Web Service - SOAP/ REST

Sistemas Distribuídos e Mobile
Prof. Gustavo Torres Custodio
gustavo.custodio@anhembi.br

O que é um Web Service?

- Um servidor sendo executado, esperando por conexões em uma porta específica.
  - Como sistemas cliente servidor que realizam troca de mensagens.
  - Oferecidos por meio de programas, banco de dados, etc.
- Web services possuem um conjunto de características que os definem.

Características de um Web Service

- Baseado em XML:
  - XML utilizado para representação e transporte de dados, evitando processos que prejudiquem a interoperabilidade.
- Baixo acoplamento:
  - Capacidade de mudar a interface sem que os clientes precisem mudar suas aplicações.
- Suporte a diferentes tipos de documentos:
  - Como são baseados em XML, Web Services permitem representar documentos complexos.

Características de um Web Service

- Síncrono / Assíncrono
  - Síncrono faz com que o cliente espere o serviço dar uma resposta antes de prosseguir.
  - Assíncrono permite que o cliente invoque o serviço e continue suas tarefas anteriores.
- Suporte para RPC:
  - Permite realizar chamadas remotas de métodos.

Características de um Web Service

- Suporte a diferentes tipos de documentos:
  - Como são baseados em XML, Web Services devem ser flexíveis quanto aos tipos de documentos que manipulam.

Web Services

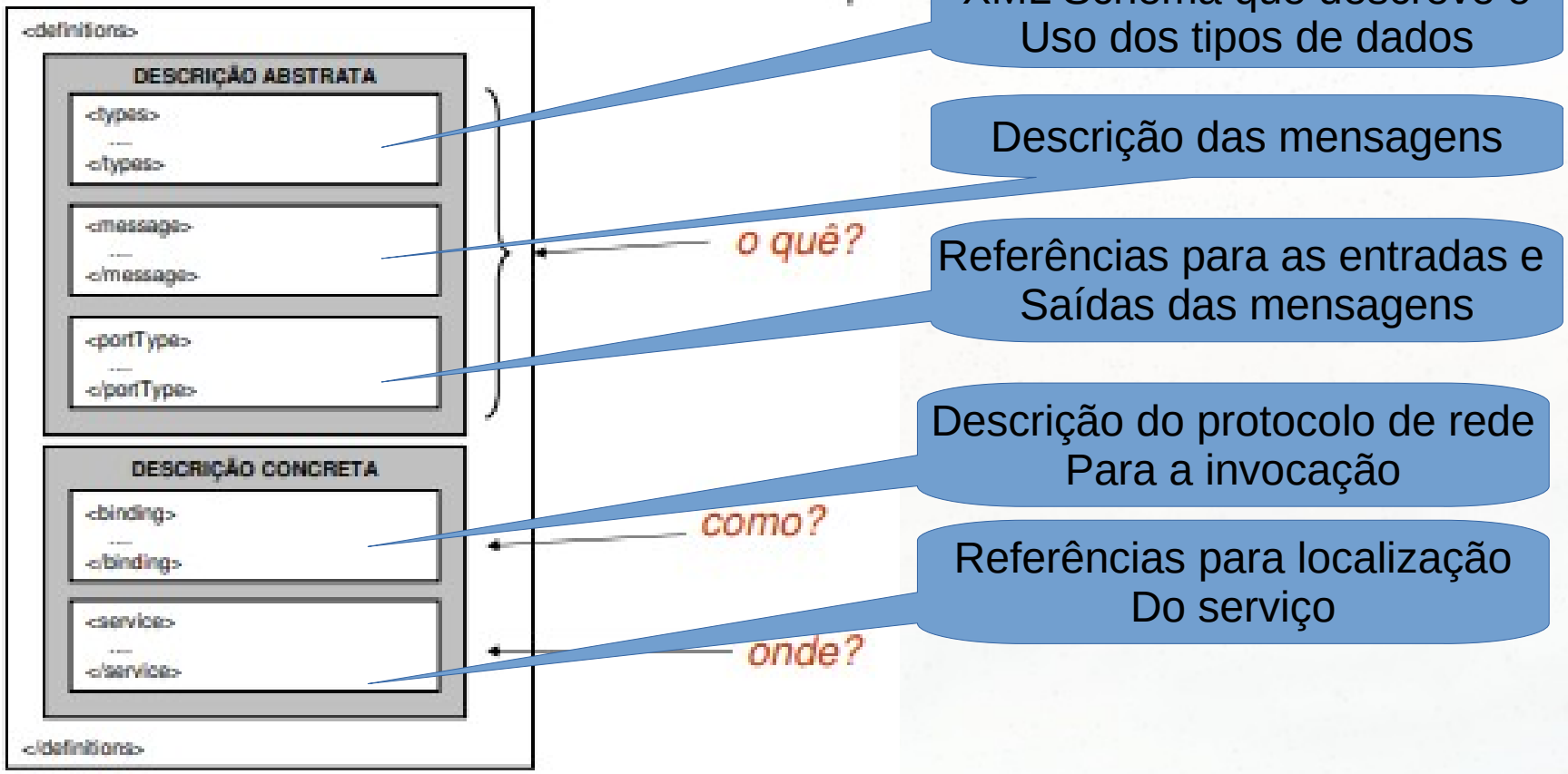
- Famosos Web Servidores:
  - Amazon;
  - Yahoo;
  - Google;
  - Ebay.
- Exemplo:
  - Amazon fornece operações para os clientes:
    - obterem informações de produtos,
    - adicionar um produto ao carrinho de compras, ou
    - verificar o status da transação

