

# Web services (Spring)



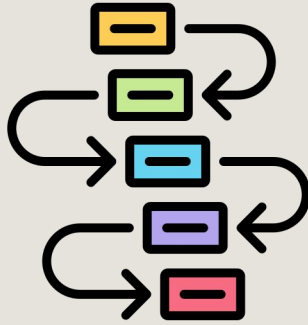
Web Development with Java



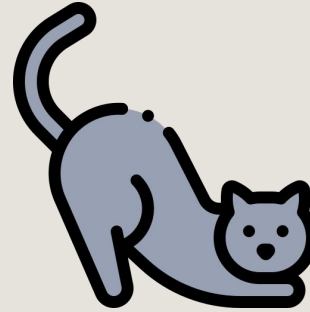
# Recap



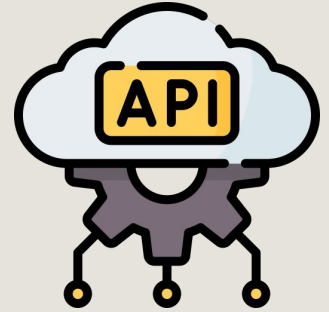
HTTP



methods



status codes

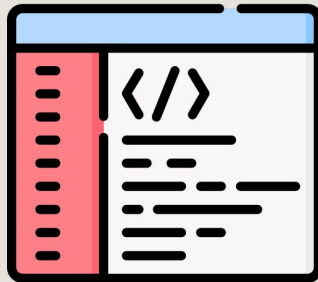


REST

# Steps



Presentation



Code

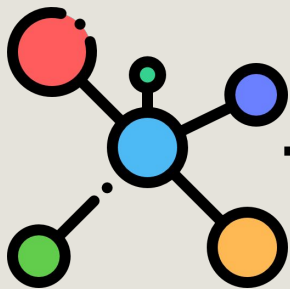


Postman

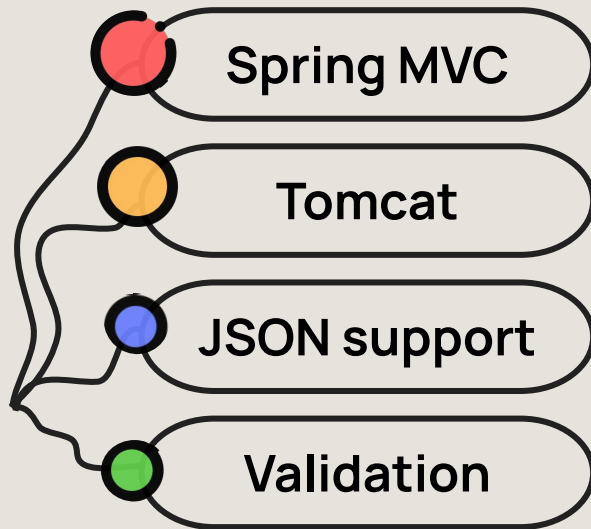
# Spring MVC

(Model View Controller)

- **Model** - data & business logic
- **View** - user interface (UI)
- **Controller** - link Model & View



"spring-boot-starter-web"



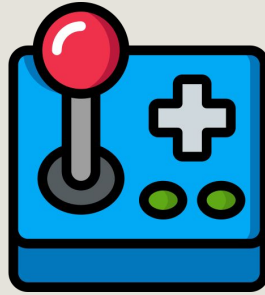
# Controller



**@RestController**

return data  
(JSON/XML)

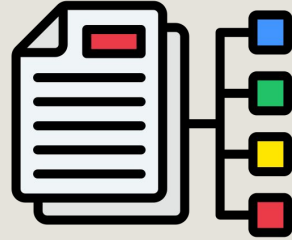
=



**@Controller**

return view  
(HTML/JSP)

+



**@ResponseBody**

formats return  
value

# Request/Response body



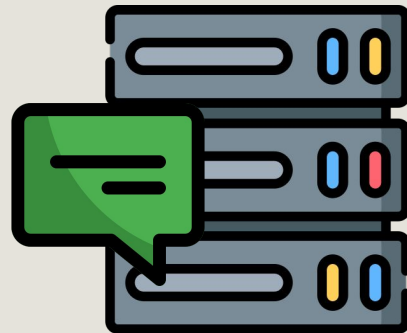
**@RequestBody**

HTTP request body



method parameter

vs



**@ResponseBody**

return value



HTTP response body

# Request mapping

- **level:** class/method
- **map:** path, method, parameter, header, media type

```
@RestController
@RequestMapping(path = "/api/shelter",
    produces = "application/json",
    consumes = "application/json")
public class ShelterController {
```

```
@RequestMapping(value = "/dogs",
    method = RequestMethod.GET,
    params = "name",
    headers = "Accept=application/json")
public Dog findByName(@RequestParam("name") String name) {
```

