Creating student list (3rd class)

Presented by ARMS（THAILAND）Co., Ltd.

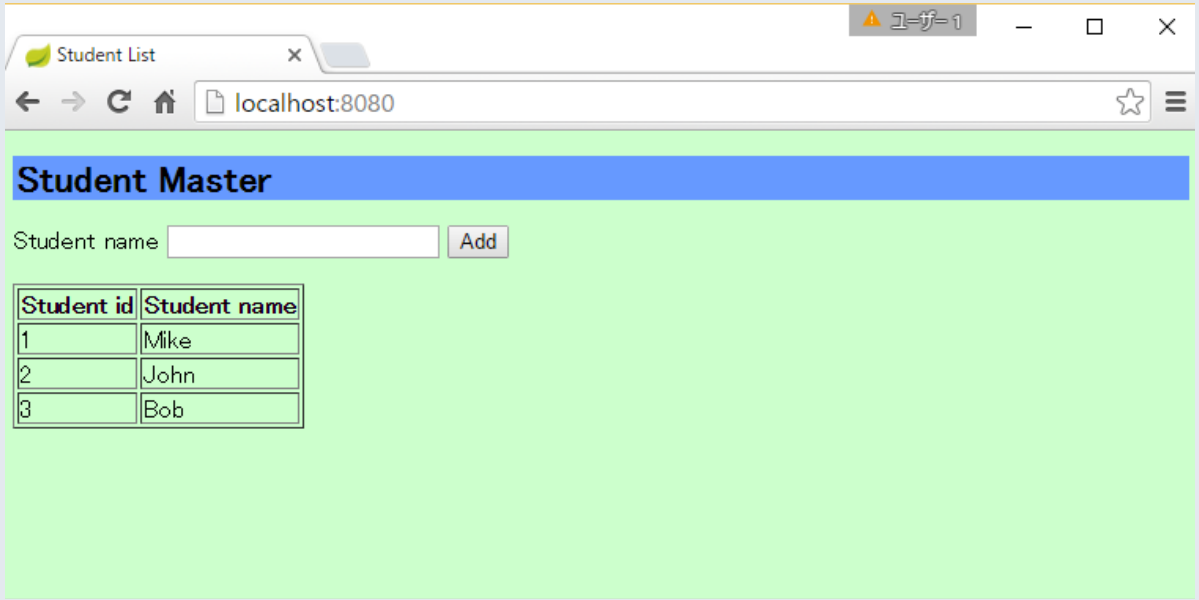
Index

1. Start creating a project
    1-1. Start Project
    1-2. Create packages
2. Create student list
    2-1. Create packages
    2-2. Create student list
    2-3. Challenge
    2-4. H2 in-memory database

1. Start creating a project

1-1. Start Project

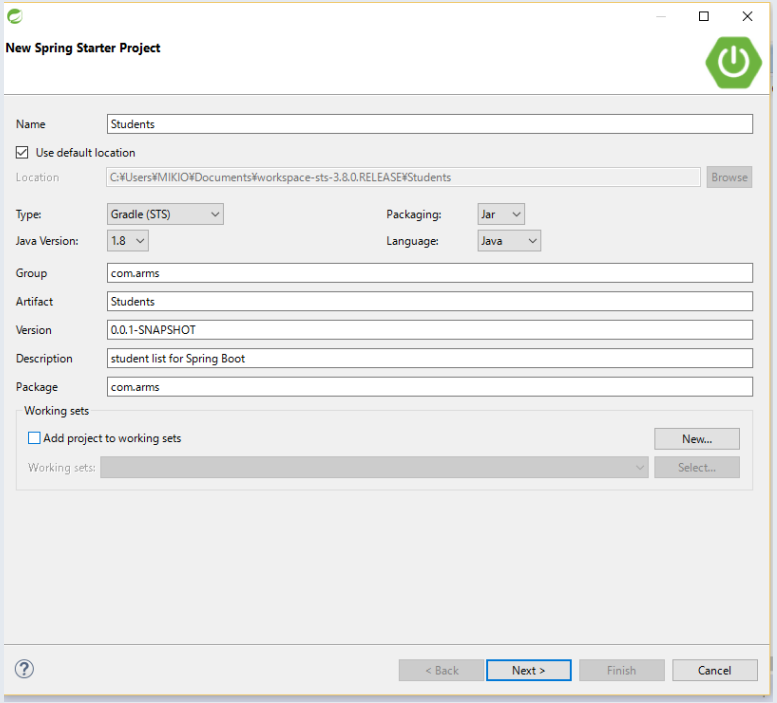
- You will create Student list Master like below



