Understanding Spring boot concept

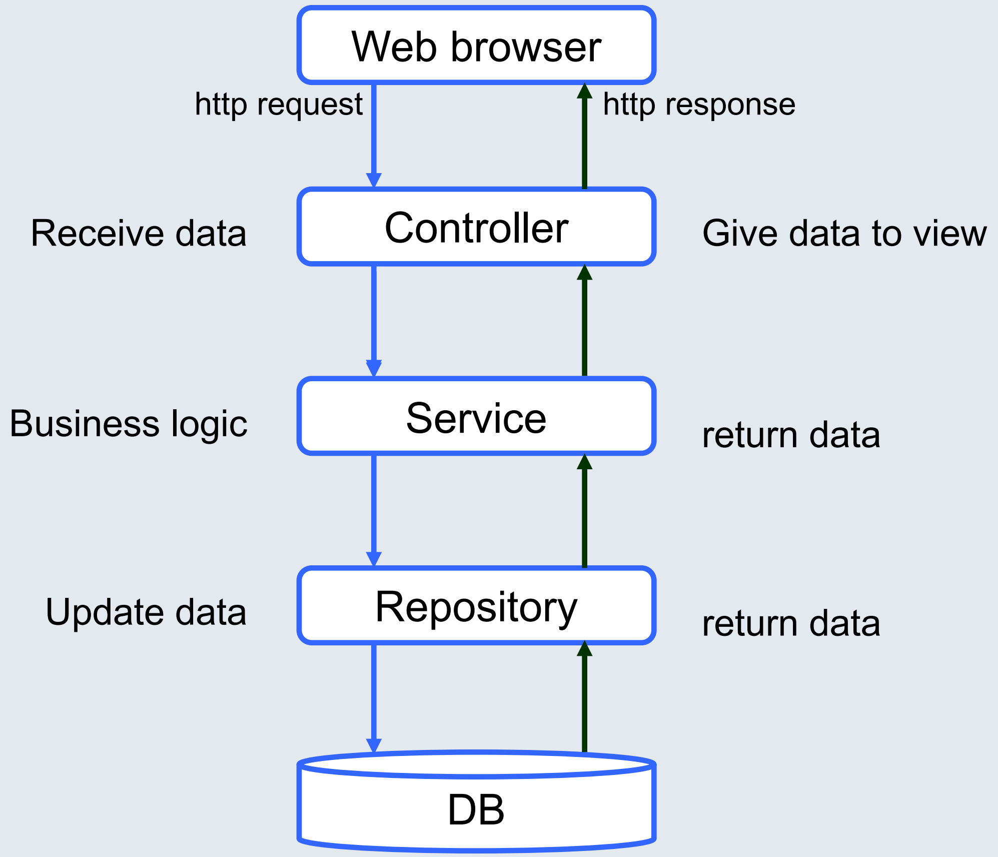
Presented by ARMS（THAILAND）Co., Ltd.

Index

1. Hierarchical(Layered) Structure

1-1. Spring boot structure

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



1. Hierarchical(Layered) Structure

1-1. Spring boot structure

• Controller – Service – Repository architecture: common design pattern in application development.

Controller: Control screen transition, call Service

Service: Provide business logic

Repository: Access database

1. Hierarchical(Layered) Structure

1-1. Spring boot structure

- Spring MVC

Model:Application Data, Database operation

View: Input from users, output to users, for example, thymeleaf part is Veiw

Controller is a bridge between Model and View. Controller gets data from Model and give it to View. Controller receives user input from View, and give it to Model

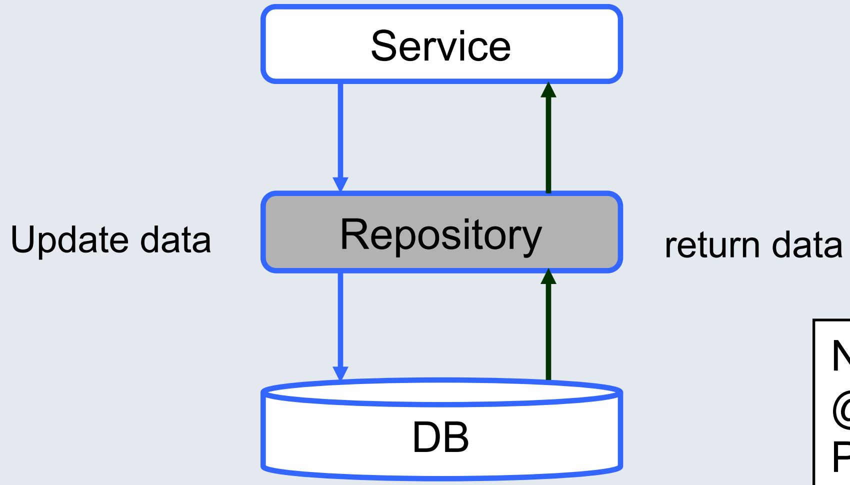
Spring MVC is created based on MVC concept, but so many libraries and complicate setting make it difficult to use……

Spring boot is easy to use this Spring MVC
Spring boot is easy to start a project

1. Hierarchical(Layered) Structure

1-2. Repository

- Receive request from Service, and update DB and get data from DB.



Note:
@Repository annotation
Persistence layer components (Data Access Layer) which can be used to get data from DB. Mainly, DB related operations are handled by Repository

1. Hierarchical(Layered) Structure

1-2. Repository

- Annotate Repository interface with @Repository and create entity class with @Entity (make fields DB properties).

