



Cento Universitário UNA

Sistemas de Informação

Tecnologias Emergentes

Práticas de Laboratório

Wesley Dias Maciel

2019/02



Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

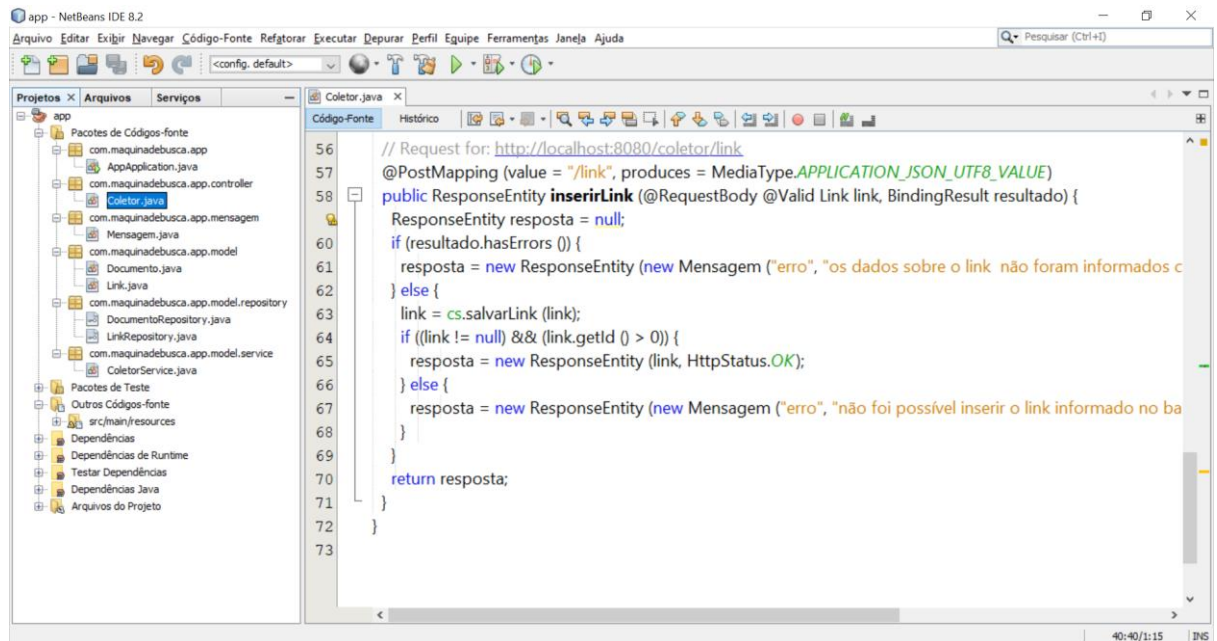
Spring Boot, Hibernate e JPA



Prática 07

- 1) Você está recebendo, juntamente com esta prática, o projeto da aplicação. Nesta versão, o projeto realiza uma validação de cada URL semente informada manualmente pelo administrador do sistema antes que elas sejam inseridas no banco de dados. A validação é realizada através da anotação **@Valid**. O resultado da validação é lido em um objeto da classe **BindingResult**. Caso o campo **url** da URL semente não seja informada pelo administrador, uma mensagem de erro no formato JSON é disparada pelo sistema.

```
// Request for: http://localhost:8080/coletor/link
@PostMapping (value = "/link", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
public ResponseEntity inserirLink (@RequestBody @Valid Link link, BindingResult resultado) {
    ResponseEntity resposta = null;
    if (resultado.hasErrors ()) {
        resposta = new ResponseEntity (new Mensagem ("erro", "os dados sobre o link não foram
informados corretamente"), HttpStatus.BAD_REQUEST);
    } else {
        link = cs.salvarLink (link);
        if ((link != null) && (link.getId () > 0)) {
            resposta = new ResponseEntity (link, HttpStatus.OK);
        } else {
            resposta = new ResponseEntity (new Mensagem ("erro", "não foi possível inserir o link
informado no banco de dados"), HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
    return resposta;
}
```

