**UNIVERSIDADE SENAI JOINVILLE**

**Felipe Rafael Rothbarth**

**OT08 - Apresentação do projeto**

**JOINVILLE**

**2024**

**Fichamento - OT13 - REST**

### **REST**

**Definição:**

* REST (Representational State Transfer) é um estilo arquitetural utilizado no desenvolvimento de software, especialmente para a criação de serviços web. REST define um conjunto de restrições e princípios que orientam a construção de sistemas que utilizam protocolos HTTP para comunicação.

**Como Surgiu:**

* REST foi introduzido em 2000 por Roy Fielding em sua tese de doutorado na Universidade da Califórnia, Irvine. Ele foi um dos principais autores da especificação do HTTP e desenvolveu REST como uma forma de modelar a comunicação entre sistemas de maneira eficiente e escalável.

### **RESTful**

**Definição**:

* RESTful é um termo usado para descrever serviços web que aderem aos princípios e restrições do estilo arquitetural REST. Um serviço é considerado RESTful se ele utilizar métodos HTTP (como GET, POST, PUT, DELETE) para manipular os recursos identificados por URIs (Uniform Resource Identifiers) e se cada operação gerar uma representação atualizada dos dados.

### **URI**

**Definição:**

* URI (Uniform Resource Identifier) é uma string que identifica de forma única um recurso na web. Em outras palavras, uma URI é um endereço que aponta para um recurso específico em um servidor, permitindo que clientes façam requisições a esse recurso.

Esses conceitos são fundamentais no desenvolvimento de APIs modernas, que utilizam REST como uma base para criar sistemas distribuídos de maneira simples, escalável e independente de plataforma.

### **Métodos HTTP**

**1. GET**

* Definição: O método GET é utilizado para solicitar dados de um servidor. Ele é usado para recuperar informações de um recurso identificado por uma URI.
* Características:
  + É um método seguro e idempotente, o que significa que a solicitação não deve causar nenhuma alteração no estado do servidor, e múltiplas requisições idênticas produzirão o mesmo resultado.
  + Os parâmetros podem ser passados na URL, mas isso limita o tamanho dos dados que podem ser enviados.
* Exemplo de Uso:
  + Um cliente faz uma requisição GET para http://example.com/produtos/123 para recuperar informações sobre o produto com o ID 123.

**2. POST**

* Definição: O método POST é utilizado para enviar dados ao servidor, geralmente para criar um novo recurso ou enviar dados que irão modificar o estado do servidor.
* Características:
  + Não é idempotente; múltiplas requisições POST podem resultar em diferentes resultados (como a criação de múltiplos recursos).
  + Os dados são geralmente enviados no corpo da requisição.
* Exemplo de Uso:
  + Um cliente faz uma requisição POST para http://example.com/produtos com os dados de um novo produto no corpo da requisição para que o servidor crie esse produto.

**3. PUT**

* Definição: O método PUT é utilizado para enviar dados ao servidor para criar ou substituir um recurso específico identificado por uma URI.
* Características:
  + É idempotente; múltiplas requisições PUT com os mesmos dados resultam no mesmo estado do servidor.
  + Utilizado geralmente para atualizar um recurso existente ou criar um recurso em uma URI específica.
* Exemplo de Uso:
  + Um cliente faz uma requisição PUT para http://example.com/produtos/123 com dados atualizados para modificar o produto com o ID 123.

**4. DELETE**

* Definição: O método DELETE é utilizado para solicitar que o servidor exclua um recurso identificado por uma URI.
* Características:
  + É idempotente; múltiplas requisições DELETE para o mesmo recurso resultam na exclusão (ou tentativa de exclusão) do mesmo recurso, sem efeitos adicionais.
* Exemplo de Uso:
  + Um cliente faz uma requisição DELETE para http://example.com/produtos/123 para excluir o produto com o ID 123 do servidor.

Esses métodos são fundamentais no desenvolvimento de APIs RESTful, permitindo que clientes e servidores interajam de maneira padronizada e previsível.

### **Framework Jersey**

**O que é:**

* Jersey é um framework de código aberto para o desenvolvimento de serviços web RESTful em Java. Ele é a implementação de referência do JAX-RS (Java API for RESTful Web Services), que é uma especificação da Java EE (Enterprise Edition) para criar serviços web baseados no estilo arquitetural REST.
* Jersey facilita o desenvolvimento de APIs RESTful ao fornecer abstrações de alto nível para lidar com requisições HTTP, mapeamento de URIs para classes Java, serialização e desserialização de objetos Java para formatos de dados como JSON e XML, entre outros recursos.

**Sua relação com Servlet:**

* Servlets são componentes Java que rodam em um servidor de aplicações, processando requisições HTTP e gerando respostas dinâmicas. Eles são a base para muitas tecnologias web em Java, incluindo frameworks como Jersey.
* Jersey, por sua vez, é construído sobre a API de Servlets. Ele usa Servlets para gerenciar o ciclo de vida das requisições HTTP. Quando um cliente faz uma requisição para um serviço RESTful criado com Jersey, a requisição é interceptada por um servlet (geralmente uma classe que estende javax.servlet.http.HttpServlet), que então delega o processamento da requisição para as classes e métodos apropriados definidos pelo desenvolvedor utilizando as anotações do JAX-RS.
* Em resumo, Jersey abstrai e simplifica a criação de APIs RESTful, enquanto os Servlets são a camada subjacente que lida com a comunicação HTTP. Jersey fornece uma maneira mais conveniente de trabalhar com serviços web RESTful sem precisar gerenciar diretamente a lógica de baixo nível dos Servlets.

**REFERÊNCIAS BIBLIOGRÁFICAS**

### REST, RESTful e URI

FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. 2000. 172 f. Tese (Doutorado em Filosofia) – Universidade da Califórnia, Irvine, 2000.

VIMERCATI, Sabrina de Capitani di et al. Security of Web Services: Standards and Research Issues. In: ACM Transactions on the Web (TWEB), 2005, v. 2, n. 2, p. 1-23.

### Métodos HTTP

FIELDING, Roy Thomas; GETTYS, Jim; MOGUL, Jeffrey C.; et al. Hypertext Transfer Protocol -- HTTP/1.1. Request for Comments: 2616. IETF, 1999. Disponível em: https://datatracker.ietf.org/doc/html/rfc2616. Acesso em: 14 ago. 2024.

MASSE, Mark. REST API Design Rulebook. 1. ed. Sebastopol: O'Reilly Media, 2011. 88 p.

### Framework Jersey

SUN MICROSYSTEMS. JAX-RS: Java API for RESTful Web Services - JSR 311. 2008. Disponível em: https://jcp.org/en/jsr/detail?id=311. Acesso em: 14 ago. 2024.

COWARD, George; LEHORS, Philippe; JORDAN, Bruce. Building RESTful Web Services with Java. 1. ed. Sebastopol: O'Reilly Media, 2010. 90 p.

### Servlets

HUNTER, Jason; CRAWFORD, William. Java Servlet Programming. 2. ed. Sebastopol: O'Reilly Media, 2001. 688 p.

COWARD, George; LEHORS, Philippe. Understanding Java Servlets. 1. ed. Sebastopol: O'Reilly Media, 2005. 112 p.

### Servidores de Aplicação Web

FARLEY, Jim. Java Distributed Computing. Sebastopol: O'Reilly Media, 1998. 484 p.

SUN MICROSYSTEMS. Java Servlet Specification. v. 2.5. 2005. Disponível em: https://jcp.org/aboutJava/communityprocess/final/jsr154/index.html. Acesso em: 14 ago. 2024.