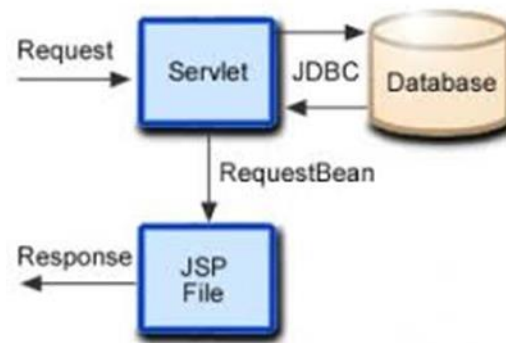


Unidade 3

Ciclo de Vida de um Servlet

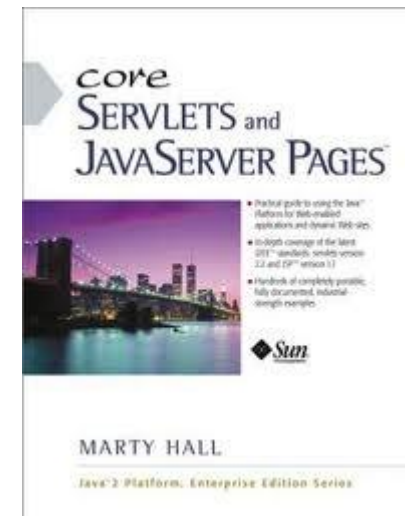
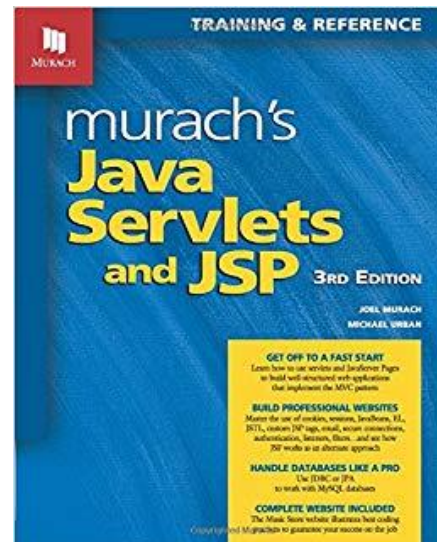


Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP



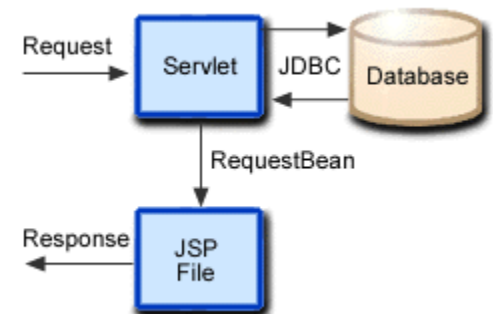
Referências

- Head First Servlets & JSP – Bryan Basham, Kathy Sierra & Bert Bates
- Core Servlets and Java Server Pages – Marty Hall
- Java Servlets and JSP –Joel Murach – 3rd Edition



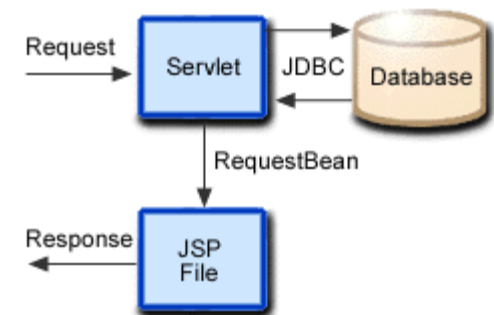
Introdução

- O servidor mantém apenas uma instância de cada **servlet**.
- Cada request de usuário resulta num thread manuseado pelos métodos **doGet** ou **doPost**.



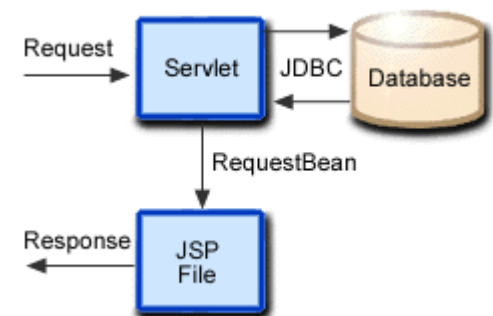
Ciclo de Vida de um Servlet

- Controlado pelo web container;
- Processo iniciado ao atender a 1ª requisição;
- Também é possível configurar para o web container carregar o Servlet automaticamente no **web.xml**.



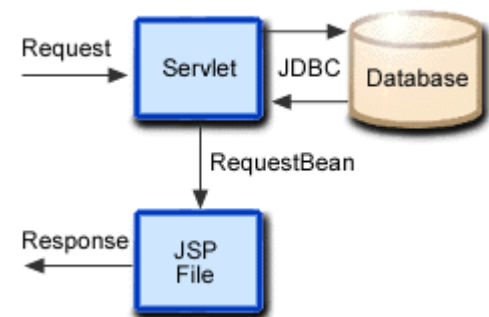
Método init()

- Quando o servlet é criado, o método **init()** é executado.
- Assim, aí é o melhor local para se instaurar código de inicialização (**setup**).



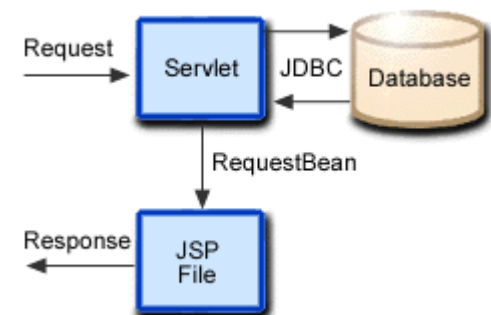
Criação de Threads

- Após a execução do método **init()**, cada requisição do usuário resulta em um thread que chama o método **service()**.
- O método **service()** checa o tipo de request HTTP (**GET**, **POST**, **PUT**, **DELETE**, etc) e chama o método apropriado para o devido tratamento.