1. Start creating a project

1-1. Start Project

- Open STS and File – New – Spring Starter Project



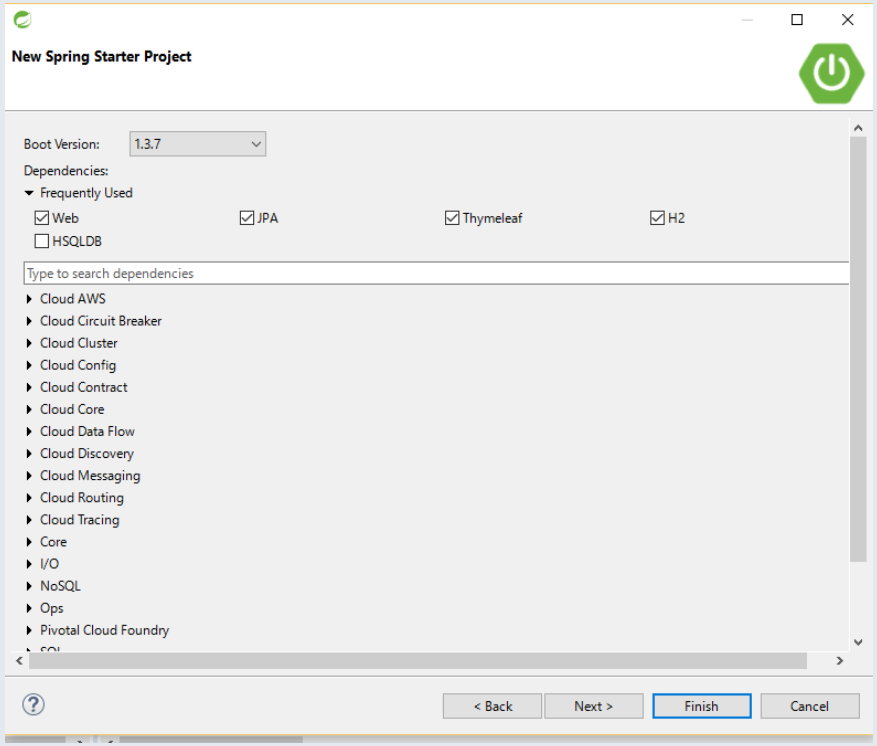
Name: Students
Type: Gradle(STS)
Group: com.arms
Artifact: Students
Package: com.arms

Press “Next>” button

1. Start creating a project

1-1. Start Project

- Open STS and File – New – Spring Starter Project



Check the following checkbox, **JPA, H2** under SQL

Thymeleaf under Template Engines.

