**Creating Your First Spring Boot Application --- Dan Bunker·Intermediate --- PLURALSIGHT**

**Controller package**

**==================**

**@ControllerAdvice**

public class ControllerExceptionHandler **{**

**@ResponseBody**

**@ResponseStatus(HttpStatus.BAD\_REQUEST)**

**@ExceptionHandler**

public ErrorMio exceptionHandler**(**ValidationException e**)** **{**

**return** **new** ErrorMio**(**"400"**,** e**.**getMessage**());**

**}**

**}**

**...**

**import** org**.**springframework**.**web**.**bind**.**MethodArgumentNotValidException**;**

**import** org**.**springframework**.**web**.**bind**.**annotation**.\*;**

**import** javax**.**validation**.**Valid**;**

**import** javax**.**validation**.**ValidationException**;**

**import** java**.**util**.**List**;**

**import** java**.**util**.**stream**.**Collectors**;**

**@RestController**

public class **FriendController** **{**

**@Autowired**

**FriendService friendService;**

**@PostMapping("/friend")**

public Friend **create(**@**Valid @RequestBody** Friend friend**)** **{**

**return** friendService**.**save**(**friend**);**

**}**

**@ResponseStatus(HttpStatus.BAD\_GATEWAY)**

**@ExceptionHandler(MethodArgumentNotValidException.class)**

List**<**FieldErrorMessage**>** **exceptionHandler(**MethodArgumentNotValidException e**)** **{**

List**<**FieldError**>** fieldErrors **=** e**.**getBindingResult**().**getFieldErrors**();**

List**<**FieldErrorMessage**>** fieldErrorMessages **=** fieldErrors**.**stream**().**map**(**fieldError **->** **new** FieldErrorMessage**(**fieldError**.**getField**(),** fieldError**.**getDefaultMessage**())).**collect**(**Collectors**.**toList**());**

**return** fieldErrorMessages**;**

**}**

**@GetMapping("/friend")**

public **Iterable<**Friend**>** read**()** **{**

**return** friendService**.**findAll**();**

**}**

**@PutMapping("/friend")**

ResponseEntity**<**Friend**>** **update(@RequestBody** Friend friend**)** **{**

**if(**friendService**.**findById**(**friend**.**getId**()).**isPresent**())**

**return** **new** ResponseEntity**(**friendService**.**save**(**friend**),** HttpStatus**.**ACCEPTED**);**

**else**

**return** **new** ResponseEntity**(**friend**,** HttpStatus**.**BAD\_REQUEST**);**

**}**

**@DeleteMapping("/friend/{id}")**

public void delete**(**@**PathVariable** Integer id**)** **{**

friendService**.**deleteById**(**id**);**

**}**

**@GetMapping("/friend/search")**

Iterable**<**Friend**>** **metodoTaylored(@RequestParam(**"uno"**)** String firstName**,** **@RequestParam(**value**=** "dos"**,** required**=false)** String lastName**){**

**return** friendService**.**findByFirstNameAndLastName**(**firstName**,** lastName**);**

**}**

**@GetMapping("/wrong")**

public void **somethingIsWrong()** **{**

**throw** **new** ValidationException**(**"Something is wrong"**);**

**}**

**@ResponseStatus(HttpStatus.BAD\_REQUEST)**

**@ExceptionHandler(ValidationException.class)**

String **exceptionHandler(**ValidationException e**)** **{**

**return** e**.**getMessage**();**

**}**

**}**

**...**

**HELPER PACKAGE**

**=============**

public class **ErrorMio** **{**

private String status**;**

private String message**;**

public ErrorMio**()** **{**

**}**

public ErrorMio**(**String status**,** String message**)** **{**

**this.**status **=** status**;**

**this.**message **=** message**;**

**}**

**...**

public class **FieldErrorMessage** **{**

private String field**;**

private String message**;**

public FieldErrorMessage**(**String field**,** String message**)** **{**

**this.**field **=** field**;**

**this.**message **=** message**;**

**}**

**...**

**MODEL PACKAGE**

**============**

**import** com**.**fasterxml**.**jackson**.**annotation**.**JsonIgnore**;**

**import** com**.**fasterxml**.**jackson**.**annotation**.**JsonManagedReference**;**

**import** com**.**fasterxml**.**jackson**.**annotation**.**JsonProperty**;**

**import** javax**.**persistence**.\*;**

**import** javax**.**validation**.**constraints**.**NotBlank**;**

**import** java**.**util**.**List**;**

**@Entity**

public class Friend **{**

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

private int id**;**

**@NotBlank**

**@JsonProperty("first-name")**

private String firstName**;**

**@NotBlank**

**@JsonProperty("last-name")**

private String lastName**;**

int age**;**

**@JsonIgnore**

boolean married**;**

**@JsonManagedReference**

**@OneToMany(mappedBy = "friend", cascade = CascadeType.ALL)**

List**<**Address**>** addresses**;**

public Friend**(@NotBlank String firstName, @NotBlank String lastName)** **{**

**this.**firstName **=** firstName**;**

**this.**lastName **=** lastName**;**

**}**

**...**

**@Entity**

public class Address **{**

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

private int id**;**

private String street**;**

private String city**;**

**@JsonBackReference**

**@ManyToOne**

Friend friend**;**

**...**

**SERVICE PACKAGE**

**===============**

**@Repository**

public interface **FriendService** **extends** **CrudRepository<**Friend**,** Integer**>** **{**

// un metodo a medida mia ademas de los que ya tiene JPA, luego me voy al controller y le llamo

// el nombre del metodo tiene que seguir esta convencion de "findByFirsName..."