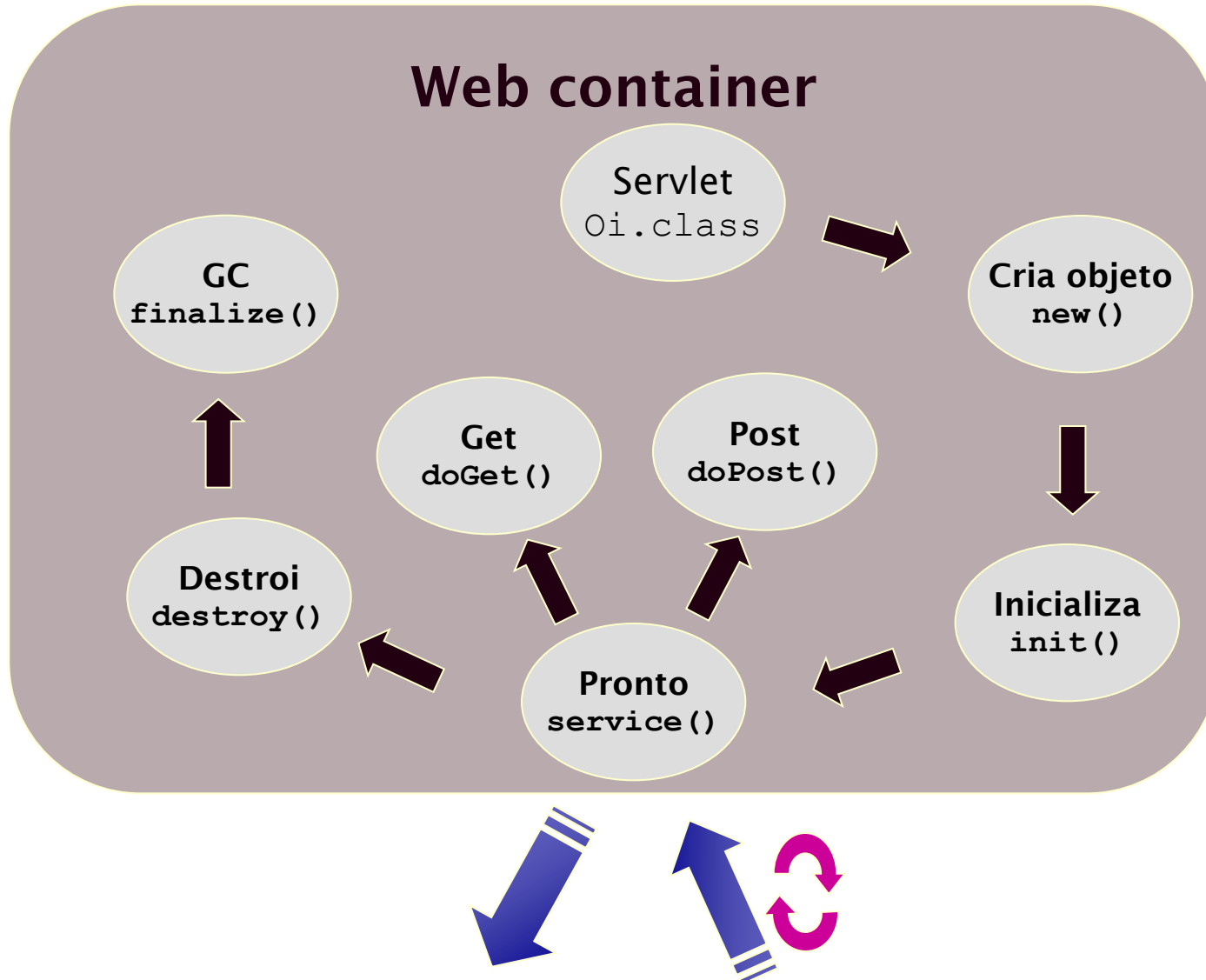


Finalização de um Servlet

- Finalmente, quando o servidor decide descarregar o servlet, ele primeiro chama o método **destroy()**.

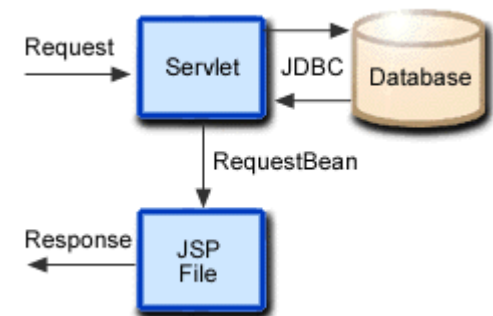


Ciclo passo a passo



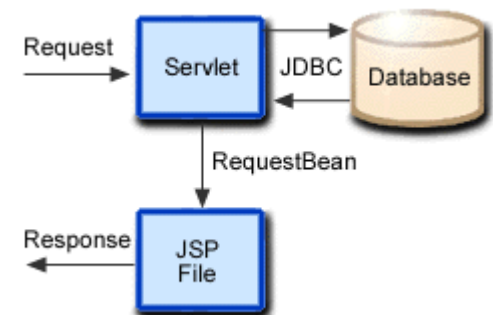
Método init()

- É chamado apenas quando o **servlet** é criado;
- **Não** é chamado novamente em cada request do usuário;
- Assim, somente é usado em tarefas de inicialização (setup).



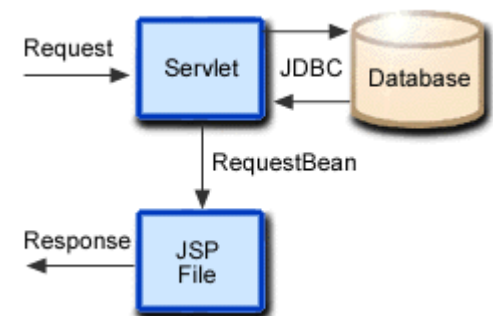
Quando o Servlet é criado ?

- Quando o usuário o invoca por meio da **URL** correspondente, ou quando o servidor for iniciado...
- Tudo depende de como o servlet foi registrado...



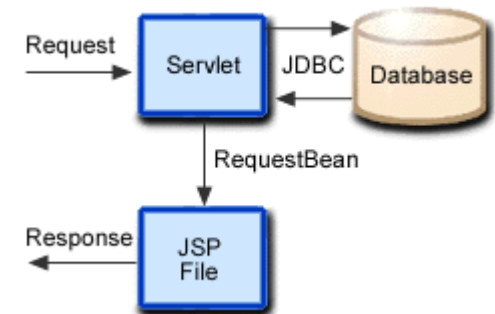
Quando o Servlet é criado ?

