

Aula 2 Servlets



Gustavo Wagner

gugawag@gmail.com

Gustavo Wagner - gustavowagner.com

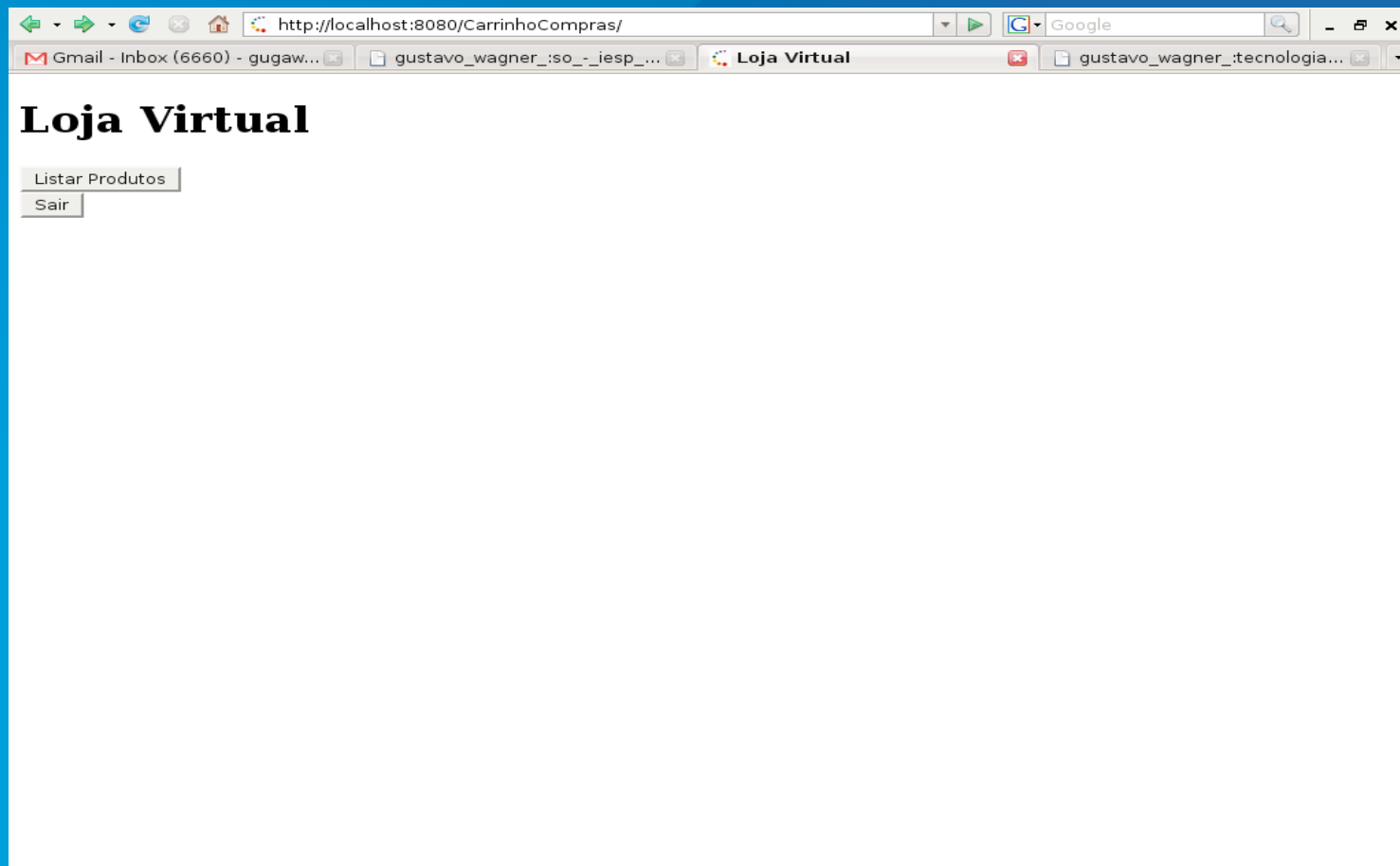
Retrospectiva

- Na aula anterior vimos uma introdução a Servlets;
- Na aula de hoje daremos mais alguns conceitos de servlet e mostraremos alguns exemplos para exemplificar o uso da API de Servlets;

