SSC-0158
Computação em Nuvem

Aula 05 – Restfull Web Services

Prof. Julio Cezar Estrella

jcezar@icmc.usp.br

Créditos

Os slides integrantes deste material foram construídos a partir dos conteúdos relacionados às referências bibliográficas descritas neste documento.

Conceitos

- É como os clientes acessam um determinado serviço. Normalmente, um serviço usará SOAP, mas se você criar um serviço REST, os clientes acessarão seu serviço com um estilo arquitetural diferente (chamadas, serialização como JSON etc.).

Conceitos

- O REST usa alguns métodos HTTP comuns para inserir / excluir / atualizar / recuperar as informações como descrevemos abaixo:
    - ***GET*** *- Solicita uma representação específica de um recurso*
    - ***PUT*** *- Cria ou atualiza um recurso com a representação fornecida*
    - ***DELETE*** *- Exclui o recurso especificado*
    - ***POST*** *- Envia dados a serem processados pelo recurso identificado*

O que é REST?

- **RE**presentational **S**tate **T**ransfer
  - Doutorado de Roy Fielding
- A Web é a aplicação de maior sucesso na Internet
  - O que torna a Web tão bem-sucedida?
    - Recursos Endereçáveis

O que é REST?

- ▫ Toda “coisa” deve ter um ID
  - ▫ Toda “coisa” deve ter um URI
- **Interface restrita**
  - ▫ Usa os métodos padrão do protocolo
  - ▫ HTTP: GET, POST, PUT, DELETE
- **Recursos com várias representações**
  - Aplicações diferentes precisam de formatos diferentes
  - Plataformas diferentes precisam de representações diferentes (XML + JSON)

O que é REST?

- Comunicação sem Estado
    - Escala de aplicação sem estado

RESTFULL

- Todo objeto possui uma URI
- De um URI, sabemos
    - O protocolo (como nos comunicamos)
    - O host / porta (onde está na rede)
    - O caminho do recurso (com que recurso estamos nos comunicando

Descrição de uma URI

http://sales.com/customers/323421

/customers/{customer-id}

- Leitura por humanos: Desejado, mas não mandatório
- Parâmetros da URI

http://sales.com/customers?zip=49009

- Parâmetros de consulta para encontrar outros recursos

http://sales.com/cars/mercedes/amg/e55;color=black

- Parâmetros da matriz para definir atributos de recursos

Implicações de uma Interface Uniforme

- **Intuitivo**
    - Você sabe quais operações o recurso suportará
- **Comportamento previsível**
    - **GET** - somente leitura e idempotente. Nunca altera o estado do recurso
    - **PUT -** uma inserção ou atualização idempotente de um recurso. Idempotente porque é repetível sem efeitos colaterais
    - **DELETE** - remoção de recurso e idempotente.
    - **POST** - não-idempotente, operação "vale tudo“
- **Clientes, desenvolvedores, administradores, operações sabem o que esperar**
    - Muito mais fácil para os administradores atribuir funções de segurança
    - Para mensagens idempotentes, os clientes não precisam se preocupar com mensagens duplicadas.

REST

- "O Representational State Transfer visa evocar uma imagem de como um aplicativo Web bem projetado se comporta: uma rede de páginas da Web (uma máquina de estado virtual), onde o usuário progride através de um aplicativo selecionando links (transições de estado), resultando em a próxima página (representando o próximo estado do aplicativo) sendo transferida para o usuário e renderizada para uso."

Porque REST?

- Menos sobrecarga (sem envelope SOAP para encerrar todas as chamadas)
- Menos duplicação (o HTTP já representa operações
- como DELETE, PUT, GET, etc., que precisam ser representados em um envelope SOAP).
- Mais padronizado - as operações HTTP são bem compreendidas e operam de forma consistente. Algumas implementações de SOAP podem ficar complicadas.

Porque RESTFULL?