- Se não for explicitamente registrado no servidor, o servlet será criado quando o usuário fizer o **primeiro request**, via **URL**.



Método init()

- Há **duas versões**;
- Uma sem argumentos e outra que tem como argumento o objeto **ServletConfig**.



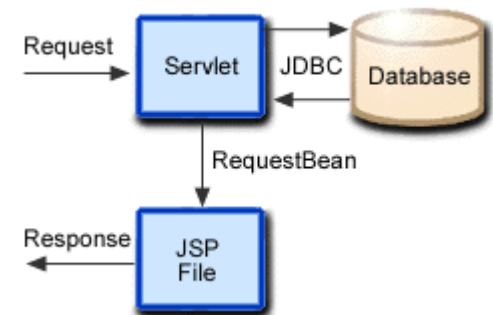
Método init() – Primeira Versão

- **Sem** argumentos;
- Usado quando o servlet não necessita de informações do servidor.



Método `init()` – Primeira Versão

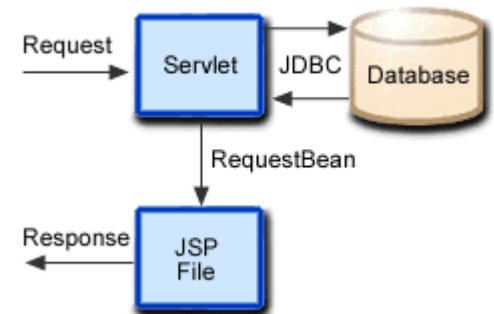
```
public void init ()  
    throws ServletException {  
    // codigo de inicializacao...  
}
```



Atividade – 3

- ⊕ Servlet utilizando o método **init()** – sem parâmetros – para imprimir uma lista de 10 números.

Obs. Utilizar o ambiente Eclipse / Tomcat



```
package br.com.qualitsys.servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Atividade_03
 */
public class Atividade_03 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private int[] tab_numeros = new int[10];

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Atividade_03() {
        super();
        // TODO Auto-generated constructor stub
    }
}
```




```
// metodo init() é chamado somente quando o  
// servlet é primeiramente carregado  
// e antes do processamento do primeiro request...  
  
public void init() throws ServletException {  
  
    for(int i=0; i < tab_numeros.length; i++) {  
        tab_numeros[i] = i;  
    }  
  
}
```



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

    response.setContentType("text/html");

    PrintWriter out = response.getWriter();

    String Titulo = "Atividade 03 com ECLIPSE";

    String docType
        = "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \" +
          \"Transitional//EN\">\n";

    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>" + Titulo + "</TITLE></HEAD>\n" +
                "<BODY BGCOLOR=\"#6699FF\">\n" +
                "<H1 ALIGN=CENTER>" + Titulo + "</H1>\n" +
                "<OL>");
```



```
        for (int i=0; i<tab_numeros.length; i++) {
            out.println( "<H1 ALIGN=CENTER>" + tab_numeros[i] + "</H1>\n" );
        }

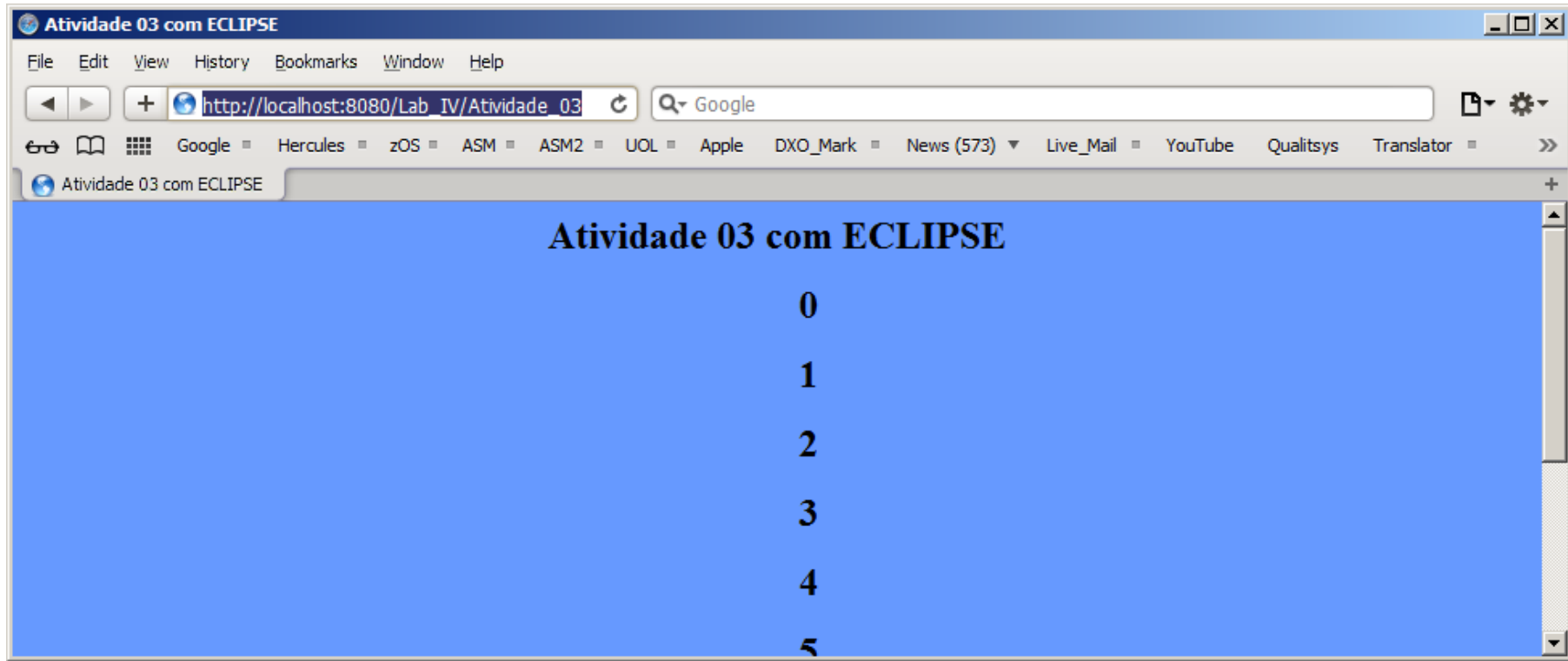
        out.println("</OL>" + "</BODY></HTML>");
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
    }
}
```



Atividade - 3

http://localhost:8080/Lab_IV/Atividade_03



Método init() – Segunda Versão

- Usada quando o servlet necessita de informações gravadas no servidor antes de completar o código de inicialização.
- Por exemplo, o servlet pode precisar de parâmetros de performance, arquivo de senhas, cookies, parâmetros de BD, etc...



Método init() – Segunda Versão

```
public void init (ServletConfig config)  
    throws ServletException {  
    // codigo de inicializacao...  
  
