INSTITUTO FEDERAL
Rio Grande do Sul

Desenvolvimento de Sistemas I

Status HTTP

Professor: *Jezer Machado de Oliveira*

Status HTTP

- Representam o código de resposta de uma solicitação feita ao servidor
- Dividido em 5 grupos distintos
  - 1xx Informativo
  - 2xx Sucesso
  - 3xx Redirecionamento
  - 4xx Erro de cliente
  - 5xx Outros erros

Status HTTP

- 1xx Informativo
  - Indicam respostas provisórias, utilizado principalmente pra requisições fracionadas ou muito longas
  - 100 Continuar
  - 101 Mudando protocolos
  - 102 Processamento (WebDAV)

Status HTTP

- 2xx Sucesso
  - Indica que a ação solicitada pelo cliente foi recebida, compreendida, aceita e processada com êxito
  - 200 OK
  - 201 Criado
  - 202 Aceito
  - 204 Nenhum conteúdo

Status HTTP

- 3xx Redirecionamento
  - O cliente deve tomar medidas adicionais para completar o pedido
  - 300 Múltipla escolha
  - 301 Movido permanentemente
  - 304 Não modificado
  - 307 Redirecionamento temporário

Status HTTP

- 4xx Erro de cliente
  - Indica que cliente cometeu algum erro na solicitação
  - 400 Requisição inválida
  - 401 Não autorizado
  - 402 Pagamento necessário
  - 403 Proibido
  - 404 Não encontrado
  - 405 Método não permitido
  - 418 Eu sou um bule de chá

Status HTTP

- 5xx outros erros
  - Erros variados, normalmente vindos do servidor
  - 500 Erro interno do servidor
  - 501 Não implementado
  - 502 Bad Gateway
  - 503 Serviço indisponível
  - 504 Gateway Time-Out
  - 505 HTTP Version not supported

Status HTTP

- https://http.cat/

Status HTTP e API Rest

Não existe uma especificação formal de quais status uma API Rest deve retornar, mas se convencionou a utilização de alguns status.

Status HTTP e API Rest

- 200 OK
  - Métodos de GET que obtém sucesso na recuperação de recursos.
- 201 Criado
  - Métodos de POST que criam recursos com sucesso e retornam o recurso criado.
- 204 Nenhum conteúdo
  - Métodos de PUT e DELETE que executam com sucesso e não retornam nada.

Status HTTP e API Rest

- 301 Movido permanentemente
  - Algum recurso da API trocou de nome de forma permanente
- 304 Não modificado
  - Utilizado em métodos GET como retorno quando o recurso não foi alterado desde a última consulta, necessita que o cliente envie a data do último acesso.

Status HTTP e API Rest

- 400 Requisição inválida
  - Erro genérico quando o cliente não envia a requisição bem formatada
- 401 Não autorizado
  - Um cliente não autentica ou com autenticação inválida, tentando acessar um recurso que necessita de autenticação

Status HTTP e API Rest

- 403 Proibido
  - A operação não é permitida ao cliente, mesmo devidamente autenticado
- 404 Não encontrado
  - O recurso não foi encontrado no momento, mas pode vir a existir no futuro
- 405 Método não permitido
  - O método não é permitido para o recurso, por exemplo, executar um POST em um recurso somente leitura

Status HTTP e API Rest

- 500 Erro interno do servidor
  - Erro inesperado na API, normalmente ocorre quando uma exceção não foi tratada
- 501 Não implementado
  - Algum recurso ou método na API que não foi implementado, mas pode ser no futuro

Status HTTP e Spring

- O Spring retorna alguns status por padrão:
  - 200 OK
  - 400 Requisição inválida
  - 401 Não autorizado
  - 404 Não encontrado
  - 405 Método não permitido
  - 500 Erro interno do servidor

Customizando status no Spring

- @ResponseStatus
  - Redefine o status quando o método é executado com sucesso
- ResponseEntity<T>
  - Encapsula o retorno em um objeto que permite configurar o status e demais elementos de cabeçalho
- Exceções
  - Permite criar exceções customizadas com status HTTP

@ResponseStatus

```
@RequestMapping(path = "/produtos/", method =RequestMethod.POST)
@ResponseStatus(HttpStatus.CREATED)
public  Produto salvar(@RequestBody Produto produto)  {
        produto.setId(0);
        Produto produtoComId = produtoDAO.save(produto);
        return produtoComId;
    }

@RequestMapping(path = "/produtos/{id}", method =RequestMethod.PUT)
@ResponseStatus(HttpStatus.NO_CONTENT)
public  void atualizar(@PathVariable int id, @RequestBody Produto produto)  {
        produto.setId(id);
        produtoDAO.save(produto);
    }
```

ResponseEntity

```
@RequestMapping(path = "/produtos/{id}", method =RequestMethod.GET)
public  ResponseEntity<Produto> recuperar(@PathVariable int id)  {
        Optional<Produto> findById = produtoDAO.findById(id);
        if(findById.isPresent()) {
            return  ResponseEntity.ok(findById.get());
        } else {
            return ResponseEntity.notFound().build();
        }
    }
```

Exceções

```
@ResponseStatus(HttpStatus.NOT_FOUND)
public class NaoEncontrado extends RuntimeException {
    public NaoEncontrado(String string) {
        super(string);
    }
}
```

Exceções

```
@RequestMapping(path = "/produtos/{id}", method =RequestMethod.DELETE)
@ResponseStatus(HttpStatus.NO_CONTENT)
public  void atualizar(@PathVariable int id)  {
      if(!produtoDAO.existsById(id)) {
          throw  new NaoEncontrado("Produto de id: "+id+" não encontrado");
      }
   produtoDAO.deleteById(id);
  }
```