Tecnologias Web Services

- WSDL (*Web Services Description Language*)
  - Todo serviço web deve ter sua interface descrita em WSDL
  - Um documento WSDL descreve a assinatura das operações (métodos) fornecidas por um serviço web, bem como sua localização, protocolo utilizado e outras informações
  - As mensagens SOAP são construídas a partir da descrição dos serviços em WSDL
  - Uma descrição em WSDL é um documento XML, podendo ser obtida de forma estática ou dinâmica pelos clientes

Tecnologias Web Services

- Estrutura do documento WSDL



Tecnologias Web Services

- WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://localhost:8080/axis/Servico.jws"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://localhost:8080/axis/Servico.jws"
    xmlns:intf="http://localhost:8080/axis/Servico.jws"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  + <wsdl:message name="somaRequest">
  + <wsdl:message name="somaResponse">
  + <wsdl:portType name="Servico">
  + <wsdl:binding name="ServicoSoapBinding" type="impl:Servico">
  + <wsdl:service name="ServicoService">
  </wsdl:definitions>
```

Tecnologias Web Services

- UDDI (*Universal Description Discovery and Integration*)
  - Protocolo avançado para publicação e descoberta de Web Services;
  - Padrão proposto pela Microsoft, IBM e Sun e visa a padronização de alguns elementos da web;
  - Define registros para Web Services:
    - Provedor;
    - Informações do serviço;
    - Acesso técnico.

SOAP e REST

- O protocolo SOAP e o REST são as duas formas mais utilizadas para implementar um web service atualmente.
- SOAP (*Simple Object Access Protocol*)
  - Permite troca e mensagem entre serviços
  - Uma mensagem SOAP é um documento XML, facilmente transportável por protocolo HTTP e SMTP (e-mail).

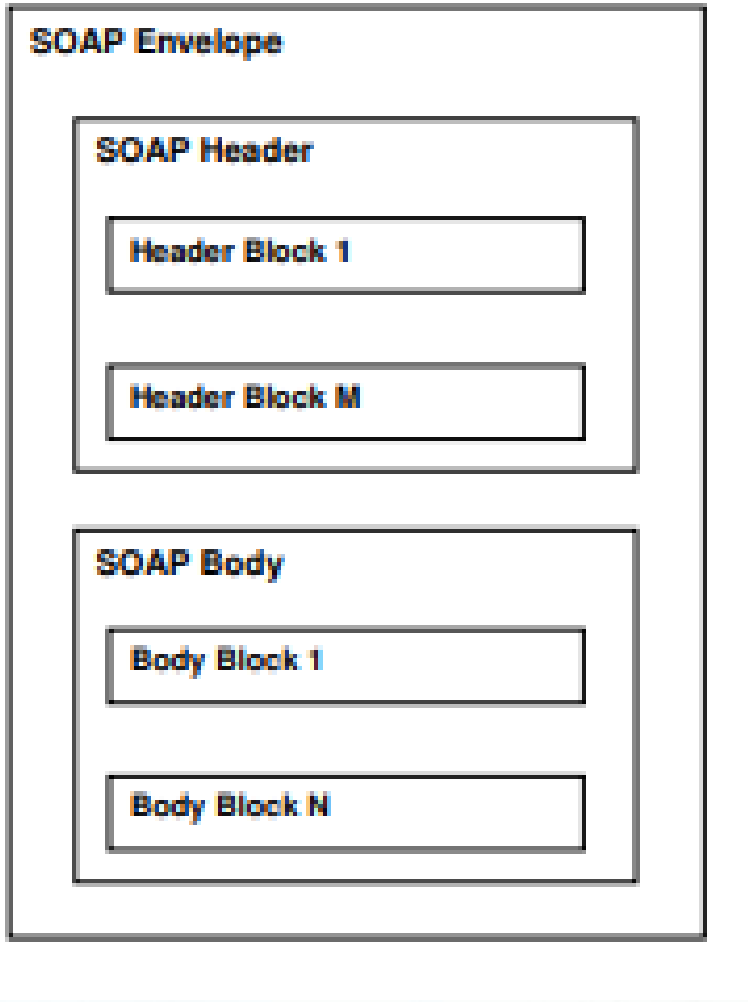
SOAP e REST

- REST (*Representational State Transfer*)
  - Não é um protocolo por si só, mas um conjuntos de regras de arquitetura.
  - Serviços REST podem ser escritos em diversas linguagens diferentes.

SOAP

SOAP

- Mensagem SOAP

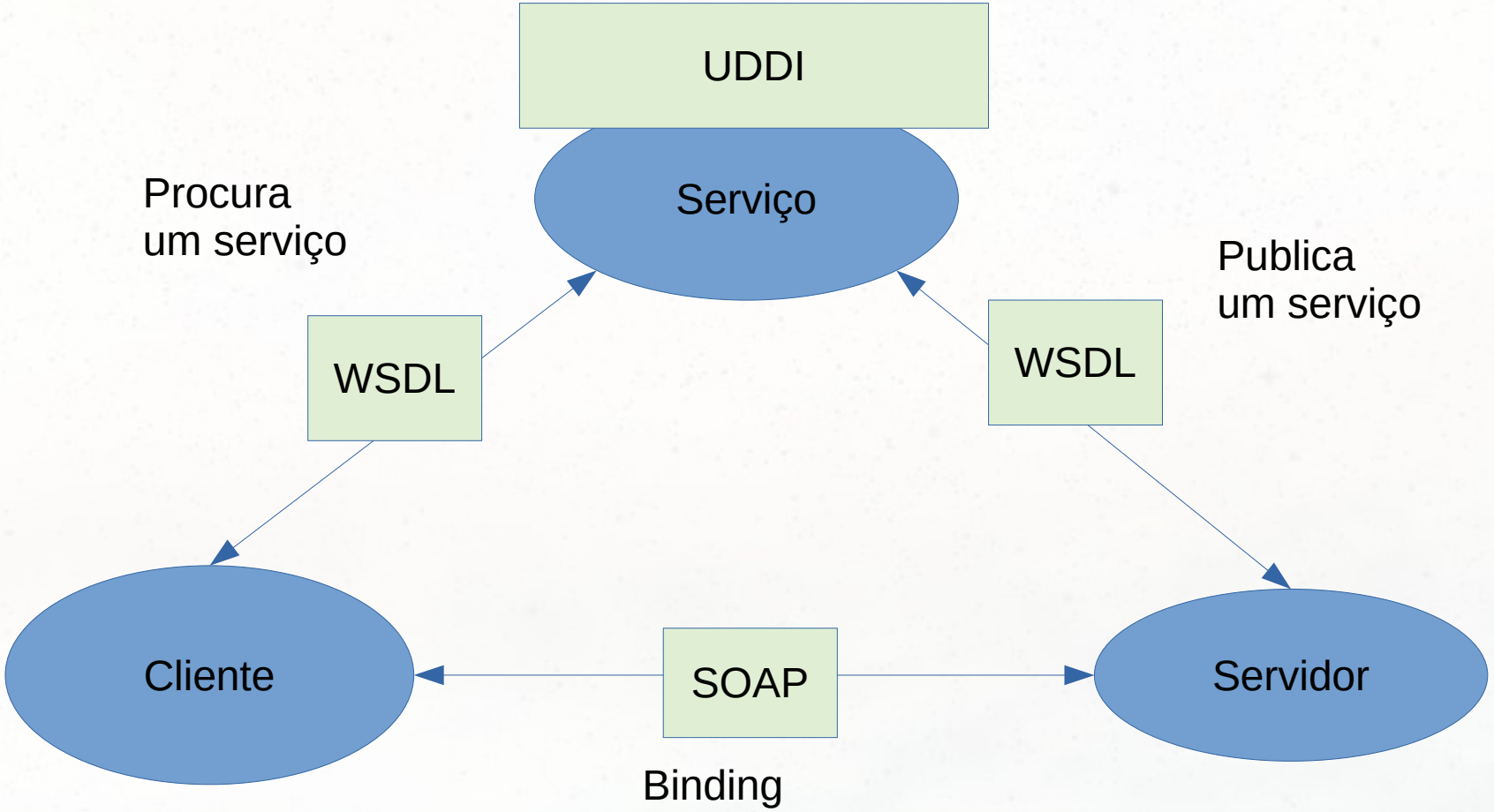


SOAP

- SOAP Request
  - O envelope SOAP é construído a partir da interface do serviço web, descrita em WSDL
