DESENVOLVIMENTO DE SERVIÇOS PARA WEB I

IFMS - Câmpus Naviraí
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Profº Marcos Rogério Ferreira

CONTEÚDO

- **Tratamento personalizado das Exceções**

SUMÁRIO

1. Tratamento de Exceções

O próprio Spring possui diretivas para tratamento e lançamento de exceções, podemos criar uma camada personalizada para tratar as exceções lançadas pelo sistema.

Tratamento de exceções
Importante: Intervir na camada que lançou a exception

A camada de serviço lançou a exceção, então faremos a intervenção a partir do lançamento.

1.1. Criando uma exceção personalizada

Erro apresentado no console da IDE

```
JavaScript
java.util.NoSuchElementException: No value present
```

Precisamos iniciar os tratamentos de exceções, vamos criar uma classe personalizada para tratar a exceção, crie uma classe com base nos dados abaixo:

Pacote: br.com.anpede.***services.exceptions***
Classe: EntityNotFoundException

```
JavaScript
public class EntityNotFoundException extends RuntimeException {
	private static final long serialVersionUID = 1L;

	public EntityNotFoundException(String msg) {
		super(msg);
	}
}
```

Agora podemos utilizar nossa classe de exceção no método que está lançando a exception.

1.1.1. Adicionando a exception personalizada

Abra o método na camada de serviço que gerou a exception

```
JavaScript
@Transactional(readOnly = true)
public AssociadoDTO findById(Long id){
	Optional<Associado> obj = repository.findById(id);
	Associado entity = obj.orElseThrow(() -> new EntityNotFoundException("O
registro não foi localizado na base de dados"));
	return new AssociadoDTO(entity);
}
```

ATENÇÃO
O java possui uma classe com o nome EntityNotFoundException, portanto verifique o import da classe, deve ser a classe que criamos na camada de serviço.

SAIBA MAIS...

`obj.orElseThrow( … );`	Retorna o valor contido, se presente, caso contrário, lança uma exceção a ser criada pelo desenvolvedor.

Documentação

- Optional - docs.oracle

Execute para verificar se o spring está lançando a nossa exceção.
Passe um ID inexistente para gerar o erro. No **console do STS** deverá ser lançado uma exceção como a linha abaixo:

```
JavaScript
br.com.anpede.services.exceptions.EntityNotFoundException: O registro não foi localizado na base de dados
```

Conseguimos passar a nossa classe de tratamento de exceção, porém o JSON que está sendo entregue no cliente ainda é gerado pelo Spring Framework e não temos muita autoridade sobre isso.

1.2. Criar uma estrutura para devolver a exception personalizada no JSON

Antes a mensagem era do Spring agora vamos enviar a nossa mensagem personalizada.

1.2.1. Capturando a exceção

PADRÃO (Não implementar)

```
JavaScript
@GetMapping(value = "/{id}")
public ResponseEntity<AssociadoDTO> findById(@PathVariable Long id){
	try {
		AssociadoDTO dto = service.findById(id);
		return ResponseEntity.ok().body(dto);
	} catch (EntityNotFoundException e) {
		//Tratamento
	}
}
```

1.2.2. Customizando a resposta de exceção

Classe responsável por manter padronizar as respostas, vamos criar a mesma estrutura apresentada pelo Spring. Quem deve enviar a resposta ao cliente é a camada de RESOURCE portanto vamos definir nessa camada a forma de envio.
Crie uma classe para formar a resposta da API, será a mesma que o Spring envia porém controlada por nós.

Pacote: br.com.anpede.***resources.exceptions***
Classe: StandartError

```
JavaScript
public class StandartError implements Serializable {
	private static final long serialVersionUID = 1L;

	private Instant timestamp;
	private Integer status;
	private String error;
	private String message;
	private String path;

	public StandartError() {
		// TODO Auto-generated constructor stub
	}

	// Getters and Setters
}
```

1.2.3. Criar uma classe para interceptar todas as exceptions.

Interceptar o lançamento da exceção e personalizar a resposta.

Pacote: br.com.anpede.***resources.exceptions***
Classe: ResourceExceptionHandler

```
JavaScript
@ControllerAdvice
public class ResourceExceptionHandler {

	@ExceptionHandler(EntityNotFoundException.class)
	public ResponseEntity<StandartError> entityNotFound(EntityNotFoundException
e, HttpServletRequest request){
		//Instanciamos uma mensagem de erro com base no padrão que criamos
		StandartError error = new StandartError();
		//Horário da ocorrencia
		error.setTimestamp(Instant.now());
		//Código HTTP que enviado, nesse caso 404.
		error.setStatus(HttpStatus.NOT_FOUND.value());
		//Titulo da exceção
		error.setError("Recurso não encontrado");
```

```
            //Captura da mensagem de erro interceptada e que foi definida na camada
de servico
            error.setMessage(e.getMessage());
            //Captura da URI que gerou a exceção
            error.setPath(request.getRequestURI());
            //Empacota a exceção no formato HTTP e entrega ao cliente
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(error);
      }
}
```

Documentação

- @ControllerAdvice - docs.spring

ATENÇÃO

O java possui uma classe com o nome EntityNotFoundException, portanto verifique o import da classe, deve ser a classe que criamos na camada de serviço.

SAIBA MAIS...

@ControllerAdvice
É uma especialização da @Component que permite tratar exceções em toda a aplicação em um componente de manipulação global. Ele pode ser visto como um interceptador de exceções lançadas por métodos anotados com @RequestMappinge similares. @ControllerAdvice é um interceptor acionado por anotação. Ele intercepta a maioria das classes que incluem @RequestMapping.

@ExceptionHandler(EntityNotFoundException.class)
A @ExceptionHandler nos dá muita flexibilidade em termos de tratamento de exceções. Para começar, para usá-lo, basta criar um método no próprio controlador ou em uma @ControllerAdvice e anotá-lo com @ExceptionHandler: O método de tratamento de exceção recebe uma exceção ou uma lista de exceções como um argumento que queremos tratar no método definido. Anotamos o método com @ExceptionHandler e @ResponseStatus para definir a exceção que queremos tratar e o código de status que queremos retornar.

entityNotFound(EntityNotFoundException e, HttpServletRequest request)
O método recebe dois parâmetros, o primeiro é a exceção lançada pela camada de serviço o segundo parâmetro passa informações da requisição que gerou o erro.

Execute e teste

Podemos testar e ver o resultado no Postman, agora as exceptions são controladas e personalizadas quem retorna são nossos controladores.

JavaScript

```
{
    "timestamp": "2023-08-07T20:35:48.516041900Z",
    "status": 404,
    "error": "Recurso não encontrado",
    "message": "O registro não foi localizado na base de dados",
    "path": "/associados/10"
}
```