Desenvolvimento Web I

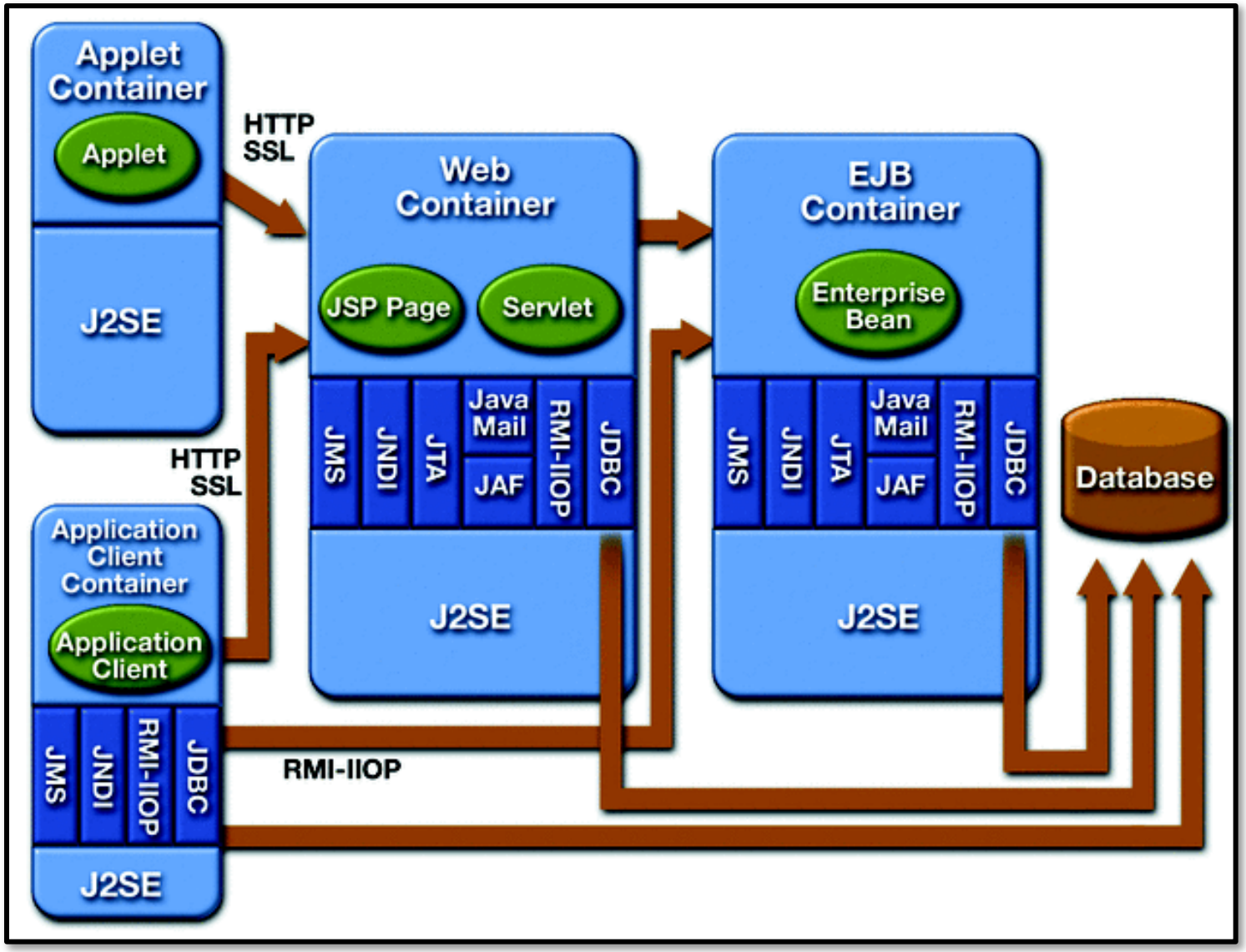
Residência de TI Aplicada
à Área Jurídica - JF e TCE

Professor: Uirá Kulesza

Outubro 2018

[Aula 2: Introdução ao Desenvolvimento de Servlets]

Arquitetura de Aplicações J2EE



Exemplos de Componentes J2EE

- Componentes Clientes
  - Applets
  - Aplicações Java Stand-Alone
- Componentes Web
  - Servlets
  - JSPs
- Componentes EJB
  - Session Beans
  - Entity Beans
  - Message-Driven Beans

