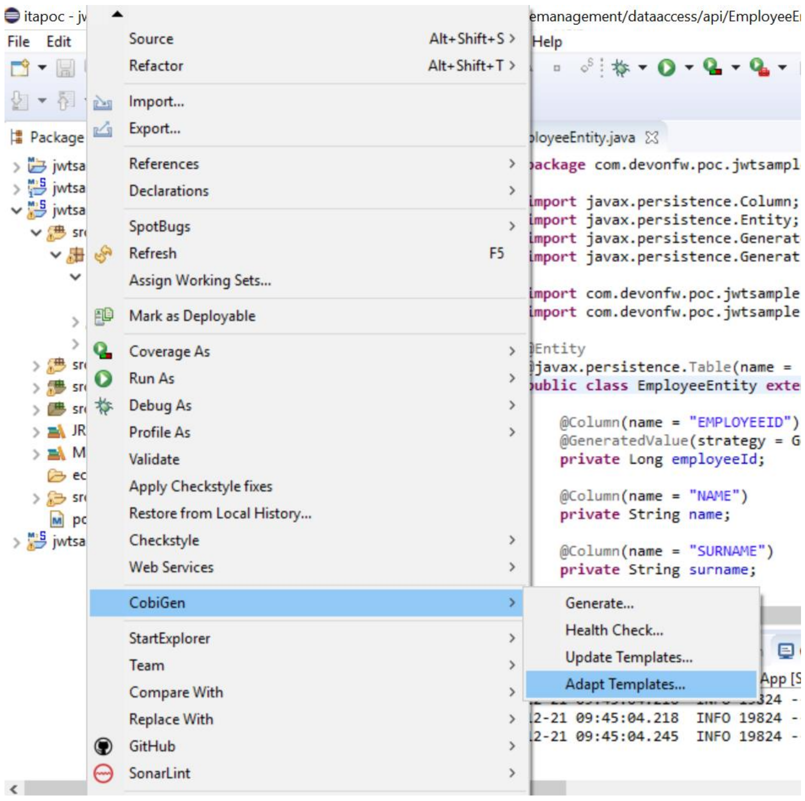
You can find your apk file in

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

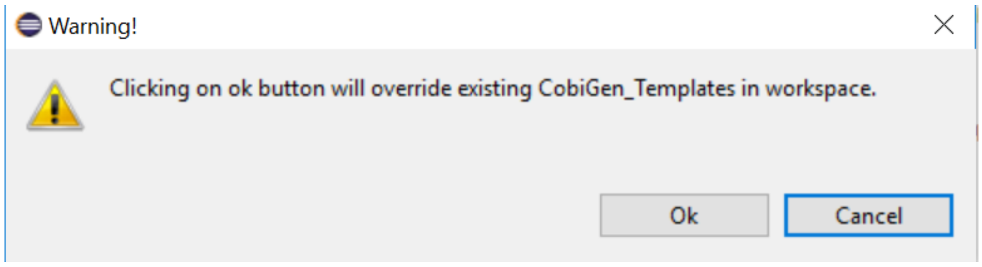
3 Adapt CobiGen_Templates:

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

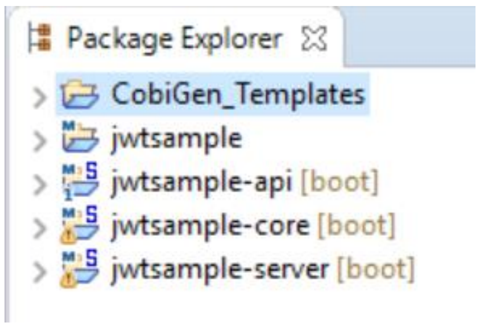


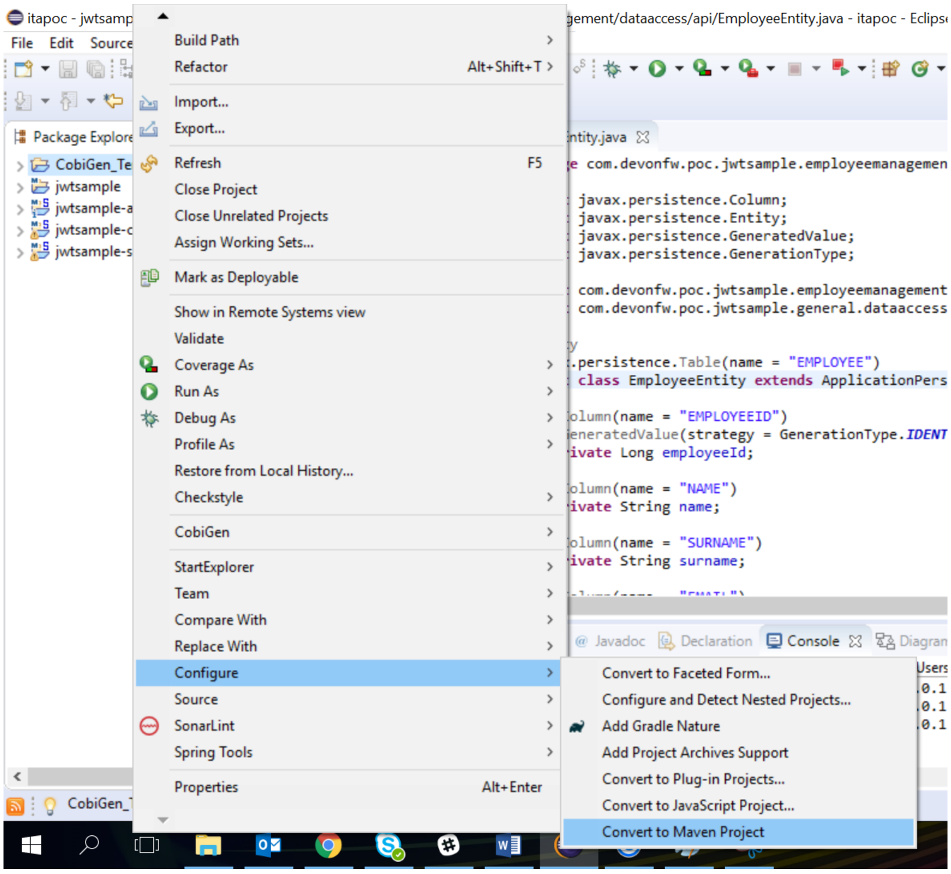
Click *Ok*:


Warning!

Clicking on ok button will override existing CobiGen_Templates in workspace.

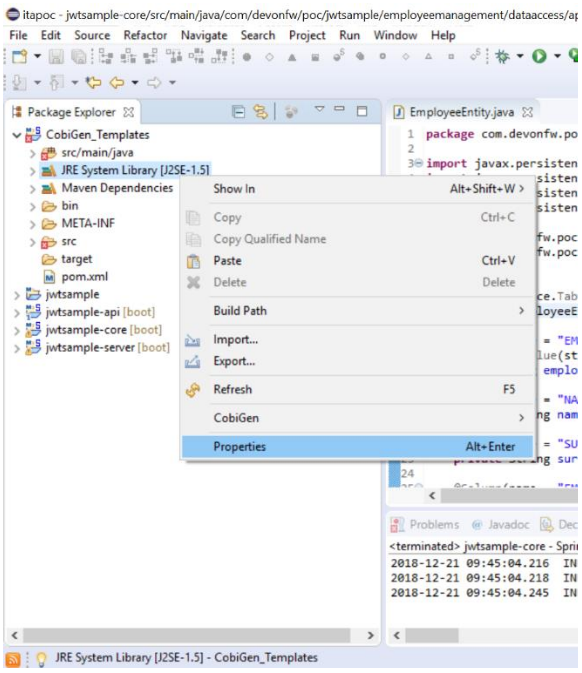
Now the CobiGen_Templates project will be automatically imported into your workspace, as shown on the image below:





```
e com.devonfw.poc.jwtsample.employeemanagemen

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

 com.devonfw.poc.jwtsample.employeemanagement
 com.devonfw.poc.jwtsample.general.dataaccess

.persistence.Table(name = "EMPLOYEE")
 class EmployeeEntity extends ApplicationPers

olumn(name = "EMPLOYEEID")
eneratedValue(strategy = GenerationType.IDENT
ivate Long employeeId;

olumn(name = "NAME")
ivate String name;

olumn(name = "SURNAME")
ivate String surname;
```