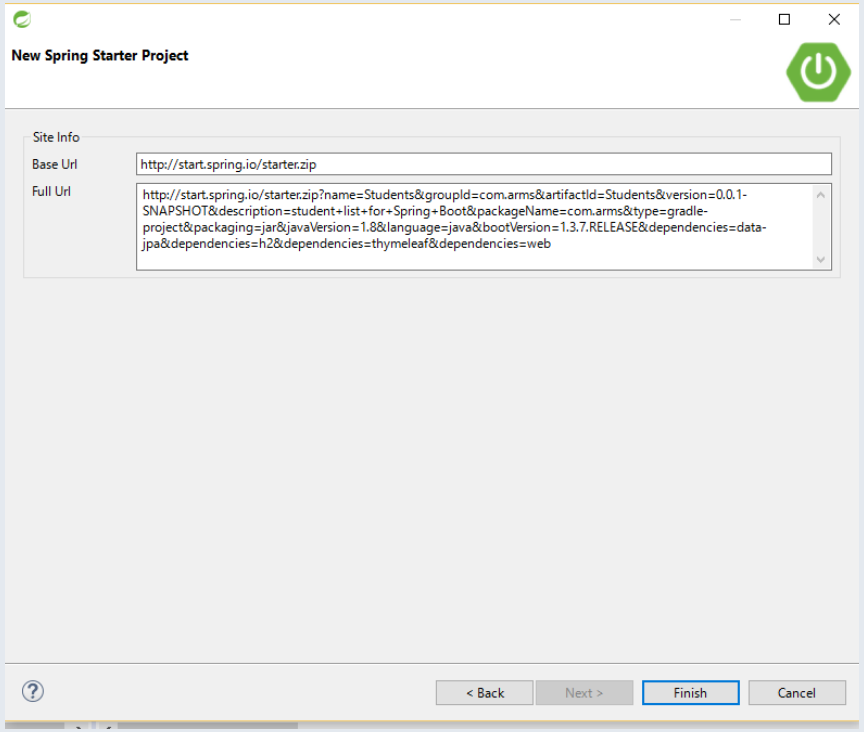
Web under Web

Press “Next>” button

1. Start creating a project

1-1. Start Project

- Open STS and File – New – Spring Starter Project

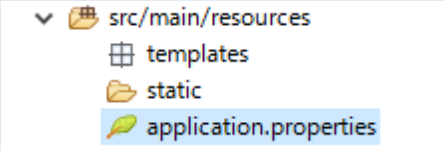


Press “Finish” button

1. Start creating a project

1-1. Start Project

- Prepare to use thymeleaf and H2 in-memory database



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
```

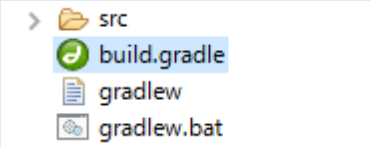
<u>spring.h2.console.enabled=true</u>
This enables h2 console

<u>spring.h2.console.path=/console</u>
This tells which URL to access H2 console
localhost:8080/console

1. Start creating a project

1-1. Start Project

- Prepare to use thymeleaf



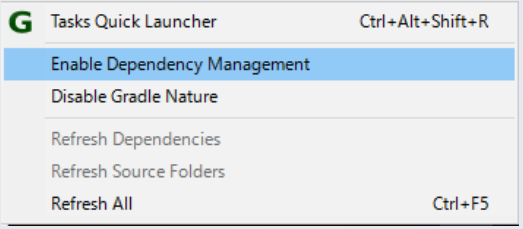
Open “build.gradle”
Add lombok and nekohtml into dependencies

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

1. Start creating a project

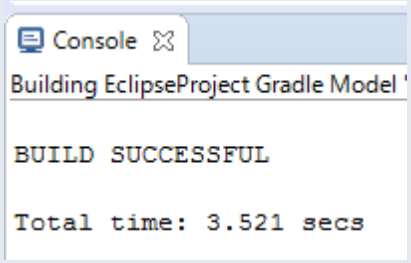
1-1. Start Project

- Right click on your project, select Gradle(STS) – Enable Dependency Management from the context menu.



This will enable “Refresh Dependencies”

Select “Refresh Dependencies” to apply changes in the previous page.



You will see “BUILD SUCCESSFUL” message in the Console screen.

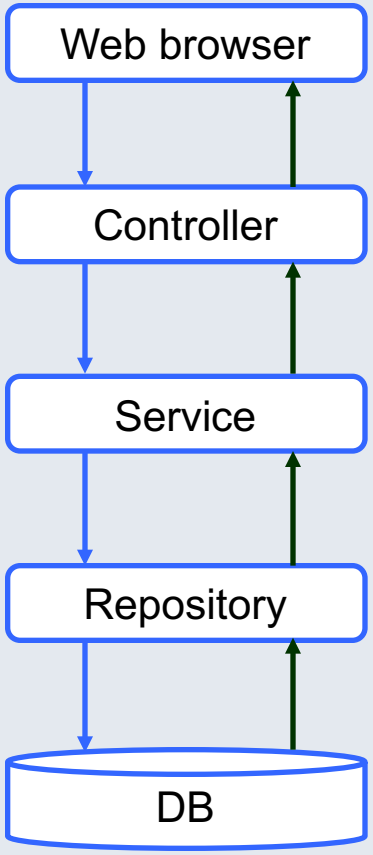
Libraries go to under Gradle Dependencies management

Currently, Eclipse doesn’t automatically update if build.gradle is updated.

2. Create student list

2-1. Create packages

- In this class, we will create this app by the following order.



student/list.html

StudentController.java

StudentService.java

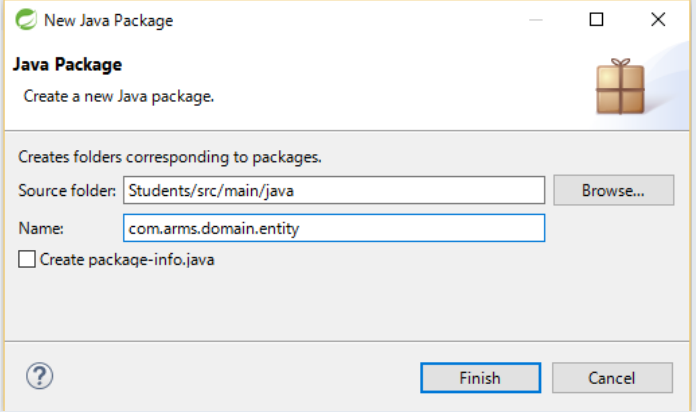
StudentRepository.java

Student.java

2. Create student list

2-1. Create packages

- Right click on “com.arms”, New - Package



Type “com.arms.domain.entity” in Name field

In the same way, add packages as below,

src/main/java

com.arms.domain.repository

com.arms.domain.service

com.arms.app.student

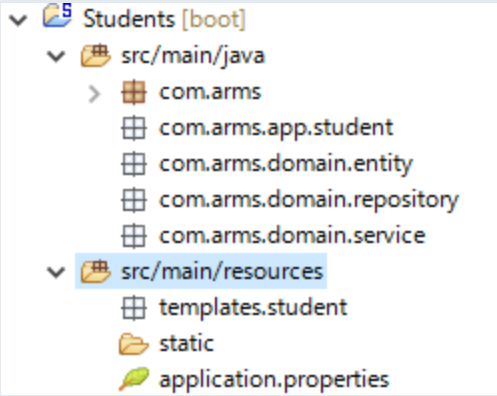
src/main/resources

templates.student

2. Create student list

2-1. Create packages

- Package structure will be like below,



Create

entity class, repository interface, service component, and controller.

And list.html

2. Create student list

2-2. Create student list

- Create Student.java

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

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

import lombok.Data;

@Entity
@Data
public class Student {

		@Id @GeneratedValue
		private int id;

		private String name;

}
```

