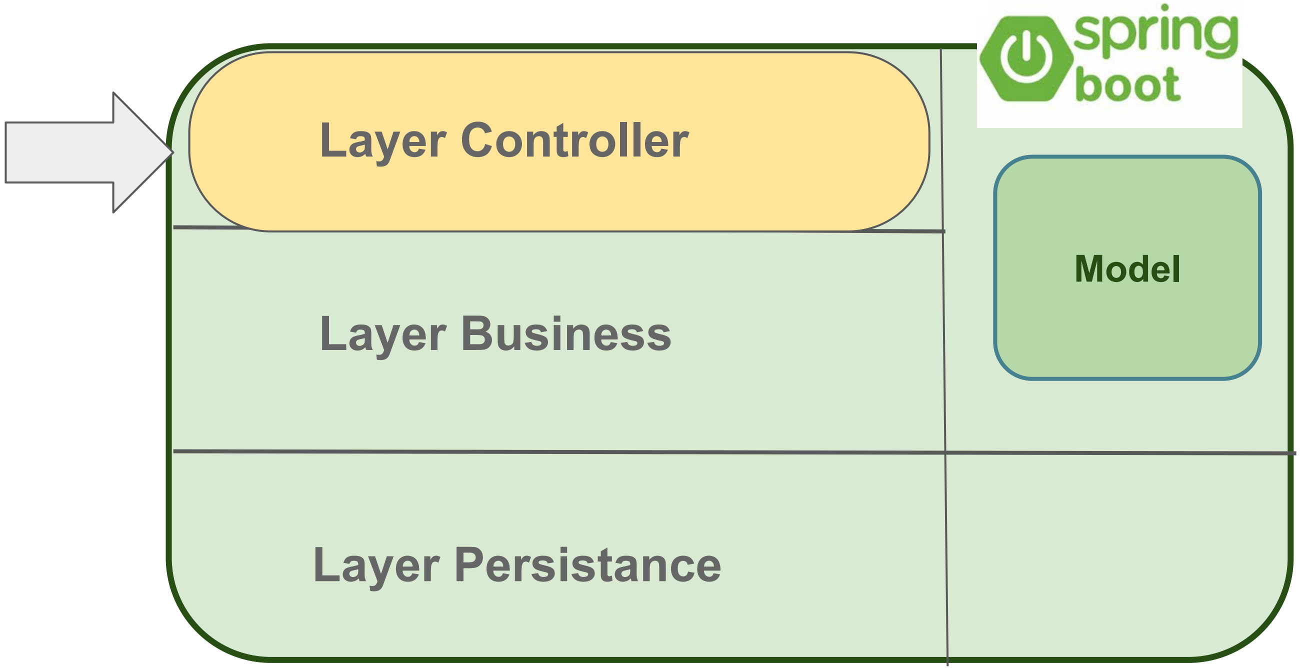
Instructor Ing. Rafael Benedettelli

Microservicios con Spring Boot


spring boot
Layer Controller
Layer Business
Layer Persistance
Model

Indice de contenidos

C
o
n
t
r
o
l
l
e
r

Open API Rest Specification

Setup de swagger en nuestro proyecto

(1) dependencias

```
<dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger2</artifactId>
        <version>2.9.2</version>
</dependency>

<dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger-ui</artifactId>
        <version>2.9.2</version>
</dependency>
```

(2) Bean Startup

```
@Configuration
@EnableSwagger2
public class SwaggerConfiguration {

          @Bean
          public Docket api() {
                    return new Docket(DocumentationType.SWAGGER_2)
                                  .select()
                                  .apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))
                                  .paths(PathSelectors.any())
                                         .build()
                                         .apiInfo(new ApiInfo("EscuelaIT", "Microservice", "1.0", "http://escuelait.com",
                                                              new Contact("Rafael Benedettelli", "escuelait.com", "escuelait@gmail.com"),
                                                              "Uso exclusivo EscuelaIT", "http://anfix.com", Collections.emptyList()));

          }

          }
```

