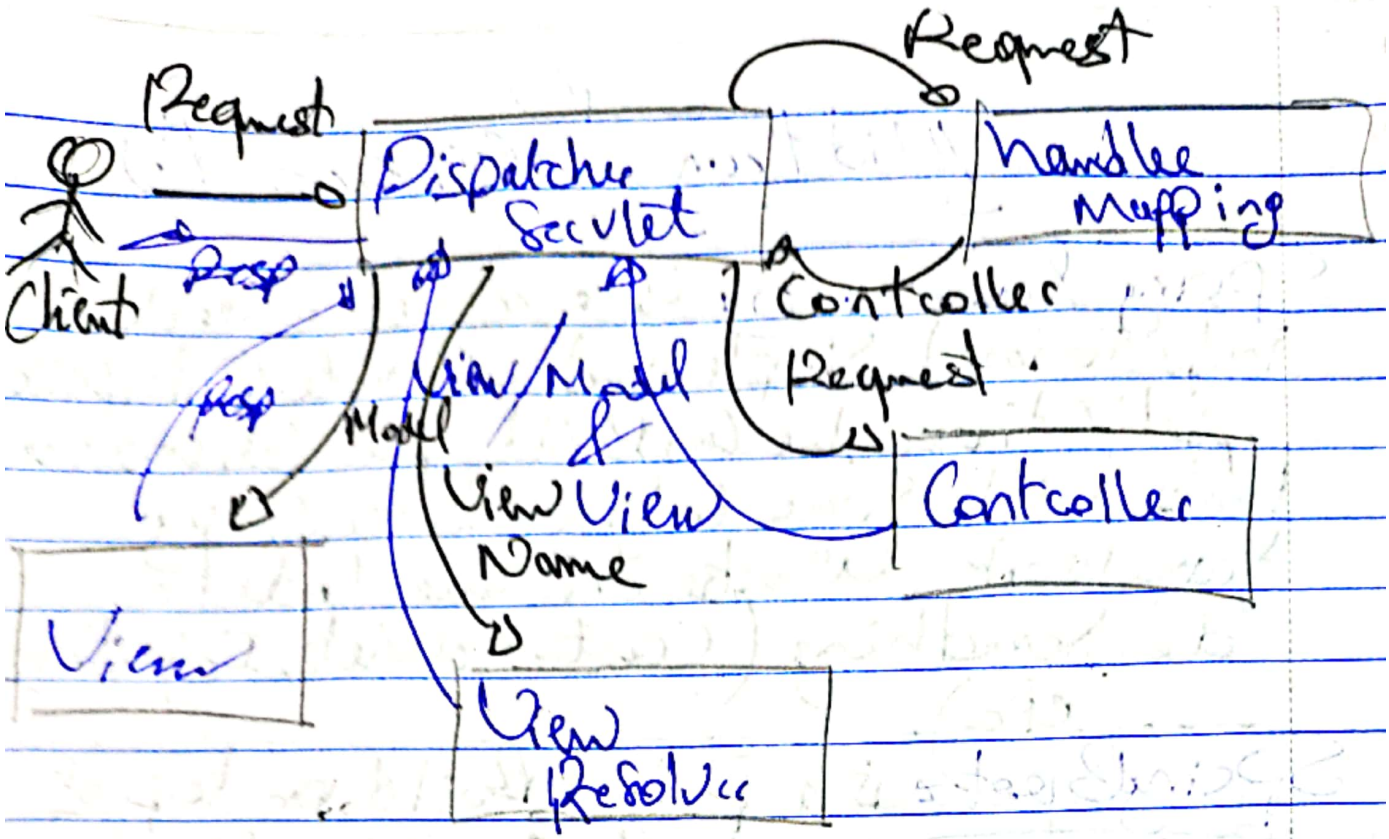
Midterm Review 1:

Spring framework: it's a framework helps to build simple and scalable enterprise applications, he can build applications from POJOs with some configuration. Spring framework works on our code to do something (Create Servlet, Injections ... etc).