Sistema: Carrinho de Compras

- Vamos desenvolver um carrinho de compras para exemplificar o uso da api de Servlets;
- O carrinho de compras funciona da seguinte forma:
 - Usuario abre tela inicial do sistema (ver proximo slide);
 - Usuario clica no botao Listar Produtos;
 - Sistema lista produtos existentes;
 - Usuario clica num produto qualquer;
 - Sistema insere esse produto no carrinho;
 - Usuario pode inserir quantos produtos quiser;
 - Usuario volta `a tela inicial para sair do sistema;
 - Sistema limpa carrinho de compras do usuario;

Tela inicial




Listagem de produtos



The screenshot shows a web browser window with the address bar displaying 'http://localhost:8080/CarrinhoCompras/ListarProdutos.do?'. The page title is 'Produtos a Venda'. Below the title is a table with two columns: 'Descricao Produto' and 'Preco'. The table contains three rows of product data.

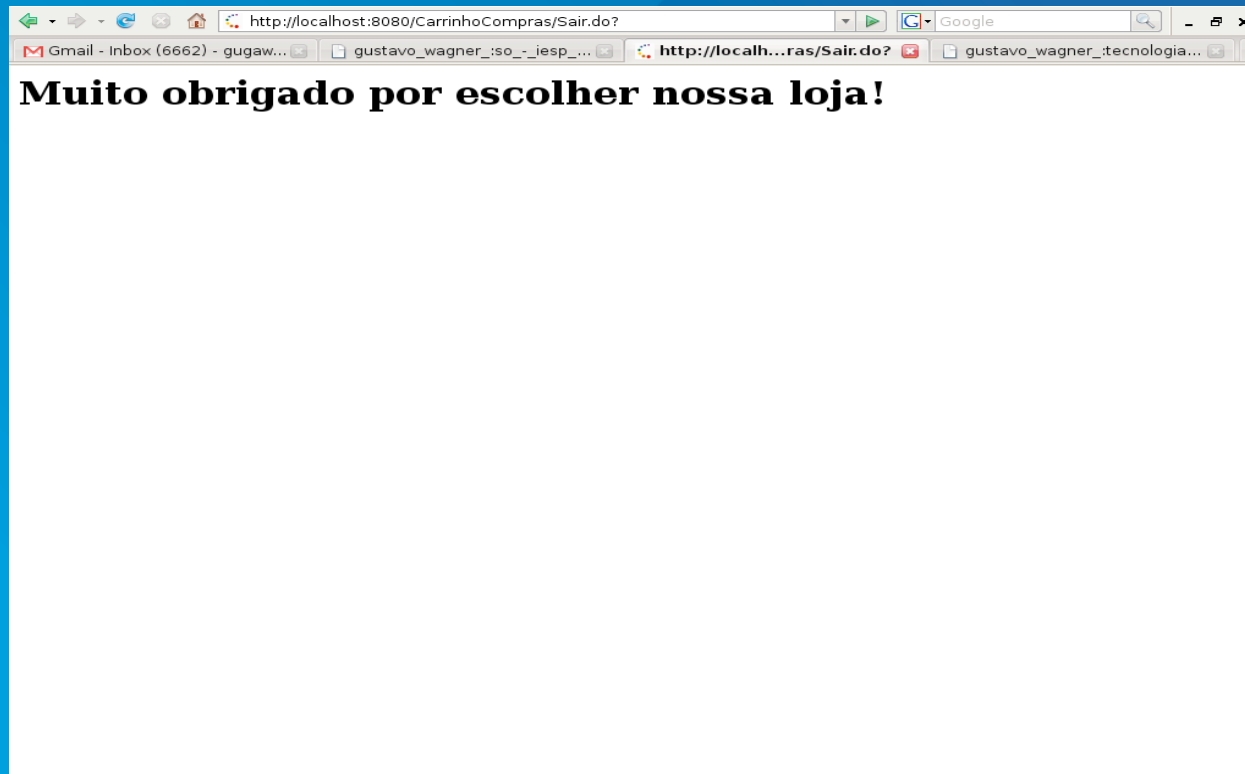
Descricao Produto	Preco
O poderoso chefao	39.99
O senhor dos aneis	30.0
O Restaurante no fim do universo	29.9

