Serviços REST e SOAP

(Web Services)

A Representational State Transfer (REST), em português Transferência de Estado Representacional, é uma abstração da arquitetura da World Wide Web (WWW), mais precisamente, é um estilo arquitetural que consiste de um conjunto coordenado de restrições arquiteturais aplicadas a componentes, conectores e elementos de dados dentro de um sistema de hipermídia distribuído.

O REST ignora os detalhes da implementação de componente e a sintaxe de protocolo com o objetivo de focar nos papéis dos componentes, nas restrições sobre sua interação com outros componentes e na sua interpretação de elementos de dados significantes.

Ele foi definido oficialmente pela W3C.

Ele é frequentemente aplicado à web services fornecendo APIs para acesso a um serviço qualquer na web. Ele usa integralmente as mensagens HTTP para se comunicar através do que já é definido no protocolo sem precisar "inventar" novos protocolos específicos para aquela aplicação.

Trabalha essencialmente com componentes, conectores e dados.

- Ele usa o protocolo HTTP (verbos, accept headers, códigos de estado HTTP, Content-Type) de forma explícita e representativa para se comunicar. URIs são usados para expor a estrutura do serviço. Utiliza uma notação comum para transferência de dados como XML ou JSON.
- Não possui estado entre essas comunicações, ou seja, cada comunicação é independente e uniforme (padronizada) precisando passar toda informação necessária.
- Ele deve facilitar o cache de conteúdo no cliente.
- Deve ter clara definição do que faz parte do cliente e do servidor. O cliente não precisa saber como o servidor armazena dados, por exemplo. Assim cada implementação não depende da outra e se torna mais escalável.
- Permite o uso em camadas também facilitando a escalabilidade, confiabilidade e segurança.

Convenhamos que isto pode servir para CRUD, mas existem tantas variações do que precisa ser feito que não é possível representar tudo só com os verbos HTTP. Ok, é possível fazer tudo parecer CRUD, mas talvez exija informações adicionais em nome do formalismo.

Tudo isto é pensado para proporcionar melhor performance, escalabilidade, simplicidade, flexibilidade, visibilidade, portabilidade e confiabilidade.
Cada um define como quiser sua API, ao contrário de SOAP onde tudo é definido oficialmente.

Pra que serve?

Serve para expor serviços http que é um protocolo de padrão aberto, trazendo assim a possibilidade de uso para qualquer tipo que aplicação cliente que possa efetuar requisições http.

Soap vs REST

Ambos possuem vantagens e desvantagens e fica a cargo do desenvolvedor determinar a melhor abordagem para cada caso em particular.

SOAP - (Simple Object Access Protocol, em português Protocolo Simples de Acesso a Objetos) é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída. Ele se baseia na Linguagem de Marcação Extensível (XML) para seu formato de mensagem, e normalmente baseia-se em outros protocolos da Camada de aplicação, mais notavelmente em Chamada de Procedimento Remoto (RPC) e Protocolo de Transferência de Hipertexto (HTTP), para negociação e transmissão de mensagens.

- É um padrão que combinado às especificações WS-* podem garantir questões de QoS(Quality of Service), Segurança, transação e outras questões presentes em integrações mais complexas.
- Uma mensagem SOAP pode ser propagada por diferentes protocolos, o que flexibiliza bastante várias integrações.
- É um padrão que está muito maduro no mercado, qualquer ferramenta de integração e Framework tem várias funcionalidades para manipular as mensagens que seguem este padrão.

REST - É simples de entender e pode ser adotado em praticamente qualquer cliente ou servidor com suporte a HTTP/HTTPS.
Suas principais vantagens são a facilidade no desenvolvimento, o aproveitamento da infra-estrutura web existente e um esforço de aprendizado pequeno.