  - A chamada ao serviço web segue empacotada em <soma> ... </soma>, juntamente com seus parâmetros, <arg0> e <arg1>

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <soapenv:Body>
    - <soma soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <arg0 xsi:type="xsd:int">2</arg0>
        <arg1 xsi:type="xsd:int">4</arg1>
      </soma>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP


UDDI
Serviço
Procura
um serviço
Publica
um serviço
WSDL
WSDL
Cliente
SOAP
Servidor
Binding

SOAP

- SOAP Response
  - A resposta do serviço web também vem dentro de um envelope SOAP, com a palavra Response anexada ao nome do método chamado

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <soapenv:Body>
    - <somaResponse
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <somaReturn xsi:type="xsd:int">6</somaReturn>
      </somaResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP

- Java e SOAP
  - Uma prática muito importante de ser seguida é sempre dividirmos a interface de um serviço da sua implementação.
  - A interface de um serviço é chamada de SEI (Service Endpoint Interface)
  - A implementação é chamada de SIB (Service Implementation Bean).

REST

REST

- REST define um conjunto de regras de arquitetura para construir sistemas distrubuídos:
  - Fáceis de escalar.
  - Fáceis de serem compreendidos.
- REST define um estilo de desenvolvimento, não um padrão.
  - Permite maior flexibilidade.

REST

- Serviços REST possuem uma gramática simples, facilmente compreensível.
- Facilmente escalável
  - Basta adicionar um novo web service.

Conceitos de REST

- Resources
- Representations
- Operations
- Hypertext
- Statelessness

Resources

Resources

Todos os recursos recebem um identificador:

- /database/select/id=1
- /country/name/spain

- Identificadores podem ser:
  - URIs e URLs

Resources

URI:

- Identifica um recurso por nome e localização.

- URL (*Uniform Resource Locator)*
  - Subconjunto da URI
    - Especifica onde o recurso mora e como recuperá-lo
    - Exemplo
      - four-hand-george, o robô, é uma coisa
      - Você pode recuperá-lo usando o HTTP:
        - http://obeautifulcode.com/game/robots/four-hand-george

Resources

- REST recomenda:
  - Identifique tudo que seja útil para os clientes.
