Universidade Federal de Uberlândia
Faculdade de Computação

Programação para Internet

Módulo 11

Introdução aos Serviços Web e ao

Desenvolvimento de Aplicações Web com Java

Prof. Dr. Daniel A. Furtado

Introdução aos Serviços Web

Web Services

- **Definição.** *Um **web service** é uma entidade de software baseada em padrões, independente de linguagem, que aceita requisições especialmente formatadas de outras entidades de softwares em máquinas remotas por meio de protocolos de comunicação universais, produzindo respostas específicas da aplicação.*

- De maneira simplista, *web services* são serviços de software que aplicações web disponibilizam para outras aplicações web utilizando padrões de comunicação universais, como HTTP com JSON ou XML;

Ref.: Ioannis G. Baltopoulos

Web Services - Exemplos

- Exemplo de *web service* para busca de informações de endereço a partir do CEP:
  - https://viacep.com.br/ws/38408100/json/

Web Services - Benefícios

- **Interoperabilidade e Integração**. Empresas frequentemente possuem sistemas diferentes, com dados próprios. ***Web services*** facilitam a comunicação e integração desses sistemas (e até mesmo entre sistemas de empresas diferentes).
- **Reusabilidade**. Uma função disponibilizada dentro de um domínio por meio de um *web service* pode ser codificada uma única vez e utilizada inúmeras vezes por outras aplicações. Por exemplo, é possível que um mesmo *web service* seja acessado por uma aplicação interna, um browser em um desktop, um aplicativo de celular ou por um navegador *mobile*.
- **Back-end/Front-end**. *Web services* viabilizam a separação do front-end do back-end em sistemas Web.

Tipos de Serviços Web

Tipos de Web Services

- Os ***web services*** são comumente categorizados de acordo com a tecnologia em que se baseiam:
  - SOAP
  - REST

Tipos de Web Services

SOAP

- Protocolo baseado na **linguagem XML**, especificado pelo W3C, para a troca de mensagens entre aplicações na Internet/Intranet
- Acrônimo para ***S****imple* ***O****bject* ***A****ccess* ***P****rotocol*
- Como é baseado na XML, é independente de plataforma ou linguagem
- Possibilita chamadas a métodos remotos (RPC), inclusive com argumentos complexos, como se fossem chamadas locais;
- Provê uma espécie de "envelope" para o envio de mensagens de serviços web através da Internet/Intranet
- Utiliza documentos **WSDL** (*Web Service Description Language*) para descrever os serviços (localização, métodos, tipos de dados, etc)

SOAP - Exemplo

1) Exemplo de mensagem de requisição SOAP para buscar no servidor a cotação de uma ação na bolsa

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding"

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

2) Exemplo de uma mensagem de resposta SOAP com o preço da ação

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Uma desvantagem clara do SOAP é o overhead com *metadados*, o que demanda uma largura de banda maior.

Adaptado de *w3schools.com*

Tipos de Web Services

REST

- ***Re**presentational **S**tate **T**ransfer;*
- **Estilo arquitetural** para comunicação entre aplicações na Web;
- Se baseia, na maioria das vezes, no **protocolo HTTP** e seus **códigos de status** e nos **métodos de requisição** (POST, GET, PUT, PATCH, DELETE, OPTIONS e HEAD);
- Diferente do SOAP, não impõe restrições ao formato da mensagem, mas apenas no comportamento das entidades envolvidas:
  → ***Mais flexível***: o desenvolvedor pode utilizar o formato que for mais apropriado, como XML, JSON, texto, etc.
- Trata objetos no servidor como recursos que podem ser criados, modificados ou removidos;
- Independente de linguagem de programação;

RESTful

- Termo comumente utilizado para designar aplicações ou APIs baseadas no estilo REST

Web Service Rest - Exemplos

- O serviço de busca de endereço mostrado anteriormente é um exemplo de serviço RESTful:
  - https://viacep.com.br/ws/38408100/json/

- Nesse exemplo, para obter a resposta no formato xml, por exemplo, basta trocar a palavra **json** na URL por **xml**

SOAP vs REST

Leitura recomendada:

http://www.java2blog.com/2016/06/difference-between-soap-and-rest-web-services.html

Conceito de Idempotência e Métodos HTTP no Contexto de Serviços RESTful

Web Services e **Idempotência**

Um requisição HTTP é dita **idempotente** quando mantem a seguinte propriedade:

- ***Executar a requisição múltiplas vezes tem o mesmo efeito que executá-la uma única vez***

Principais Métodos HTTP para serviços *RESTful*

POST

- Utilizado para **criar** novos recursos (por exemplo, em conjunto com a operação *INSERT* da SQL)
- Em geral, altera o estado da aplicação no servidor
- Em caso de sucesso, deve-se retornar o código HTTP 201 (Created)
- Por definição, não é idempotente. Isto significa que sucessivas requisições* utilizando o POST podem ter efeitos diferentes (por exemplo, podem resultar na criação de dois ou mais recursos contendo a mesma informação).