Carrinho contendo produtos escolhidos pelo usuario



Descricao Produto	Preco
O poderoso chefao	39.99
O Restaurante no fim do universo	29.9
Valor total: R\$ 69,89	

Usuario clicou no bota Sair da tela inicial: sistema limpa carrinho de compras



Escopo

- Sabemos que HTTP eh stateless, ou seja, não guarda estado;
- Isso significa que o servidor não reconhece o cliente entre requisicoes:
- Dessa forma, como guardar as informacoes num carrinho dos itens que o cliente escolheu e mostra-los, como na figura do slide 6?

Escopo

- Para entendermos como isso eh possivel, precisamos aprender quais são os possiveis escopos de um objeto numa aplicacao web;
- Existem 3 escopos:
 - **request**: os objetos nesse escopo sobrevivem apenas durante o tratamento de uma requisicao;
 - **session**: os objetos nesse escopo sobrevivem durante uma secao, ou seja, durante um tempo predefinido no servidor ou ateh que uma secao seja invalidada. Todos os servlets de uma aplicacao pode enxergar uma sessao; Soh existe uma secao por cliente (browser) por aplicacao;

Escopo

- **context**: nesse escopo objetos vivem durante todo o ciclo de vida de uma aplicação (ateh que ela seja retirada do servidor de aplicação)

Como mostrar uma lista de produtos?

- Há várias formas de se fazer isso:
 - buscar de um banco de dados;
 - ler de arquivos;
 - etc
- Iremos, para esse exemplo, colocar uma lista de produtos no escopo **context**, ou seja, no escopo da aplicação;
 - Fizemos assim pois toda vez que uma aplicação for implantada ela lerá essa lista;
- Mas como saber que uma aplicação foi implantada

Listeners

- Para isso precisamos de listeners;
- Um listener “escuta” um determinado momento da vida de uma aplicação;
- No nosso caso específico, queremos colocar uma lista de produtos no escopo **context** quando a aplicação for implantada;
- Iremos usar para isso um *ServletContextListener*

Implementando um listener: AplicacaoListener

```
Aplicações Locais Sistema Gustavo Wagner Qui 06 Mar, 16:12
Java EE - CarrinhoCompras/src/com/gugawag/tw/listeners/AplicacaoListener.java - Red Hat Developer Studio
File Edit Source Refactor Navigate Search Project Run Window Help

GeraNumero.java index.htm Produto.java *AplicacaoListener.j index.htm

import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import com.gugawag.tw.modelo.Produto;

public class AplicacaoListener implements ServletContextListener {

    /**
     * Coloca no escopo da aplicacao (context) a lista de produtos.
     */
    public void contextInitialized(ServletContextEvent event) {
        List<Produto> listaProdutos = new ArrayList<Produto>();

        int codigoProduto = 1;
        listaProdutos.add(new Produto(codigoProduto++, "0 poderoso chefao", 39.99));
        listaProdutos.add(new Produto(codigoProduto++, "0 senhor dos aneis", 30.00));
        listaProdutos.add(new Produto(codigoProduto++, "0 Restaurante no fim do universo"));
        event.getServletContext().setAttribute("produtos", listaProdutos);
    }

    public void contextDestroyed(ServletContextEvent event) {
    }
}
```

Um ServletContextListener precisa implementar dois metodos: contextInitialized, chamado pelo container quando a aplicacao for implantada, e contextDestroyed, chamado quando a aplicacao for desinstalada

Monta-se uma lista de produtos e a insere no escopo da aplicacao (context) atraves do metodo setAttribute(). O primeiro parametro eh o nome que se dah ao objeto, o segundo o proprio objeto. Pega-se o context atraves do evento gerado pelo servidor e enviado como parametro do metodo.