- É mais elegante, pois utiliza ao máximo o protocolo HTTP, evitando a construção de protocolos adicionais;
- Tem o potencial de ser bem mais simples que uma implementação com WSDL/SOAP;
- Tende a ser mais performático;
- Mais ou menos 80% das integrações utilizam o protocolo HTTP;
- A possibilidade de ter diversas representações de um mesmo recurso, por exemplo, uma dada entidade pode ser representada em diferentes formatos como Json, xml, html e text/plain, dependendo da requisição feita pelo cliente(Content-Negotiation)
- Possibilidade de navegar entre relacionamentos (Links web) de vários recursos de forma dinâmica. seguindo a usabilidade de qualquer sistema web.

Curiosidades

- Ambas as tecnologias podem ser misturadas e combinadas. O REST é fácil de entender e extremamente acessível porém faltam padrões, e a tecnologia é considerada apenas uma abordagem arquitetural.
- Em comparação, o SOAP é um padrão da indústria, com protocolos bem definidos e um conjunto de regras bem estabelecidas.

Onde o REST funciona bem ?

- Situações em que há limitação de recursos e de largura de banda: A estrutura de retorno é em qualquer formato definido pelo desenvolvedor e qualquer navegador pode ser usado. Isso porque a abordagem REST usa o padrão de chamadas GET, PUT, POST e DELETE. O REST também pode usar objetos XMLHttpRequest (a base do velho AJAX) que a maioria dos navegadores modernos suportam.
- Operações totalmente sem-estado: se uma operação precisa ser continuada, o REST não será a melhor opção. No entanto, se forem necessárias operações de CRUD stateless (Criar, Ler, Atualizar e Excluir), o REST seria a melhor alternativa.
- Situações que exigem cache: se a informação pode ser armazenada em cache, devido à natureza da operação stateless do REST, esse seria um cenário adequado para a tecnologia.

Onde o SOAP funciona bem?

- Processamento e chamada assíncronos: se o aplicativo precisa de um nível garantido de confiabilidade e segurança para a troca de mensagens, então o SOAP 1.2 oferece padrões adicionais para este tipo de operação como por exemplo o WSRM (WS-Reliable Messaging).

- Contratos formais: se ambos os lados (fornecedor e consumidor) têm que concordar com o formato de intercâmbio de dados, então o SOAP 1.2 fornece especificações rígidas para esse tipo de interação.
- Operações stateful: para o caso de o aplicativo precisar de informação contextual e gerenciamento de estado com coordenação e segurança, o SOAP 1.2 possui uma especificação adicional em sua estrutura que apoia essa necessidade (segurança, transações, coordenação etc.). Comparativamente, usar o REST exigiria que os desenvolvedores construírem uma solução personalizada.

Os verbos e o uso de cada um

Get (Http Get) - Usado comumente para efetuar requisições com retorno de dados.

GET **http://example.com/cliente/1** - O exemplo acima utiliza o verbo GET, para retornar um cliente com ID de valor 1(um);

Post (Http Post) - Usado para efetuar requisições que tem necessidade de um envio maior ou mais complexo de dados na requisição, pode receber retorno de dados ou não.

POST **http://example.com/cliente**

Corpo da requisição:

{"id":"0", "nome","teste"}

Neste exemplo, uma requisição com verbo POST informa o objeto JSON no corpo da requisição que deverá ser gravado.

Put (Http Put) - Usado para fazer alterações de dados, os dados da requisição também são enviadas no corpo;

PUT **http://example.com/cliente/1**

Corpo da requisição:

{"id":"1", "nome","teste alterando"}

Delete (Http Delete) - Assim como o nome do verbo sugere, é usado para efetuar exclusões de dados.

DELETE **http://example.com/cliente/1**

Repare que a URI acima tem exatamente o mesmo formato, porém é requisitada com verbo diferente, o DELETE, desta forma o servidor entenderá que deverá ser feita a exclusão do registro com o ID de valor 1(valor);

A conclusão é simples, cada um dos protocolos tem suas vantagens e desvantagens, deve ser analisado o cenário ao utilizar cada uma delas.