*Requisições idênticas
Lista dos códigos de status HTTP: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
Mais detalhes sobre POST: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

Principais Métodos HTTP para serviços *RESTful*

GET

- Utilizado normalmente para operações de ***leitura*** de recursos (por exemplo, em conjunto com a operação *SELECT* da SQL);
- Não altera o estado da aplicação no servidor;
- Em caso de sucesso, retorna-se uma representação do recurso no formato *XML, JSON* ou *Texto* e o código HTTP 200 (OK)
- Por definição, **é idempotente**. Isto significa que um usuário (ou alguma ferramenta de software) pode executar a mesma requisição inúmeras vezes sem se preocupar em produzir efeitos diversos no servidor.
- Leitura complementar: https://stackoverflow.com/questions/18395523/what-is-difference-between-http-methods-get-post-put-and-delete

Mais detalhes sobre o método GET:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET

Principais Métodos HTTP para serviços *RESTful*

PUT

- Utilizado para operações de ***atualização/substituição*** por inteiro de um recurso (muitas vezes, envolve a operação *UPDATE* da SQL);
- Semelhante ao método POST, porém com a diferença de ***ser idempotente***;
- Em outras palavras, se um recurso é atualizado por meio de uma requisição PUT e, na sequência, a mesma requisição é repetida outras vezes, então todas elas terão o mesmo resultado, uma vez que o recurso atualizado permanecerá no mesmo estado que estava logo após a primeira requisição;
- Em caso de sucesso, os possíveis códigos de status de retorno são 200, 201, 204
- Leitura complementar: https://stackoverflow.com/questions/23777714/actual-use-of-get-put-delete-post-methods-in-http

Mais detalhes sobre o método PUT:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT

Principais Métodos HTTP para serviços *RESTful*

PATCH

- Frequentemente utilizado para operações de ***atualização parcial*** de um recurso (por exemplo, atualização de um dado em particular de um cliente, como número de telefone ou estado civil);
- Diferente do PUT, **PATCH** não é idempotente, o que significa que requisições sucessivas utilizando PATCH podem ter efeitos diferentes;

Mais detalhes sobre o método PATCH:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH

Principais Métodos HTTP para serviços *RESTful*

DELETE

- Utilizado para ***remover*** um recurso (por exemplo, envolvendo a operação *DELETE* da SQL);
- Em caso de sucesso, deve-se retornar o código HTTP 200 (OK)
- **É idempotente**. Repetidas requisições para remoção do mesmo recurso devem ter o mesmo resultado (200 – OK), ou seja, o recurso foi apagado e continua apagado.

Uma descrição completa de todos os métodos HTTP pode ser obtida em:
https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

Métodos Suportados em Formulários HTML e Código de Retorno com PHP

- Vale destacar que, para submissão de formulários HTML utilizando a tag <form>, os únicos métodos suportados são GET e POST (**não** se deve utilizar *method='put',* por exemplo);
- Entretanto, os demais métodos podem ser utilizados por meio de requisições HTTP utilizando o objeto ***XMLHttpRequest*** ou o métod jQuery ***$.ajax;***
- Ver exemplo anexo.

Aplicações Web com Java

Exemplo Introdutório com Eclipse, Tomcat, Maven e JSP

Algumas Tecnologias Envolvidas

***Eclipse** for Java EE Developers*

Ambiente de desenvolvimento integrado (IDE) para desenvolvimento em Java e diversas outras linguagens.

Apache Tomcat

Servidor Web de código aberto desenvolvido pela *Apache Software Foundation* que implementa uma série de especificações para desenvolvimento de aplicações Web com Java.