# HTTP Methods

## @GetMapping

```
@GetMapping(🌐📄 "/dogs")  
public List<Dog> getAllDogs() {
```

```
@GetMapping(value = 🌐📄 "/dogs", params = "name",  
            headers = "Accept=application/json")  
public Dog findByName(@RequestParam("name") String name) {
```



# Variables

## @RequestParam

/api/dogs?breed=Labrador

```
@GetMapping(🌐📄 "/dogs")
```

```
public String getDogByBreed(@RequestParam("breed") String breed) {
```

```
@GetMapping(🌐📄 "/dogs/{breed}")
```

```
public String getDogByBreed(@PathVariable("breed") String breed) {
```

## @PathVariable

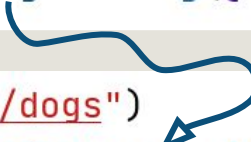
/api/dogs/Labrador

# HTTP Methods

## @PostMapping

```
@PostMapping(🌐📄"/dogs")  
public Dog addDog(@RequestBody Dog dog) {
```

```
@PostMapping(🌐📄"/dogs")  
public ResponseEntity<Dog> createDog(@RequestBody Dog dog) {  
    Dog savedDog = dogService.save(dog);  
    return new ResponseEntity<>(savedDog, HttpStatus.CREATED);  
}
```



# HTTP Methods

## @PutMapping

```
@PutMapping(🌐📄 "/dogs/{id}")  
public Dog updateDog(@PathVariable Long id, @RequestBody Dog dog) {
```

ResponseBody  
ResponseEntity<Void>

```
return ResponseEntity.ok().build();
```

# HTTP Methods

## @PatchMapping

```
@PatchMapping(🌐📄 "/dogs/{id}")  
public Dog partialUpdateDog(@PathVariable Long id,  
                             @RequestBody Map<String, Object> dogUpdate) {
```



```
@RequestBody DogDTO dogUpdate)
```

# DTO

(Data Transfer Object)

- simple objects
- encapsulate & transfer

```
public class DogDTO {  
    no usages  
    private String name;  
    no usages  
    private Integer age;  
  
    // setters & getters  
}
```

```
@Data  
public class DogDTO {
```

“lombok”

setters & getters

equals & hashCode

RequiredArgsConstructor

toString

# Validation

**@NotNull**

**@NotEmpty** - *not (null) empty collections*

**@NotBlank** - *min 1 non-whitespace char*

**@Size(min=, max=)** - *length*

**@Min(..) @Max(..)** - *number range*

**@Email**

**@Pattern(regexp=..)**



# Validation

```
@Data
public class DogDTO {
    @NotBlank(message = "Name is mandatory")
    private String name;

    @NotNull
    @Min(value = 0)
    private Integer age;

    @Size(max = 200)
    private String description;
}
```



```
@Valid @RequestBody
DogDTO dogUpdate) {
```

# HTTP Methods

## @DeleteMapping

```
@DeleteMapping(🌐✓ "/dogs/{id}")  
public void deleteDog(@PathVariable Long id) {
```





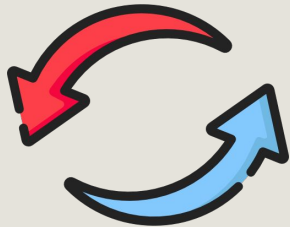
# Mapper

## DogDto

```
@Data
public class DogDTO {
    @NotBlank
    private String name;

    @NotNull
    @Min(value = 0)
    private Integer age;

    @Size(max = 200)
    private String breed;
```



## Dog

```
public class Dog {
    no usages
    private Long id;
    no usages
    private String name;
    no usages
    private Integer age;
    no usages
    private String breed;
```

# Mapper - classic

```
public class DogMapper {  
    no usages    new *  
    public static Dog dtoToDog(DogDTO dogDTO) {  
        Dog dog = new Dog();  
        dog.setName(dogDTO.getName());  
        dog.setBreed(dogDTO.getBreed());  
        dog.setAge(dogDTO.getAge());  
        // and many more fields to map  
        return dog;  
    }  
}
```

# Mapper - Mapstruct

```
@Mapper(componentModel = "spring")  
public interface DogMapper2 {  
    no usages  
    Dog dtoToDog(DogDTO dogDTO);  
    no usages  
    DogDTO dogToDto(Dog dog);  
}
```

# Mapstruct

- **code generator** (less boilerplate)
- **simpler mapping** (nested types, collections)
- **type safety** (compile-time checks)

“mapstruct”



```
@Mapping(source = "dogName", target = "name")
@Mapping(source = "breedCode", target = "breed", qualifiedByName = "breedCodeToBreed")
Dog toDog(DogDTO dogDTO);
```

no usages

```
@Named("breedCodeToBreed")
static String breedCodeToBreed(String breedCode) {
```



# Thank you!

Radina Nuneva