Servidores J2EE

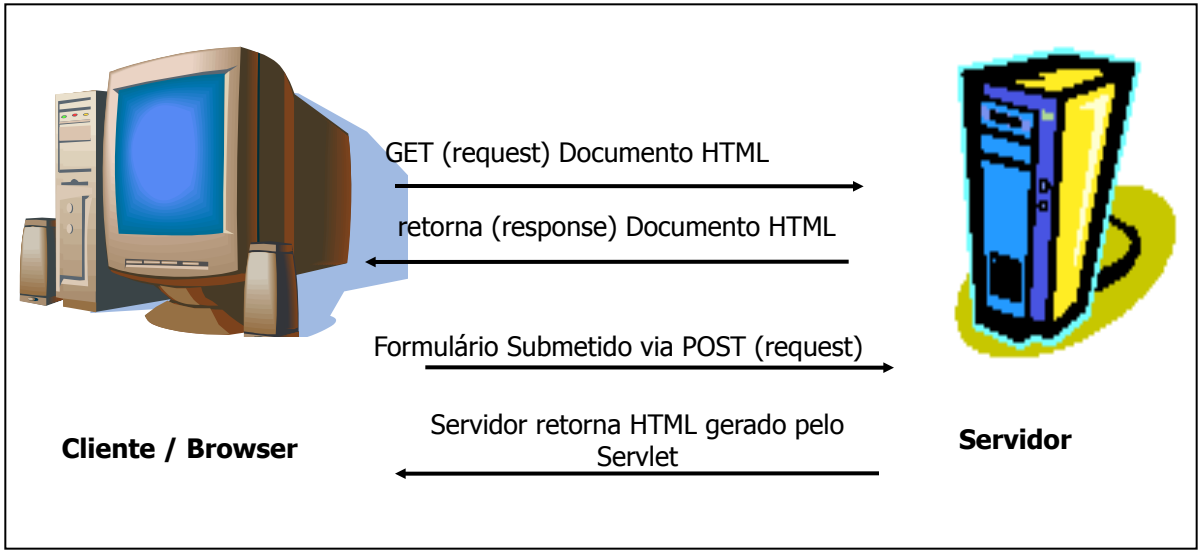
- Componentes J2EE são instalados em servidores
- Servidores:
  - Oferecem infra-estrutura para gerenciamento do ciclo de vida dos componentes
  - Incluem containers Web e EJB
  - Provêem diversos serviços para os componentes
    - ✓ Transações
    - ✓ Persistência
    - ✓ Segurança
    - ✓ Nomes
    - ✓ Distribuição

Java Servlets

O que é um Servlet?

- Aplicação Java que é executada do lado do servidor que estende a capacidade do servidor web.
  - Alternativa Java para os scripts CGI
  - Gerenciado pelo módulo web de um servidor
- Vantagens
  - Melhor aproveitamento dos recursos do sistema
  - Fácil de programar
  - Portável

