Técnico em Desenvolvimento de Sistemas

Programação WEB 3

Introdução às tecnologias Web Services
SOA, SOAP, REStful, WSDL e UDDI

Prof. Laércio Silva

laercio.silva31@etec.sp.gov.br

lndsilva@Hotmail.com

O que é REST?

- Conceitualmente falando, o modelo REST (*REpresentational State Transfer*) representa nada mais que uma “nova” possibilidade para a criação de web services
- Cujas principais diferenças em relação ao modelo tradicional (SOAP) estão na utilização semântica dos
- Métodos HTTP (GET, POST, PUT e DELETE), na leveza dos pacotes de dados transmitidos na rede e na simplicidade, fazendo desnecessária a criação de camadas intermediárias para encapsular os dados.

O que é REST?

- No mundo REST, uma **requisição HTTP** é equivalente a uma chamada de um método (operação) em um objeto (recurso) residente no servidor.

Como principais características de uma requisição REST, podemos destacar:

- O método HTTP é utilizado para determinar a operação a ser realizada em um determinado recurso.
- Em geral, utiliza-se:
  - GET para recuperar,
  - POST para criar,
  - PUT para alterar e
  - DELETE para apagar;

- O recurso, por sua vez, é indicado na URL da requisição;
- Parâmetros podem ser passados na própria URL e/ou no corpo na requisição;
- Os tipos de dados utilizados na requisição e na resposta devem ser acordados entre o servidor e o cliente **JSON** e **XML** estão entre os tipos mais utilizados.

As fundações de REST

- O “marco zero” de REST e o **recurso**.
- Em REST, tudo e definido em termos de recursos, sendo estes os conjuntos de dados que são trafegados pelo protocolo.
- Os recursos são representados por URI’s.
- Note que, na web, URI’s e URL’s sao essencialmente a mesma coisa.

Qual a diferença entre uma URL e uma URI?

- URL significa Universal Resource Locator e URI, *Universal Resource Identifier*.
- Uma URI, como diz o próprio nome, pode ser utilizada para identificar qualquer coisa – dar um caminho para um determinado conteúdo, dar nome a este, etc.
- Já uma URL pode ser utilizada apenas para fornecer caminhos – sendo que uma URL e, portanto, uma forma de uma URI.
- E mais natural que URI's que não sejam URL's sejam utilizadas em outros contextos, como fornecimento de namespaces XML.

A URL utilizada para localização da listagem em varias partes:

http://localhost/cervejaria/clientes

- **http://** - Indica o protocolo que esta sendo utilizado (no caso, HTTP);
- **localhost:80** - Indica o servidor de rede que esta sendo utilizado e a porta (quando a porta não e especificada, assume-se que e a padrão - no caso do protocolo HTTP, 80);
- **cervejaria** - Indica o **contexto** da aplicação, ou seja, a raiz pela qual a aplicação esta sendo fornecida para o cliente. Vou me referir a esta, daqui em diante, como **contexto da aplicação** ou apenas **contexto**;
- **clientes** - E o endereço, de fato, do recurso - no caso, a listagem de clientes.
- Vou me referir a este, daqui em diante, como **endereço do recurso**.

O protocolo

- O protocolo, em realidade, não e uma restrição em REST - em teoria.
- Na prática, o HTTP e o único protocolo 100% compatível conhecido.
- Vale destacar que o HTTPS (HyperText Transfer Protocol over Secure Sockets Layer) não e uma variação do HTTP
- Mas apenas adição de uma camada extra - o que mantem o HTTPS na mesma categoria que o HTTP, assim como qualquer outro protocolo que seja utilizado sobre o HTTP, como SPDY.

O protocolo

- As possíveis causas para esta “preferência” podem ser apontadas:
- O autor do HTTP também e o autor de REST
- O protocolo HTTP e um dos mais utilizados no mundo (sendo que a web, de maneira geral, e fornecida por este protocolo).
- Estes fatos promoveriam uma aceitação grande e rápida absorção de REST pela comunidade de desenvolvedores.

A URL

- A URL escolhida deve ser única por recurso.
- Isto significa que, sempre que desejar obter a representação de todos os clientes, você deve utilizar a URL http://localhost/cervejaria/clientes .
- Realizar uma consulta nesta URL pode retornar dados da seguinte maneira:

```
<clientes>
        <cliente id="1">
                <nome>João</nome>
                <dataNascimento>1999-12-01</dataNascimento>
        </cliente>
        <cliente id="2">
                <nome>Gustavo</nome>
                <dataNascimento>2000-11-01</dataNascimento>
        </cliente>
</clientes>
```