Java Standard Edition (SE) Development Kit (JDK)

Kit para desenvolvimento de aplicações Java

Maven, POM

Algumas Tecnologias Envolvidas

Maven

Ferramenta desenvolvida pela Apache para gerenciar as dependências do projeto (tais como bibliotecas e frameworks utilizados pelo mesmo) e automatizar o processo de compilação (build)

pom.xml

É o arquivo XML utilizado para configuração do Maven. POM é um acrônimo de *Project Object Model*.

```
<project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchem
a-instance"
xsi:schemaLocation="http://maven.apache.or
g/POM/4.0.0
http://maven.apache.org/xsd/maven-
4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>
<groupId>com.mycompany.app</groupId>
<artifactId>my-app</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>

<name>Maven Quick Start Archetype</name>
<url>http://maven.apache.org</url>

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.8.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
</project>
```

Exemplo de arquivo pom.xml

Algumas Tecnologias Envolvidas

JSP

- Uma forma de criar web sites dinâmicos com Java é utilizando a tecnologia JSP
- **JSP** é um acrônimo de **J**ava**S**erver **P**ages;
- Um arquivo JSP pode conter código HTML, CSS, JavaScript e **Java** propriamente dito (de maneira semelhante a um arquivo PHP);
- Arquivos JSP são convertidos automaticamente em programas Java denominados ***servlets****

Scriptlet JSP

- É o código Java escrito entre as tags **<%** e **%>** de um arquivo JSP;

Expressão JSP

- Utilizada para avaliar uma expressão Java simples;
- Colocada entre <%= *expressão* %>

*Com a crescente popularidade dos frameworks MVC (como Spring), a tecnologia **JSP** com ***Servlets*** já *não é* amplamente utilizada no desenvolvimento de novas aplicações Web. Entretanto, o entendimento dos conceitos envolvidos continua sendo fundamental para formação sólida da base de conhecimento.

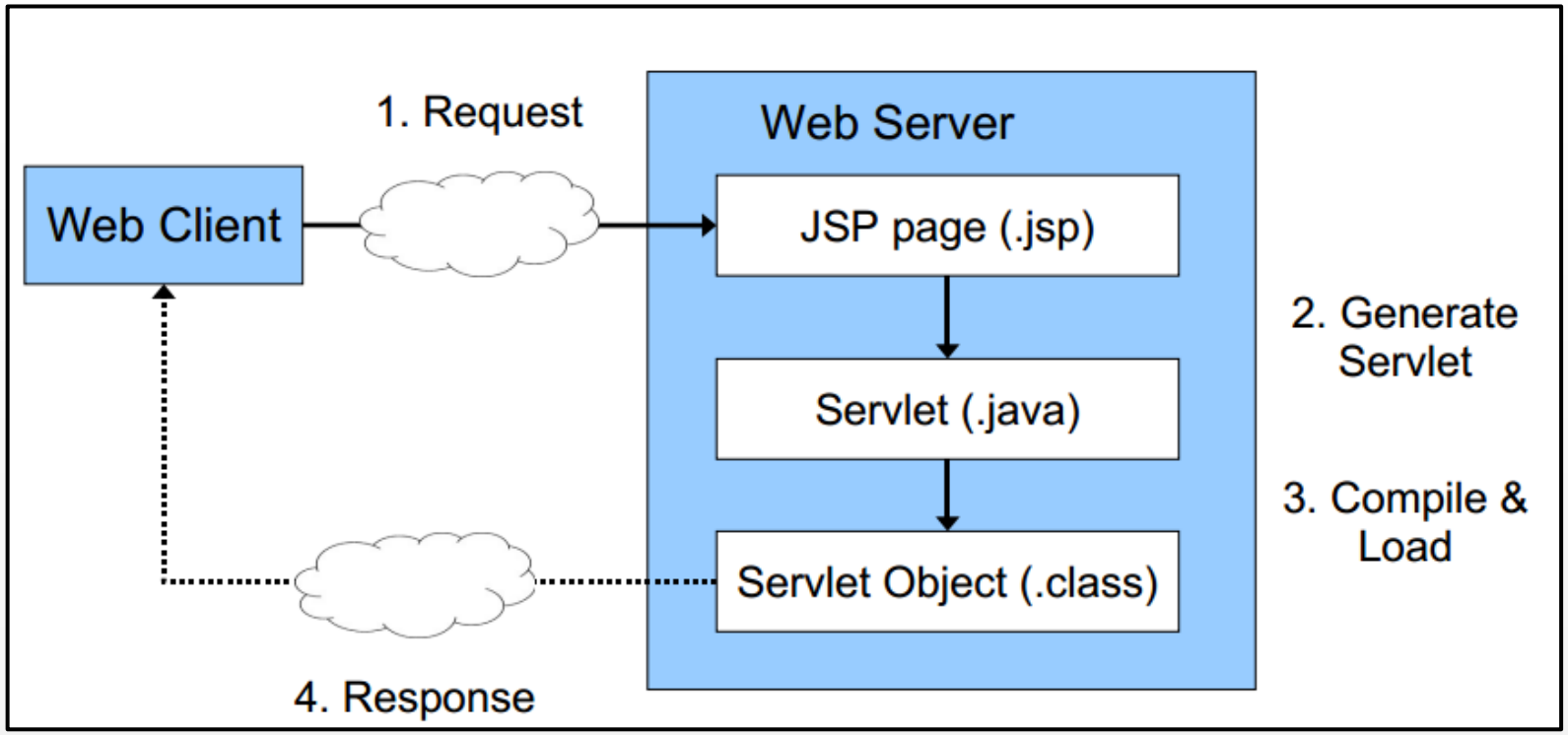
Exemplo de arquivo JSP

```
<html>
<body>
<h2>Hello World!</h2>
 <%
    for (int i = 0; i < 10; i++)
        out.println("<h3>Texto gerado dinamicamente com Java</h3>");
 %>
</body>
</html>
```