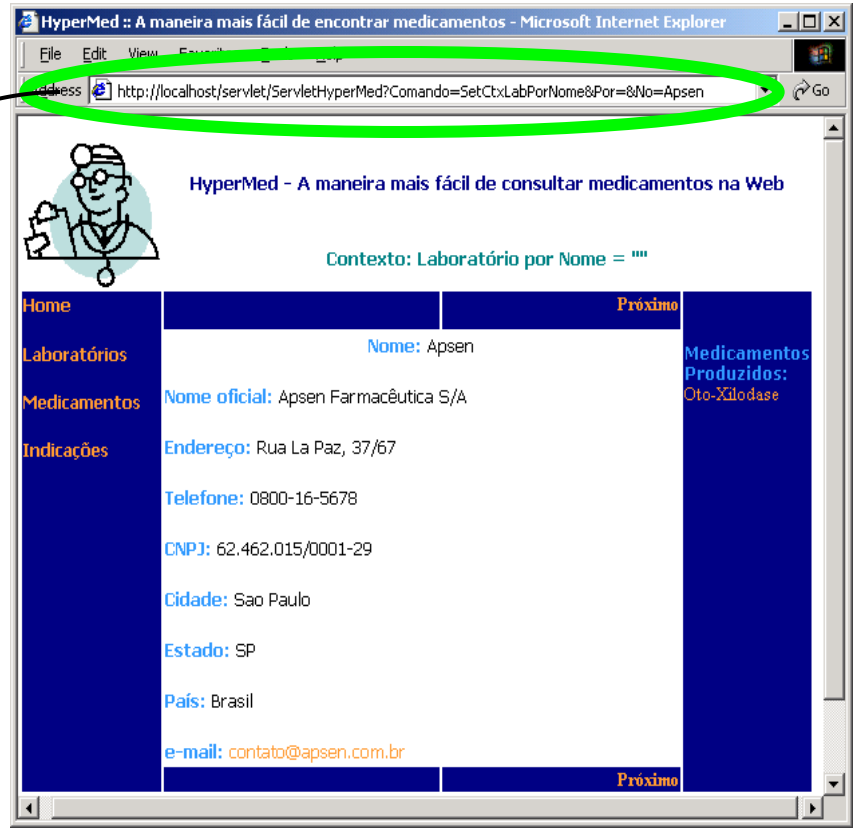
Arquitetura Cliente-Servidor



Arquitetura Cliente-Servidor

- Arquitetura baseada na divisão do processamento em processos distintos:
  - Servidor
    - responsável pela manutenção da informação.
  - Clientes
    - responsáveis pela obtenção dos dados
    - envio de pedidos ao processo servidor
- Cliente faz requisições utilizando protocolo HTTP
- Para cada requisição o servidor cria uma nova Thread

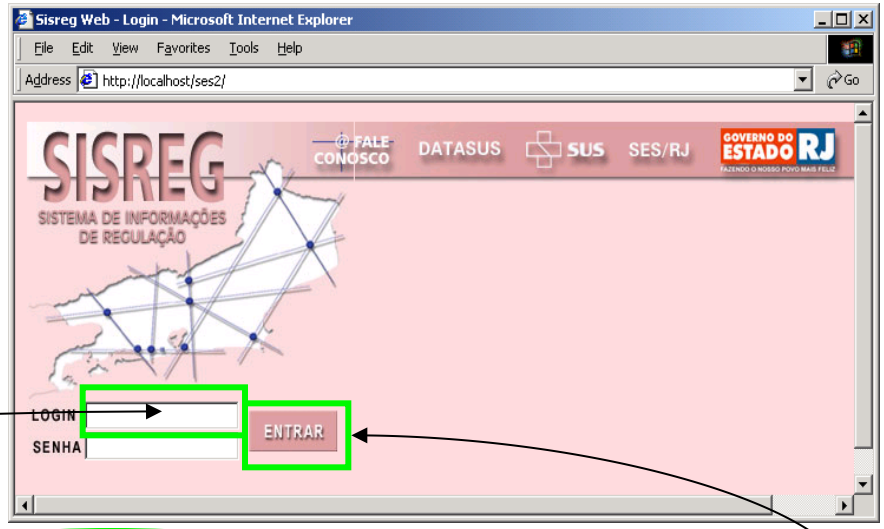
Requisitando um Servlet via GET



GET (HTTP)

http://localhost/servlet/ServletHyperMed?Comando=SetCtxLabPorNome&Por=&No=Apsen

Requisitando um Servlet via POST




```
<html>
        <!-- (...) -->
        <form method="POST" action="/servlet/ServletSes">
            <!-- (...) -->
                <input type="text" name="Login" size="18" tabindex="0">
            <!-- (...) -->
                <input border="0" src="BtnEntrar.jpg" name="I1" type="image">
            <!-- (...) -->
                <input type="password" name="Senha" size="18" tabindex="0">
            <!-- (...) -->
          <input type="hidden" name="Comando" value="Login">
        </form>
        <!-- (...) -->
</html>
```

A API de Servlet

- A API de Servlet é um conjunto de classes Java que define uma interface padrão entre o cliente web e o servidor web.

```
java.lang.Object
   |
   +----javax.servlet.GenericServlet
           |
           +----javax.servlet.http.HTTPServlet
                              |
                              +----br.ufrn.dimap.MeuServlet
```

HttpServlet

- Tratador de requisições HTTP
- Trata métodos HTTP específicos:
  - doGet(HttpServletRequest req, HttpServletResponse resp)
  - doPost(HttpServletRequest req, HttpServletResponse resp)
- doGet e doPost são chamados pelo método service()
- Subclasses reescrevem os métodos doGet(), doPost() e podem reescrever os métodos init() e destroy()

Ciclo de Vida do Servlet

- Criação e inicialização
  - Realizado uma única vez.
  - init(ServletConfig config)
  - Sempre chama o método da super classe primeiro
    - super.init(config)
  - Pode sinalizar a exceção *UnavailableException* caso ocorra algum erro durante o processo de inicialização

- Os métodos service(), doGet(), doPost() são chamados para atender requisições de clientes
  - Cada requisição é atendida por uma nova thread

- destroy() e Coleta de lixo
  - Executado somente uma vez
  - Alguns servidores somente removem o servlet quando ocorre o *shutdown*.
  - Utilizado para liberar recursos.