}
```



Atividade – 4

Servlet utilizando o método **init()** utilizando parâmetros de inicialização.

Obs. Utilizar o ambiente Eclipse / Tomcat



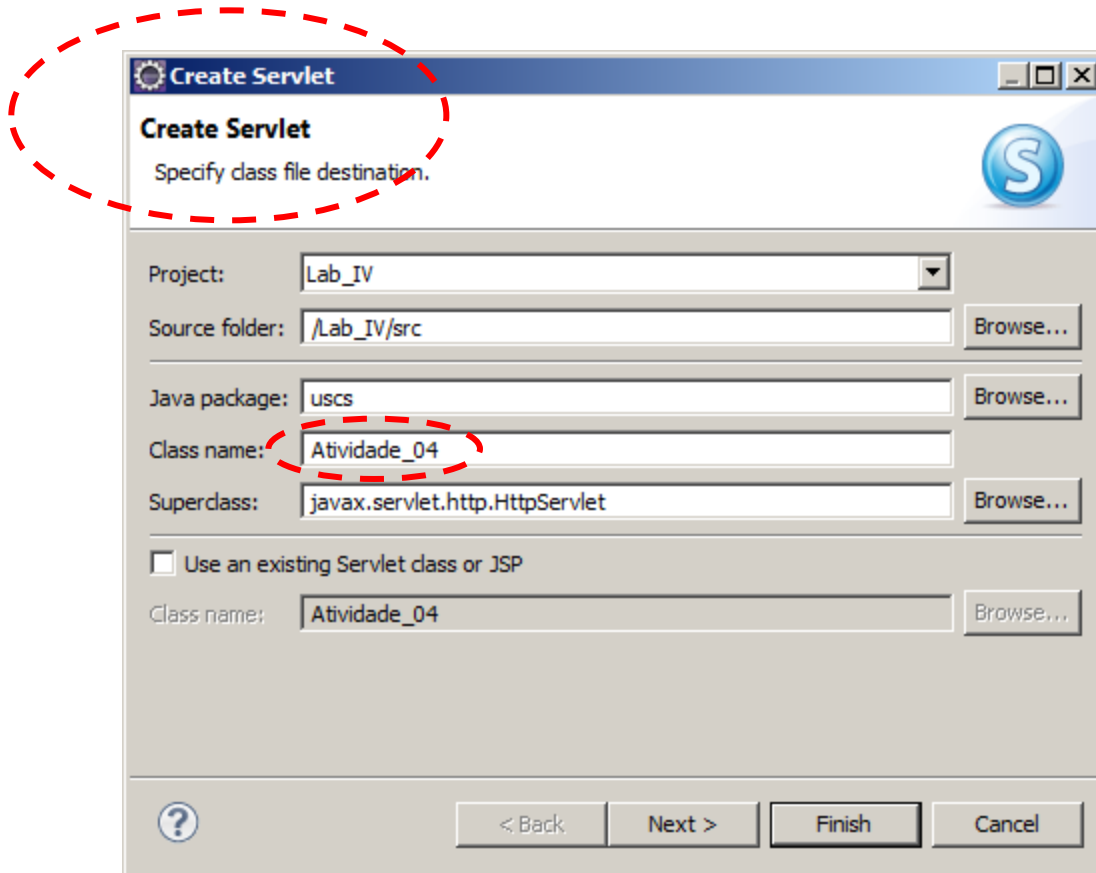
Atividade – 4

- Escrever um servlet (**Atividade_04**) que efetua leitura de duas variáveis definidas no servidor, por meio do arquivo web.xml.
- As variáveis são: **PARAM1 = “USCS”**
PARAM2 = 10
- O valor de **PARAM2** indica quantas vezes o valor de **PARAM1** deve ser exibido na saída.



Criação do Servlet – Eclipse

Clique no botão **Next**



Create Servlet
Specify class file destination.

Project: Lab_IV

Source folder: /Lab_IV/src Browse...

Java package: uscs Browse...

Class name: Atividade_04

Superclass: javax.servlet.http.HttpServlet Browse...

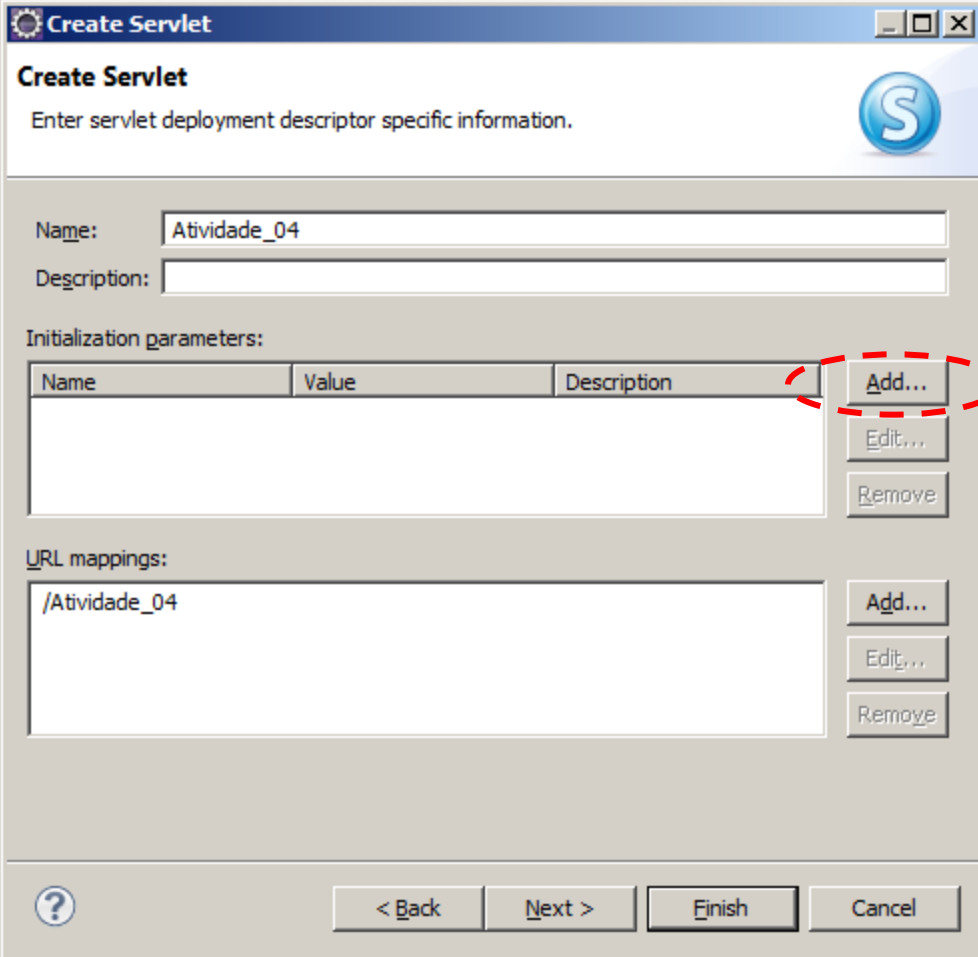
☐ Use an existing Servlet class or JSP

Class name: Atividade_04 Browse...

? < Back Next > Finish Cancel



Clique no botão **Add**



Create Servlet

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

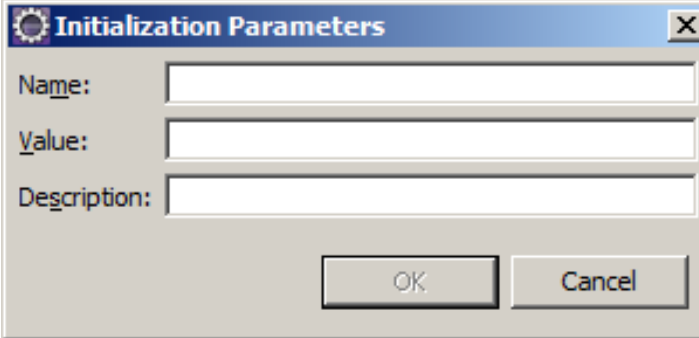
Name	Value	Description			

URL mappings:

/Atividade_04



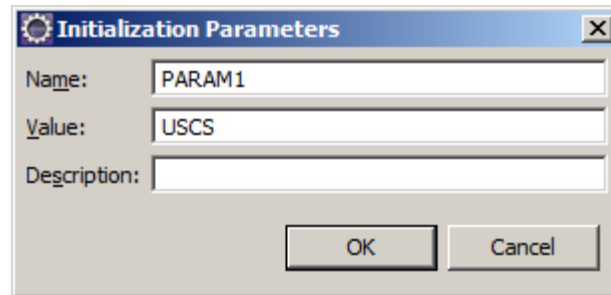
Entre com os parâmetros, conforme Atividade...



