Creating OneToMany/ManyToOne (6th class)

Presented by ARMS (THAILAND) Co., Ltd.

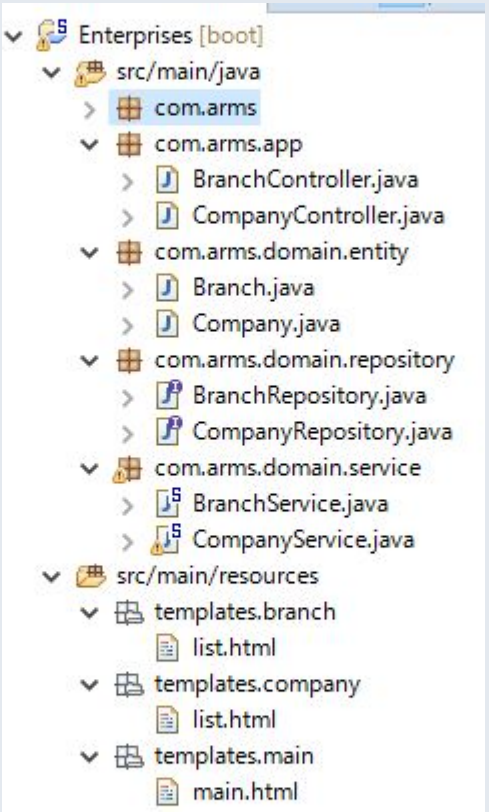
Index

1. Start creating a project
   - 1-1. Start Project
   - 1-2. Create packages
2. Create OneToMany/ManyToOne
   - 2-1. Create company list
   - 2-2. Create branch list
   - 2-3. Create OneToMany/ManyToOne
   - 2-4. H2 in-memory database

1. Start creating a project

1-1. Start Project

- We will create a project as below



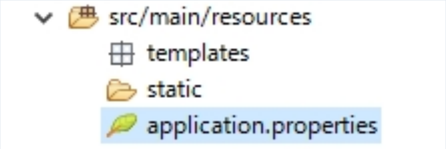
```
Enterprises [boot]
  src/main/java
    com.arms
    com.arms.app
      BranchController.java
      CompanyController.java
    com.arms.domain.entity
      Branch.java
      Company.java
    com.arms.domain.repository
      BranchRepository.java
      CompanyRepository.java
    com.arms.domain.service
      BranchService.java
      CompanyService.java
  src/main/resources
    templates.branch
      list.html
    templates.company
      list.html
    templates.main
      main.html
```

See the left figure and create your project.

1. Start creating a project

1-1. Start Project

- Prepare to use thymeleaf and H2 in-memory database



src/main/resources
templates
static
application.properties

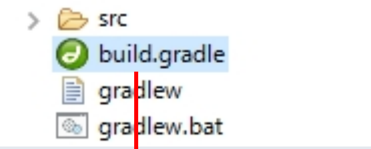
Open “application.properties”, and add the following lines

```
spring.thymeleaf.mode=LEGACYHTML5
spring.thymeleaf.cache=false
spring.thymeleaf.encoding=UTF-8

spring.h2.console.enabled=true
spring.h2.console.path=/console
```

1. Start creating a project

1-1. Start Project

- Modify or add dependencies in build.gradle



```
> src
  build.gradle
  gradlew
  gradlew.bat
```

```
dependencies {
        compile('org.springframework.boot:spring-boot-starter-data-jpa')
        compile('org.springframework.boot:spring-boot-starter-thymeleaf')
        compile('org.projectlombok:lombok:1.16.6')
        compile('org.springframework.boot:spring-boot-starter-web')
        runtime('com.h2database:h2')
        compile('net.sourceforge.nekohtml:nekohtml')
        testCompile('org.springframework.boot:spring-boot-starter-test')
}
```

2. Create OneToMany/ManyToOne

2-1. Create comapny list

Create Company.java and modify this POJO to make it work for Company list database.

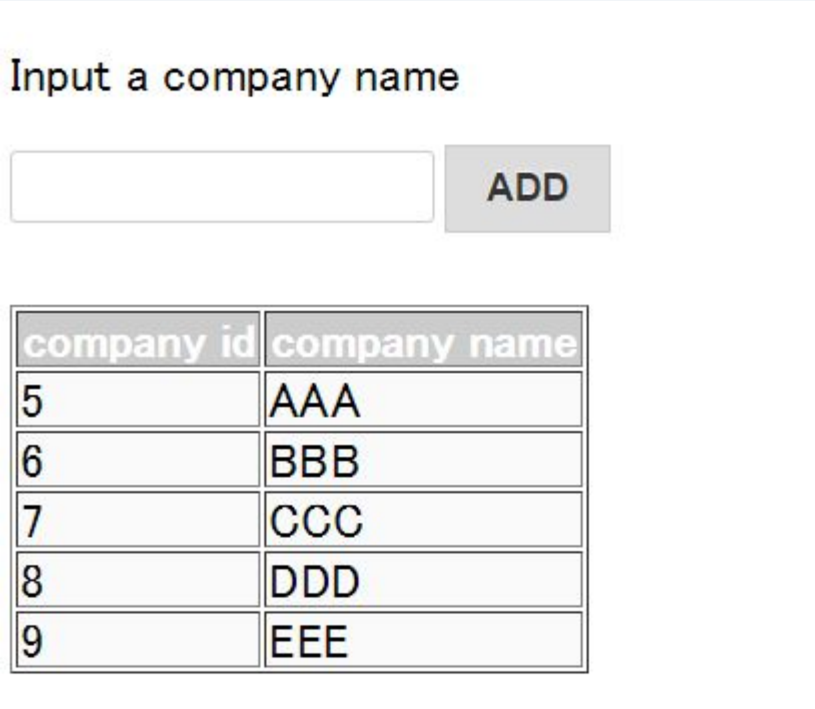
```
package com.arms.domain.entity;

public class Company {

                    private int id;

                    private String name;

}
```

