Web services (Spring)
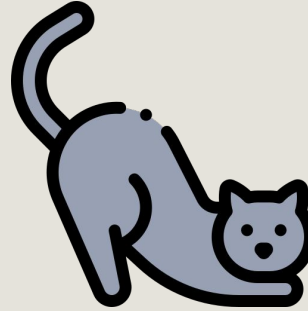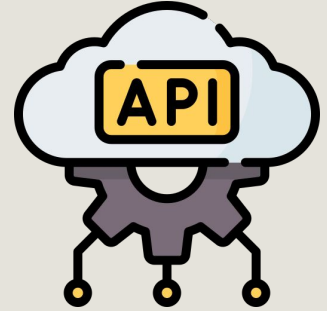
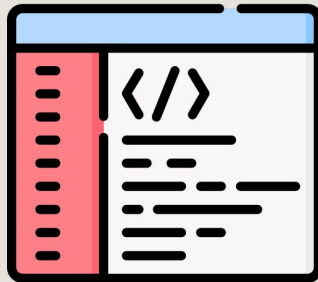Dreamix

Recap

HTTP

methods

status codes

REST

Steps
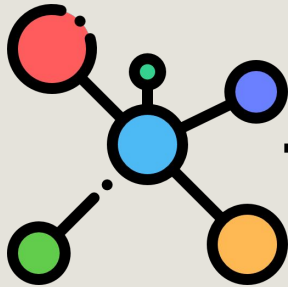
Presentation

Code

Postman

Spring MVC

(Model View Controller)

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

"spring-boot-starter-web"

Spring MVC

Tomcat

JSON support

Validation
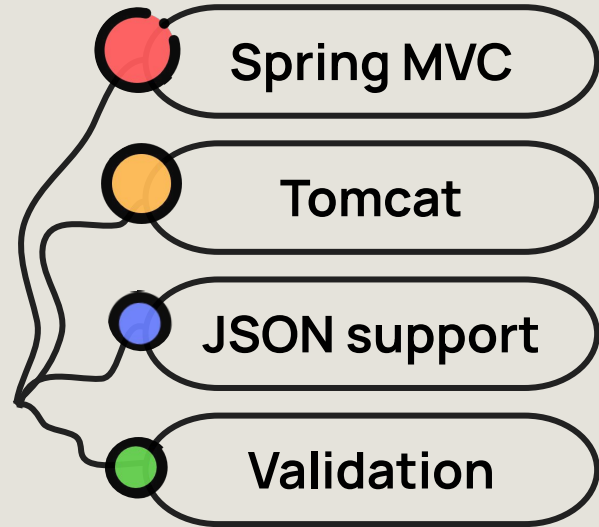
Controller

@RestController
return data
(JSON/XML)

=

@Controller
return view
(HTML/JSP)

+
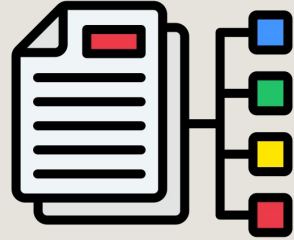
@ResponseBody
formats return
value

Request/Response body

@RequestBody

vs

@ResponseBody

HTTP request body
↓
method parameter
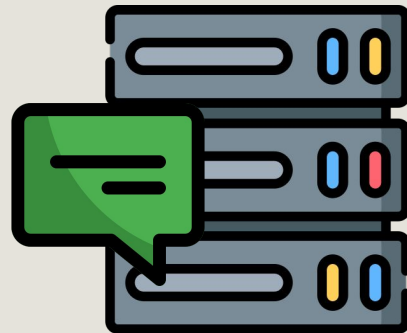
return value
↓
HTTP response body

Request mapping

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

```java
@RestController
@RequestMapping(path = "/api/shelter",
        produces = "application/json",
        consumes = "application/json")
public class ShelterController {
```
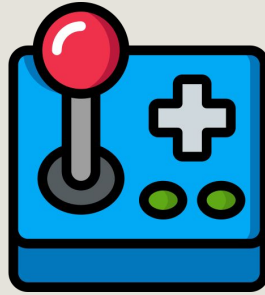
```java
@RequestMapping(value = "/dogs",
                method = RequestMethod.GET,
                params = "name",
                headers = "Accept=application/json")
public Dog findByName(@RequestParam("name") String name) {
```

HTTP Methods

@GetMapping

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

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

Variables

@RequestParam

/api/dogs**?breed=Labrador**

@PathVariable

/api/dogs**/Labrador**

```java
@GetMapping(🌐∨"/dogs")
public String getDogByBreed(@RequestParam("breed") String breed) {
```

```java
@GetMapping(🌐∨"/dogs/{breed}")
public String getDogByBreed(@PathVariable("breed") String breed) {
```
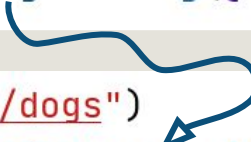
HTTP Methods

@PostMapping

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

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

HTTP Methods

@PutMapping

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

ResponseEntity<Void>

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

HTTP Methods

@PatchMapping

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

```java
@RequestBody DogDTO dogUpdate)
```

DTO

(Data Transfer Object)

- simple objects
- encapsulate & transfer

```java
public class DogDTO {
    no usages
    private String name;
    no usages
    private Integer age;


    // setters & getters
}
```

```java
@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=..)

"**hibernate-validator**"

Validation

```java
@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;
}
```

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

HTTP Methods

@DeleteMapping

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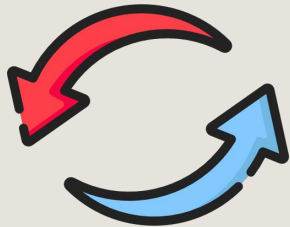
Mapper

DogDto

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

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

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

Dog

```java
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

```java
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

```java
@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)

```java
@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

Maven dependencies

Spring MVC (slide 4 - 16)

Lombok (slide 13)

Validation (slide 14 - 15)

Mapstruct (slide 19 -20) AND ALSO:

Mapstruct processor (slide 19 - 20)