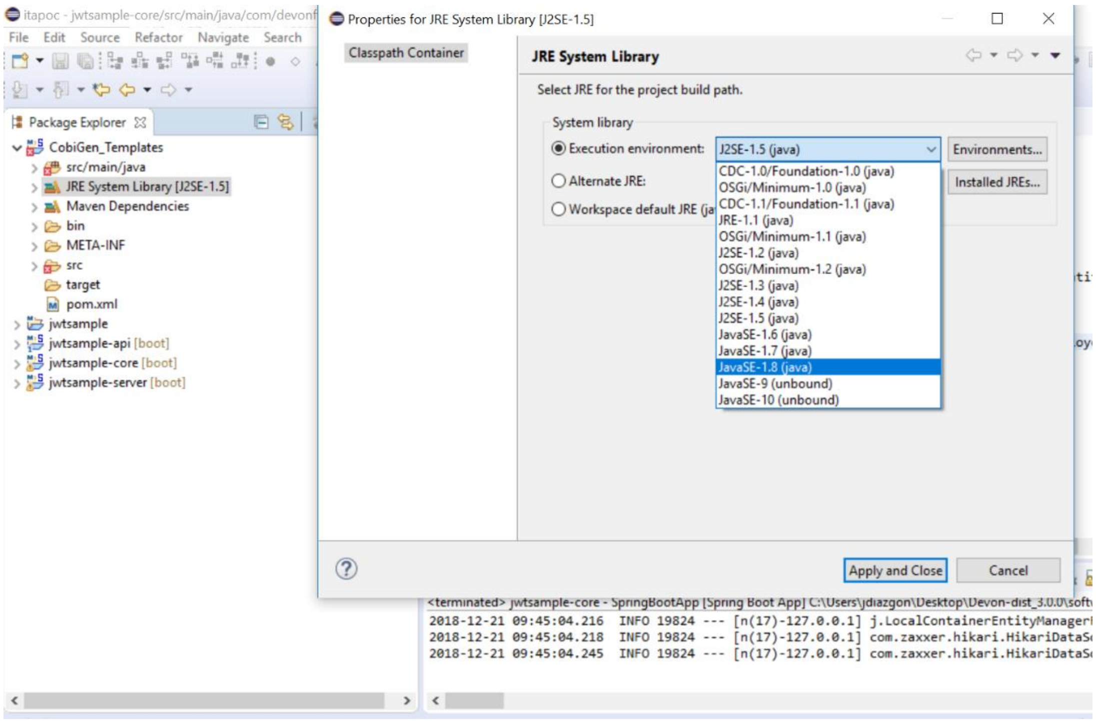
Now you just need to change the Java version of the project to JRE 1.8. Right click on the JRE system library, and then on *Properties:*



```
package com.devonfw.po

import javax.persisten
```

```
<terminated> jwtsample-core - Spri
2018-12-21 09:45:04.216  IN
2018-12-21 09:45:04.218  IN
2018-12-21 09:45:04.245  IN
```

Now change the version to Java 1.8



Properties for JRE System Library [J2SE-1.5]

Classpath Container

JRE System Library

Select JRE for the project build path.