Recapitulando...

- Para criar um Servlet é necessário:
  - Estender a classe HttpServlet
    - pacote javax.servlet.http do JSDK
  - Implementar os métodos doGet ou doPost

Request e Response

- Os métodos doGet(), doPost() recebem dois parâmetros:
  - HttpServletRequest
    - encapsula os parâmetros da requisição
  - HttpServletResponse
    - encapsula a resposta ao cliente

HttpServletRequest

- Interface que encapsula a requisição feita pelo cliente através do protocolo HTTP

- Possui métodos que permitem recuperar os dados da requisição:
  - cabeçalho
  - dados de formulários ou parâmetros enviados
  - informações sobre a sessão do cliente

HttpServletRequest

- Alguns métodos definidos em HttpServletRequest
  - Manipulação de Parâmetros
    - java.util.Enumeration getParameterNames()
      - Obtém listagem, contendo nomes de parâmetros da requisição
    - java.lang.String[] getParameterValues(java.lang.String name)
      - Obtém valores do parâmetro “name” da requisição
    - java.lang.String getParameter(java.lang.String name)
      - Obtém valor do parâmetro “name” da requisição

HttpServletResponse

- Interface que encapsula a resposta ao cliente
- Possui um método responsável por recuperar o canal de resposta com o cliente
  - getWriter()
    - retorna um PrintWriter
    - Ex: PrintWriter out = response.getWriter();
      out.println(“Escrevendo no cliente”);
- Também é possível configurar qual é o tipo dos dados que estão sendo enviados de volta ao cliente
  - setContentType(String type)
  - Os tipos mais comuns são text/html, text/xml
  - O método setContentType deve ser chamado antes do método getWriter

Hello World!

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Arquivo web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         id="WebApp_ID" version="3.1">
  <display-name>primeiroServlet</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>MeuServlet</servlet-name>
    <servlet-class>br.dimap.MeuServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MeuServlet</servlet-name>
    <url-pattern>/MeuServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Segundo Exemplo

```
@WebServlet("/ExemploParametros")
public class ExemploParametros extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void service(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
        Enumeration<String> enumer = request.getParameterNames();
        PrintWriter out = response.getWriter();

        String name, value;
        while (enumer.hasMoreElements()) {
            name = enumer.nextElement();
            value = request.getParameter(name);
            out.println(name + " = " + value);
        }

    }

}
```

Gerenciamento de sessão (I)

- O protocolo HTTP é um protocolo sem estado

- Não é possível a princípio conhecer o estado do cliente anterior à uma requisição
- O gerenciamento de sessão provê uma maneira de identificar usuários através de várias requisições HTTPs e recuperar suas informações
  - Dados do estado do cliente

- Para obter uma sessão:
  - HttpSession getSession()
    - Obtém sessão existente, caso não exista retorna uma nova sessão.
  - HttpSession getSession(boolean create)
    - Obtém sessão existente ou cria uma nova sessão dependendo do valor do parâmetro create

Gerenciamento de sessão (II)

- Principais métodos
  - Métodos para manipulação de objetos em sessão
    - void removeAttribute(java.lang.String name)
      - Retira objeto da sessão
    - void setAttribute(java.lang.String name, java.lang.Object value)
      - Coloca ou sobreescreve objeto na sessão identificado por name
    - java.lang.Object getAttribute(java.lang.String name)
      - Obtém objeto da sessão
    - java.util.Enumeration getAttributeNames()
      - Obtém os nomes de todos os objetos armazenados na sessão

Gerenciamento de sessão (III)

- Tempo de duração da sessão
  - As sessões em geral tem intervalo máximo de tempo que podem ficar inativas
  - public int getMaxInactiveInterval()
  - public void setMaxInactiveInterval(int interval)
- As sessões também podem ser encerradas explicitamente pelo programador:
  - public void invalidate()