- Quando um arquivo JSP é acessado pela primeira vez, o servidor Tomcat converte o arquivo JSP em um programa Java, denominado ***servlet.*** Logo em seguida, o Tomcat compila o servlet e o executa;
- Nos próximos acessos o processo é mais rápido, uma vez que a conversão e a compilação não são necessárias (exceto se o arquivo JSP sofrer alguma alteração)

JavaServer Pages (JSP)

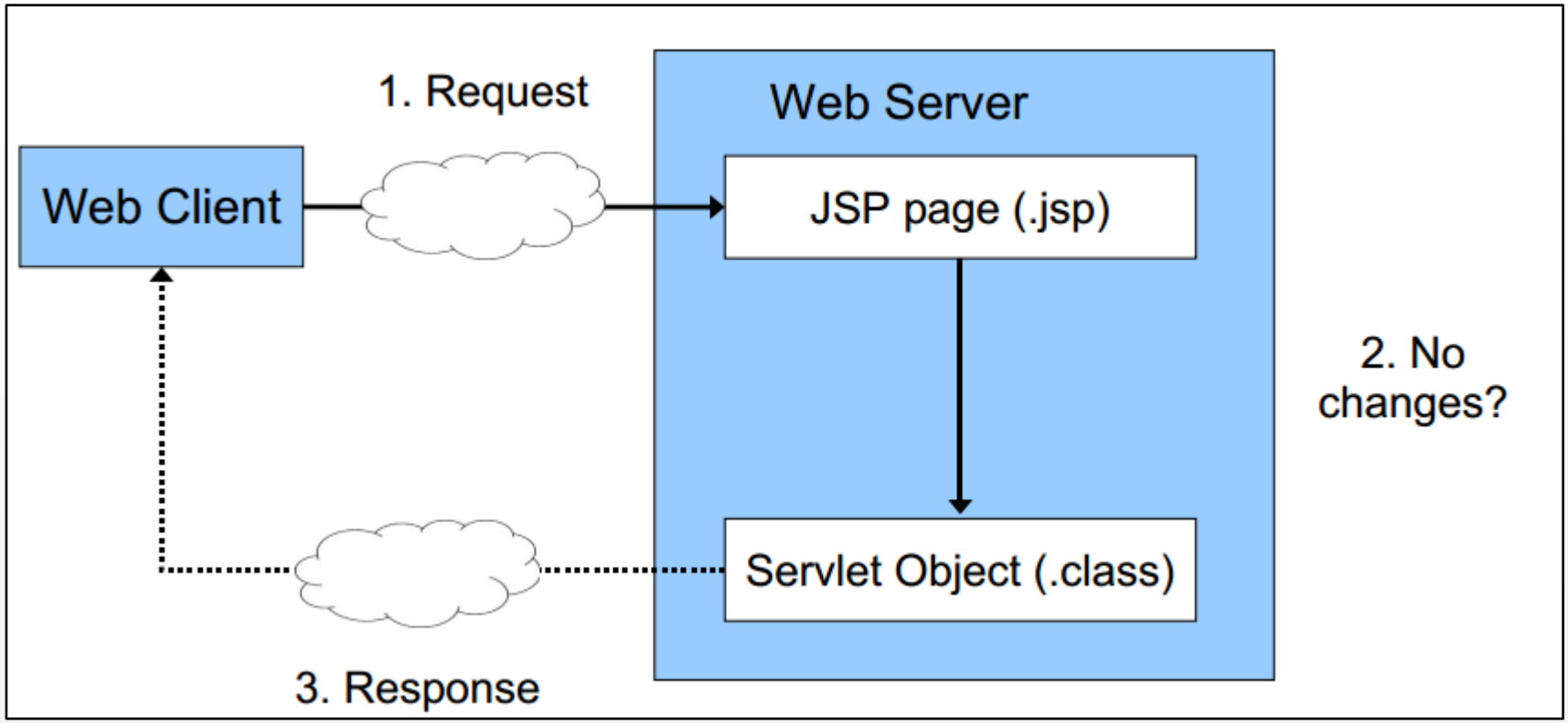
Primeira Requisição



Ref.: https://wtad640002.wordpress.com/2013/02/07/how-jsp-works/

JavaServer Pages (JSP)

Próximas Requisições



Ref.: https://wtad640002.wordpress.com/2013/02/07/how-jsp-works/

Servlet

- Um ***servlet*** normalmente contém pelo menos três métodos:
  - *init()*
    - Inicializa o *servlet*;
    - Executado quando o *servlet* é carregado.
  - *service()*
    - Executado toda vez em que a página é requisitada (HTTP request);
    - Processa a requisição HTTP e fornece uma resposta (por exemplo, em HTML) que é enviada para o cliente HTTP (navegador do usuário, por exemplo)
  - *destroy()*
    - Executado antes do *servlet* ser encerrado e removido da memória

JSP request.getParameter

- Parâmetros da requisição podem ser resgatados com o método ***getParameter***, seja por GET ou POST
- Exemplo:
  - String usuario = request.getParameter("usuario");

Baixando os Pré-requisitos

- Java Standard Edition (SE) Development Kit (JDK)
  - http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Eclipse IDE for Java EE Developers
  - https://www.eclipse.org/downloads/eclipse-packages/
- Apache Tomcat (servidor web)
  - https://tomcat.apache.org/download-80.cgi
  - Baixe o arquivo zip (core) e descompacte para a pasta C:/Tomcat

