REST

○ Definição

○ Como surgiu

○ Definição de RESTful

○ Definição de URI

REST (Representational State Transfer) é um estilo arquitetural que define um conjunto de princípios para projetar redes escaláveis e interconectadas. Ele foi introduzido por Roy Fielding em sua dissertação de doutorado em 2000. Fielding era um dos principais autores da especificação HTTP (Hypertext Transfer Protocol) e, portanto, suas ideias foram influenciadas pelo funcionamento desse protocolo.

O conceito de RESTful refere-se à aplicação dos princípios do REST na concepção e desenvolvimento de sistemas e serviços na web. Esses princípios incluem:

Cliente-Servidor: O sistema é composto por um cliente que faz requisições e um servidor que fornece recursos e serviços.

Sem Estado (Stateless): Cada requisição do cliente para o servidor deve conter toda a informação necessária para compreender e processar a requisição. Isso significa que o servidor não deve manter nenhuma informação do estado da sessão do cliente entre requisições.

Cacheável: As respostas do servidor devem ser explicitamente marcadas como cacheáveis ou não-cacheáveis. Isso ajuda a melhorar a eficiência, escalabilidade e desempenho da rede.

Interface Uniforme: Os recursos dentro do sistema são identificados por URIs (Uniform Resource Identifiers) e manipulados usando um conjunto uniforme e restrito de operações, como GET, POST, PUT e DELETE. Isso promove a simplicidade, escalabilidade e interoperabilidade.

Sistema em Camadas: O sistema é organizado em camadas hierárquicas, onde um cliente não precisa saber se está se comunicando diretamente com um servidor ou com um intermediário. Isso permite a escalabilidade e a segurança do sistema.

Uma URI (Uniform Resource Identifier) é uma cadeia de caracteres que identifica um recurso específico. Em outras palavras, é um identificador único que é usado para acessar um recurso na web. URIs são uma parte fundamental do funcionamento da web e são frequentemente usados em conjunto com os métodos HTTP para acessar e manipular recursos na web. Exemplos comuns de URIs incluem URLs (Uniform Resource Locators), que são um tipo de URI que especifica o local de um recurso na internet.

● Métodos HTTP

○ GET

○ POST

○ PUT

○ DELETE

Os métodos HTTP são os verbos que indicam a ação que deve ser realizada em um recurso específico. Aqui estão os métodos HTTP mais comuns:

GET: O método GET é usado para solicitar dados de um recurso específico no servidor. Ele é usado para recuperar informações e não deve causar efeitos colaterais no servidor. Por exemplo, ao digitar uma URL em um navegador, o navegador faz uma solicitação GET para recuperar a página da web associada à URL.

POST: O método POST é usado para enviar dados para o servidor para serem processados. Ele é frequentemente utilizado quando você está enviando dados de um formulário HTML para o servidor. Diferentemente do método GET, o método POST pode causar alterações no estado do servidor ou criar novos recursos.

PUT: O método PUT é usado para atualizar um recurso específico no servidor com os dados fornecidos na solicitação. Ele é utilizado para substituir completamente o conteúdo de um recurso existente ou criar um recurso novo se ele não existir.

DELETE: O método DELETE é usado para remover um recurso específico do servidor. Ele é utilizado para excluir o recurso identificado pela URI na solicitação.

Esses métodos HTTP, juntamente com outros como OPTIONS, HEAD, PATCH e outros, formam o conjunto de operações padronizadas que podem ser realizadas em recursos na web, seguindo os princípios do protocolo HTTP e do estilo arquitetural REST.

● Framework Jersey

○ O que é

○ Sua relação com Servlet

Jersey é um framework open source para desenvolvimento de serviços RESTful em Java. Ele fornece uma implementação robusta e eficiente das especificações JAX-RS (Java API for RESTful Web Services), que permite aos desenvolvedores criar APIs web de forma simples e escalável.

Em termos simples, o Jersey facilita a criação de serviços web RESTful em Java, abstraindo grande parte da complexidade subjacente e fornecendo uma estrutura coesa para lidar com solicitações HTTP, roteamento de URI, serialização e desserialização de dados, entre outras funcionalidades.

Quanto à sua relação com Servlet, é importante entender que Servlet é uma especificação do Java que lida com solicitações e respostas entre um cliente e um servidor. Os servlets são classes Java que são usadas para estender a capacidade de um servidor. Eles são usados para criar aplicativos web dinâmicos.

Jersey, por outro lado, pode ser implementado em cima do Servlet Container, como Tomcat ou Jetty. Ele usa os servlets Java subjacentes para manipular solicitações HTTP, mas fornece uma camada de abstração que simplifica o desenvolvimento de serviços RESTful.

Resumindo, Jersey não substitui Servlet, mas sim utiliza a funcionalidade dos servlets Java subjacentes para lidar com solicitações HTTP, enquanto fornece uma abstração de alto nível para facilitar o desenvolvimento de serviços web RESTful em Java.