UI Generada + api-docs

Documentación Rest

@ApiModel
@ApiModelProperty
@Api
@ApiOperation
@ApiParam
@ApiResponses

Documentación Rest

```
@ApiModel(description = “System user”)
public class User{ … }

@ApiModelProperty(notes = "Unique identifier of the
User.", example = "1", required = true, position = 0)
private Integer id;
```

Documentación Rest

```
@Api(tags = “User API Rest”)
public class UserControllerRest{ … }

@ApiOperation(notes=”Retrieve one user system by
id”,value=”Get user by id”)
private ResponseEntity<UserDTO> getUserById(Integer id){};
```

Documentación Rest

```
getUserById
(
 @ApiParam(example = "1",value = "Identifier for User",allowableValues =
"1,2,3,4",required = true)
@PathVariable Integer id
)
```

Documentación Rest

```
@ApiResponses(value = {
        @ApiResponse(code = 200,message = "Response ok if the operation was successful"),
        @ApiResponse(code = 404,message = "Response not found if the resource could not be found")
})
```

Hateoas

```
{
    "departmentId": 10,
    "departmentName": "Administration",
    "locationId": 1700,
    "managerId": 200,
    "links": [
        {
            "href": "10/employees",
            "rel": "employees"
        }
    ]
}
```

Hateoas

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-hateoas</artifactId>
</dependency>
```

Hateoas

```
@Bean
public LinkDiscoverers discoverers() {
        List<LinkDiscoverer> plugins = new ArrayList<>();
        plugins.add(new CollectionJsonLinkDiscoverer());
        return new LinkDiscoverers(SimplePluginRegistry.create(plugins));


}
```

Hateoas

```
public class UserDTO extends RepresentationModel<UserDTO> {
... }
```

Hateoas

```
import org.springframework.hateoas.Link;
import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.linkTo;
import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.methodOn;
```

Hateoas (Navegación self resource)

```
        Link withSelfRel =
linkTo(methodOn(UsersControllerRest.class).getUserById(userDTO.getId())).withSelfRel();
        userDTO.add(withSelfRel);
```

tipo de relación

Hateoas (Navegación self list)

```
Link link = linkTo(methodOn(UserController.class).getAllUsers()).withSelfRel();
    CollectionModel<UserDTO> result = CollectionModel.of(users, link);
        return ResponseEntity.ok(result);
```

Hateoas (Navegación inter-resource)

```
 Link accountsRel =
linkTo(methodOn(UsersControllerRest.class).getUserAccountById(userDTO.getId())).withRel("accounts");
        userDTO.add(accountsRel);
```

Java

Instructor Ing. Rafael Benedettelli

Java Bean Validation 2.0

JSR 380

Capacitacion Anfix

Spring Version 2.3.3 debemos explicitar la dependencia

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

JSR 380

Validators

@Validated
@Valid

@NotNull (null)
@NotBlank (“”)
@Size (string)
@Min (int)
@Max (int)
@Regex()
@AssertTrue (bool)

New Validators in JSR 380

@Validated
@Valid

@Email
@Positive
@PositiveOrZero
@Negative
@NegativeOrZero
@PastOrPresent
@FutureOrPresent

Custom Error Messages

```
import java.util.List;
import org.springframework.http.HttpStatus;
import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class ApiError {

        private HttpStatus status;
        private String message;
        private List<String> errors;