  - Exemplo:
    - Vendas:
      - /game/spaceships/upgrades/56789
    - Negociações
      - /game/negotations/truces/12345
  - Coleções também são identificadas:
    - /game/robots (all robots)
    - /game/spaceships?visability=invisible (all spaceships that are invisible)

Resources

- Dicas:
  - Não use identificadores como nomes de métodos ou parâmetros.
  - Não coloque detalhes técnicos nos nomes, eles podem mudar com o tempo.

Resources

- Benefícios da URI como identificador:
  - Fácil de entender
  - Fácil de debugar
  - Fácil dos clientes utilizarem

Representations

Representations

- Sistemas RESTful permitem que os clientes perguntem em uma forma que eles possam entender;
- Exemplo: HTTP Header
  - GET /pages/archiveHTTP/1.1
  - Host: obeautifulcode.com
  - Accept: text/html

Representations

- O formato/representação dos dados requisitados é especificado pelo **MIME type** (i.e. text/html)
  - Existem muitos padrões MIME diferentes.
- Possível pedir um arquivos XML ou JSON

Representations

- REST recomenda criar um identificador que pode ser renderizado de diferentes formas.
  - GET *reports*2015/qtr/3/salesHTTP/1.1
  - HOST: obeautifulcode.com
  - Accept: application/vnd.ms-excel

Operações

Operações

- REST define 4 operações padrão (invocadas por HTTP).
  - GET: recupera algum recurso
  - PUT: cria um recurso em um URI ou atuliza um recurso existente.
  - POST: cria um recurso, mas deixa o server deicidr a URL.
  - DELETE: exclui um recurso.

Operações

- Quando tratamos recursos como coisas para manipular através de uma URL e nos restringimos a um pequeno conjunto de operações,
  - há menos necessidade de descrever um serviço com detalhes.

Exemplo - Operações REST

- Queremos voar com qual espaço-nave?
  - “ship-123”
- Para onde?
  - Marte
- Como voar (id = “ship-123”, destino = Planet.Mars)?
  - PUT para /ships/ship-123/flightpath?destination=mars

Hypertext

Hypertext

- No REST, o estado de um aplicativos é transfeiro e descoberto por meio de mensagens de hipertexto.
- O cliente REST tem menos necessidade de saber como interagir com qualquer serviço.
- Por meio de hipertexto, é possível definir o que os navegadores Web devem fazer.

Hypertext

- Exemplo de XML:

```
<robot self='http://obeautifulcode.com/game/robots/123' >
 <age>3</age>
 <money
ref='http://obeautifulcode.com/game/bank/accounts/567' />
 <lastBattle ref='http://obeautifulcode/game/battles/890' />
</robot>
```

Statelessness

Statelessness

- O REST estabelece que o servidor nãomantém nenhuma informação sobre o estado da sessão do cliente.
  - A requisição que o cliente faz deve conter toda a informação necessária para entender a request.
- O cliente é responsável por mandar informações do estado para o servidor sempre que necessário.

Benefícios - Statelessness

- Clientes são isolados contra alterações no servidor.
- Statelessness promove redundância.
  - Os clientes não precisam falar com o mesmo servidor para pedidos consecutivos.
  - Se um servidor ficar inativo, basta carregar os dados do cliente para outros servidores.

Códigos de Status

- O HTTP tem um conjunto bem definido de códigos de status para respostas, algumas das quais indicam um erro.
- Os clientes não precisam necessariamente lidar com cada um dos códigos individuais.
  - Um erro pode ser aumentado com mais detalhes sobre o que causou o erro simplesmente incluindo texto no corpo da resposta.
  - Então, ao lançar e manipular erros, os sistemas RESTful conversam usando um padrão conhecido.

Códigos de Erro

- Códigos de status HTTP retornados no cabeçalho de resposta:
  - 200 OK
    - O recurso foi lido, atualizado ou excluído.
  - 201 Criado
    - O recurso foi criado.
  - 400 Solicitação Inválida
    - Os dados enviados na solicitação foram mau.
  - 403 Não Autorizado
    - O servidor não foi autorizou execução.
  - 404 não encontrado
    - O recurso não existe.
  - 409 Conflito
    - Um recurso duplicado não pode ser criado.
  - 500 Internal Server Error
    - Ocorreu um erro de serviço.

REST X SOAP

1)SOAP é considerado um protocolo, enquanto REST é um estilo arquitetural.

2)REST é muito mais simples que SOAP. Criar clients, desenvolver APIs e documentar é muito mais fácil e simples em REST.

3)REST permite diferentes formatos para dados (JSON, XML, etc,), enquanto SOAP permite somente XML. JSON por exemplo, é enxuto e tem processamento rápido, além de ter suporte nativo em JavaScript.

REST X SOAP

4)REST tem uma melhor performance e escalabilidade, além de poder ser cacheado. SOAP não pode ser cacheado

5)Uma das vantagens do SOAP é o suporte à WS-Security que adiciona camadas de segurança extras além das suportadas através de HTTP, SSL, etc. Isso lhe dá algumas vantagens em determinadas finalidades no mundo “Enterprise”.

6)SOAP suporta transações ACID (Atomicity, Consistency, Isolation, Durability) – mesmo conceito encontrado em transações de banco de dados. REST também suporta transações mas não abrangente como o SOAP.

Netbeans

REST com NetBeans

- Como criar um aplicativo da Web anotado por Jersey a partir do NetBeans IDE.
  - No NetBeans IDE, crie um aplicativo da Web simples. Este exemplo cria um aplicativo da Web "Hello, World" muito simples.

1. Abra o NetBeans IDE.
2. Selecione Arquivo-> Novo Projeto.
3. Em Categorias, selecione Java Web com Maven. Em Projetos, selecione Aplicativo da Web com Maven. Clique em Next.

REST com NetBeans

4. Digite um nome de projeto, HelloWorldApplication, clique em Avançar.

   Verifique se o servidor é Sun GlassFish v3 (no mínimo v3).

5. Clique em Concluir. Você pode ser solicitado para o seu nome de usuário e senha do administrador do servidor. Em caso afirmativo, insira essas informações.

   O projeto será criado. O arquivo index.jsp será exibido no painel Origem.

6. Clique com o botão direito do mouse no projeto, selecione Novo e selecione Serviços Web RESTful com Padrões.

7. Selecione o recurso raiz simples. Clique em Next.

REST com NetBeans

8. Digite um nome do pacote de recursos, como helloWorld.

9. Digite helloworld no campo Path. Digite HelloWorld no campo Nome da Classe. Para o tipo MIME, selecione text / html.

10. Clique em Concluir.

11. A página Configuração de Recursos REST é exibida. Selecione OK.

12. Um novo recurso, HelloWorld.java, é adicionado ao projeto e exibido no painel Origem. Este arquivo fornece um modelo para criar um serviço da web RESTful.t.

REST com NetBeans

13. Em HelloWorld.java, encontre o método getHtml (). Substitua o // TODO comentário e a exceção com o seguinte texto, para que o produto final se pareça com o método a seguir.

- Nota
  - Como o tipo MIME que produz é HTML, você pode usar tags HTML em sua instrução de retorno.