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

……………………..
@Repository
public interface ProjectRepository extends JpaRepository<Project, Integer> {

}
```

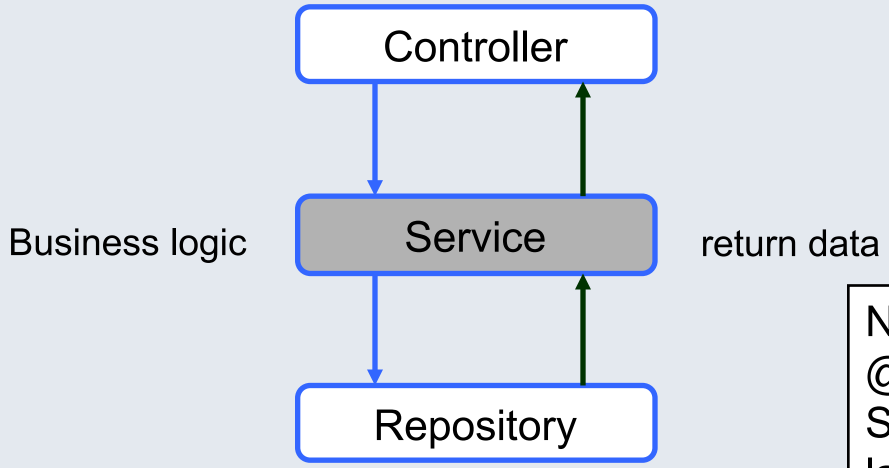
```
package com.arms.domain.entity;
……………………………………………..
@Entity
@Table(name= "project")
public class Project {
    @GeneratedValue @Id
    private int id;

    @NotEmpty
    private String name;
```

1. Hierarchical(Layered) Structure

1-3. Service

- Receive request from Controller, and update data to Repository and get data from Repository.



Note:
@Service annotation
Service components (in business layer) which can be used to check and compare values from Controller, request Repository to update data.
Mainly business logic is here.

1. Hierarchical(Layered) Structure

1-3. Service

- Annotate with @Service, and call methods in Repository.

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

…………………………………………………..
@Service
@Transactional
public class ProjectService {

    @Autowired
    ProjectRepository projectRepository;


    public List<Project> findAllProject() {
        return projectRepository.findAll();
    }
  …………………………………………………..
```

@Service annotation

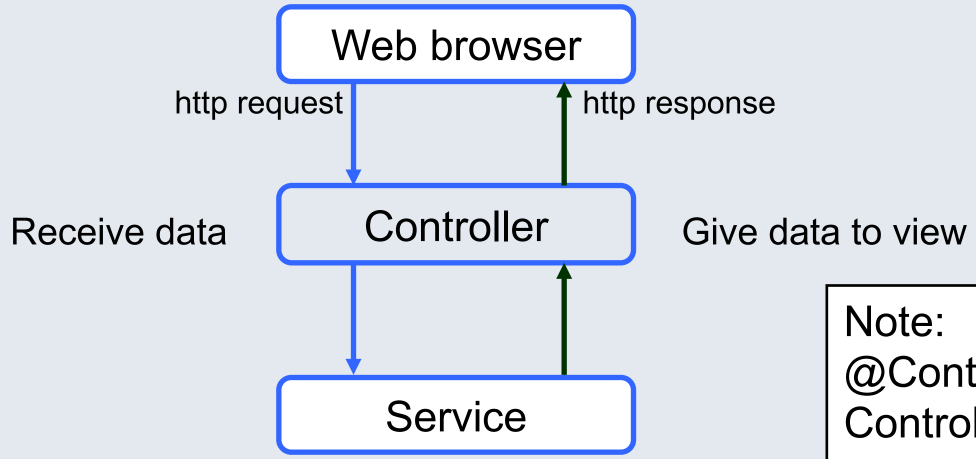
Reference to Repository and autowires to it, so you can access without instantiation.(no need to use new operator)

Call methods in Repository and return requested values.

1. Hierarchical(Layered) Structure

1-4. Controller

- Receive values from browser, and request Service to deal with the values, and then the result from Service is given to the view and renders html.



Note:
@Controller annotation
Controller components (in presentation layer) which can be used to map requests to the View and forward and call Service.

1. Hierarchical(Layered) Structure

1-4. Controller

- Annotate with @Controller, and map request to html

```
package com.arms.app.project;

………………………………………………………….
@Controller
public class ProjectController {

    @Autowired
    ProjectService projectService;

    @RequestMapping(value = “/", method = RequestMethod.GET)
    public String index(Model model) {
        List<Project> projectList = projectService.findAllProject();
        model.addAttribute("projectList", projectList);
        return "project/list";
    }
```

@Controller annotation

Autowire and call methods in Service

Accessing “/” by GET method is mapped to project/list.html

Model (interface)class is a bridge between Controller and View, result of projectList is stored in Model and give it to view.

1. Hierarchical(Layered) Structure

1-5. View Template

- Declare namespace to use Thymeleaf template engine.

```
……………………………………………….
@Controller
public class ProjectController {

   @RequestMapping(value = "/", method = RequestMethod.GET)
   public String index(Model model) {
        List<Project> projectList = projectService.findAllProject();
        model.addAttribute("projectList", projectList);
        return "project/list";
   }
```

project/list.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:th="http://www.thymeleaf.org">
<head>
</head>
<body>
div class="col-lg-4" style="padding: 0;" th:each="project, stat : ${projectList}">
<span th:text="${project.name}">project0001</span>
…………………………………………
</body>
</html>
```

ARMS

Your Idea Leads Your Ideals

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