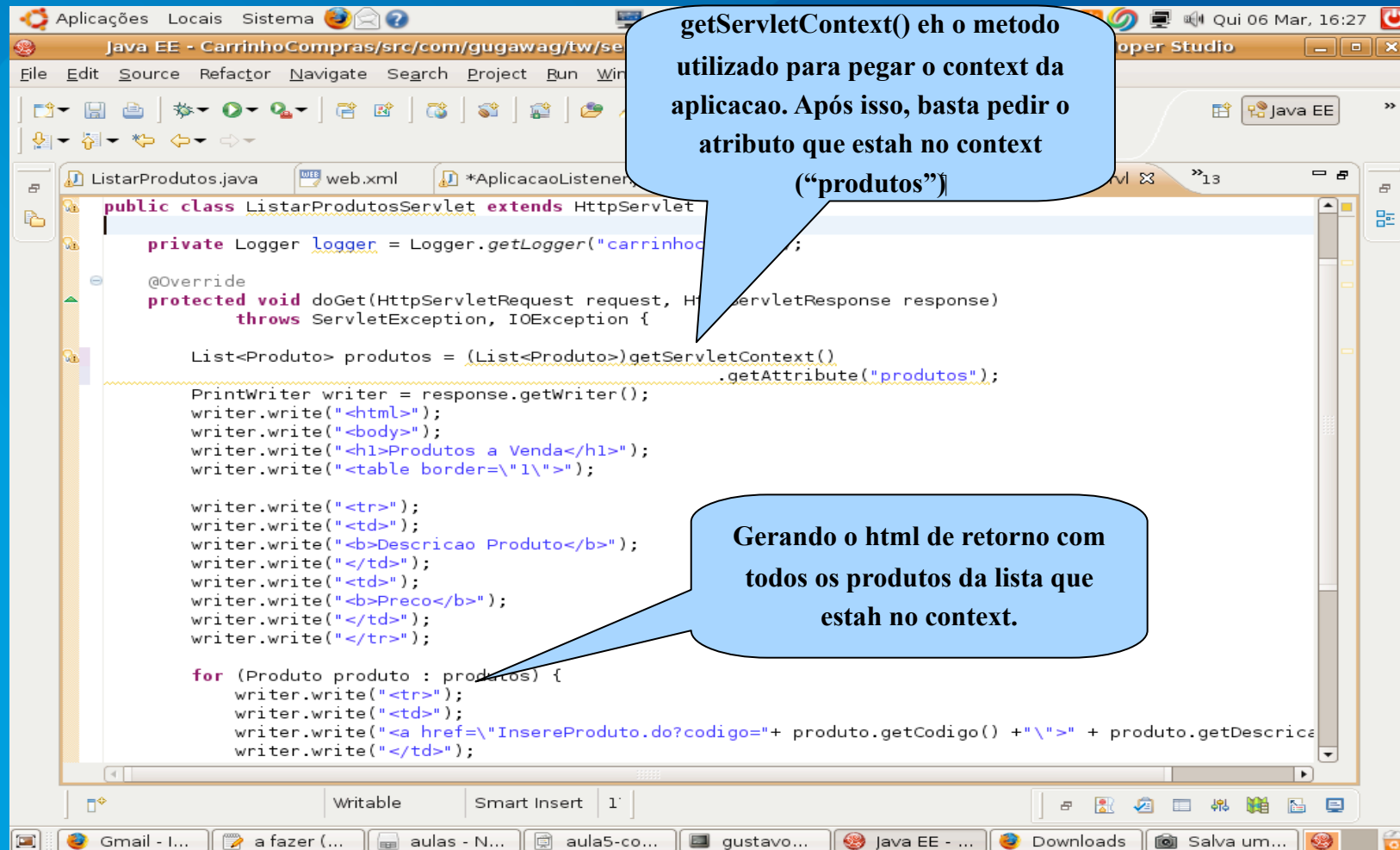
Declarando um listener

- Há duas formas de se declarar um listener:
 - Colocando no arquivo web.xml;

```
...  
<listener>  
  <listener-class>com.gugawag.tw.listeners.AplicacaoListener</listener-class>  
</listener>  
...
```

- Anotando a classe com @WebListener
- A segunda forma é mais prática, e só apareceu após o JEE6.