SpringBoot is a Project Build on top of Spring framework it provides simple and faster way to setup, config, and run web-based Applications.

Spring MVC: Model that specifically designed to implement the model View Controller design pattern.

Dispatcher Servlet is a Single Central Servlet that dispatches requests to controllers integrated with Spring Container, has webApplicationContext inherits all the beans defined in the rootApplicationContext.



Inversion of control : (I.o.C)
it's a principe that doing things with a group of patterns (DI, factory, Strategy ---), Ioc container is the one who manages the beans (objects, Inject
Ioc is implemented using Dependency Injection

Dependancy Injection : it's all about doing less code and have more flexibility to accomplish more.

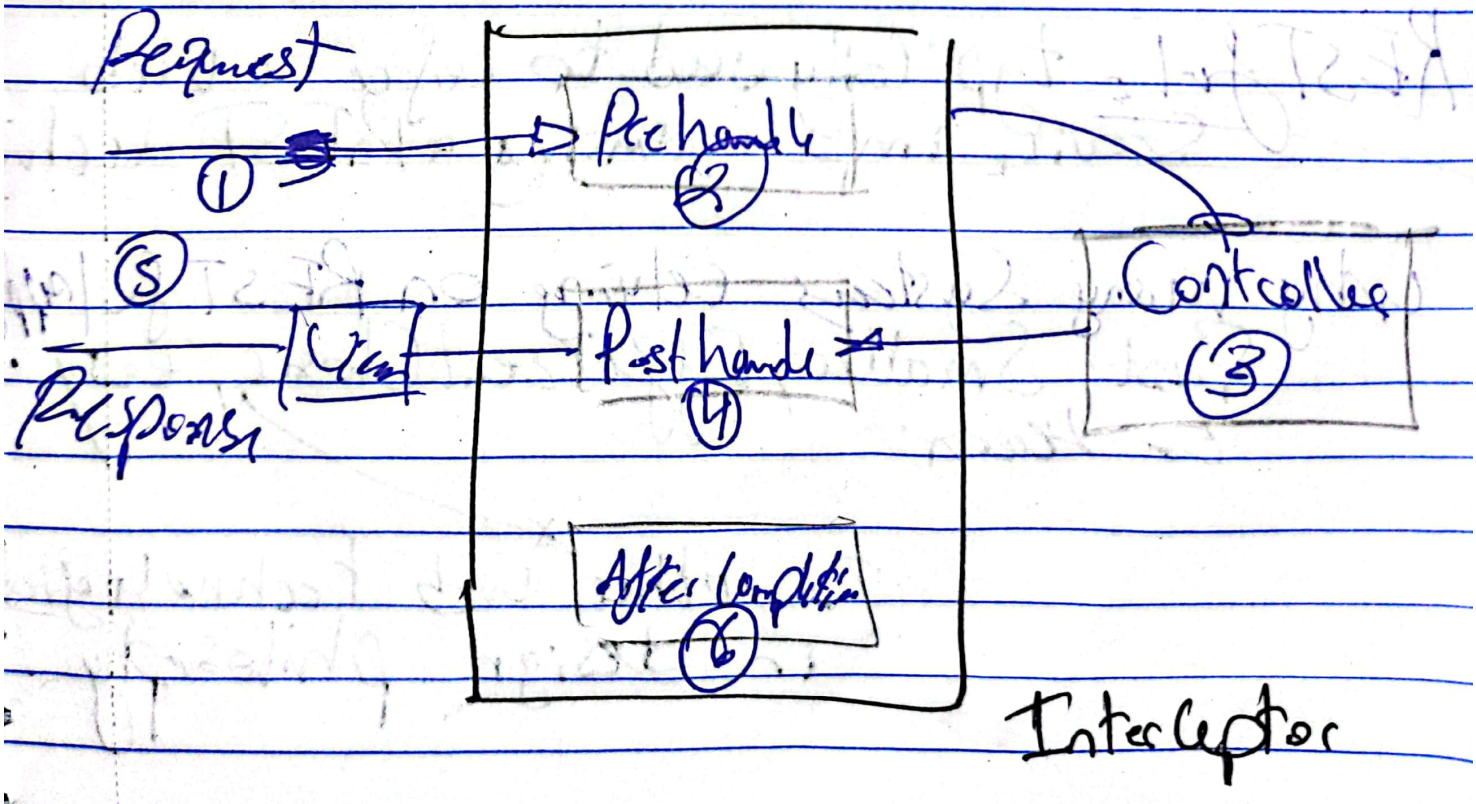
Types : Setter Injection
Constructor Injection
Autowiring.