2. Create student list

2-2. Create student list

- Create StudentRepository interface with JpaRepository extended.

```
package com.arms.domain.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.arms.domain.entity.Student;

@Repository
public interface StudentRepository extends JpaRepository<Student, Integer> {

}
```

2. Create student list

2-2. Create student list

- Create StudentService.java

```
package com.arms.domain.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.arms.domain.entity.Student;
import com.arms.domain.repository.StudentRepository;

@Service
public class StudentService {

		@Autowired
		StudentRepository studentRepository;

		public List<Student> findAllStudent() {
				return studentRepository.findAll();
		}

		public void create(String name){
				Student student = new Student();
				student.setName(name);
				studentRepository.save(student);
		}

}
```

2. Create student list

2-2. Create student list

- Create StudentController.java

```
package com.arms.app.student;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import com.arms.domain.service.StudentService;

@Controller
public class StudentController {

		@Autowired
		StudentService studentService;

		@RequestMapping(value = "/", method = RequestMethod.GET)
		public ModelAndView index(ModelAndView mav){
				mav.addObject("list", studentService.findAllStudent());
				mav.setViewName("student/list");
				return mav;
		}
```

Continue on next page

2. Create student list

2-2. Create student list

- Create StudentController.java

```
@RequestMapping(value = "/", method = RequestMethod.POST)
public String createStudent(@RequestParam("name") String name){
        studentService.create(name);
        return "redirect:/";
}

}
```

2. Create student list

2-2. Create student list

- Create list.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link href="/css/style.css" th:href="@{/css/style.css}" rel="stylesheet" />
        <title>Student List</title>
</head>

<body>
        <h1>Student Master</h1>
        <form th:action="@{/}" method="POST">
                <label for="name">Student name</label>
                <input th:text="${name}" id="name" type="text" name="name">
                <button type="submit">Add</button>
        </form>

        <br />
```

Continue on next page

2. Create student list

2-2. Create student list

- Create list.html

```
                <table border="1">
                <tr>
                        <th>Student id</th>
                        <th>Student name</th>
                </tr>
                <tr th:each="student: ${list}" th:object=${student}>
                        <td th:text="*{id}">Student id</td>
                        <td th:text="*{name}">Student name</td>
                </tr>
                </table>
</body>