Listando os produtos do context



```
public class ListarProdutosServlet extends HttpServlet {
    private Logger logger = Logger.getLogger("carrinhoc");

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        List<Produto> produtos = (List<Produto>)getServletContext()
            .getAttribute("produtos");

        PrintWriter writer = response.getWriter();
        writer.write("<html>");
        writer.write("<body>");
        writer.write("<h1>Produtos a Venda</h1>");
        writer.write("<table border=\"1\">");

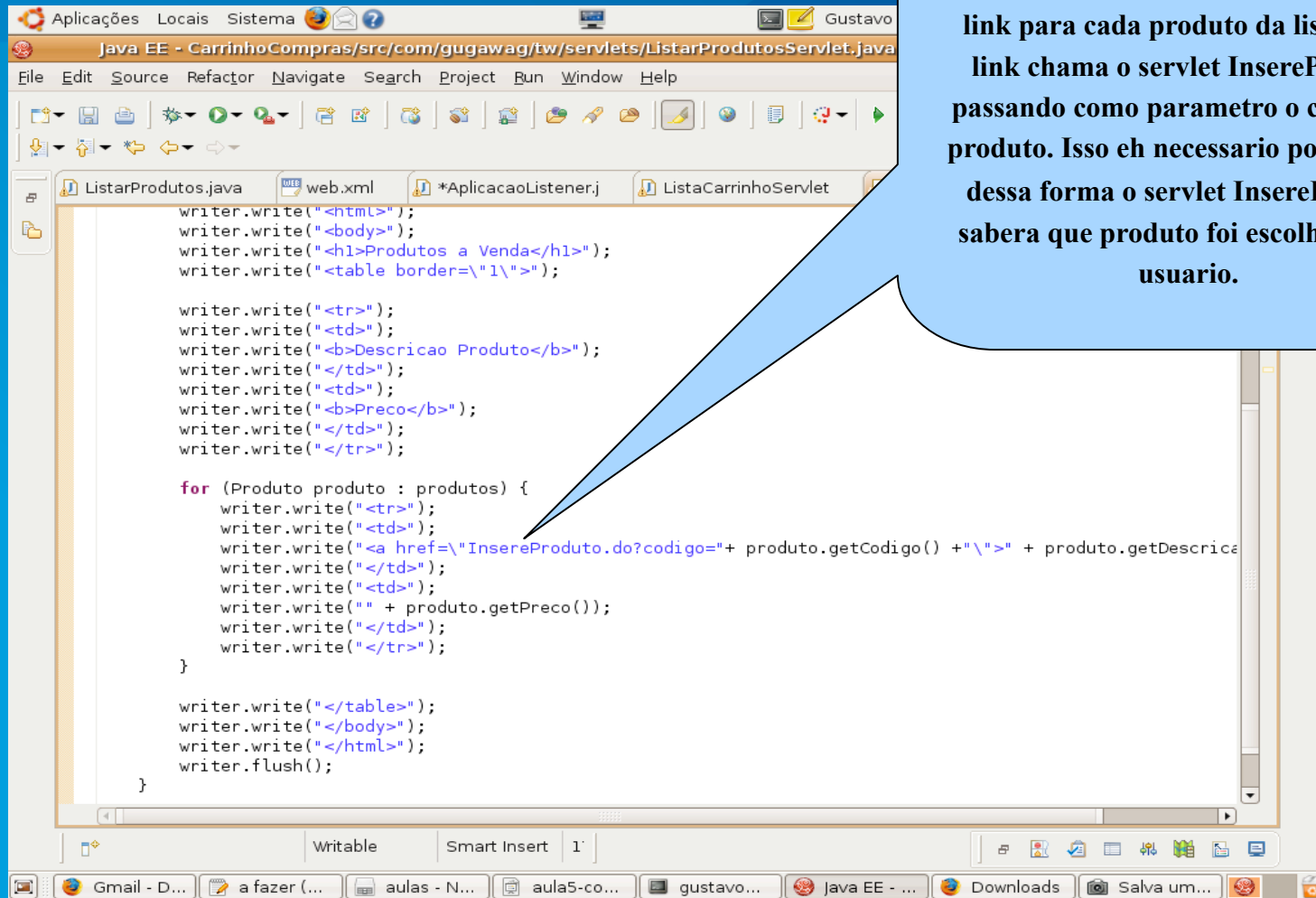
        writer.write("<tr>");
        writer.write("<td>");
        writer.write("<b>Descricao Produto</b>");
        writer.write("</td>");
        writer.write("<td>");
        writer.write("<b>Preco</b>");
        writer.write("</td>");
        writer.write("</tr>");

        for (Produto produto : produtos) {
            writer.write("<tr>");
            writer.write("<td>");
            writer.write("<a href=\"InsererProduto.do?codigo="+ produto.getCodigo() +"\">"+ produto.getDescricao());
            writer.write("</td>");
        }
    }
}
```