```
/ **
    * Recupera a representação de uma instância do helloWorld.HelloWorld
    * @return uma instância de java.lang.String
    * /
   @GET
   @Produces ("MediaType.TEXT_HTML")
   public String getHtml () {
      return "<html><body><h1>Olá, Mundo !!</body></h1></html>";
   }
```

REST com NetBeans

14. Teste o serviço da web. Para isso, clique com o botão direito do mouse no nó do projeto e clique em Testar Serviços Web RESTful.

- Esta etapa implantará o aplicativo e exibirá um cliente de teste no navegador.

  Quando o cliente de teste for exibido, selecione o recurso helloworld no painel esquerdo e clique no botão Testar no painel direito.

  As palavras Olá, mundo !! serão exibidas na janela Response abaixo.

REST com NetBeans

Caso o painel do cliente não apareça, é possível navegar para o recurso diretamente pela url

- localhost:8080/HelloWorld/resources/helloWorld

REST com NetBeans

15. Implante e execute o aplicativo
   - Defina as propriedades de execução. Para fazer isso, clique com o botão direito do mouse no nó do projeto, selecione Propriedades e, em seguida, selecione a categoria Executar. Defina a URL relativa como o local do serviço da Web RESTful relativo ao Caminho de Contexto, que para este exemplo é resources/helloworld.

REST com NetBeans

- Dica:
  - Você pode encontrar o valor da URL relativa na janela do navegador Testar serviços da Web RESTful. Na parte superior do painel direito, após Resource, está a URL para o serviço da web RESTful que está sendo testado. A parte que segue o Caminho do Contexto (http://localhost:8080/HelloWorldApp) é o URL relativo que precisa ser inserido aqui.
  - Se você não definir essa propriedade, por padrão, o arquivo index.jsp será exibido quando o aplicativo for executado. Como esse arquivo também contém o Hello World como seu valor padrão, talvez você não perceba que seu serviço da Web RESTful não está em execução, portanto, fique atento a esse padrão e à necessidade de definir essa propriedade ou atualize index.jsp para fornecer um link para o serviço da Web RESTful.
- Clique com o botão direito do mouse no projeto e selecione Implantar.
- Clique com o botão direito do mouse no projeto e selecione Executar.
- Uma janela do navegador é aberta e exibe o valor de retorno de Olá, Mundo !!

Exercícios

Exercícios

1) Faça um web service para validar um CPF. Utilize java e REST.

Para exemplificar o cálculo, vamos imaginar o número de um CPF hipotético:

123.456.789-10

São os dois últimos dígitos que atestam a validade do CPF, e estes são calculados baseando-se nos 9 primeiros dígitos.

Exercícios

Calculando o Primeiro Dígito Verificador

O primeiro dígito verificador do CPF é calculado baseando-se no seguinte algoritmo.

Distribua os 9 primeiros dígitos do CPF na primeira linha de uma tabela, e na linha abaixo distribua os pesos 10, 9, 8, 7, 6, 5, 4, 3, 2 conforme representação abaixo:

1	2	3	4	5	6	7	8	9
10	9	8	7	6	5	4	3	2

Multiplique os valores de cada coluna:

1	2	3	4	5	6	7	8	9
10	9	8	7	6	5	4	3	2
10	18	24	28	30	30	28	24	18

Exercícios

Calcule a somatória dos resultados (10+18+...+24+18) = 210

O resultado obtido (210) será divido por 11. Considere como quociente apenas o valor inteiro obtido na divisão, o resto da divisão será responsável pelo cálculo do primeiro dígito verificador.

O resto da divisão é 1. Para calcular o dígito verificador, você deve subtrair o resto encontrado de onze.

11 – 1 = 10

Se o resultado da subtração for maior que 9, o dígito verificador é ZERO. Caso contrário, o dígito verificador é o resultado dessa subtração. Neste caso, o primeiro dígito verificador é ZERO.

Já temos portanto parte do CPF válido, confira: 123.456.789-0X.

Exercícios

2) Faça um web service para

- Construir na aplicação REST para obter
  a) título e assunto
  b) ano de publicação
- Exportar serviço de Autor como um WebService REST (permitir listar todos os autores, listar um autor pelo ID, filtrar autores pelo nome, inserir, remover e atualizar).
- Utilize java e REST.

Referências

- https://www.infoq.com/minibooks/emag-03-2010-rest
- http://blog.obeautifulcode.com/API/Learn-REST-In-18-Slides/
- http://courses.ischool.berkeley.edu/i290-rmm/s12/slides/Lecture3%20REST.pdf

Obrigado!
Bom Dia!
Boa Tarde!
Boa Noite!

gustavo.custodio@anhembi.br


ecossistema
ânima