A screenshot of a Java Swing dialog box titled "Initialization Parameters". The dialog has a standard title bar with a close button (X). Inside, there are three text input fields labeled "Name:", "Value:", and "Description:". At the bottom right, there are two buttons: "OK" and "Cancel".



Definição dos Parâmetros...



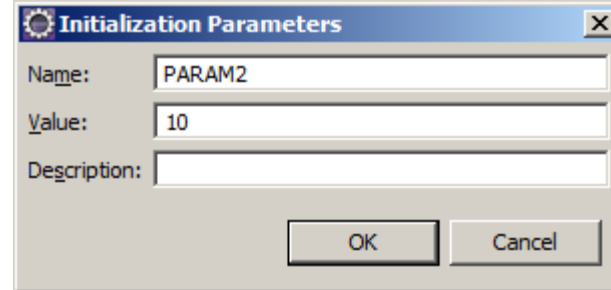
Initialization Parameters

Name:

Value:

Description:

OK Cancel



Initialization Parameters

Name:

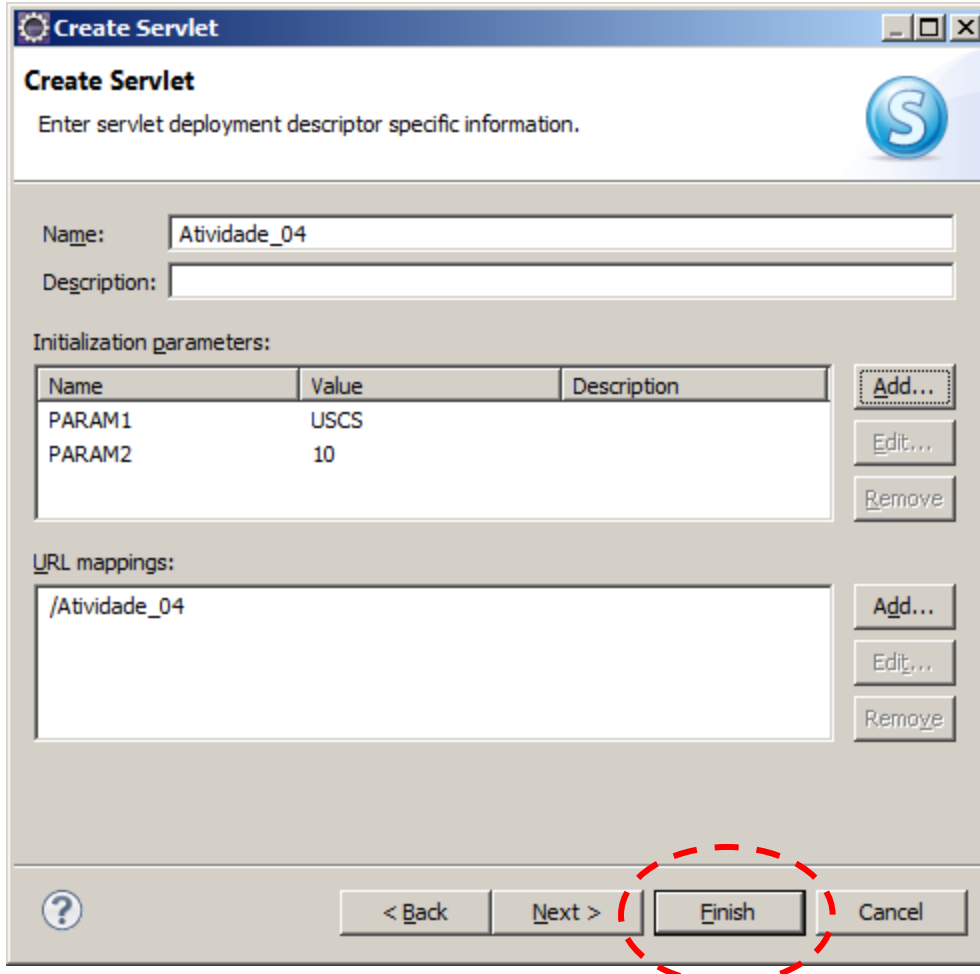
Value:

Description:

OK Cancel



Clique no botão **Finish** ...



Create Servlet

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description
PARAM1	USCS	
PARAM2	10	

Buttons: Add..., Edit..., Remove

URL mappings:

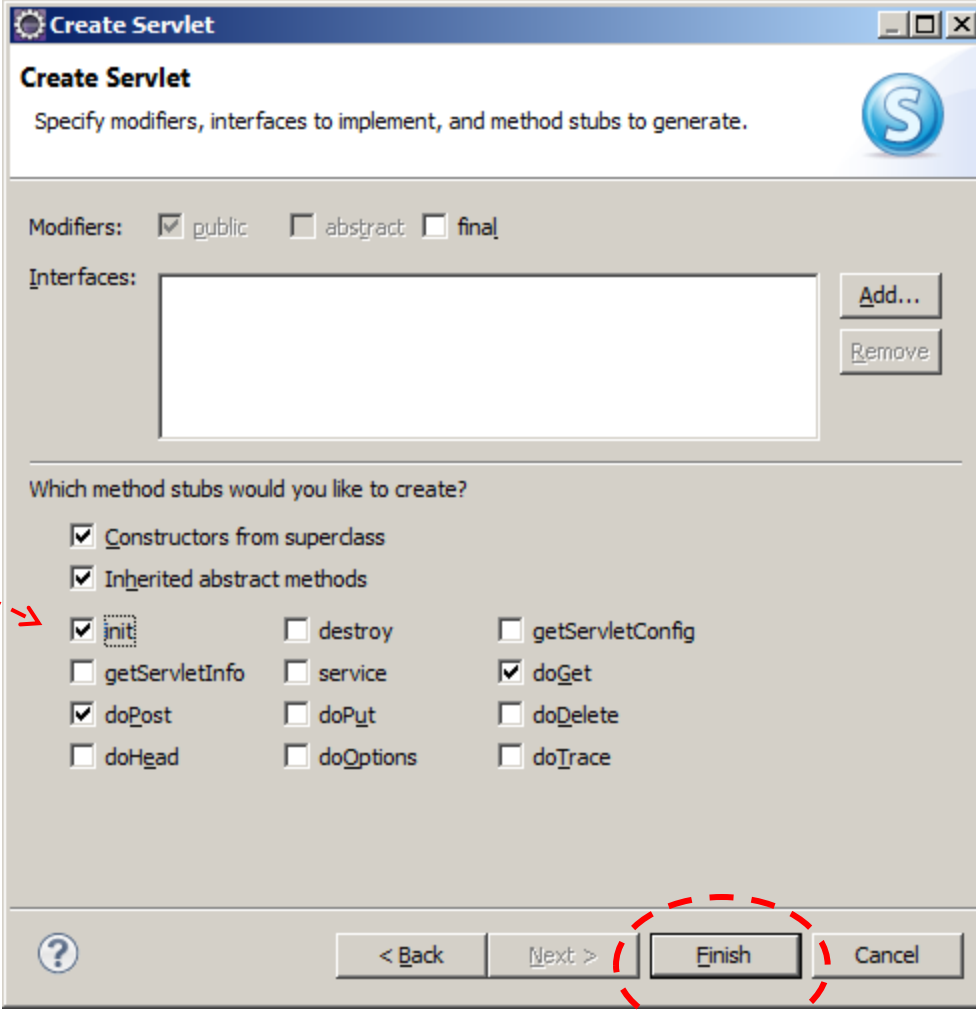
/Atividade_04

Buttons: Add..., Edit..., Remove

Navigation buttons: < Back, Next >, **Finish** (circled in red), Cancel



Marque que irá usar o Método **init()** e **Finish**



The image shows the "Create Servlet" dialog box in an IDE. It has a title bar "Create Servlet" and a subtitle "Specify modifiers, interfaces to implement, and method stubs to generate." Below the subtitle, there are sections for "Modifiers" (with checkboxes for `public`, `abstract`, and `final`), "Interfaces" (with an empty list box and "Add..." and "Remove" buttons), and "Which method stubs would you like to create?". The "Which method stubs" section contains a grid of checkboxes for various methods: `Constructors from superclass`, `Inherited abstract methods`, `init` (highlighted with a red dashed arrow), `destroy`, `getServletConfig`, `getServletInfo`, `service`, `doGet` (checked), `doPost`, `doPut`, `doDelete`, `doHead`, `doOptions`, and `doTrace`. At the bottom, there are buttons for "< Back", "Next >", "Finish" (circled with a red dashed line), and "Cancel".

Which method stubs would you like to create?

<input checked="" type="checkbox"/> Constructors from superclass	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input checked="" type="checkbox"/> Inherited abstract methods	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input checked="" type="checkbox"/> init	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace
<input checked="" type="checkbox"/> doPost		
<input type="checkbox"/> doHead		

< Back Next > **Finish** Cancel



Código do Servlet