- Se você desejar, portanto, retornar um cliente especifico, você deve utilizar uma URL diferente.
- Suponha, por exemplo, que você deseja retornar o cliente com id igual a 1. Neste caso, a URL seria http://localhost/cervejaria/clientes/1

```
<cliente id="1">
        <nome>João</nome>
        <dataNascimento>1999-12-01</dataNascimento>
</cliente>
```

O que são recursos REST

- Um recurso pode ser definido como um elemento vital a ser referenciado em um sistema cliente-servidor.
- A arquitetura REST trata todo o seu conteúdo como um recurso, que inclui páginas Html, imagens, arquivos de texto, vídeos, etc.
- O acesso aos recursos é fornecido pelo servidor REST onde o cliente REST é usado para acessar e também modificar os recursos.
- Todos os seus recursos são identificados por meio de URI, que é abreviado como Uniform Resource Identifier.

Representação de recursos REST

- Existem várias formas de representação de recursos REST, como XML, baseado em texto, JSON, etc.
- As duas formas mais populares de representação de recursos são o uso de JSON e XML. REST não impõe nenhuma restrição para um formato específico na representação de recursos em REST.
- Tudo depende da escolha e exigência do projeto e do desenvolvedor.
- Portanto, você pode usar qualquer formato para representar recursos em REST.

O exemplo a seguir é mostrado no formato XML que mostra as informações do perfil do usuário:

```
<root>
   <Data>
      <UserGUID>4f1764fe-b3d3-ce7c-0f08-0ef09d834b7e</UserGUID>
      <FullName>Alex</FullName>
      <ProfilePic>http://localhost/9c0977f4-877a-8138-4fbb-a524edf20437.jpg</ProfilePic>
   </Data>
   <Message>Success</Message>
   <ResponseCode>200</ResponseCode>
</root>
```

O mesmo exemplo de recurso REST agora é mostrado usando o formato

JSON:


```
{
  "ResponseCode": 200,
  "Message": "Success",
  "Data": {
    "UserGUID": "4f1764fe-b3d3-ce7c-0f08-0ef09d834b7e",
    "FullName": "Alex",
    "ProfilePic": "http:\/\/localhost\/9c0977f4-877a-8138-4fbb-a524edf20437.jpg"
  }
}
```

Desvantagem com a formatação do XML

Existem algumas desvantagens com XML para formatar os recursos REST. Esses são:

- **Falta de tipo de dados** : os elementos XML não expressam o tipo de dados ou a ordem de empregá-los. Para isso, seria necessário utilizar XML Schema.
- **Falta de listas** : Não há como expressar nativamente as listas. Isso torna confuso se um determinado elemento é uma lista ou um objeto.

Para o desenvolvimento de recursos em REST

Alguns pontos fundamentais precisam ser mantidos em mente ao construir um recurso REST. Esses são:

- **Completude de significado** : seu formato deve representar completamente o recurso que você pretende criar. O recurso pode ser simples ou complexo ou aninhado, que compreende um recurso contendo outro recurso.

- **Estrutura compreensível** : Você deve enquadrar o recurso REST de forma que possa ser compreensível tanto pelo servidor quanto pelo cliente, para que o REST possa utilizar o formato do seu recurso.
- **Capacidade de link** : pode haver algumas situações em que seu REST pode ter um recurso que está vinculado a outro recurso. Em tais situações, você deve gerenciá-lo e estruturá-lo adequadamente.

Recursos básicos de HTTP para REST

O HTTP é um protocolo que nos permite enviar e receber arquivos pela internet, utilizando uma arquitetura cliente-servidor e é mais fácil de monitorar. Os recursos essenciais do HTTP são:

- HTTP é um protocolo sem conexão.
- O HTTP não tem estado porque o servidor e o cliente controlam a última solicitação.
- Todos os dados podem ser transmitidos e recebidos por meio do protocolo HTTP, portanto, independente do tipo.

Recursos básicos de HTTP para REST

Para identificar quaisquer recursos implementados usando REST ou em geral, bem como estabelecer qualquer conexão, o HTTP faz uso do **Uniform Resource Identifier (URI)** .

A solicitação e a resposta de HTTP consistem nos quatro itens a seguir:

- Uma linha de partida
- Um ou mais cabeçalhos
- Uma linha em branco que indica o término do(s) campo(s) do cabeçalho
- E, finalmente, um corpo de mensagem opcional

O que é HATEOS ?

- O termo HATEOAS é abreviado como "Hipermídia como o mecanismo do estado do aplicativo" é um atributo do REST que limita a construção do aplicativo, de modo que sua maneira de projetar a arquitetura RESTful parece única em relação a todas as outras arquiteturas de aplicativos de rede diferentes.
- O HATEOAS fornece uma conexão através da hipermídia.

