


Guia de nomenclatura de recursos REST

Última atualização :23 de novembro de 2021  Por: Lokesh Gupta

1. O que é um Recurso?

Em REST, a representação de dados primária é chamada de **recurso**. Ter uma estratégia de nomenclatura de recursos REST consistente e robusta – provará ser uma das melhores decisões de design a longo prazo.

A abstração chave de informações em REST é um recurso. Qualquer informação que possa ser nomeada pode ser um recurso: um documento ou imagem, um serviço temporal (por exemplo, “tempo de hoje em Los Angeles”), uma coleção de outros recursos, um objeto não virtual (por exemplo, uma pessoa) e assim por diante. sobre.

Em outras palavras, qualquer conceito que possa ser alvo de referência de hipertexto de um autor deve se encaixar na definição de um recurso.

Um recurso é um mapeamento conceitual para um conjunto de entidades, não a entidade que corresponde ao mapeamento em um determinado momento.

— *Dissertação de Roy Fielding*

1.1. Recursos de Singleton e Coleção

Um **recurso** pode ser um **singleton** ou uma **coleção**.

Por exemplo, “**customers**” é um recurso de coleção e “**customer**” é um recurso singleton (em um domínio bancário).

Podemos identificar **customers** recurso de coleção “ ” usando o URI “**/customers**”. Podemos identificar um único **customer** recurso “ ” usando o URI

“ `/customers/{customerId}`”.

1.2. Recursos de coleção e subcoleção

Um **recurso também pode conter recursos de subcoleção** .

Por exemplo, o recurso de subcoleção “ `accounts`” de um determinado “ `customer`” pode ser identificado usando o URN “ `/customers/{customerId}/accounts`” (em um domínio bancário).

Da mesma forma, um recurso singleton “ `account`” dentro do recurso de subcoleção “ `accounts`” pode ser identificado da seguinte forma: “ `/customers/{customerId}/accounts/{accountId}`”.

1.3. URI

As APIs REST usam Identificadores Uniformes de Recursos (URIs) para endereçar recursos. Os designers da API REST devem criar URIs que transmitam o modelo de recursos de uma API REST para os clientes em potencial da API. Quando os recursos são bem nomeados, uma API é intuitiva e fácil de usar. Se for mal feita, essa mesma API pode ser um desafio para usar e entender.

A restrição de uma interface uniforme é parcialmente abordada pela combinação de URIs e verbos HTTP e usando-os de acordo com os padrões e convenções.

Abaixo estão algumas dicas para você começar a criar os URIs de recursos para sua nova API.

2. Práticas recomendadas

2.1. Use substantivos para representar recursos

O URI RESTful deve se referir a um recurso que é uma coisa (substantivo) em vez de se referir a uma ação (verbo) porque os substantivos têm propriedades que os verbos não têm – da mesma forma, os recursos têm atributos. Alguns exemplos de recurso são:

- Usuários do sistema
- Contas de usuário
- Dispositivos de rede etc

e seus URIs de recursos podem ser projetados como abaixo:

```
http://api.example.com/device-management/managed-devices
http://api.example.com/device-management/managed-devices/{device-id}
http://api.example.com/user-management/users
http://api.example.com/user-management/users/{id}
```

Para maior clareza, vamos dividir os **arquétipos de recursos** em quatro categorias (documento, coleção, armazenamento e controlador). Então seria melhor se **você sempre colocasse um recurso em um arquétipo e então usasse sua convenção de nomenclatura de forma consistente** .

Por uma questão de uniformidade, resista à tentação de projetar recursos que sejam híbridos de mais de um arquétipo.

2.1.1. documento

Um recurso de documento é um conceito singular que é semelhante a uma instância de objeto ou registro de banco de dados.

No REST, você pode visualizá-lo como um único recurso dentro da coleção de recursos. A representação de estado de um documento normalmente inclui campos com valores e links para outros recursos relacionados.

Use o nome “singular” para denotar o arquétipo do recurso do documento.

```
http://api.example.com/device-management/managed-devices/{device-id}
http://api.example.com/user-management/users/{id}
http://api.example.com/user-management/users/admin
```

2.1.2. coleção

Um recurso de coleção é um diretório de recursos gerenciado pelo servidor.

Os clientes podem propor novos recursos a serem adicionados a uma coleção. No entanto, cabe ao recurso de coleção optar por criar ou não um novo recurso.

Um recurso de coleção escolhe o que deseja conter e também decide os URIs de cada recurso contido.

Use o nome “plural” para denotar o arquétipo do recurso de coleção.

```
http://api.example.com/device-management/managed-devices  
http://api.example.com/user-management/users  
http://api.example.com/user-management/users/{id}/accounts
```

2.1.3. armazenar

Uma loja é um repositório de recursos gerenciado pelo cliente. Um recurso de armazenamento permite que um cliente de API insira recursos, recupere-os e decida quando excluí-los.

Uma loja nunca gera novos URIs. Em vez disso, cada recurso armazenado tem um URI. O URI foi escolhido por um cliente quando o recurso o colocou inicialmente na loja.

Use o nome “plural” para denotar o arquétipo do recurso de armazenamento.

```
http://api.example.com/song-management/users/{id}/playlists
```

2.1.4. controlador

Um recurso de controlador modela um conceito de procedimento. Os recursos do controlador são como funções executáveis, com parâmetros e valores de retorno, entradas e saídas.

Use “verbo” para denotar o arquétipo do controlador.

```
http://api.example.com/cart-management/users/{id}/cart/checkout  
http://api.example.com/song-management/users/{id}/playlist/play
```

2.2. Consistência é a chave

Use convenções de nomenclatura de recursos consistentes e formatação de URI para ambiguidade mínima e legibilidade e manutenção máximas. Você pode implementar as dicas de design abaixo para obter consistência:

2.2.1. Use barra (/) para indicar relacionamentos hierárquicos

O caractere de barra (/) é usado na parte do caminho do URI para indicar um relacionamento hierárquico entre os recursos. por exemplo

```
http://api.example.com/device-management
http://api.example.com/device-management/managed-devices
http://api.example.com/device-management/managed-devices/{id}
http://api.example.com/device-management/managed-devices/{id}/scripts
http://api.example.com/device-management/managed-devices/{id}/scripts/{id}
```

2.2.2. Não use barra à direita (/) em URIs

Como o último caractere dentro do caminho de um URI, uma barra (/) não adiciona nenhum valor semântico e pode confundir. É melhor retirá-lo do URI.

```
http://api.example.com/device-management/managed-devices/
http://api.example.com/device-management/managed-devices /*This is much better
version*/
```

2.2.3. Use hífen (-) para melhorar a legibilidade dos URIs

Para facilitar a leitura e interpretação de seus URIs, use o caractere de hífen (-) para melhorar a legibilidade dos nomes em segmentos de caminho longo.

```
http://api.example.com/device-management/managed-devices/
http://api.example.com/device-management/managed-devices /*This is much better
version*/
```

2.2.4. Não use sublinhados (_)

É possível usar um sublinhado no lugar de um hífen para ser usado como separador – Mas dependendo da fonte do aplicativo, é possível que o caractere sublinhado (_) fique parcialmente obscurecido ou completamente oculto em alguns navegadores ou telas.

Para evitar essa confusão, use hífen (-) em vez de sublinhados (_).

```
http://api.example.com/inventory-management/managed-entities/{id}/install-script-location
//More readable

http://api.example.com/inventory-management/managedEntities/{id}/installScriptLocation
//Less readable
```

2.2.5. Usar letras minúsculas em URIs

Quando conveniente, as letras minúsculas devem ser consistentemente preferidas nos caminhos de URI.

```
http://api.example.org/my-folder/my-doc //1
HTTP://API.EXAMPLE.ORG/my-folder/my-doc //2
http://api.example.org/My-Folder/my-doc //3
```

Nos exemplos acima, 1 e 2 são iguais, mas 3 não é, pois usa **My-Folder** em letras maiúsculas.

2.3. Não use extensões de arquivo

As extensões de arquivo parecem ruins e não adicionam nenhuma vantagem. Removê-los também diminui o comprimento dos URIs. Não há razão para mantê-los.

Além do motivo acima, se você deseja destacar o tipo de mídia da API usando a extensão de arquivo, deve confiar no tipo de mídia, conforme comunicado por meio do **Content-Type** cabeçalho, para determinar como processar o conteúdo do corpo.

```
http://api.example.com/device-management/managed-devices.xml /*Do not use it*/
http://api.example.com/device-management/managed-devices /*This is correct URI*/
```

2.4. Nunca use nomes de função CRUD em URIs

Não devemos usar URIs para indicar uma função CRUD. Os URIs devem ser usados apenas para identificar exclusivamente os recursos e não qualquer ação sobre eles.

Devemos usar métodos de solicitação HTTP para indicar qual função CRUD é executada.

```
HTTP GET http://api.example.com/device-management/managed-devices //Get all devices
HTTP POST http://api.example.com/device-management/managed-devices //Create new Device

HTTP GET http://api.example.com/device-management/managed-devices/{id} //Get device for given Id
HTTP PUT http://api.example.com/device-management/managed-devices/{id} //Update device for given Id
```

```
HTTP DELETE http://api.example.com/device-management/managed-devices/{id}  
//Delete device for given Id
```

2.5. Use o componente de consulta para filtrar a coleção de URI

Muitas vezes, você encontrará requisitos em que precisará de uma coleção de recursos classificados, filtrados ou limitados com base em algum atributo de recurso específico.

Para este requisito, não crie novas APIs – em vez disso, habilite os recursos de classificação, filtragem e paginação na API de coleta de recursos e passe os parâmetros de entrada como parâmetros de consulta. por exemplo

```
http://api.example.com/device-management/managed-devices  
http://api.example.com/device-management/managed-devices?region=USA  
http://api.example.com/device-management/managed-devices?region=USA&brand=XYZ  
http://api.example.com/device-management/managed-devices?  
region=USA&brand=XYZ&sort=installation-date
```