Criando um Hello World

- Veja os passos necessários para se criar uma aplicação Web utilizando as tecnologias Java, JSP, Eclipse, Maven e Tomcat:
  - https://crunchify.com/how-to-create-dynamic-web-project-using-maven-in-eclipse/

Exemplo de Arquivo JSP

```
1.  <%@ page import="java.util.*" %>

2.  <!DOCTYPE html>
3.  <html>
4.  <head>
5.  <title>Insert title here</title>
6.  </head>
7.  <body>

8.  <%
9.      String usuario = request.getParameter("usuario");
10.     String senha = request.getParameter("senha");
11.
12.     Date horaLogin = new Date();
13. %>

14. <h1>Bem vindo, <%=usuario%>! </h1>
15. <h2>Sua senha eh: <%=senha%> </h2>
16. <h3>Data e hora do login: <%=horaLogin %></h3>

17. </body>
18. </html>
```

Framework Spring Boot

- Materiais auxiliares
  - https://javabrains.io/courses/spring_bootquickstart/ (Curso excelente)
  - https://www.baeldung.com/spring-vs-spring-boot
  - https://dzone.com/articles/spring-boot-vs-spring-mvc-vs-spring-how-do-they-compare
  - https://www.youtube.com/watch?reload=9&v=cNUQZnRRMco
  - https://www.youtube.com/watch?v=rPMt8GhZkA0
  - https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-first-application.html

Referências

- www.w3schools.com
- https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
- http://nordicapis.com/rest-vs-soap-nordic-apis-infographic-comparison/
- http://www.restapitutorial.com/lessons/httpmethods.html
- https://www.cl.cam.ac.uk/~ib249/teaching/Lecture1.handout.pdf
- https://www3.ntu.edu.sg/home/ehchua/programming/java/JSPByExample.html
- https://www.caelum.com.br/apostila-java-web/javaserver-pages/
- https://httpd.apache.org/docs/current/howto/htaccess.html