- A frase "hipermídia" pode ser definida como qualquer conteúdo que contenha conexões com vários outros tipos de mídia, como imagens, texto, videoclipes e também filmes.
- O REST fornece uma abordagem arquitetônica que permitirá que você use links hipermídia nesses conteúdos de mídia para que seus clientes possam navegar para todos os recursos adequados em tempo de execução à medida que atravessam esses links hipermídia.
- O conceito é o mesmo para navegação em páginas da web por meio de hiperlinks exatos para alcançar os recursos desejados.

- Exemplo, usando a resposta JSON, API como HTTP GET pode ser formada:

http://api.domain_name.com/manage/designations/20


```
{
  "designationID": 20,
  "designationName": "Tech-writer",
  "location": 346,
  "managerID": 180,
  "links": [
    {
      "href": "/demoApp/employees/20",
      "rel": "employees",
      "type": "GET"
    }
  ]
}
```

Outra representação:

- Um meio simples de fornecer tantos links conectados quanto possível em relação à origem é usar a resposta.
- Aqui está um caso de pedido de registro para um novo funcionário em uma organização.

Outra representação:

- A resposta usual dos detalhes do funcionário terá um caminho:

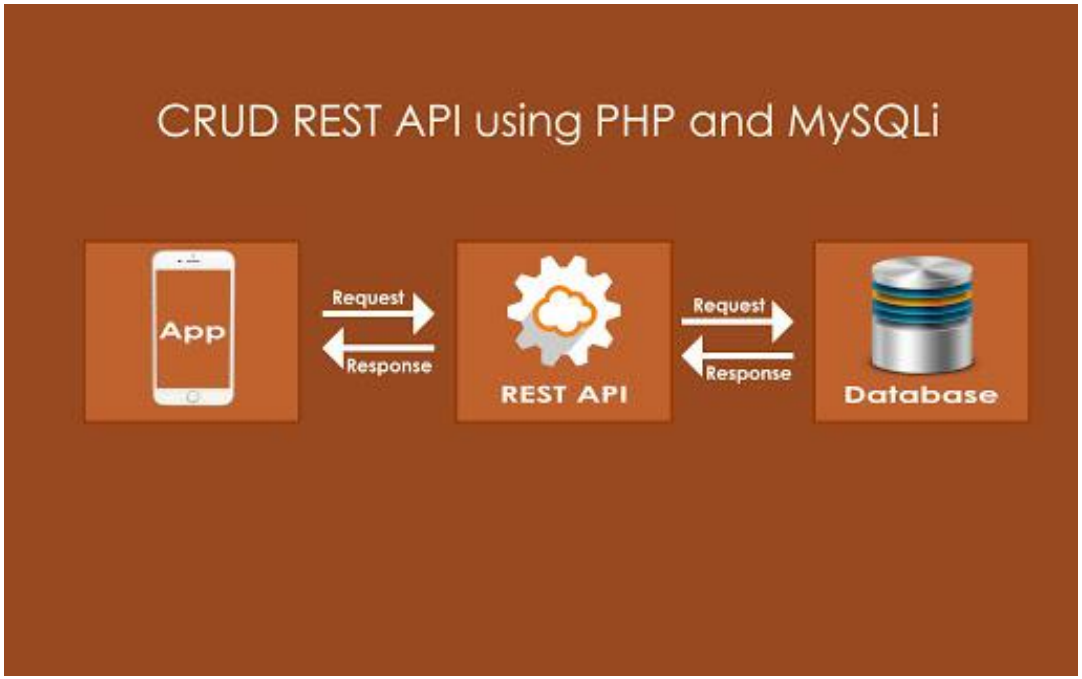
/demoApp/workers/ 06


```
{
    "id": "10",
    "fName": "Dennis",
    "lName": "Ritchie",
    "age": "46"
}
```

No caso de você fornecer qualquer informação extra ao cliente, vamos supor, para qual perfil de trabalho essa pessoa está sendo contratada ou inscrita, o link adicional será algo como este:

```
{
   "id": "10",
   "fName": "Dennis",
   "lName": "Ritchie",
   "age": "46",
   "link": {
      "rel": "designation",
      "href": "/demoApp/employees/6/designations"
   }
}
```

HATEOAS é uma percepção ou modelo para fornecer links dos sub-recursos associados para o recurso principal solicitado por qualquer cliente, o que torna mais fácil construir chamadas adicionadas à sua API REST.



Atividade 2 – Rest API

- Desenvolver uma pesquisa **sobre como a representação HATEOS é importante para o protocolo de serviços Rest API**.
    - Para complementar sua pesquisa você deverá colocar os locais de referência das pesquisas realizadas, com data da pesquisa e fonte ou nome de quem escreveu.

Muito Obrigado

Até a Próxima Aula

Prof. Laércio Silva
Email: lndsilva@hotmail.com