Iterable**<**Friend**>** **findByFirstNameAndLastName(**String firstName**,** String lastName**);**

**}**

**...**

**GENERAL PACKAGE**

**===============**

**@SpringBootApplication**

public class FriendsApplication **{**

public static void **main(**String**[]** args**)** **{**

SpringApplication**.**run**(**FriendsApplication**.**class**,** args**);**

**}**

**}**

**...**

**application.properties**

**======================**

spring**.**datasource**.**url**=**jdbc**:**mysql**:**//localhost/friends

spring**.**datasource**.**username**=**root

spring**.**datasource**.**password**=**1234

spring**.**datasource**.**driver**-**class**-**name**=**com**.**mysql**.**jdbc**.**Driver

spring**.**jpa**.**properties**.**hibernate**.**dialect**=**org**.**hibernate**.**dialect**.**MySQL5Dialect

spring**.**jpa**.**hibernate**.**ddl**-**auto**=**create**-**drop

spring**.**jpa**.**show**-**sql**=true**

spring**.**datasource**.**url**=**jbdc**:**mysql**:**//localhost/friends

spring**.**datasource**.**username**=**admin

spring**.**datasource**.**password**=**1234

spring**.**datasource**.**driver**-**class**-**name**=**com**.**mysql**.**jdbc**.**Driver

spring**.**jpa**.**properties**.**hibernate**.**dialect**=**org**.**hibernate**.**dialect**.**MySQL5Dialect

spring**.**jpa**.**hibernate**.**ddl**-**auto**=**update

spring**.**jpa**.**show**-**sql**=true**

**...**

**TEST**

**=================================================**

**1- Test de la aplicacion**

**@RunWith(SpringRunner.class)**

**@SpringBootTest**

public class FriendsApplicationTests **{**

**@Autowired**

FriendController friendController**;**

**@Test**

public void contextLoads**()** **{**

Assert**.**assertNotNull**(**friendController**);**

**}**

**}**

**SERVICE TEST**

**===============================================**

**2- SYSTEM TEST**

public class **SystemTests** **{**

**@Test**

public void **testCreateReadDelete()** **{**

RestTemplate restTemplate **=** **new** RestTemplate**();**

String url **=** "http://localhost:8080/friend"**;**

Friend friend **=** **new** Friend**(**"Gordon"**,** "Moore"**);**

ResponseEntity**<**Friend**>** entity **=** restTemplate**.**postForEntity**(**url**,** friend**,** Friend**.**class**);**

Friend**[]** friends **=** restTemplate**.**getForObject**(**url**,** Friend**[].**class**);**

Assertions**.**assertThat**(**friends**).**extracting**(**Friend**::**getFirstName**).**containsOnly**(**"Gordon"**);**

restTemplate**.**delete**(**url **+** "/" **+** entity**.**getBody**().**getId**());**

Assertions**.**assertThat**(**restTemplate**.**getForObject**(**url**,** Friend**[].**class**)).**isEmpty**();**

**}**

**@Test**

public void **testErrorHandlingReturnsBadRequest()** **{**

RestTemplate restTemplate **=** **new** RestTemplate**();**

String url **=** "http://localhost:8080/wrong"**;**

**try** **{**

restTemplate**.**getForEntity**(**url**,** String**.**class**);**

**}** **catch** **(**HttpClientErrorException e**)** **{**

Assert**.**assertEquals**(**HttpStatus**.**BAD\_REQUEST**,** e**.**getStatusCode**());**

**}**

**}**

**}**

**CONTROLLER TEST**

**===============================================**

**3- INTEGRATION TEST**

**@RunWith(SpringRunner.class)**

**@SpringBootTest**

public class **IntegrationTests** **{**

**@Autowired**

FriendController friendController**;**

**@Test**

public void testCreateReadDelete**()** **{**

Friend friend **=** **new** Friend**(**"Gordon"**,** "Moore"**);**

Friend friendResult **=** friendController**.**create**(**friend**);**

Iterable**<**Friend**>** friends **=** friendController**.**read**();**

Assertions**.**assertThat**(**friends**).**first**().**hasFieldOrPropertyWithValue**(**"firstName"**,** "Gordon"**);**

friendController**.**delete**(**friendResult**.**getId**());**

Assertions**.**assertThat**(**friendController**.**read**()).**isEmpty**();**

**}**

@Test**(**expected **=** ValidationException**.**class**)**

public void errorHandlingValidationExceptionThrown**()** **{**

friendController**.**somethingIsWrong**();**

**}**

**}**

**===============================================**

**4- STANDALONE TEST**

**@RunWith(SpringRunner.class)**

**@WebMvcTest(FriendController.class)**

public class StandaloneControllerTests **{**

**@MockBean**

FriendService friendService**;**

**@Autowired**

MockMvc mockMvc**;**

**@Test**

public void **testCreateReadDelete()** **throws** **Exception** **{**

Friend friend **=** **new** Friend**(**"Gordon"**,** "Moore"**);**

List**<**Friend**>** friends **=** Arrays**.**asList**(**friend**);**

Mockito**.**when**(**friendService**.**findAll**()).**thenReturn**(**friends**);**

**mockMvc.perform(**get**(**"/friend"**))**

**.**andExpect**(**status**().**isOk**())**

**.**andExpect**(**jsonPath**(**"$"**,** Matchers**.**hasSize**(**1**)))**

**.**andExpect**(**jsonPath**(**"$[0].firstName"**,** Matchers**.**is**(**"Gordon"**)));**

**}**

**}**