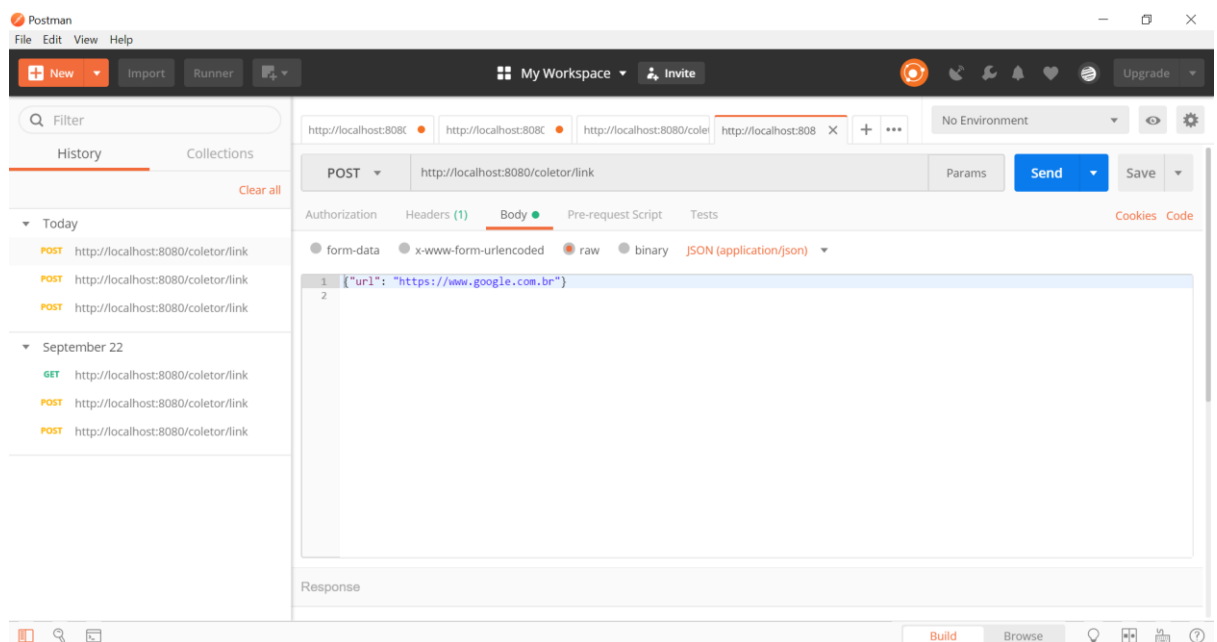


```
56 // Request for: http://localhost:8080/coletor/link
57 @PostMapping (value = "/link", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
58 public ResponseEntity inserirLink (@RequestBody @Valid Link link, BindingResult resultado) {
59     ResponseEntity resposta = null;
60     if (resultado.hasErrors ()) {
61         resposta = new ResponseEntity (new Mensagem ("erro", "os dados sobre o link não foram informados c
62     } else {
63         link = cs.salvarLink (link);
64         if ((link != null) && (link.getId () > 0)) {
65             resposta = new ResponseEntity (link, HttpStatus.OK);
66         } else {
67             resposta = new ResponseEntity (new Mensagem ("erro", "não foi possível inserir o link informado no ba
68         }
69     }
70     return resposta;
71 }
72 }
73 }
```