Autowired: is to wire Beans together. Make an object of the autowired object, instantiated and the Injected

Types: XML: Name, type, Constructor
Annotations: Name, type, Constructor

Interceptors: is like a filter, Can be used for Cross-Cutting Concerns and to avoid repetitive handler code like Logging, changing globally used parameters in Spring model etc. It has three 3 Methods:

Pre handle: Before going to the Ctrl
Post handle: After going to the Ctrl
After completion: After View was generated



Indempotent: When we make a multiple Request has the same effects as making a single Request, then that Rest API is called Indempotent.

REST Representational State Transfer is a new Communication type between different languages, is to take data and sent it to another object.
REST use HTTP Operations.

3 Acchitectural Constraints:
Uniform Interface: Individual resources are Identified in requests.
Client-Server: Separation of Concerns
Cacheable: clients can cach responses.

RESTful: typically used to refer to web servile implementing a REST Acchi

Why many Systems relying on RESTful API?
Fast, Smaller, Efficient Closer, easy to learn

To other web technologies in design philosophy.

@RequestParam & @PathVariable

Extract data found in query parameters / data passed in the URL

States in JPA ORM:

Transient: Created but has no DB id

Managed: Created and has DB id

Detached: has a db id, but not managed by the current persistence context

Removed: data removed from db.

Fetch:

Eager: Return data right now

Lazy: when first called, he gives just a proxy of the object.

What does the CrudRepository?

CrudRepository is an interface extends the Spring data Repository Interface it provides generic Crud functions like findAll, findById, Save, deleteById

Validation

Steps to Implement Bean Validation

1. - define a default error message
2. - Create a Constraint annotation
3. - Implement a Validator.

Need for Custom Validation Annotation

To make sure all user input is correct and make sure he doesn't gives wrong information.

Types (Annotations)

@Past

@NotNull / @NotBlank / @NotEmpty

@DateTimeFormat(pattern = "mm-dd-yyyy

@Size(min = 2, max = 5, message = "{0} field ... {1} alphab

@Valid → check all the validation in Class Address

private Address address;

@Email

@Digits(integer = 3, fraction = 0, msg = "...")

Query:

```
@Query("Select p from Post p where p.author = :author")
List<Post> findByAuthor(@Param("author") String author);

@Query("Select u.id, u.name from User u where u.id = :id")
public String getIDAndName(@Param("id") long id)

@Query("Select u.posts from User u where u.id = :id")
List<Posts> findPostsForUser(@Param("id") String id);
```

Controller

```
@RestController
@RequestMapping("/Posts")
public class PostsController {
    @Autowired
    PostsService postsService;
    // getting all the posts :
    @GetMapping
    @ResponseBody
    public List<Post> getPosts() {
        return postsService.getAll();
    }
    // getOnePost
    @GetMapping("/{id}")
    public Post getPost(@PathVariable("id")
        long id) {
        return postsService.getPost(id);
    }
    // update
    @PutMapping("/{id}")
    public void updatePost(@PathVariable("id")
        @RequestBody Post p) {
        postsService.updatePost(id, p);
    }
    // add Post
    @PostMapping
    public Post addPost(@RequestBody Post p)
    {
        postsService.addPost(p);
        return p;
    }
}
```

@Entity

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Entity
public class User {
    @Id
    long id;
    String name;
    @OneToMany
    @JoinTable(name = "user_posts")
    List<Post> posts
}

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Entity
public class Post {
    @Id
    long id;
    String title;
    String author;
    String content;
}