Exemplo – Carrinho de compras (I)
Adicionando parâmetros à Sessão

```
@WebServlet("/ServletSessaoHttp")
public class ServletSessaoHttp extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {


        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String comando = request.getParameter("comando");

        if (comando.equalsIgnoreCase("Login")) {
            HttpSession session = request.getSession(true);
            String login = request.getParameter("login");
            session.setAttribute("login", login);

            ListaCompras carrinho = new ListaCompras();
            session.setAttribute("listaCompras", carrinho);

            this.retornarMensagem("Login realizado: " + login, out);
```

Exemplo – Carrinho de compras (II)
Recuperando parâmetros da Sessão

```
if (comando.equalsIgnoreCase("Login")) {
    HttpSession session = request.getSession(true);
    String login = request.getParameter("login");
    session.setAttribute("login", login);

    ListaCompras carrinho = new ListaCompras();
    session.setAttribute("listaCompras", carrinho);

    this.retornarMensagem("Login realizado: " + login, out);


} else if (comando.equalsIgnoreCase("Comprar")) {
    HttpSession session = request.getSession(false);

    String login = (String) session.getAttribute("login");
    ListaCompras carrinho = (ListaCompras) session.getAttribute("listaCompras");

    String nome = request.getParameter("produto");
    String preco = request.getParameter("preco");

    Produto produto = new Produto(nome, preco);
    carrinho.addProduto(produto);

    this.retornarMensagem(login + " selecionou o produto: " + nome + " - " + preco, out);
```

Exemplo – Carrinho de compras (III)
Recuperando parâmetros da Sessão

```
} else if (comando.equalsIgnoreCase("Comprar")) {
    HttpSession session = request.getSession(false);

    String login = (String) session.getAttribute("login");
    ListaCompras carrinho = (ListaCompras) session.getAttribute("listaCompras");

    String nome = request.getParameter("produto");
    String preco = request.getParameter("preco");

    Produto produto = new Produto(nome, preco);
    carrinho.addProduto(produto);

    this.retornarMensagem(login + " selecionou o produto: " + nome + " - " + preco, out);
```

Exemplo – Carrinho de compras (IV)
Encerrando a Sessão

```
if ( comando.equalsIgnoreCase( "Logout" ) )
{
   System.out.println( "logout2" );
   HttpSession session = request.getSession( false );
   session.invalidate();
}
out.close();
```

Colocando para Funcionar

- Para que o servlet “funcione” é preciso um servidor web/ aplicação com suporte a Servlets/JSPs que escute as requisições do cliente

- Ex: JBoss, GlassFish, Geronimo, Tomcat

- Alguns destes servidores mantém o Tomcat como Servlet container

Jakarta Tomcat

- Servidor gratuito baseado em java
- Configurável de maneira simples via XML
- Oferece suporte a Java (Servlets e JSP)

Configurando o Tomcat

- Onde colocar as classes dos Servlets?
  - Em geral as classes são colocadas no diretório:
    - %TOMCAT_HOME%\webapps\XXX\WEB-INF\classes
  - A partir deste diretório devem ser colcadas em seus próprios pacotes

Configurando o Tomcat

- Registrando os Servlets
  - Os servlets devem ser chamados com seu nome completo (pacotes e classe)
  - Muitas vezes o nome se torna grande demais
  - Pode-se então dar um nome ao servlet através do qual o servidor redirecionará a chamada à classe específica

Configurando o Tomcat

- Registrando Servlets
  - O arquivo web.xml
    - Localizado em geral no diretório:
      - %TOMCAT_HOME%\webapps\ROOT\WEB-INF
    - Registrando um servlet:

```
<web-app>
    <servlet>
        <servlet-name>ServletSes</servlet-name>
        <servlet-class>interfPesquisa.ServletSes</servlet-class>
    </servlet>
</web-app>
```

Exercício

- Eclipse IDE for Java EE Developers (Eclipse WTP
  - http://www.eclipse.org/downloads/
- Aprender a usar o ambiente
  - Use o help e tutorial listado no próximo slide
- Executar os primeiros Servlets

Referências

- Tutorial para criação e configuração de Servlets/JSPs no Eclipse WTP
  - http://www.vogella.com/tutorials/EclipseWTP/article.html
- Java EE 7 Tutorial
  - http://docs.oracle.com/javaee/7/tutorial/doc/