# METODY PROGRAMOWANIA

Mateusz Czarnowski

14.10.2024

## REST CONTROLLER

```
@RestController
@RequestMapping("books-rest")
public class SimpleBookRestController {

    @GetMapping("/{id}", produces = "application/json")
    public Book getBook(@PathVariable int id) {
        return findBookById(id);
    }

    private Book findBookById(int id) {
        // ...
    }
}
```

## REQUEST BODY

```
@PostMapping("/request")
public ResponseEntity postController(
  @RequestBody LoginForm loginForm) {

    exampleService.fakeAuthenticate(loginForm);
    return ResponseEntity.ok(HttpStatus.OK);
}

public class LoginForm {
    private String username;
    private String password;
    // ...
}
```

## LOGINFORM JSON

```
{
„username” : „login”,
„password” : „password”
}
```

Json musi się zgadzać z obiektem.

## RESPONSE BODY

```
@PostMapping("/response")
    @ResponseBody
    public Car postResponseController(
      @RequestBody LoginForm loginForm) {
        return new Car();
     }
```

Z RestController niepotrzebne.

## PATH VARIABLE

```
@RequestMapping(path="/{name}/{age}") public String
getMessage(@PathVariable("name") String name,
@PathVariable("age") String age) {
```

## REQUEST PARAM

```
@GetMapping("/api/foos")
@ResponseBody
public String getFoos(@RequestParam String id) {
    return "ID: " + id;
}
```

```
http://localhost:8080/api/foos?id=abc
```

## JDBC

SPRING

JDBC

## H2

## JDBC TEMPLATE

```
@Autowired
JdbcTemplate jdbcTemplate;

jdbcTemplate.execute("CREATE TABLE customers(" + "id
SERIAL, first_name VARCHAR(255), last_name
VARCHAR(255))");
```

## JDBC TEMPLATE

```
@Autowired
JdbcTemplate jdbcTemplate;

jdbcTemplate.queryForObject( "SELECT id, first_name,
last_name FROM customers WHERE first_name = ?",
new Object[] { "Josh" }, (rs, rowNum) ->
new Customer(
rs.getLong("id"),
rs.getString("first_name"),
rs.getString("last_name")) );
```

# SPRING JPA

+

## SPRING JPA

```
@Entity public class Customer {
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
private Long id;
private String firstName;
private String lastName;

protected Customer() {}

public Customer(String firstName, String lastName) {
		this.firstName = firstName;
		this.lastName = lastName;
}
```

## SPRING JPA - REPOSITORY

```
public interface CustomerRepository extends
CrudRepository<Customer, Long> {
		List<Customer> findByLastName(String lastName);
}
```

## SPRING JPA - CRUDREPOSITORY

save(), findById(), delete(), count(), exists(), etc.

## SPRING JPA

```
repository.save(new Customer("Jack", "Bauer"));

Customer customer = repository.findById(1L);
```