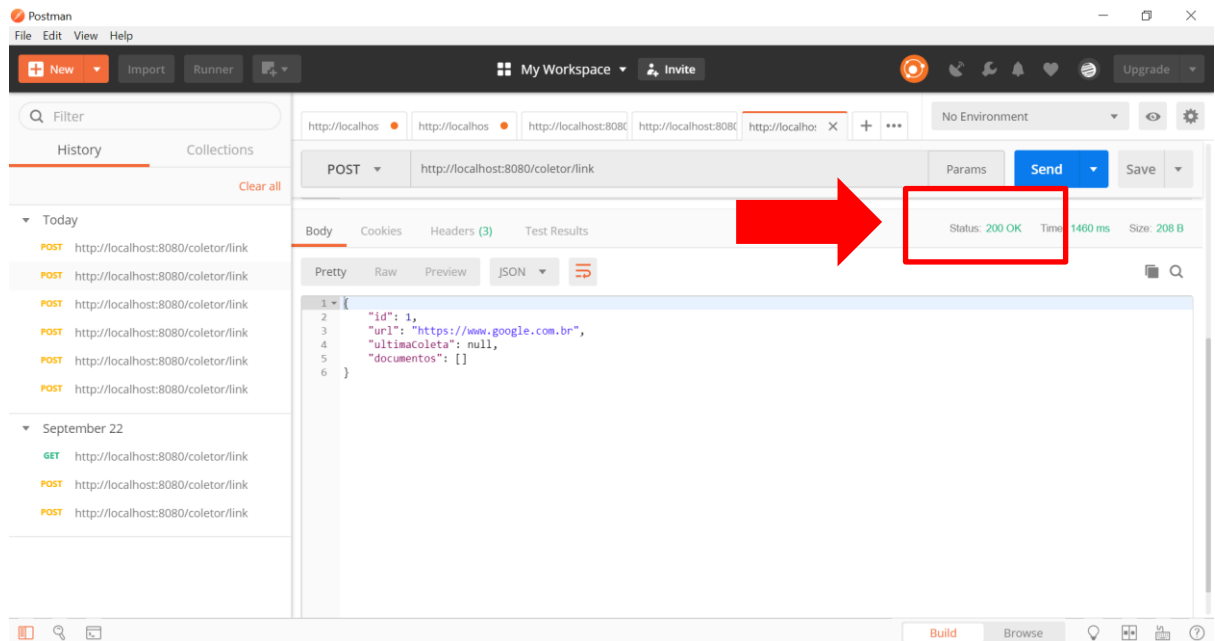
Para testar a API de cadastro das URLs sementes com validação, use um programa para teste de APIs REST, como o Postman (<https://www.getpostman.com/>).

Execute o projeto e observe as respostas geradas pelo servidor no Postman.

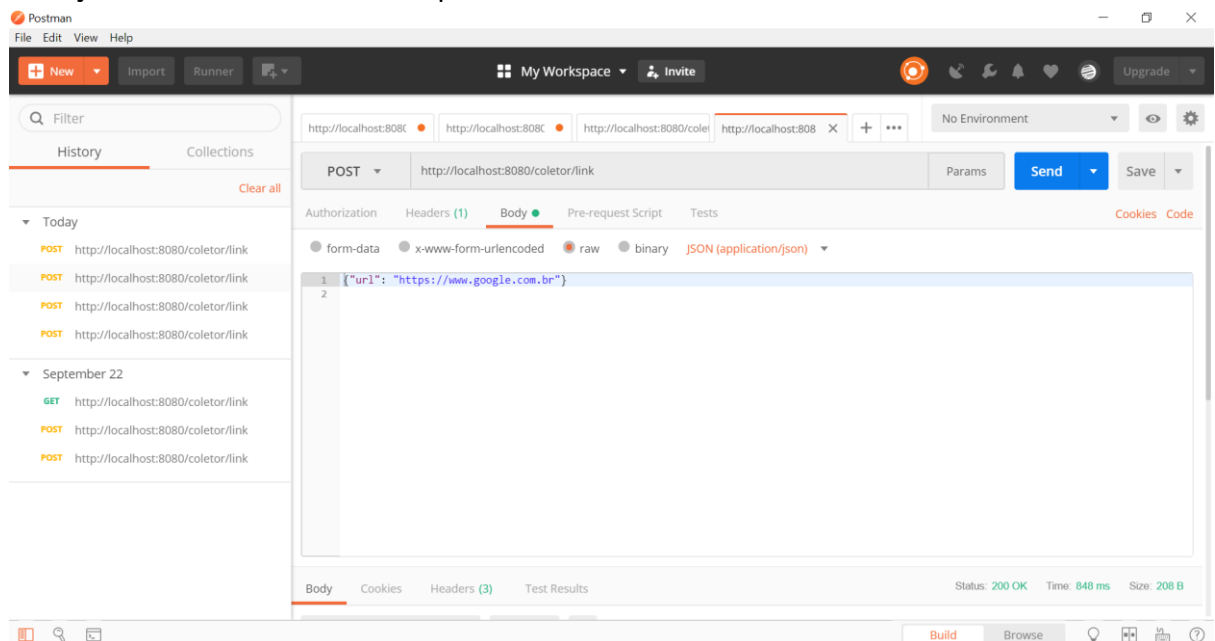
Inserção de uma nova URL semente:



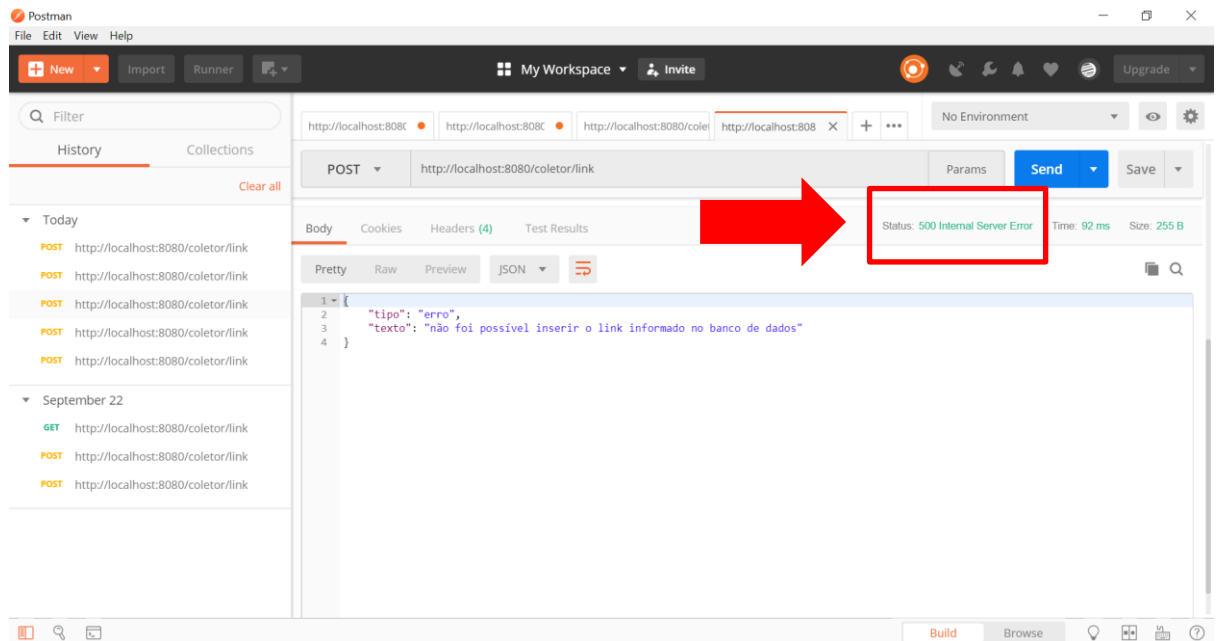
Observe a mensagem de sucesso retornada pelo servidor:



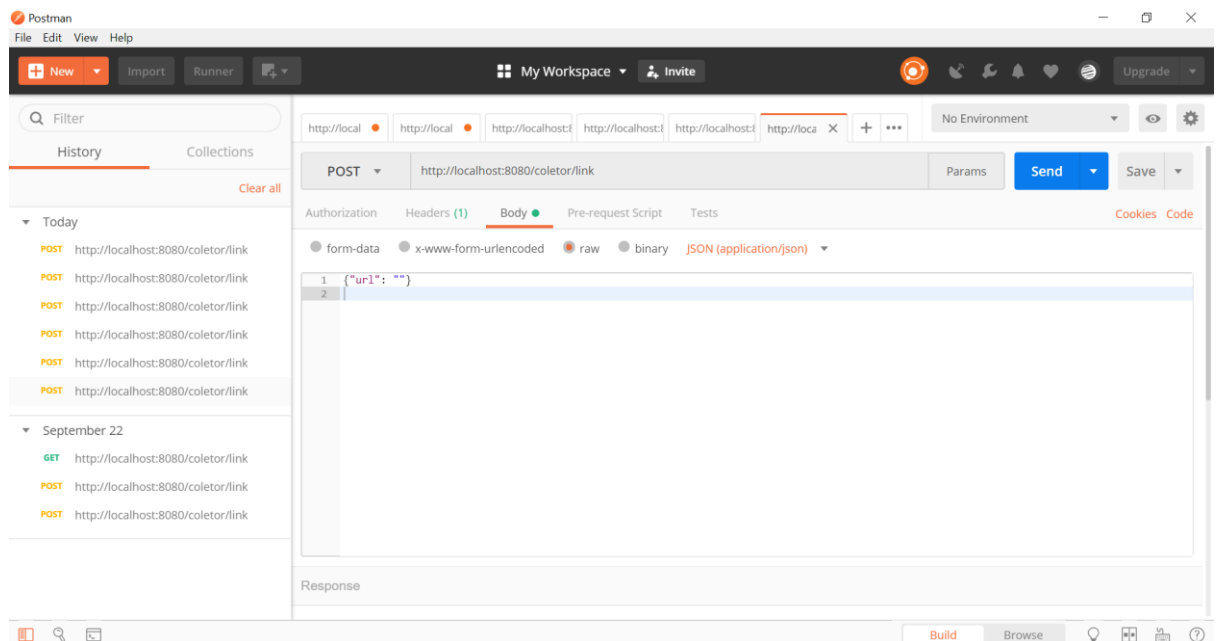
Inserção de uma URL semente repetida:



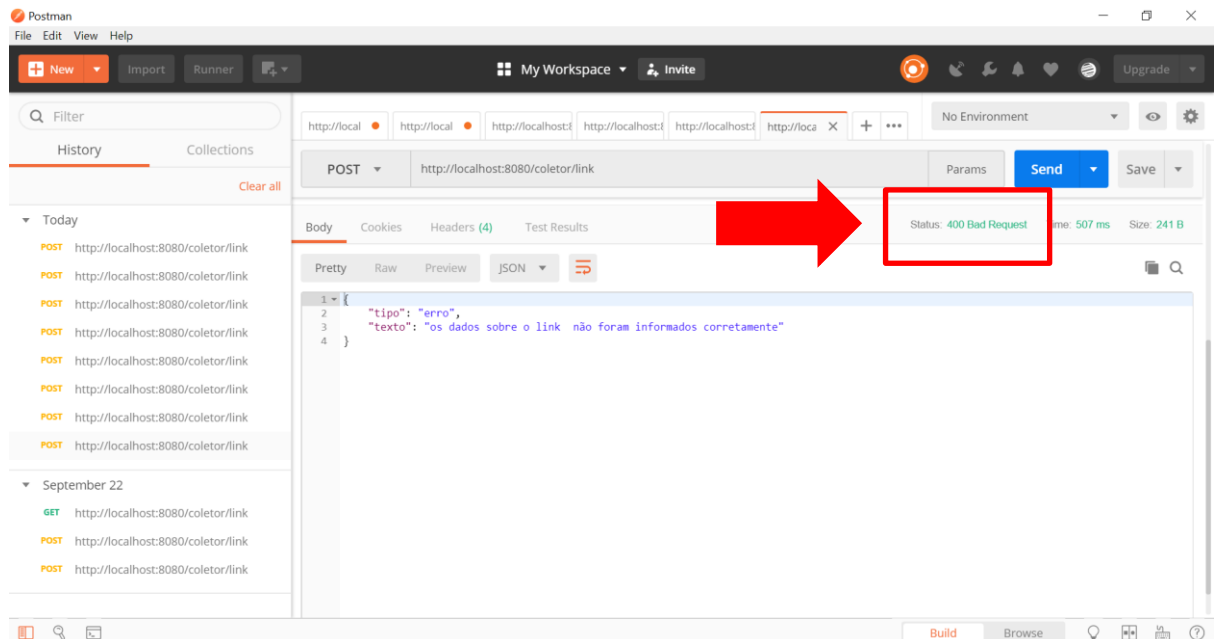
Observe a mensagem de erro retornada pelo servidor:



Inserção de uma nova URL semente com o campo `url` vazio:



Observe a mensagem de erro retornada pelo servidor:



- 2) Em seu projeto, realize a validação de parâmetros recebidos de aplicações cliente que fazem interface com usuário.
- 3) Analise a API do seu projeto. Sempre que necessário, faça alterações para melhoria da API, adequando-a ao padrão arquitetural REST (Representational State Transfer).
- 4) Para todos os métodos que interagem com aplicações cliente, retorne respostas com mensagens significativas para os clientes da aplicação, obedecendo os códigos adequados do protocolo HTTP.

Lista de códigos de status HTTP:

1xx Informativa

- 100 Continuar
- 101 Mudando protocolos
- 102 Processamento (WebDAV) (RFC 2518)
- 122 Pedido-URI muito longo

2xx Sucesso

- 200 OK
- 201 Criado
- 202 Aceito
- 203 não-autorizado (desde HTTP/1.1)
- 204 Nenhum conteúdo
- 205 Reset
- 206 Conteúdo parcial



207-Status Multi (WebDAV) (RFC 4918)

3xx Redirecionamento

- 300 Múltipla escolha
- 301 Movido
- 302 Encontrado
- 303 Consulte Outros
- 304 Não modificado
- 305 Use Proxy (desde HTTP/1.1)
- 306 Proxy Switch
- 307 Redirecionamento temporário (desde HTTP/1.1)
- 308 Redirecionamento permanente (RFC 7538[2])

4xx Erro de cliente

- 400 Requisição inválida
- 401 Não autorizado
- 402 Pagamento necessário
- 403 Proibido
- 404 Não encontrado
- 405 Método não permitido
- 406 Não Aceitável
- 407 Autenticação de proxy necessária
- 408 Tempo de requisição esgotou (Timeout)
- 409 Conflito
- 410 Gone
- 411 comprimento necessário
- 412 Pré-condição falhou
- 413 Entidade de solicitação muito grande
- 414 Pedido-URI Too Long
- 415 Tipo de mídia não suportado
- 416 Solicitada de Faixa Não Satisfatória
- 417 Falha na expectativa
- 418 Eu sou um bule de chá
- 422 Entidade improcessável (WebDAV) (RFC 4918)
- 423 Fechado (WebDAV) (RFC 4918)
- 424 Falha de Dependência (WebDAV) (RFC 4918)
- 425 coleção não ordenada (RFC 3648)
- 426 Upgrade Obrigatório (RFC 2817)
- 450 bloqueados pelo Controle de Pais do Windows
- 499 cliente fechou Pedido (utilizado em ERPs/VPs)

5xx outros erros (erro de servidor)

- 500 Erro interno do servidor (Internal Server Error)
- 501 Não implementado (Not implemented)
- 502 Bad Gateway
- 503 Serviço indisponível (Service Unavailable)
- 504 Gateway Time-Out
- 505 HTTP Version not supported

OBS:

- 1) RFC (Request for Comments - "pedido de comentários"): são documentos técnicos desenvolvidos e mantidos pelo IETF (Internet Engineering Task Force), uma instituição que especifica os padrões que serão implementados e utilizados em toda a Internet.



Centro Universitário UNA
Sistemas de Informação
Tecnologias Emergentes
Prática de Laboratório
Wesley Dias Maciel
2019/02

- 2) WebDAV (Web-based Distributed Authoring and Versioning): é uma extensão do protocolo HTTP para transferência de arquivos; suporta bloqueio de recursos. Quando uma pessoa está editando um arquivo, ele fica bloqueado, impedindo que outras pessoas façam alterações ao mesmo tempo.