```
package uscs;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Atividade06
 */
public class Atividade_04 extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private String mensagem;
    private String mensagem_padrao = "Mensagem não lida do servidor...";
    private int repete;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Atividade_04() {
        super();
        // TODO Auto-generated constructor stub
    }
}
```



Código do Servlet

```
public void init(ServletConfig config)
    throws ServletException {

    super.init(config);
    mensagem = config.getInitParameter("PARAM1");
    if (mensagem == null) {
        mensagem = mensagem_padrao;
    }
    try {
        String repeteString =
            config.getInitParameter("PARAM2");
        repete = Integer.parseInt(repeteString);
    } catch (NumberFormatException nfe) {
    }

}
```



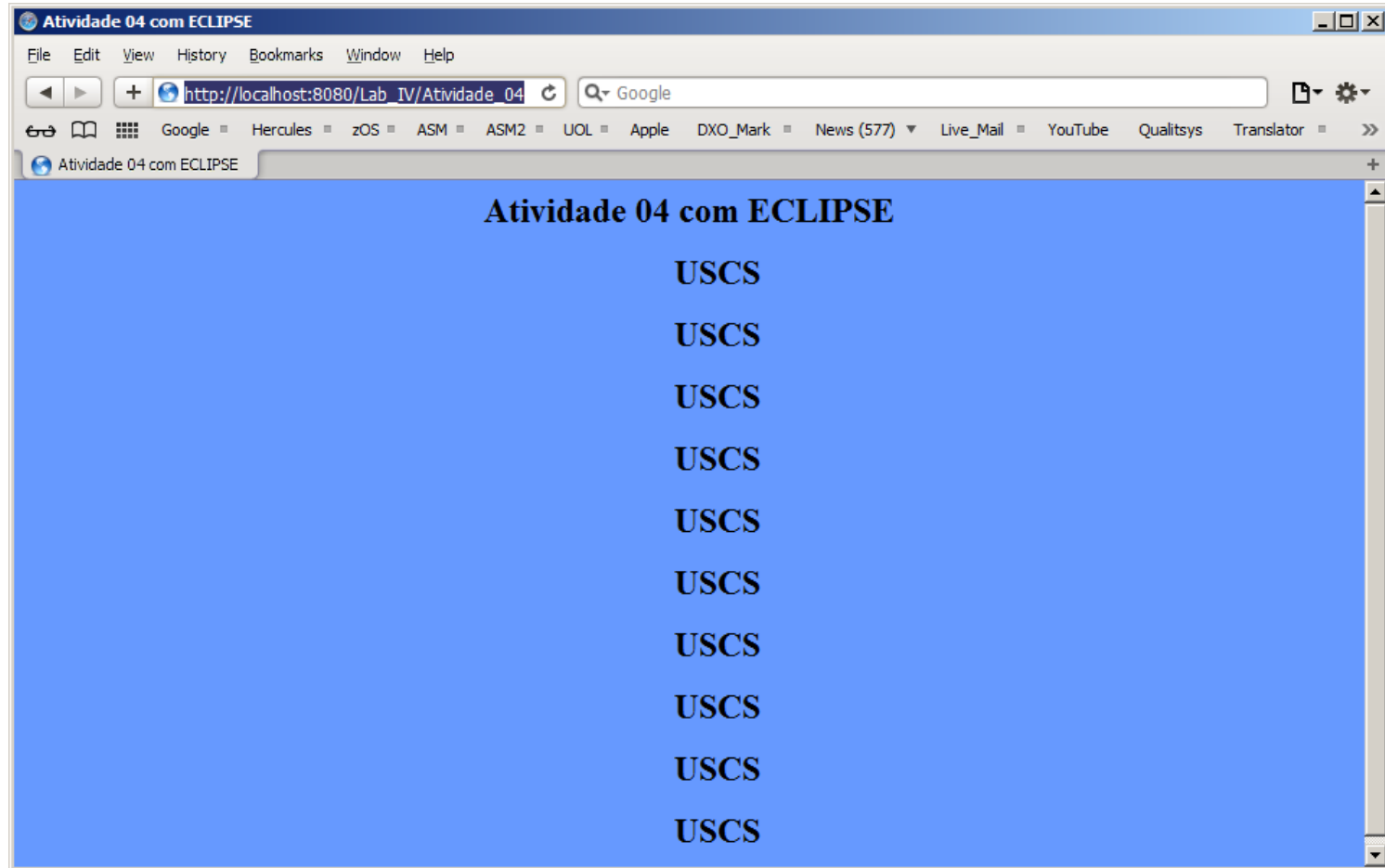

```
protected void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, IOException {
// TODO Auto-generated method stub

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

String Titulo = " Atividade 04 com ECLIPSE ";
String docType =
    "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \" +
    \"Transitional//EN\">\n";
out.println(docType +
    "<HTML>\n" +
    "<HEAD><TITLE>" + Titulo + "</TITLE></HEAD>\n" +
    "<BODY BGCOLOR=\"#6699FF\">\n" +
    "<H1 ALIGN=CENTER>" + Titulo + "</H1>\n" +
    "<OL>");
for(int i=0; i<repete; i++) {
    out.println( "<H1 ALIGN=CENTER>" + mensagem + "</H1>\n" );
}
out.println("</OL>" +
    "</BODY></HTML>");
}
}
```

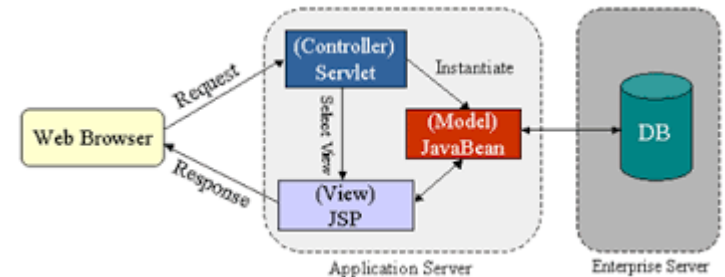


http://localhost:8080/Lab_IV/Atividade_04



Método service()

- Cada vez que o servidor recebe um request para um servlet, ele cria um novo thread e chama o método **service()**;
- O método **service()** checa o método **HTTP** do request (**GET,POST,PUT, DELETE,etc...**) e chama o método apropriado para o tratamento do request.



Método service()

```
public void service(  
    HttpServletRequest request,  
    HttpServletResponse response)  
    throws ServletException,  
           IOException {  
    // código Servlet ...  
  