getServletContext() eh o metodo utilizado para pegar o context da aplicacao. Após isso, basta pedir o atributo que estah no context ("produtos")

Gerando o html de retorno com todos os produtos da lista que estah no context.

Cont. Listando os produtos do context



```
writer.write("<html>");
writer.write("<body>");
writer.write("<h1>Produtos a Venda</h1>");
writer.write("<table border='1'>");

writer.write("<tr>");
writer.write("<td>");
writer.write("<b>Descricao Produto</b>");
writer.write("</td>");
writer.write("<td>");
writer.write("<b>Preco</b>");
writer.write("</td>");
writer.write("</tr>");

for (Produto produto : produtos) {
writer.write("<tr>");
writer.write("<td>");
writer.write("<a href='InsereProduto.do?codigo=' + produto.getCodigo() + '>' + produto.getDescricao");
writer.write("</td>");
writer.write("<td>");
writer.write(" + produto.getPreco());
writer.write("</td>");
writer.write("</tr>");
}

writer.write("</table>");
writer.write("</body>");
writer.write("</html>");
writer.flush();
}
```

Perceba que o html de retorno tem um link para cada produto da lista. Esse link chama o servlet `InsereProduto` passando como parametro o codigo do produto. Isso eh necessario pois apenas dessa forma o servlet `InsereProduto` sabera que produto foi escolhido pelo usuario.

Atributos em secao

- Nosso proximo passo serah acrescentar um carrinho ao escopo de secao com os itens selecionados pelo usuario:
 - queremos manter o carrinho do usuario enquanto ele estiver navegando pelo site
- Para isso, ao clicar no link gerado pelo servlet visto no ultimo slide:
 - o servlet InsereProduto colocarah o produto selecionado no Carrinho;
 - colocarah esse Carrinho na secao, caso ainda não esteja;
 - chamarah um servlet para listar os itens do Carrinho;

Trabalhando com escopo de secao: InsereProdutoServlet

```
Aplicações Locais Sistema Gustavo Wagner Qui 06 Mar, 16:57
Java EE - CarrinhoCompras/src/com/gugawag/tw/servlets/InsereProdutoServlet.java - Red Hat Developer Studio
File Edit Source Refactor Navigate Search Project Run Window Help

public class InsereProdutoServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int codigo = Integer.parseInt(request.getParameter("codigo"));
        List<Produto> produtos = (List<Produto>)getServletContext().getAttribute("produtos");

        //A posicao do produto na lista eh o numero do seu codigo -1
        Produto produtoAInserir = produtos.get(codigo - 1);

        HttpSession secao = request.getSession(true);
        Carrinho carrinho = null;
        if (!secao.isNew()){
            carrinho = (Carrinho)secao.getAttribute("carrinho");
        }

        //Nao havia ainda secao
        if (carrinho == null){
            carrinho = new Carrinho();
        }
        carrinho.insereProduto(produtoAInserir);
        secao.setAttribute("carrinho", carrinho);

        RequestDispatcher dispatcher = request.getRequestDispatcher("ListaCarrinho.do");
        dispatcher.forward(request, response);
    }
}
```