</html>
```

2. Create student list

2-2. Create student list

- Create style.css and design by yourself

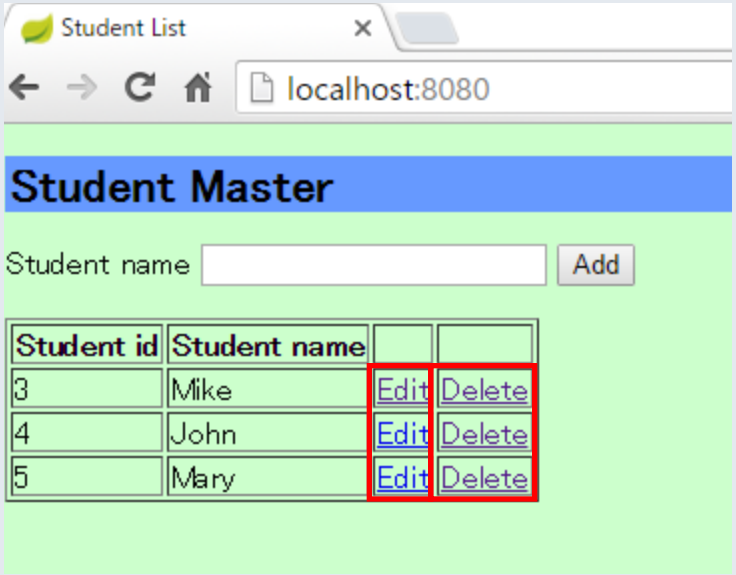
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link href="/css/style.css" th:href="@{/css/style.css}" rel="stylesheet" />
        <title>Student List</title>
</head>
```

Make sure to put it under static folder.

2. Create student list

2-3. Challenge

- Create “edit” button
- Create “delete” button



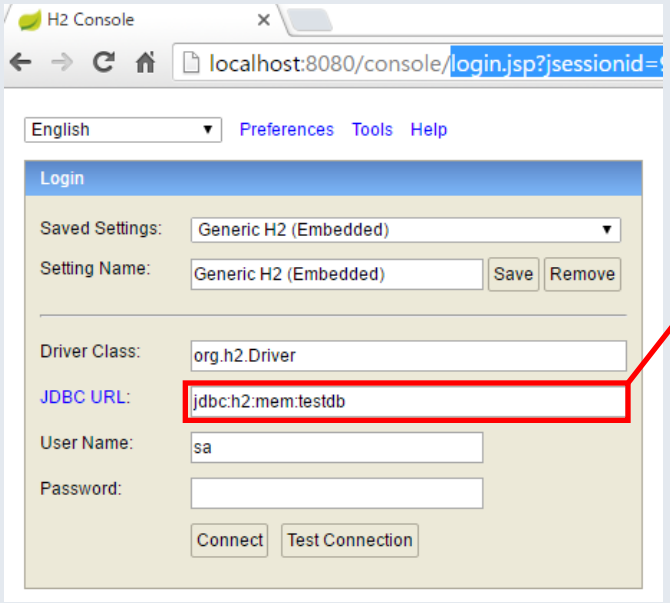
2. Create student list

2-4. H2 in-memory database

- Let's access H2 in-memory database since we configured it. Access localhost:8080/console

application.properties

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



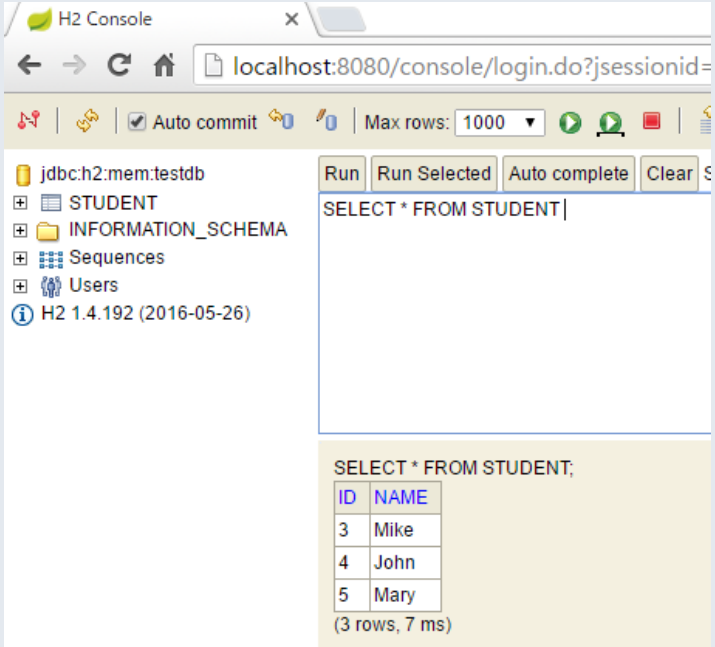
Change JDBC URL:
jdbc:h2:mem:testdb

And press "Connect" button

2. Create student list

2-4. H2 in-memory database

- You can run SQL command to see your student list database



SELECT * FROM STUDENT

Your Idea Leads Your Ideals

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