After that, create
Repository interface,
Service,
Controller,
And company/list.html

2. Create OneToMany/ManyToOne

2-2. Create company list

You will have the page as below



Input a company name

company id	company name
5	AAA
6	BBB
7	CCC
8	DDD
9	EEE

2. Create OneToMany/ManyToOne

2-2. Create branch list

Create Branch.java and modify this POJO to make it work for banch list database.

```
package com.arms.domain.entity;

public class Branch {

            private int branchId;

            private String branchName;

}
```

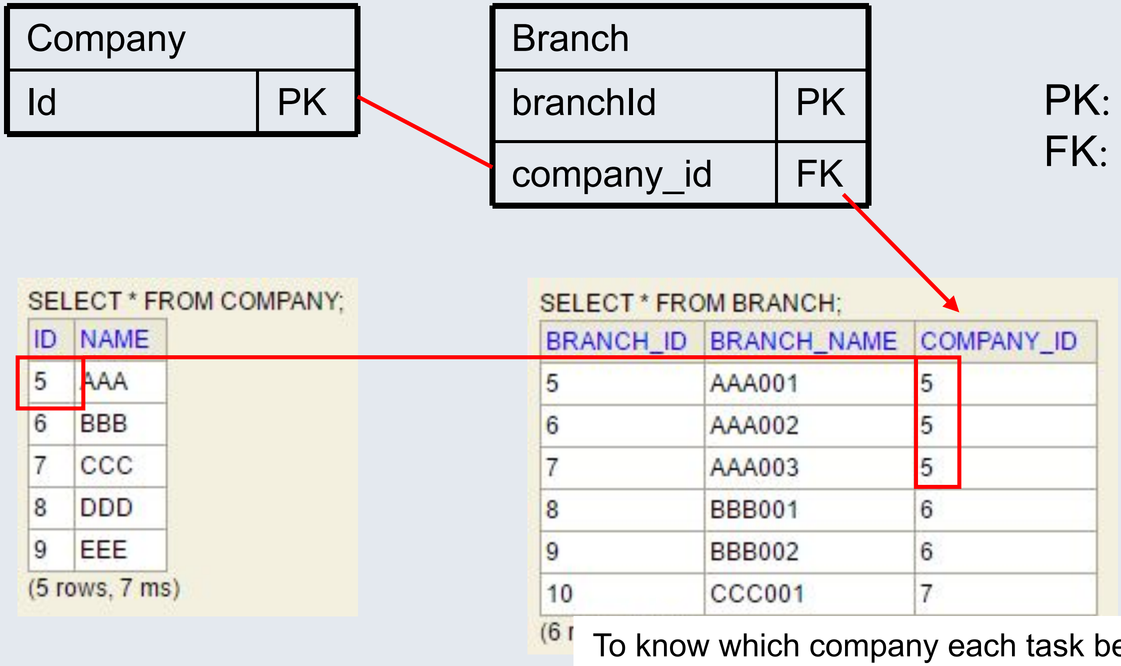
After that, create
Repository interface,
Service,
Controller,
And branch/list.html

2. Create OneToMany/ManyToOne

2-3. Create OneToMany/ManyToOne

Create OneToMany/ManyToOne relationship

One company can have many branches. Many branches can belong to one company.



Company	
Id	PK

Branch	
branchId	PK
company_id	FK

PK: primary key
FK: foreign key

SELECT * FROM COMPANY;

ID	NAME
5	AAA
6	BBB
7	CCC
8	DDD
9	EEE

(5 rows, 7 ms)

SELECT * FROM BRANCH;

BRANCH_ID	BRANCH_NAME	COMPANY_ID
5	AAA001	5
6	AAA002	5
7	AAA003	5
8	BBB001	6
9	BBB002	6
10	CCC001	7

(6 r

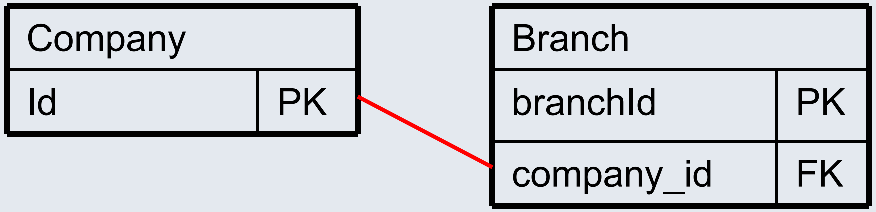
To know which company each task belongs to, this adds COMPANY_ID as a foreign key!!!

2. Create OneToMany/ManyToOne

2-3. Create OneToMany/ManyToOne

Create OneToMany/ManyToOne relationship

1. Non-owner decides the owner by “mappedBy”.
2. mappedBy designates non-owner property or field name which the owner refers to.
3. In OneToMany/ManyToOne relationship, mappedBy can not be used in ManyToOne side.



Company	
Id	PK

Branch	
branchId	PK
company_id	FK

PK: primary key
FK: foreign key

ARMS

Your Idea Leads Your Ideals

homepage: http://arms-asia.com/
facebook: https://www.facebook.com/arms.asia?fref=ts