}
```

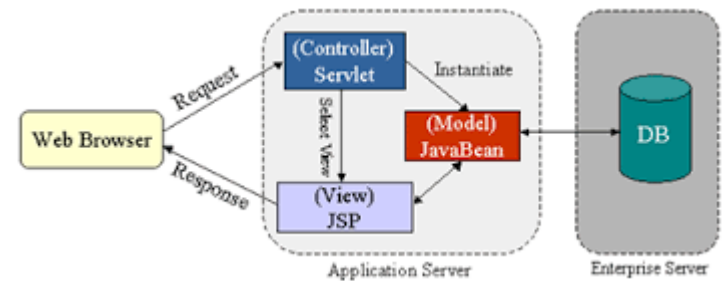


Método GET

⊕ O método **GET** é usado quando queremos pesquisar ou passar informações para uma outra página usando a **URL** da página.

⊕ Por exemplo:

<http://www.uscs.br/busca.jsp?cod=548981>



Método GET

<http://www.uscs.br/busca.jsp?cod=548981>

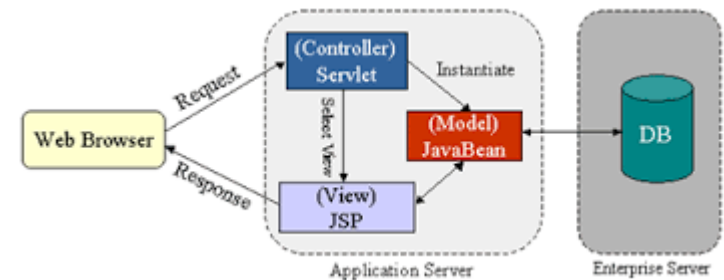


- ⊕ Tudo que é inserido após “?” é considerado **Form Data** ou **QueryData** e é a forma mais simples de se passar dados pela WEB.
- ⊕ A informação pode ser acessada pela combinação nome=valor, onde no caso anterior nome = **cod** e valor = **548981**.



Transferindo dados ao servidor

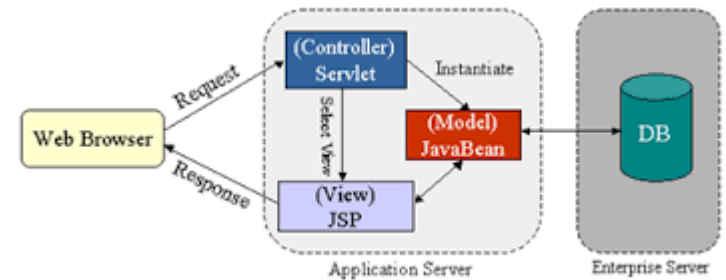
- ⊕ Os dados são lidos por meio do método **getParameter** da classe **HttpServletRequest**.
- ⊕ Os parâmetros (case-sensitives) são passados como argumentos ao método **getParameter** e separados por “&”.
- ⊕ A função é usada da mesma forma tanto para o método **GET** quanto para o **POST**.



Atividade – 5

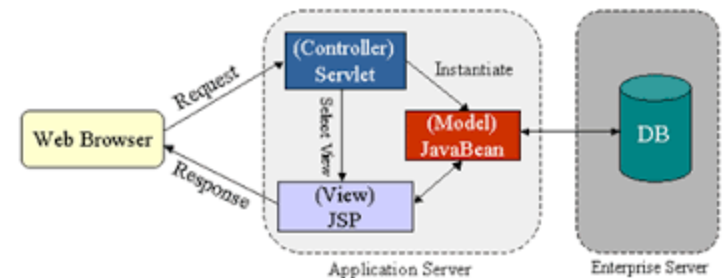
Lendo dois parâmetros de forma explícita (via URL)

Obs. Utilizar o ambiente ECLIPSE / Tomcat



Atividade – 5

1. Escrever um **servlet** (Atividade_05) que efetua leitura de dois parâmetros (**PARAM1** e **PARAM2**) de forma explícita – via URL - e os imprime em uma lista bullet.
2. Obs. Os parâmetros devem ser informados através de argumentos da **URL**.



Código do Servlet

```
package qualit;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
/**  
 * Servlet implementation class Servlet_05  
 */  
public class Atividade_05 extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
  
    public Atividade_05() {  
        super();  
        // TODO Auto-generated constructor stub  
    }  
}
```

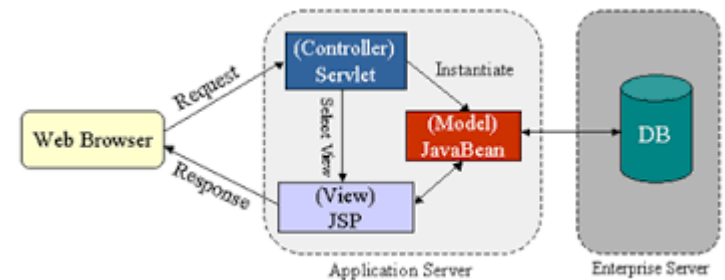


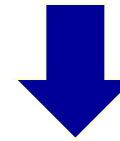
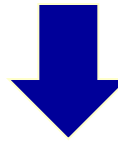
Código do Servlet

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
// TODO Auto-generated method stub

response.setContentType("text/html");
PrintWriter out = response.getWriter();
String Titulo = "Atividade 5 com Eclipse - Lendo parâmetros via URL...";
String docType = "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \" +
    \"Transitional//EN\">\n";

out.println(docType +
    "<HTML>\n" +
    "<HEAD><TITLE>" + Titulo + "</TITLE></HEAD>\n" +
    "<BODY BGCOLOR=\"#6699FF\">\n" +
    "<H1 ALIGN=CENTER>" + Titulo + "</H1>\n" +
    "<UL>\n" +
    " <LI><B> PARAM1: </B>:  "
+ request.getParameter("PARAM1") + "\n" +
    " <LI><B> PARAM2: </B>:  "
+ request.getParameter("PARAM2") + "\n" +
    "</UL> \n" +
    "</BODY></HTML>");
```





http://localhost:8080/Lab_IV/Atividade_05?PARAM1=QUALIT&PARAM2=Computacao