- Mais legível e testável por humanos (mais difícil de testar o SOAP com apenas um navegador).
- Não é necessário usar XML (bem, você também não precisa usar SOAP, mas isso dificilmente faz sentido, pois você já está analisando o envelope).
- As bibliotecas tornaram o SOAP (mais ou menos) fácil. Mas você está abstraindo muita redundância, como observei. Sim, em teoria, o SOAP pode repassar outros transportes, para evitar que suba em uma camada fazendo coisas semelhantes, mas, na realidade, quase todo o trabalho SOAP que você fará é sobre HTTP.

A idéia por traz de REST

- Simplicidade é melhor
- A Web funciona e muito bemworks and works well
- Os serviços web devem seguir o estilo da Web

Serviços RESTFULL

- **Recursos como URI**
    - Utilize URI exclusivo para referenciar todos os recursos em sua API
- **Operações como métodos HTTP**
    - GET – Consultas
    - POST – Consultas
    - PUT, DELETE - Inserir, atualizar e excluir
- **Conexão e descoberta**
    - Como a Web, as respostas HTTP contêm links para outros recursos

Exemplo de REST API

URL	http://del.icio.us/api/[username]/book marks/	
Method	GET	
Querystring	tag=	Filter by tag
	dt=	Filter by date
	start=	The number of the first bookmark to return
	end=	The number of the last bookmark to return
Returns	200 OK & XML (delicious/bookmarks+xml)	
	401 Unauthorized	
	404 Not Found	

Exemplo de REST API

URL	http://del.icio.us/api/[username]/bookmarks/
Method	POST
Request Body	XML (delicious/bookmark+xml)
Returns	201 Created & Location
	401 Unauthorized
	415 Unsupported Media Type

Exemplo de REST API

URL	http://del.icio.us/api/[username]/bookmarks/[hash]
Method	DELETE
Returns	204 No Content
	401 Unauthorized
	404 Not Found

O desenho de um recurso com interface uniforme

- Ná duvida, defina um novo recurso
- */orders*
  - *GET - list all orders*
  - *POST - submit a new order*
- */orders/{order-id}*
  - *GET - get an order representation*
  - *PUT - update an order*
  - *DELETE - cancel an order*
- */orders/average-sale*
  - *GET - calculate average sale*
- */customers*
  - *GET - list all customers*
  - *POST - create a new customer*
- */customers/{cust-id}*
  - *GET - get a customer representation*
  - *DELETE- remove a customer*
- */customers/{cust-id}/orders*
  - *GET - get the orders of a customer*

JSON

- Notação de Objeto JavaScript
- Sintaxe leve para representar dados
- Mais fácil de analisar o código do cliente JavaScript
- Alternativa ao XML em aplicativos AJAX

[{"Email":"bob@example.com","Name":"Bob"},{"Email
":"mark@example.com","Name":"Mark"},{"Email":
"j ohn@example.com","Name":"John"}]

Recursos com Múltiplas Representações

- Cabeçalhos HTTP gerenciam essa negociação
    - CONTENT-TYPE: especifica o tipo MIME do corpo da mensagem
    - ACCEPT: lista delimitada por vírgula de um ou mais tipos MIME que o cliente gostaria de receber como resposta
    - No exemplo a seguir, o cliente está solicitando uma representação do cliente no formato xml ou json