}
```

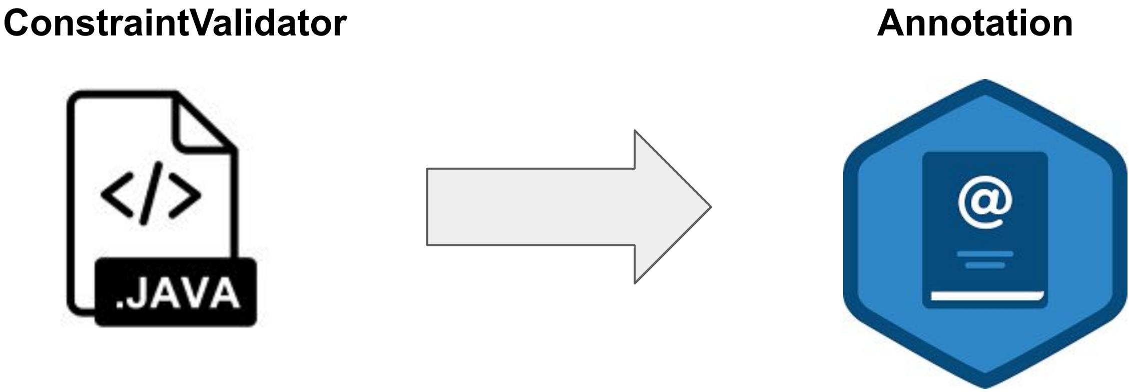
Custom Error Messages

```
import java.util.ArrayList;
import java.util.List;

import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

@ControllerAdvice
public class CustomRestExceptionHandler extends ResponseEntityExceptionHandler {

	@Override
	protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
					HttpHeaders headers, HttpStatus status, WebRequest request) {

		List<String> errors = new ArrayList<String>();

		for (FieldError error : ex.getBindingResult().getFieldErrors()) {
			errors.add(error.getField() + ": " + error.getDefaultMessage());
		}

		for (ObjectError error : ex.getBindingResult().getGlobalErrors()) {
			errors.add(error.getObjectName() + ": " + error.getDefaultMessage());
		}

		ApiError apiError = new ApiError(HttpStatus.BAD_REQUEST, ex.getMessage(), errors);
		return handleExceptionInternal(ex, apiError, headers, apiError.getStatus(), request);
	}

}
```

Externalización e Internacionalización I18N

```
import java.util.Locale;

import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ReloadableResourceBundleMessageSource;
import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;

@Configuration
public class ResourceMessageConfiguration implements WebMvcConfigurer {

        /**
         * Establece un archivo de mensajes por default
         */
        @Bean
        public MessageSource messageSource() {

                ReloadableResourceBundleMessageSource messageSource = new ReloadableResourceBundleMessageSource();
                messageSource.setBasename("classpath:messages");
                messageSource.setDefaultEncoding("UTF-8");
                return messageSource;
        }

        /*
         * Proporciona la posibilidad de que los mensajes de los validator accedan al archivo de claves message
         */
        @Bean
        public LocalValidatorFactoryBean validator(MessageSource messageSource) {
                LocalValidatorFactoryBean bean = new LocalValidatorFactoryBean();
                bean.setValidationMessageSource(messageSource);
                return bean;
        }

        /*
         * Establece un locale por default
         */
        @Bean
        public SessionLocaleResolver localeResolver() {
                SessionLocaleResolver localeResolver = new SessionLocaleResolver();
                localeResolver.setDefaultLocale(Locale.ENGLISH);
                return localeResolver;
        }

        /*
         * Permite cambiar el local en cada llamada por el param lang
         */
        @Bean
        public LocaleChangeInterceptor localeChangeInterceptor() {
                LocaleChangeInterceptor lci = new LocaleChangeInterceptor();
                lci.setParamName("lang");
                return lci;
        }

        @Override
        public void addInterceptors(InterceptorRegistry registry) {
                registry.addInterceptor(localeChangeInterceptor());
        }


}
```

Custom Validators



Custom Validators

```
import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

public class CIFValidator implements ConstraintValidator<CIF, String>{

	@Override
	public boolean isValid(String value, ConstraintValidatorContext
context) {

		if (value == null) {
			return false;
		}

		return value.length() == 9;
	}

}
```

```
import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
import javax.validation.Constraint;
import javax.validation.Payload;

@Documented
@Constraint(validatedBy = CIFValidator.class)
@Target( { ElementType.FIELD })
@Retention(RetentionPolicy.RUNTIME)
public @interface CIF {
   String message() default "Invalid CIF Number";
   Class<?>[] groups() default {};
   Class<? extends Payload>[] payload() default {};
}
```

Validación por grupos

```
public interface OnUpdate{}
public interface OnCreate{}
@Validate(OnUpdate.class)
@NotNull(groups=OnUpdate.class)
```