Pega-se o codigo do produto vindo no request

Pega-se a lista de produtos do contexto para selecionar o produto escolhido pelo usuario

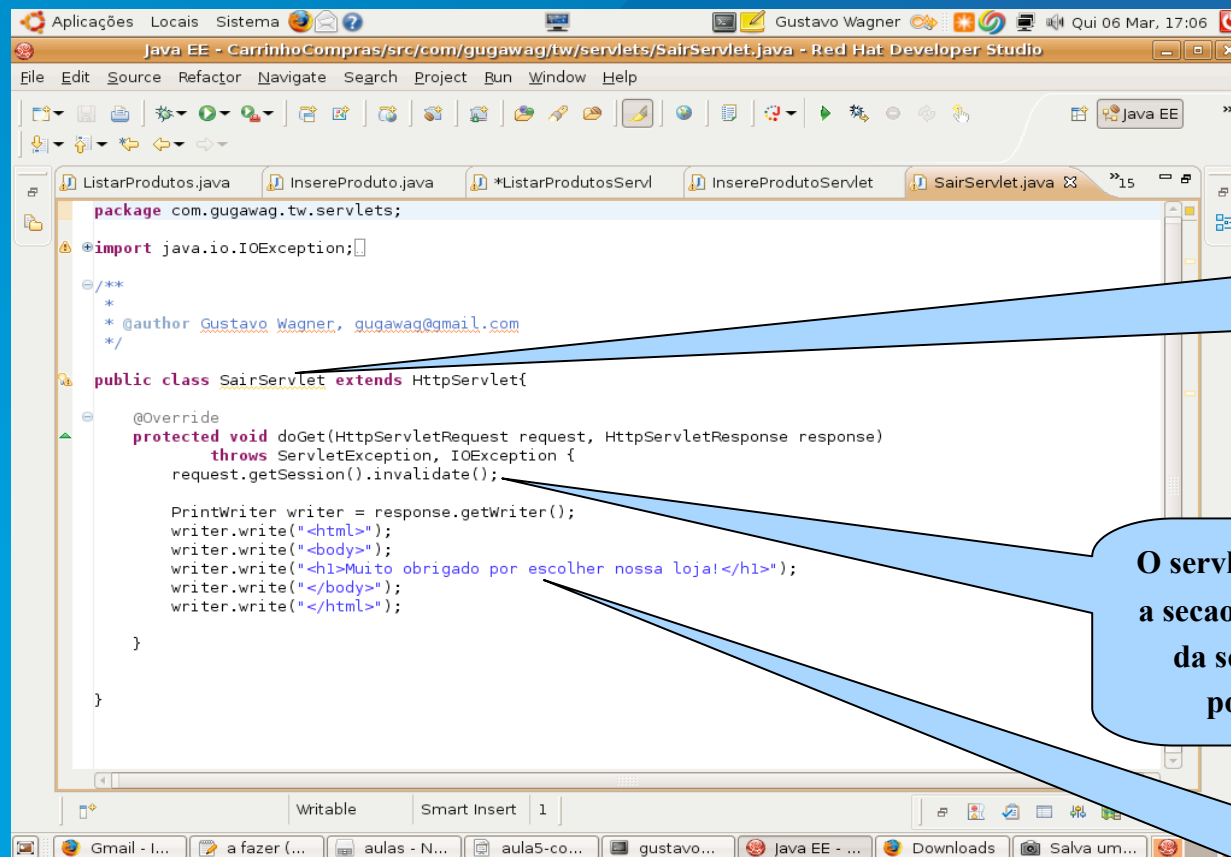
Forma de pegar uma secao. Se a secao não existir, serah criada uma: isso eh dito passando true como paramentro

Se a secao jah existir (usndo-se o metodo isNew() de session), pega-se o atributo "carrinho".