GET /customers/33323

ACCEPT: application/xml,application/json

Recursos com Múltiplas Representações

- Preferências são suportadas e definidas pela especificação do protocolo HTTP

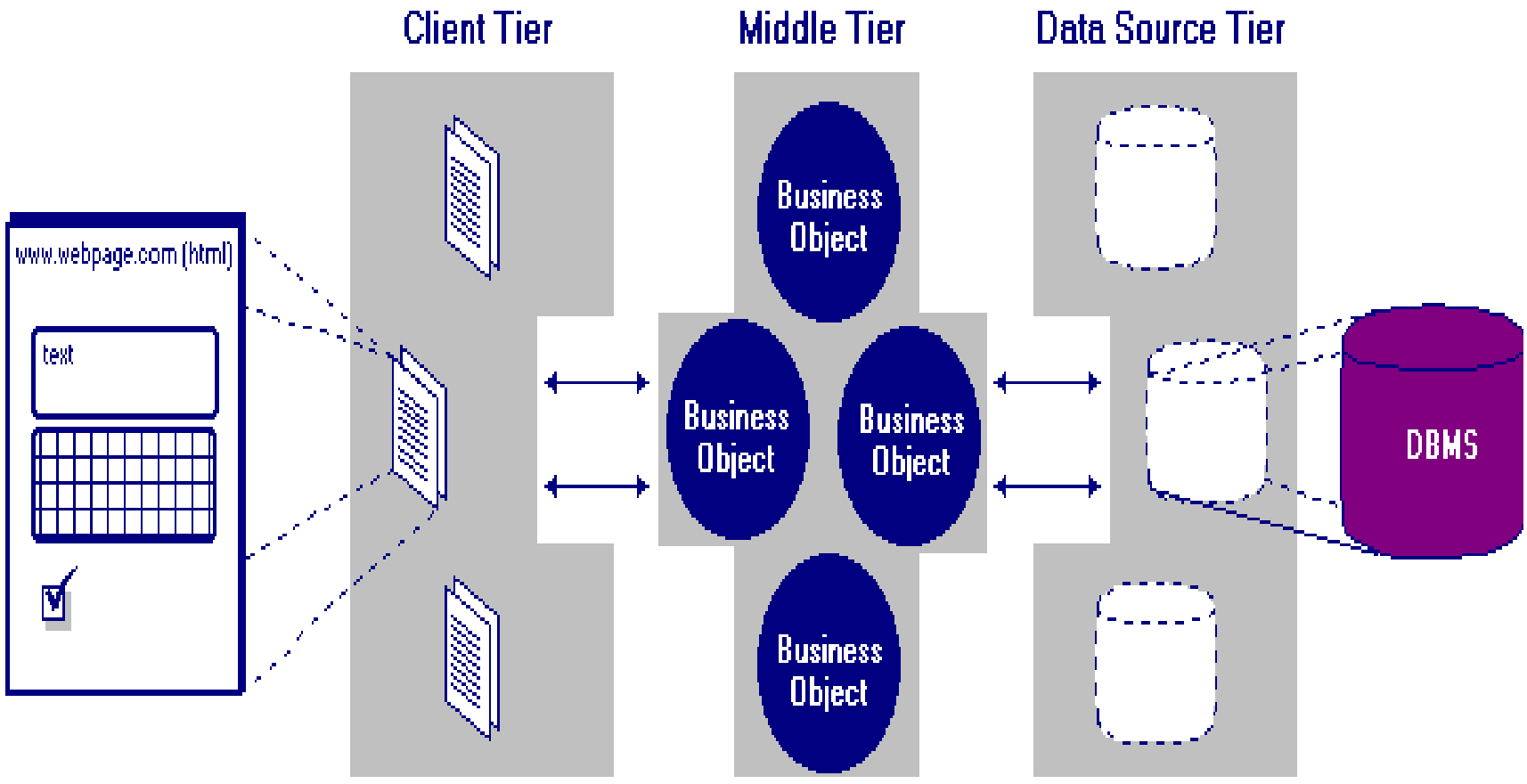
GET /customers/33323 ACCEPT: text/html;q=1.0, application/json;q=0.5;application/xml;q=0.7

Publicar e Consumir Serviços REST

- Facebook Graph API
- Google Custom Search API

Arquitetura de N Camadas

- Os serviços da Web baseados em SOAP e REST permitem que a arquitetura de três camadas seja estendida em n camadas.



Referências

- https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Dúvidas