Spring Boot

Anna Pilot

- **What is Annotation?**

provide developer the metadata (information) to the program elements

- **Why we use SpringBoot?**

build a web application for backend fast and easily

- **How to start a SpringBoot Project?**

Show how to start a hello world project

- **localhost IP address**

is same with ip 127.0.0.1

- **What is Maven?**

in pom.xml

- **What is Dependency?**

- **Build Spring Boot Project / Migration**

- **What is Restful API?**

It’s an interface that two computer systems use to exchange information securely over the internet.

RESTful API is an interface that two computer systems use to exchange information securely over the internet. Most business applications have to communicate with other internal and third-party applications to perform various tasks. For example, CRUD operations can be made by calling APIs.

- **What is CRUD operation?**

Create, read, update, delete.

Create a Demo project, including folders:

- Entity (Models - setter and getter)
- Repository (use a map to mock Data Storage – manipulate multiple **Dao**)
- Service (business logic - call repository method to manipulate data)
- Controller (Setting API communicated interface)

@RequestMapping

@GetMapping

@PathVariable

@RequestParam

Homework 6

- Create your first SpringBoot project
- Migrate your SpringMVC code into SpringBoot and boot it up

- **How to define the path/url**
- **How to use Postman Tool**
- @RequestBody
- @ResponseBody
- **Model Defines for APIs**
- **Two entity has an association example**

Homework 7

Complete your SpringBoot project with two entity’s association.

- **IOC (Inversion of Control)**

framework will create it for you when you need it.

Then no need to initialize one by developer, like new item.

(Give the one IPad example)

- **What is Container**

(Graph and understanding)

- **What is Configuration?**

- **Bean**

for the constructor method in configuration class

@Bean annotation which is applied on a method to specify that it returns a bean to be managed by Spring IoC Container

- **DI (Dependency Injection)** - is responsible to create objects

Dependency injection is a pattern we can use to implement IoC, where the control being inverted is setting an object's dependencies.

Connecting objects with other objects, or “injecting” objects into other objects, is done by an assembler rather than by the objects themselves.

- **3 methods of DI** - on field, on setter, on constructor

- **@Autowired**

is used for automatic dependency injection. Spring framework is built on dependency injection and we inject the class dependencies through spring bean configuration file.

- **@Component**

is an annotation that allows Spring to automatically detect our custom beans.

Spring will scan our application for classes annotated with @Component. Instantiate them and inject any specified dependencies into them.

- **@Controller** - indicates that a particular class serves the role of a controller.
- **@Service** - indicates that a particular class serves the role of a controller.
- **@Repository** - indicates that a particular class serves the role of a controller.

- **@Qualifier**(value = “ipad1”)

and how to identify two instances with similar name (Give 2 IPad in a Room example)

- **Java Reflection**

allows Java program to manipulate internal fields of the program. And it's possible for java to obtain the names of all its fields members and display them.

ex: give a customized class, and initialize and call

```
Class c = cat.getClass();
```

and from c, it can get all internal fields name from cat Class

and it also can change values through the Class

That's how internal Spring works to get/ set value for objects