System library

(●) Execution environment: J2SE-1.5 (java)

( ) Alternate JRE:

( ) Workspace default JRE

CDC-1.0/Foundation-1.0 (java)
OSGi/Minimum-1.0 (java)
CDC-1.1/Foundation-1.1 (java)
JRE-1.1 (java)
OSGi/Minimum-1.1 (java)
J2SE-1.2 (java)
OSGi/Minimum-1.2 (java)
J2SE-1.3 (java)
J2SE-1.4 (java)
J2SE-1.5 (java)
JavaSE-1.6 (java)
JavaSE-1.7 (java)
[✓ JavaSE-1.8 (java)]
JavaSE-9 (unbound)
JavaSE-10 (unbound)

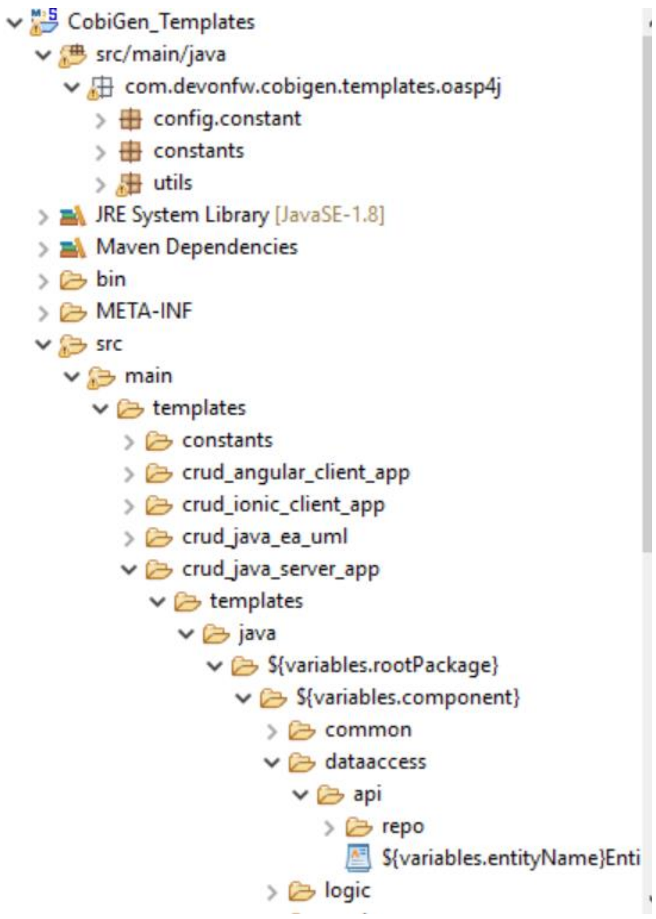
Package Explorer

- CobiGen_Templates
  - src/main/java
  - JRE System Library [J2SE-1.5]
  - Maven Dependencies
  - bin
  - META-INF
  - src
  - target
  - pom.xml
- jwtsample
- jwtsample-api [boot]
- jwtsample-core [boot]
- jwtsample-server [boot]

```
<terminated> jwtsample-core - SpringBootApp [Spring Boot App] C:\Users\jdiazgon\Desktop\Devon-dist_3.0.0\soft
2018-12-21 09:45:04.216  INFO 19824 --- [n(17)-127.0.0.1] j.LocalContainerEntityManager
2018-12-21 09:45:04.218  INFO 19824 --- [n(17)-127.0.0.1] com.zaxxer.hikari.HikariDataS
2018-12-21 09:45:04.245  INFO 19824 --- [n(17)-127.0.0.1] com.zaxxer.hikari.HikariDataS
```

itapoc - jwtsample-core/src/main/java/com/devonf
File Edit Source Refactor Navigate Search
Environments...
Installed JREs...
Apply and Close
Cancel

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:



- CobiGen_Templates
  - src/main/java
    - com.devonfw.cobigen.templates.oasp4j
      - config.constant
      - constants
      - utils
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - bin
  - META-INF
  - src
    - main
      - templates
        - constants
        - crud_angular_client_app
        - crud_ionic_client_app
        - crud_java_ea_uml
        - crud_java_server_app
          - templates
            - java
              - ${variables.rootPackage}
                - ${variables.component}
                  - common
                  - dataaccess
                    - api
                      - repo
                      - ${variables.entityName}Enti
                  - logic

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