Chama-se o dispatcher do request(para passar o trabalho de responder ao cliente para outro servlet), chamando o servlet ListarCarrinho e depois da-se um forward passando-se o request e response

Coloca-se o produto no carrinho e insere-se o carrinho na secao

Limpendo a secao



```
package com.gugawag.tw.servlets;

import java.io.IOException;

/**
 *
 * @author Gustavo Wagner, gugawag@gmail.com
 */
public class SairServlet extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        request.getSession().invalidate();

        PrintWriter writer = response.getWriter();
        writer.write("<html>");
        writer.write("<body>");
        writer.write("<h1>Muito obrigado por escolher nossa loja!</h1>");
        writer.write("</body>");
        writer.write("</html>");

    }

}
```

Ao clicar no botao sair, esse servlet serah chamado.

O servlet SairServlet invalida a secao (limpa todos os dados da secao e o usuario não podera mais ve-la);

Retorna mensagem de agradecimento ao usuario

Problema com sessões

- O sistema de carrinho de compras mostrado acima só funciona se o usuário não desabilitar cookies no seu browser;
 - Isso porque como o http não guarda estado, é necessário guardar algum dado no lado cliente para que o servidor reconheça quem está fazendo a requisição
 - Esse dado é o sessionid (identificação da sessão), armazenada no cliente como um cookie:
- HTTP response:
 - HTTP/1.1 OK
 - Set-Cookie: JSESSIONID=098AB65DE43
 - ...

Problema com sessões

- Dessa forma, se o usuário desabilitar cookies, não será mais possível guardarmos o estado de um carrinho no servidor, pois o servidor não conseguirá mais reconhecer o usuário;
- A solução para esse problema:
 - URL rewriting;
- Dessa forma, antes de escrever um link num HTML de resposta para um servlet, precisamos fazer um URL rewriting;

URL Rewriting

Para que o metodo encodeURL pegue um numero de sessao e faça URL rewriting corretamente (veja quadro abaixo), eh necessario que uma sessao esteja ativa.

Para se fazer URL rewriting, deve-se chamar o metodo encodeURL do objeto response, passando como parametro a o servlet e seus parametros. Para que isso funcione, eh necessario que jah exista uma sessao. Veja quadro acima.

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

    List<Produto> produtos = (List<Produto>) request.getAttribute("produtos");

    request.getSession(true);
    PrintWriter writer = response.getWriter();
    writer.write("<html>");
    writer.write("<body>");
    writer.write("<h1>Produtos a Venda</h1>");
    writer.write("<table border='1'>");

    writer.write("<tr>");
    writer.write("<td>");
    writer.write("<b>Descricao Produto</b>");
    writer.write("</td>");
    writer.write("<td>");
    writer.write("<b>Preco</b>");
    writer.write("</td>");
    writer.write("</tr>");

    for (Produto produto : produtos) {
        writer.write("<tr>");
        writer.write("<td>");
        writer.write("<a href='\" + response.encodeURL(\"InserereProduto.do?codigo="+
            produto.getCodigo() + "\">\" + produto.getDescricao()+"</a>");
        writer.write("</td>");
        writer.write("<td>");
        writer.write("<td>");
        writer.write("<b>Preco</b>");
        writer.write("</td>");
        writer.write("</tr>");
    }
}
```

URL Rewriting

Perceba que agora o link gerado tem um JSESSIONID na url, separado por ;. não confunda isso com um atributo do request. Esse ; serah tratado pelo servidor web antes que seja repassado para o servlet requisitado. Desta forma, não serah mais necessario guardar cookies no cliente.

Descricao Produto	Preco
O poderoso chefao	39.99
O senhor dos aneis	30.0
O Restaurante no fim do universo	29.9

http://localhost:8080/CarrinhoCompras/InsereProduto.do;jsessionid=FB3270E64C53923DC898380393515165?codigo=1

Discussao cookies

- Alguns sites, como gmail, necessitam que o usuario habilite cookies
- Veja a mensagem abaixo, após desabilitar cookies:

