Programação WEB

Java Server Pages e Servlets

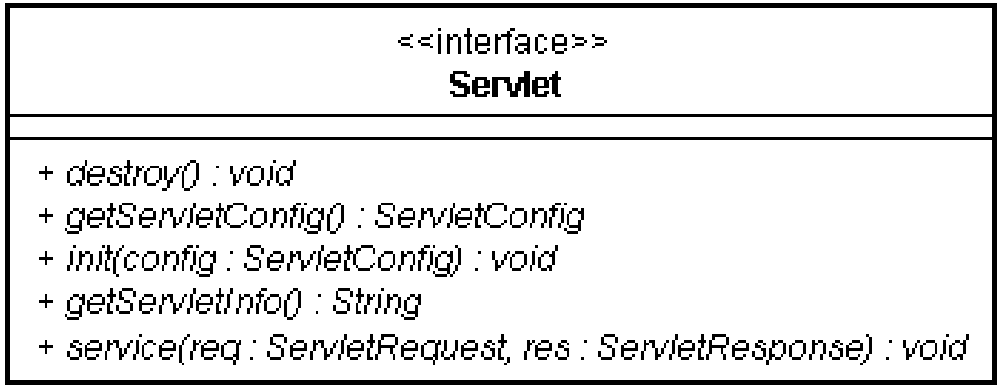
Douglas Nassif Roma Junior

douglas.junior@grupointegrado.br

Conteúdo

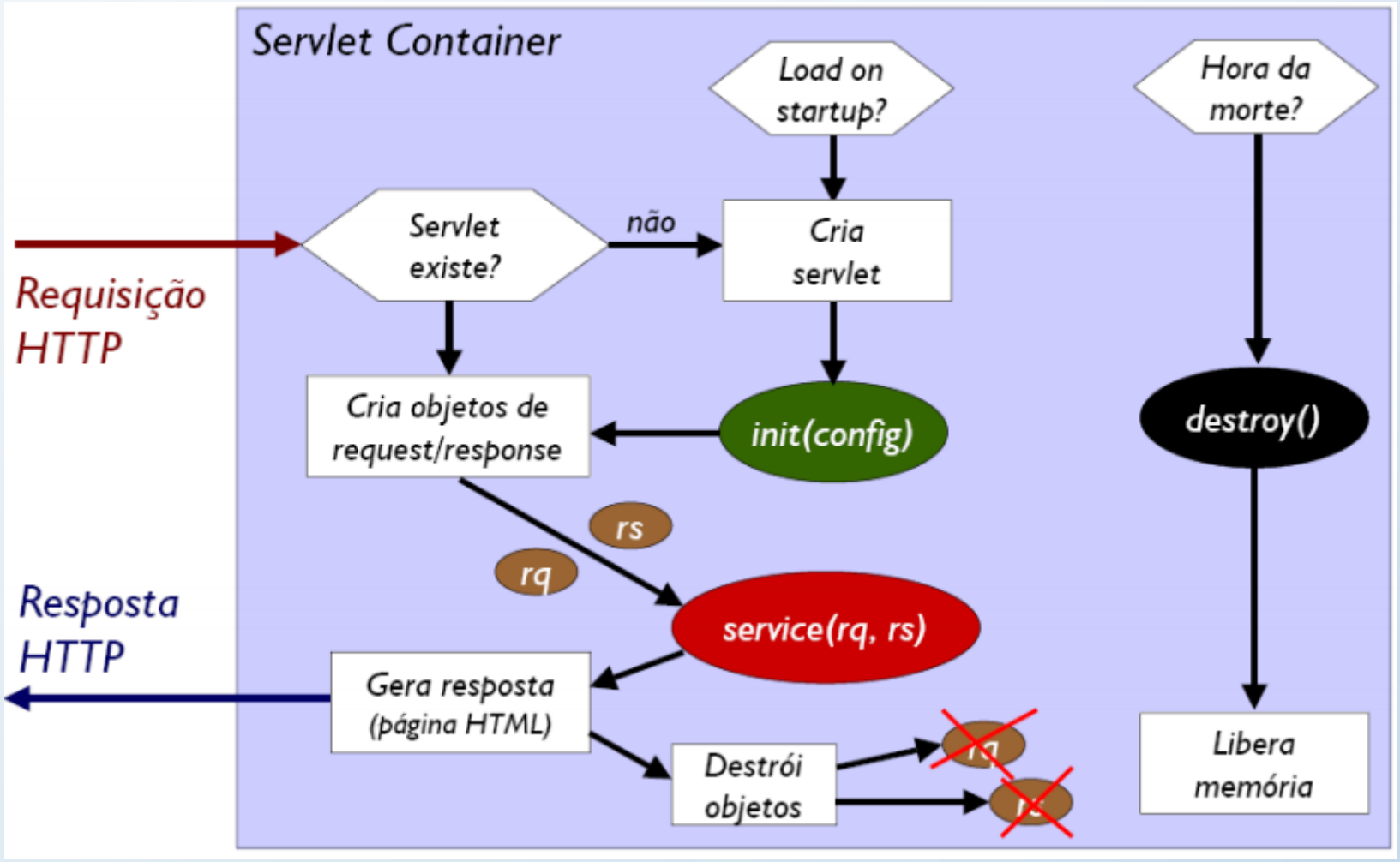
- O pacote javax.servlet;
- O ciclo de vida do servlet;
- Como obter informações de configuração;
- Como compartilhar informações entre servlets;
- A interface ServletRequest;
- A interface ServletResponse;
- A classe GenericServlet.

O pacote javax.servlet

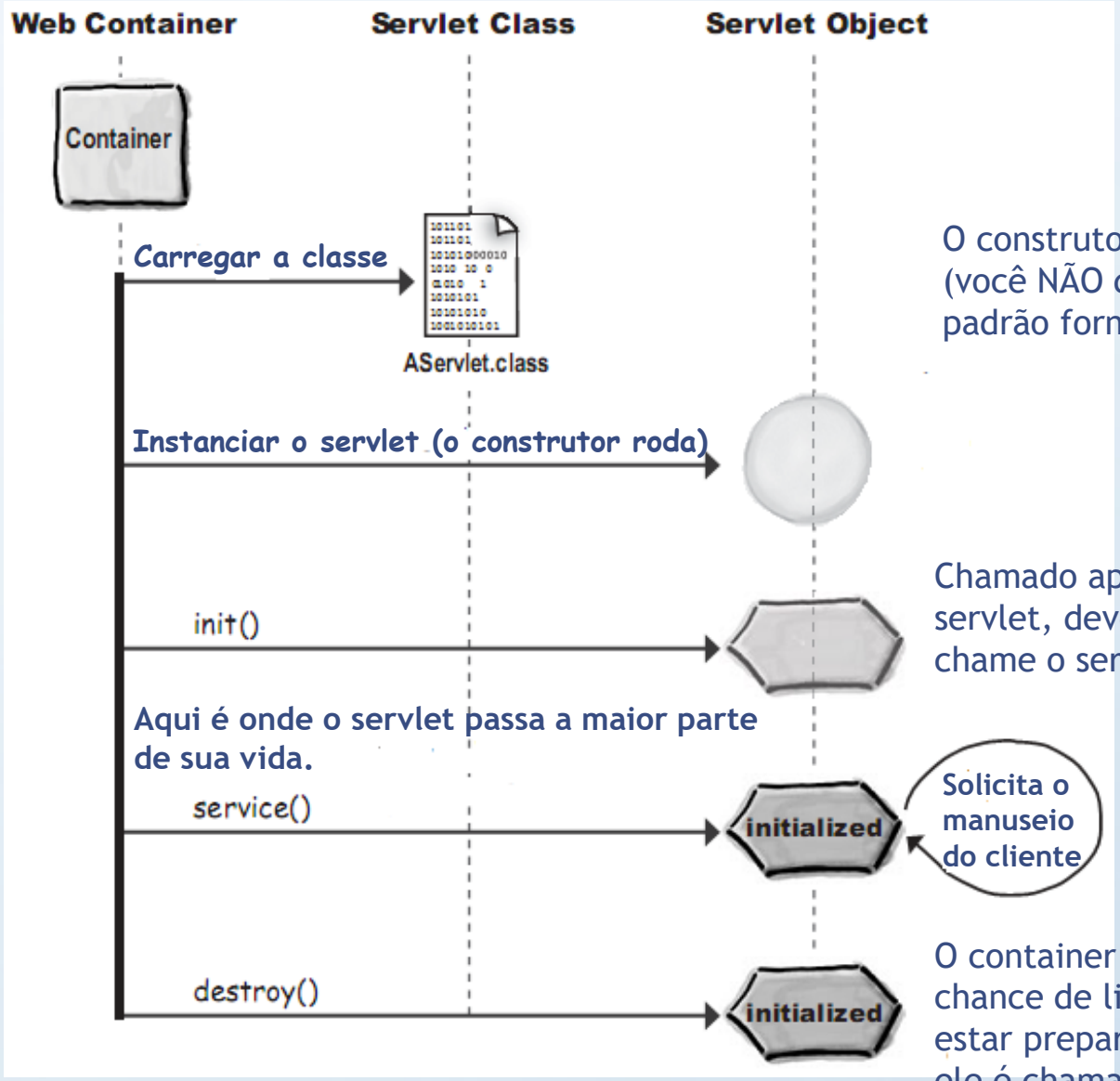


`void destroy()` Método chamado pelo servlet container para indicar que o servlet esta sendo desalocado.
`ServletConfig getServletConfig()` Retorna um objeto ServletConfig que contem parâmetros de inicialização para este servlet.
`String getServletInfo()` Retorna informações sobre o servlet, tal como autor, versão e copyright.
`void init(ServletConfig config)` Método chamado pelo servlet container para indicar que o servlet esta sendo colocado em serviço.
`void service(ServletRequest req, ServletResponse res)` Método chamado pelo servlet container para permitir ao servlet responder uma requisição.

O ciclo de vida do servlet



O ciclo de vida do servlet



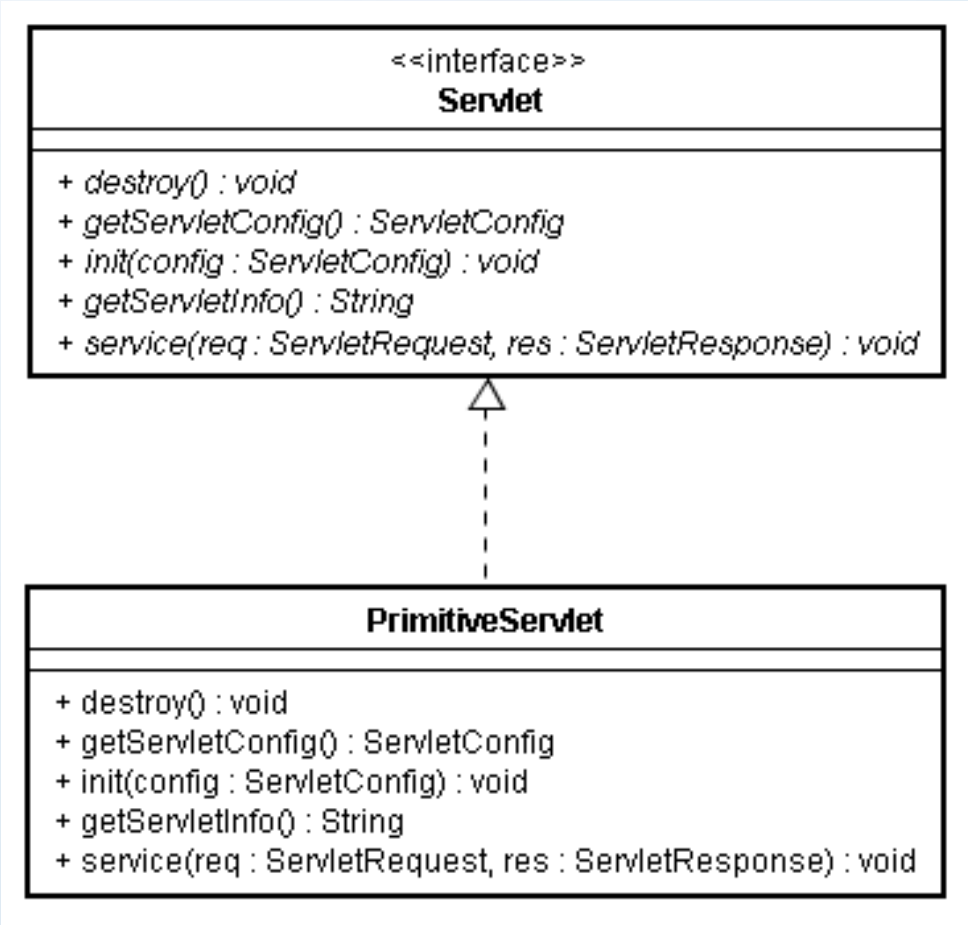
O construtor padrão da sua classe Servlet roda (você NÃO deve escrever um construtor. Use o padrão fornecido com o seu compilador.

Chamado apenas uma vez durante a vida do servlet, deve completar antes que o container chame o service().

Trata as solicitações do cliente doGet(), doPost(), etc. (Cada solicitação roda em uma thread separada).

O container chama para dar ao servlet uma chance de limpar antes de morrer(quer dizer, estar preparado para virar lixo). Como o init(), ele é chamado apenas uma vez.

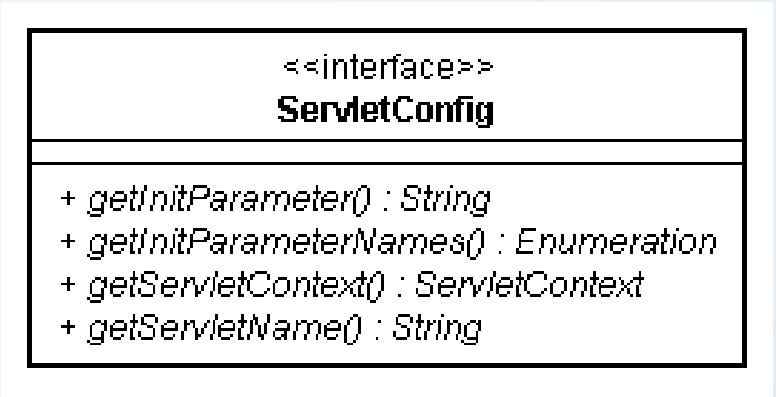
Implemente `PrimitiveServlet`



PrimitiveServlet

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

public class PrimitiveServlet implements Servlet {
    public void init(ServletConfig config) throws ServletException {
        System.out.println("Init");
    }

    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
        System.out.println("Service");
    }

    public void destroy() {
        System.out.println("Destroy");
    }

    public ServletConfig getServletConfig() {
        return null;
    }

    public String getServletInfo() {
        return null;
    }
}
```

ServletConfig



`String getInitParameter(String name)` Retorna uma String contendo o valor do parâmetro de inicialização `name`, ou `null` se o parâmetro não existe.
`Enumeration getInitParameterNames()` Retorna os nomes dos parâmetros de inicialização do servlet como uma Enumeration de objetos String, ou uma `Enumeration` vazia se o servlet não apresentar parâmetros de inicialização.
`ServletContext getServletContext()` Retorna uma referência ao ServletContext em que o chamador esta executando.
`String getServletName()` Retorna o nome da instância do servlet.

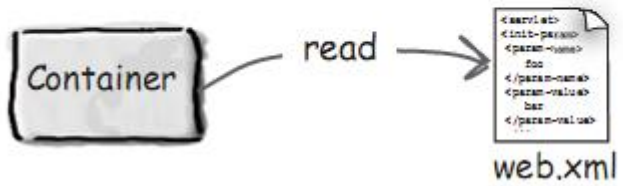
Por que o uso de parâmetros de inicialização é uma estratégia interessante?

web.xml

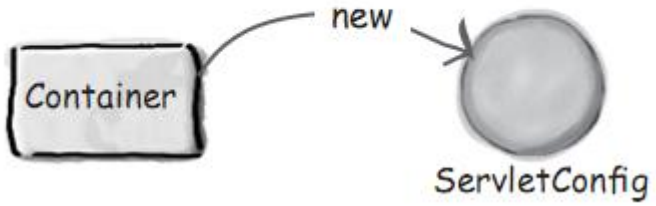
```
<servlet>
  <description></description>
  <display-name>ConfigDemoServlet</display-name>
  <servlet-name>ConfigDemoServlet</servlet-name>
  <servlet-class>ConfigDemoServlet</servlet-class>
  <init-param>
     <param-name>adminEmail</param-name>
     <param-value>douglas.nassif@grupointegrado.br</param-value>
  </init-param>
  <init-param>
     <param-name>adminContactNumber</param-name>
     <param-value>9982-4317</param-value>
  </init-param>
</servlet>
```

ServletConfig

1. O container lê o Deployment Descriptor para este servlet, incluindo os parâmetros init (<init-param>)

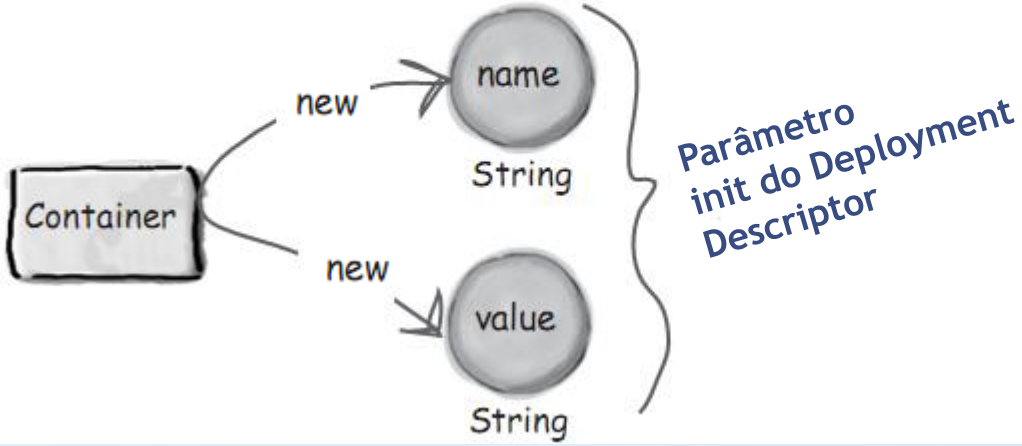


2. O Container cria uma nova instância ServletConfig para este servlet.

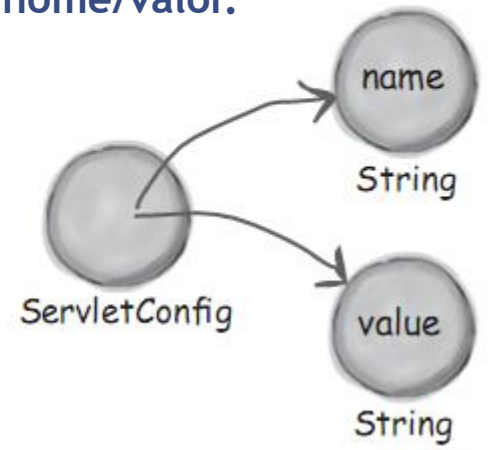


ServletConfig

③ O container cria um par de strings nome/valor para cada parâmetro init do servlet. Considere que temos apenas um.

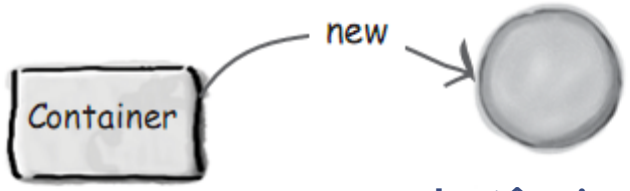


④ O Container dá ao ServletConfig as referências para os parâmetros init nome/valor.



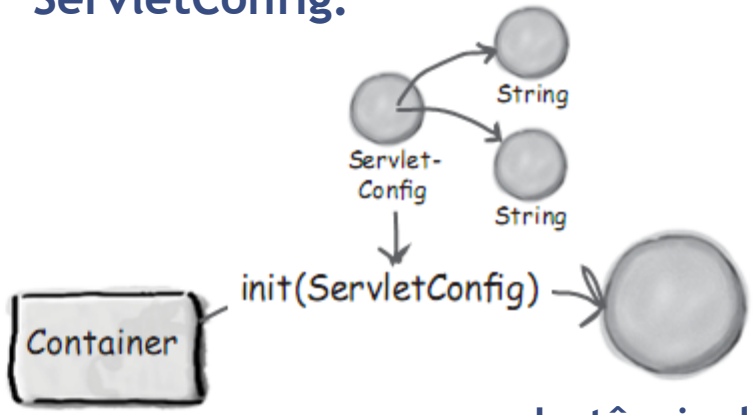
ServletConfig

5 O container cria uma nova instância da classe servlet.



Instância de MyServlet.class

6 O container chama o método init() do Servlet, passando a referência para o ServletConfig.



Instância de MyServlet.class

Exercício

- Criar uma classe chamada `ConfigDemoServlet` que implementa a interface `Servlet`.

- `ConfigDemoServlet` deve exibir no console os parâmetros de inicialização configurados no `web.xml`.

ConfigDemoServlet

```
import javax.servlet.*;
import java.io.IOException;
import java.util.Enumeration;

public class ConfigDemoServlet implements Servlet {
    public void init(ServletConfig config) throws ServletException {
        Enumeration parameters = config.getInitParameterNames();
        while(parameters.hasMoreElements()){
            String parameter = (String) parameters.nextElement();
            System.out.println("Parameter name:" + parameter);
            System.out.println("Parameter value:" +
                config.getInitParameter(parameter));
        }
    }
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
    }

    public void destroy(){
    }

    public String getServletInfo(){
        return null;
    }

    public ServletConfig getServletConfig(){
        return null;
    }
}
```

Exercício

- **Problema:** Preciso exibir/utilizar os parâmetros de inicialização no método `service()`.

- **Solução:** Crie um objeto `ServletConfig` para armazenar uma cópia do objeto `ServletConfig` recebido no método `init()`.

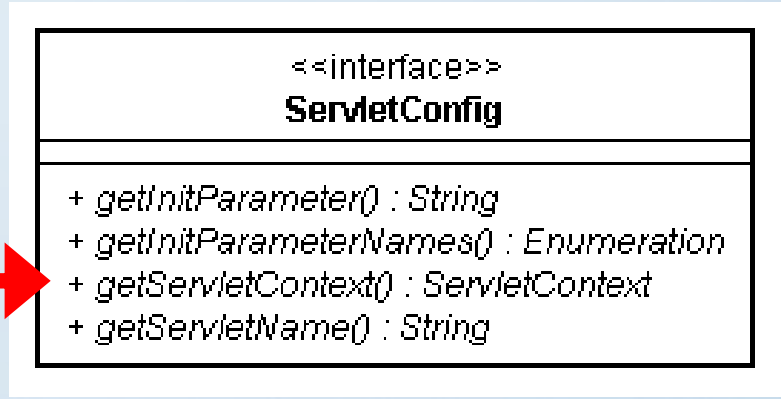
- Crie uma classe chamada `ReserveConfigServlet` que implementa a solução acima.

ReserveConfigServlet

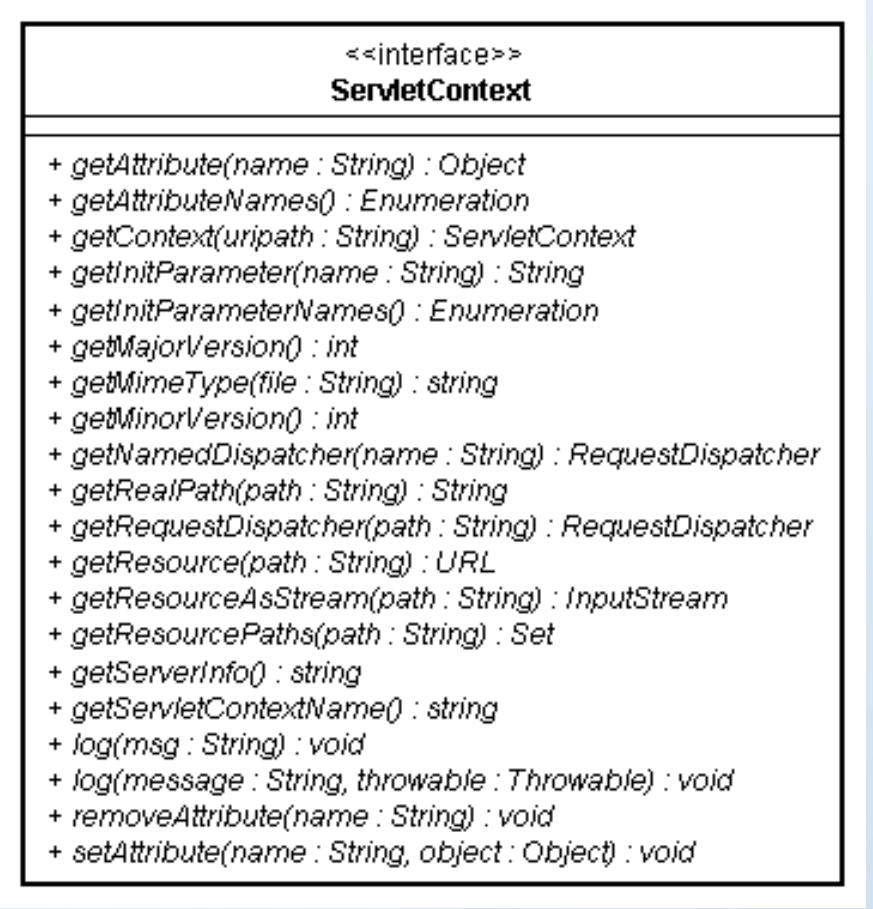
```
import javax.servlet.*;
import java.io.IOException;
import java.util.Enumeration;

public class ReserveConfigServlet implements Servlet {
    ServletConfig servletConfig;

    public void init(ServletConfig config) throws ServletException {
        this.servletConfig=config;
    }

    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
        Enumeration parameters = this.getServletConfig().getInitParameterNames();
        while(parameters.hasMoreElements()){
            String parameter = (String) parameters.nextElement();
            System.out.println("Parameter name:" + parameter);
            System.out.println("Parameter value:" + this.getServletConfig().getInitParameter(parameter));
        }
    }
    public void destroy() {
    }

    public ServletConfig getServletConfig() {
        return this.servletConfig;
    }

    public String getServletInfo() {
        return null;
    }
}
```

O contexto de um `servlet`

- O contexto de um servlet é o ambiente onde o servlet executa.
- A interface `ServletContext` define um conjunto de métodos que um servlet usa para se comunicar com o servlet container.
- Você consegue uma referência ao objeto `ServletContext` pelo método `getServletContext()` da interface `ServletConfig`.



A interface `ServletContext`



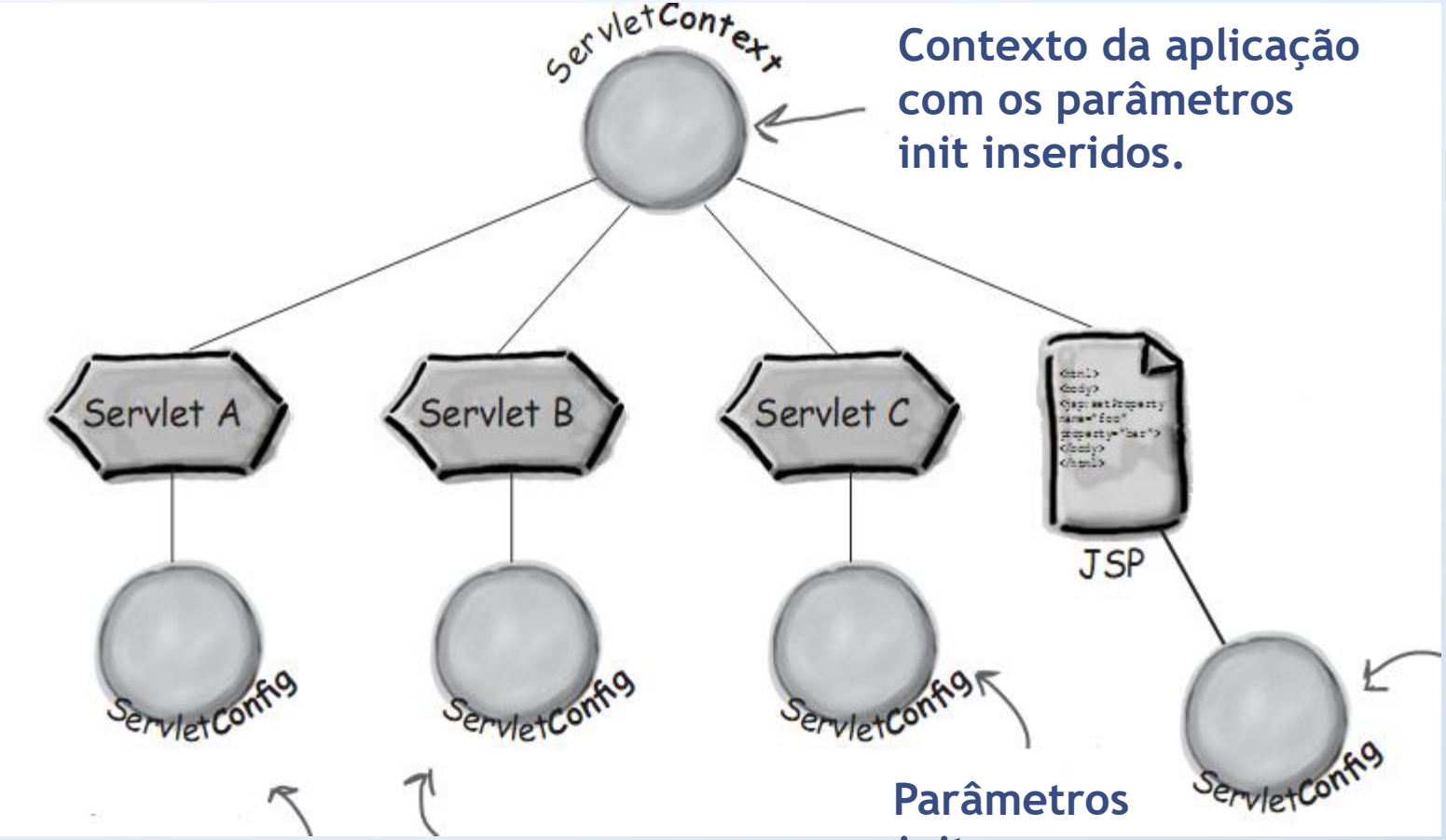
Os parâmetros `init` do contexto

```
(...)
<context-param>
  <param-name>adminEmail</param-name>
  <param-value>douglas.nassif@grupointegrado.br</param-value>
</context-param>
(...)
```

- Exercício:
  - Exiba os parâmetros iniciais do contexto.

Exercício

```
import java.io.IOException;
import java.util.Enumeration;
import javax.servlet.*;

public class ContextServlet implements Servlet {
    private ServletConfig config;
    public void init(ServletConfig config) throws ServletException {
        this.config = config;
    }
    public ServletConfig getServletConfig() {
        return config;
    }
    public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
        ServletContext context = this.getServletConfig().getServletContext();
        Enumeration e = context.getInitParameterNames();
        while(e.hasMoreElements()){
            String parameter = (String) e.nextElement();
            System.out.println("Parameter:"+parameter);
            System.out.println("Value:"+context.getInitParameter(parameter));
        }
    }
    public String getServletInfo() {
        return null;
    }

    public void destroy() {
    }
}
```



Contexto da aplicação com os parâmetros init inseridos.

Sim, os JSP são transformados em Servlets de primeira classe e também recebem seus próprios ServletConfig

Parâmetros init para o Servlet A inseridos

Parâmetros init para o Servlet B inseridos

Parâmetros init para o Servlet C inseridos

A interface ServletContext

`Object getAttribute(String name)`
Retorna o atributo relacionado ao `name` dado em um objeto, ou `null` se não existir nenhum atributo para o `name`.

`Enumeration getAttributeNames()`
Retorna uma `Enumeration` contendo os nomes dos atributos disponíveis dentro o contexto.

`ServletContext getContext(String urpath)`
Retorna um objeto ServletContext que corresponde a uma específica URL do servidor.

`int getMajorVersion()`
Retorna a maior versão da API Java Servlet que este servlet container suporta.

`String getMimeType(String file)`
Retorna o tipo MIME do arquivo especificado, ou null se o tipo MIME não for conhecido.

`int getMinorVersion()`
Retorna a menor versão da API Java Servlet que este servlet container suporta.

`String getRealPath(String path)`
Retorna uma string contendo o caminho real para um dado caminho virtual.

`String getServerInfo()`
Retorna o nome e a versão do servlet container que o servlet esta rodando.

`String getServletContextName()`
Retorna o nome desta aplicação web correspondente a este `ServletContext` como especificado no descritor `web.xml` pelo elemento `display-name`.

`void log(String msg)`
Escreve uma mensagem ao arquivo de log do servlet log file.

`void removeAttribute(String name)`
Remove o atributo expresso em `name` para este contexto.

`void setAttribute(String name, Object obj)`
Este método armazena um objeto no `ServletContext` e liga o objeto ao nome dado. Se o nome já existir no contexto, o antigo objeto ligado da lugar ao objeto passado por este método.

Exemplos

```
public void service(ServletRequest req, ServletResponse res)
        throws ServletException, IOException {
    ServletContext context = this.getServletConfig().getServletContext();
    //Cria atributos na aplicação
    context.setAttribute("login","admin");
    context.setAttribute("pass","xxx");
    context.setAttribute("removeAttribute", "yyy");
    //Remove atributos na aplicação
    context.removeAttribute("removeAttribute");
    //Retorna todos os atributos da aplicação
    Enumeration e = context.getAttributeNames();
    while(e.hasMoreElements()){
        String attributte = (String) e.nextElement();
        String value = context.getAttribute(attributte).toString();
        System.out.println("Attribute: "+attributte);
        System.out.println("Value: "+value);
    }
    //Retorna o caminho da aplicação
    System.out.println("Path:"+context.getContextPath());
    //Retorna o caminho absoluto dado um caminho relativo.
    System.out.println("Real Path: "+context.getRealPath(context.getContextPath()));
    //Retorna a maior e menor versão que o container suporta.
    System.out.println("Major Version: "+context.getMajorVersion());
    System.out.println("Minor Version: "+context.getMinorVersion());
    //Retona o nome do Servlet que foi definido na DD pela tag <display-name>
    System.out.println("Context Name: "+context.getServletContextName());
    //Escreve no log do servidor um registro.
    context.log("Arquivo x acessado com sucesso");
    //Retorna o Tipo MIME de um arquivo no servidor.
    System.out.println("Tipo MIME:"+context.getMimeType("editar.png"));
}
```

Exercício

- Criar uma classe chamada `ContextDemoServlet` que implementa a interface `Servlet`.

- `ContextDemoServlet` deve exibir no console os atributos os atributos definidos pelo servlet container.

- Além disso exiba as informações da API Java Servlet e do servidor.

ContextDemoServlet

```
import java.io.IOException;
import java.util.Enumeration;
import javax.servlet.*;

public class ContextDemoServlet implements Servlet {
    ServletConfig servletConfig;
    public void init(ServletConfig config) throws ServletException{
        this.servletConfig=config;
    }
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
        ServletContext servletContext = this.getServletConfig().getServletContext();
        Enumeration attributes = servletContext.getAttributeNames();
        while (attributes.hasMoreElements()){
            String attribute = (String)attributes.nextElement();
            System.out.println("Attribute name:"+attribute);
            System.out.println("Attribute value:"+servletContext.getAttribute(attribute));
        }
        System.out.println("Major version:"+servletContext.getMajorVersion());
        System.out.println("Minor version:"+servletContext.getMinorVersion());
        System.out.println("ServerInfo:"+servletContext.getServerInfo());
    }
    public ServletConfig getServletConfig() {
        return this.servletConfig;
    }
    public void destroy() {
    }
    public String getServletInfo() {
        return null;
    }
}
```

Compartilhar informações entre servlets

- Crie duas classes que implementam servlet:
  - `AttributeSetterServlet;`
  - `DisplayAttributeServlet.`

- `AttributeSetterServlet` em seu método `init` deve criar o atributo "`password`" com valor "`dingdong`";

- Em seu método `service` deve exibir no console o valor do atributo.

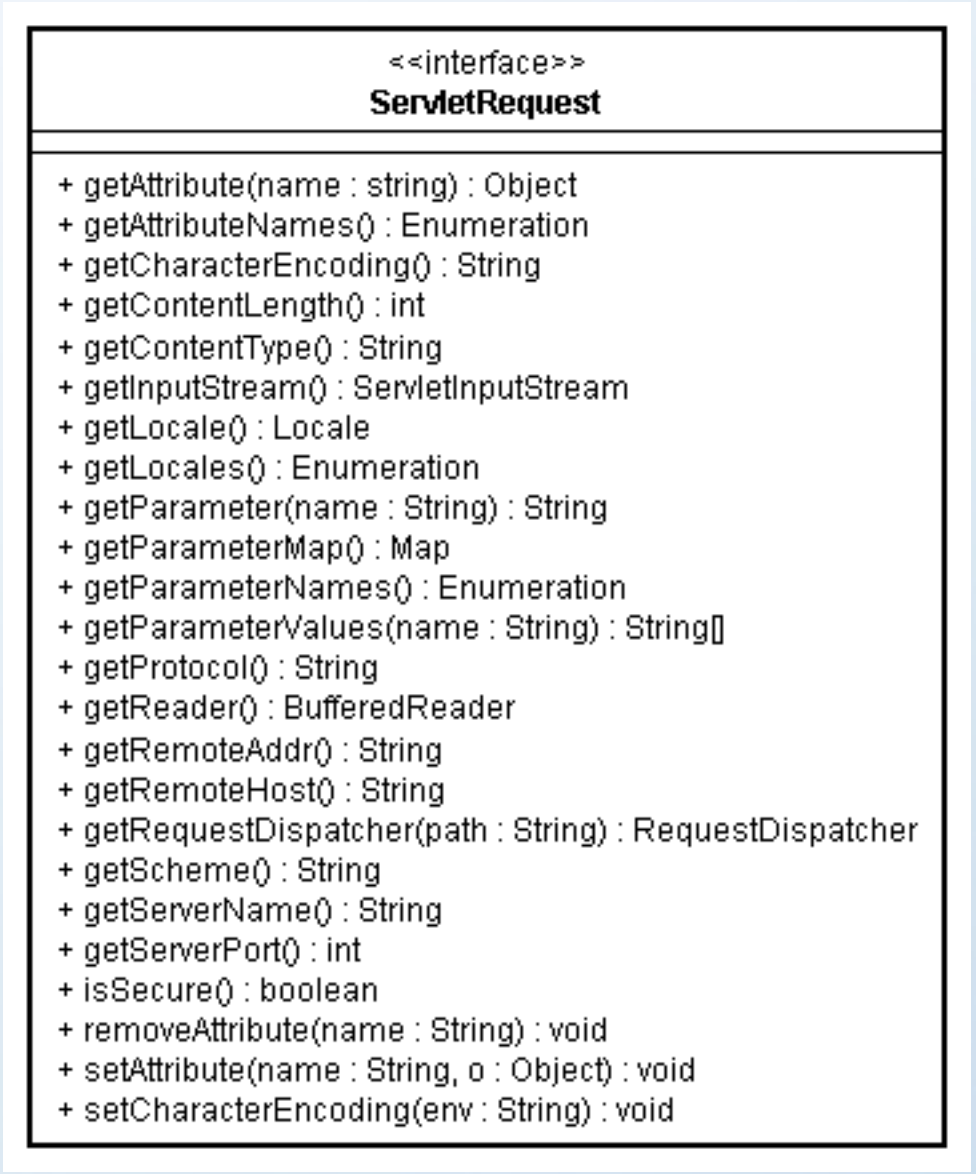
AttributeSetterServlet

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

public class AttributeSetterServlet implements Servlet {
    public void init(ServletConfig config) throws ServletException {
        ServletContext servletContext = config.getServletContext();
        servletContext.setAttribute("password", "dingdong");
    }
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
    }

    public void destroy() {
    }

    public ServletConfig getServletConfig() {
        return null;
    }

    public String getServletInfo() {
        return null;
    }
}
```

DisplayAttributeServlet

```
import java.io.IOException;
import java.util.Enumeration;
import javax.servlet.*;

public class DisplayAttributeServlet implements Servlet {
    ServletConfig servletConfig;
    public void init(ServletConfig config) throws ServletException {
        this.servletConfig = config;
    }
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
        ServletContext servletContext = this.getServletConfig().getServletContext();
        Enumeration attributes = servletContext.getAttributeNames();
        while (attributes.hasMoreElements()){
            String attribute = (String)attributes.nextElement();
            System.out.println("Attribute name:"+attribute);
            System.out.println("Attribute value:"+servletContext.getAttribute(attribute));
        }
    }
    public void destroy() {
    }
    public ServletConfig getServletConfig() {
        return servletConfig;
    }
    public String getServletInfo() {
        return null;
    }
}
```

A interface `ServletRequest`



A interface ServletRequest

`Object getAttribute(String name)` Retorna o valor do atributo `name` como um `Object`, ou `null` se o atributo `name` não existir.'
`Enumeration getAttributeNames()` Retorna uma `Enumeration` contendo os nome dos atributos disponíveis na requisição.
`String getCharacterEncoding()` Retorna os nomes dos *character encoding* usados no corpo da requisição.
`int getContentLength()` Retorna o tamanho, em bytes, do corpo da requisição.
`String getContentType()` Retorna o tipo MIME do corpo da requisição, ou `null` se o tipo for desconhecido.
`ServletInputStream getInputStream()` Fornece o corpo da requisição em dados binários usando um `ServletInputStream`.
`String getParameter(String name)` Retorna o valor de um parâmetro da requisição como uma String, ou `null` se o parâmetro não existir.
`Map getParameterMap()` Retorna a `Map` de parâmetros desta requisição.
`Enumeration getParameterNames()` Retorna uma Enumeration de objetos String contendo os nomes dos parâmetros da requisição.
`String[] getParameterValues(String name)` Retorna um array de objetos String contendo todos os valores do parâmetro `name`, ou `null` se o parâmetro não existir.

A interface ServletRequest

`String getProtocol()` Retorna o nome e a versão do protocolo que a requisição usa formulário, por exemplo (protocol/majorVersion.minorVersion) HTTP/1.1.
`BufferedReader getReader()` Fornece o corpo da requisição em caracteres usando um BufferedReader.
`String getRemoteAddr()` Retorna o endereço de IP do cliente que enviou a requisição.
String `getRemoteHost()` Retorna nome do cliente que enviou a requisição.
String `getScheme()` Retorna o nome do esquema usado nesta requisição, por exemplo, http, https, or ftp.
String `getServerName()` Retorna o nome do servidor que recebe a requisição.
int `getServerPort()` Retorna o numero da porta em que esta requisição foi recebida.
boolean `isSecure()` Retorna um `boolean` indicando se esta requisição foi feita usando um canal seguro, tal como HTTPS.
void `removeAttribute(String` name) Remove um atributo desta requisição.
void `setAttribute(String` name, Object o) Armazena um atributo nesta requisição.

Exercício

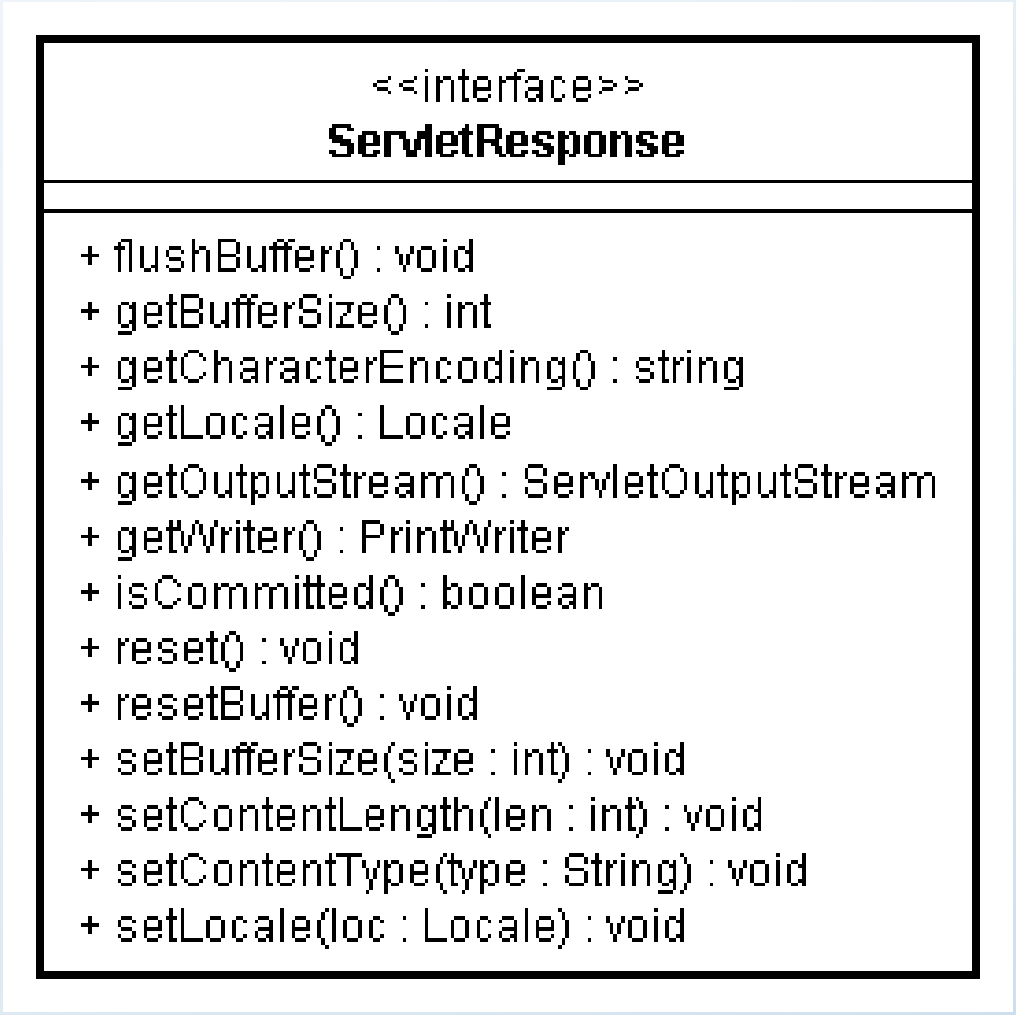
- Crie um formulário a seguir:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Página Inicial</title>
</head>
<body>
    <form action="RequestDemoServlet" method="post">
        <label>
            Author:<br>
            <input type="text" name="author">
        </label>
        <input type="submit" name="submit" />
        <input type="reset" name="reset" />
    </form>

</body>
</html>
```

Exercício

- Criar uma classe chamada `RequestDemoServlet` que implementa a interface `Servlet`.
- `RequestDemoServlet` deve exibir no console o parâmetro definido no formulário. Além disso, explore os métodos da interface `ServletRequest`

A interface `ServletResponse`



A interface ServletResponse

`void flushBuffer()` Força qualquer conteúdo do buffer ser escrito para o cliente.
`int getBufferSize()` Retorna o tamanho do buffer usado para resposta
`String getCharacterEncoding()` Retorna o nome do *character encoding* usado para o envio do corpo nesta resposta.
`String getContentType()` Retorna o tipo de conteúdo usado para o envio do corpo nesta resposta.
`Locale getLocale()` Retorna o `locale` especificado para esta resposta usando o método `setLocale`.
`ServletOutputStream getOutputStream()` Retorna o `ServletOutputStream` adequado para a escrita de dados binários na resposta.
`PrintWriter getWriter()` Retorna um objeto `PrintWriter` que pode enviar caracteres para o cliente.
`boolean isCommitted()` Retorna um *boolean* indicando se a resposta foi *committed*.

`void reset()` Limpa qualquer dado que existe em um buffer bem como o código de status e cabeçalhos.
`void resetBuffer()` Limpa o conteúdo do buffer sem limpar o código de status e cabeçalhos.
`void setBufferSize(int size)` Configura o tamanho do buffer para o corpo da resposta.
`void setCharacterEncoding(String charset)` Configura o *character encoding* da resposta que será enviada ao cliente, por exemplo, UTF-8.
`void setContentLength(int len)` Configura o tamanho do conteúdo do corpo na resposta.
`void setContentType(String type)` Configura o tipo de conteúdo a ser enviado ao cliente, se a resposta ainda não foi *committed*.
`void setLocale(Locale loc)` Configura o `locale` da resposta, se a resposta não foi `committed` ainda.

A interface ServletResponse

- Crie um formulário a seguir:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <form action="ResponseDemoServlet" method="post">
        <label>
            Author:<br>
            <input type="text" name="author">
        </label>
        <input type="submit" name="submit" />
        <input type="reset" name="reset" />
    </form>
</body>
</html>
```

A interface `ServletResponse`

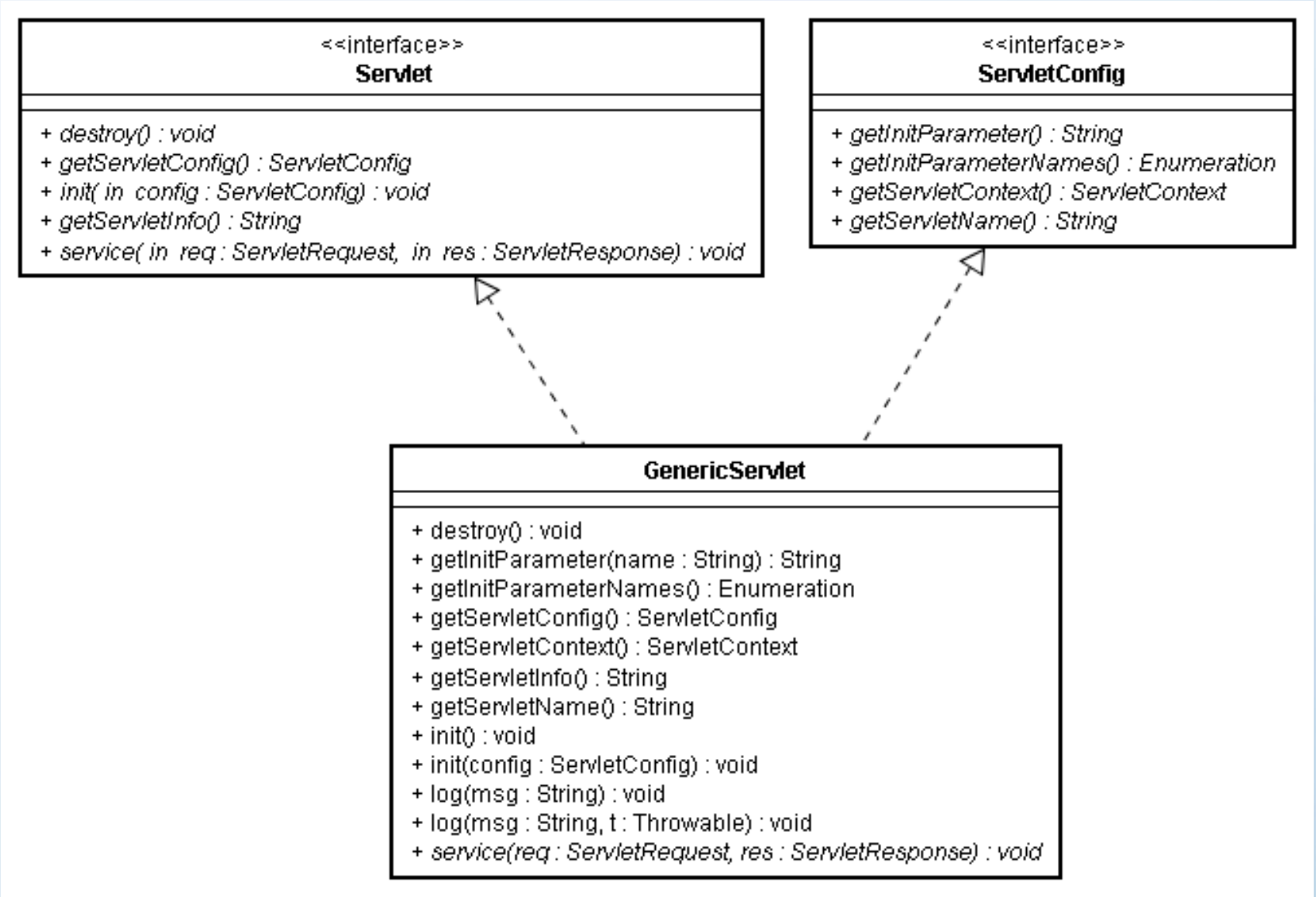
- Criar uma classe chamada `ResponseDemoServlet` que implementa a interface `Servlet`.

- `ResponseDemoServlet` deve exibir no navegador os parâmetro definido no formulário, bem como elementos do cabeçalho HTTP existentes no `ServletRequest`.

GenericServlet

- Problemas:
  - Você precisa oferecer implementação para todos os cinco métodos da interface Servlet.
  - O objeto `ServletConfig` é passado ao init. Você precisa preservar este objeto para usá-lo a partir de outros métodos.
- Criar uma classe que implemente estas interfaces e nossos servlets herdarão esta classe.

GenericServlet

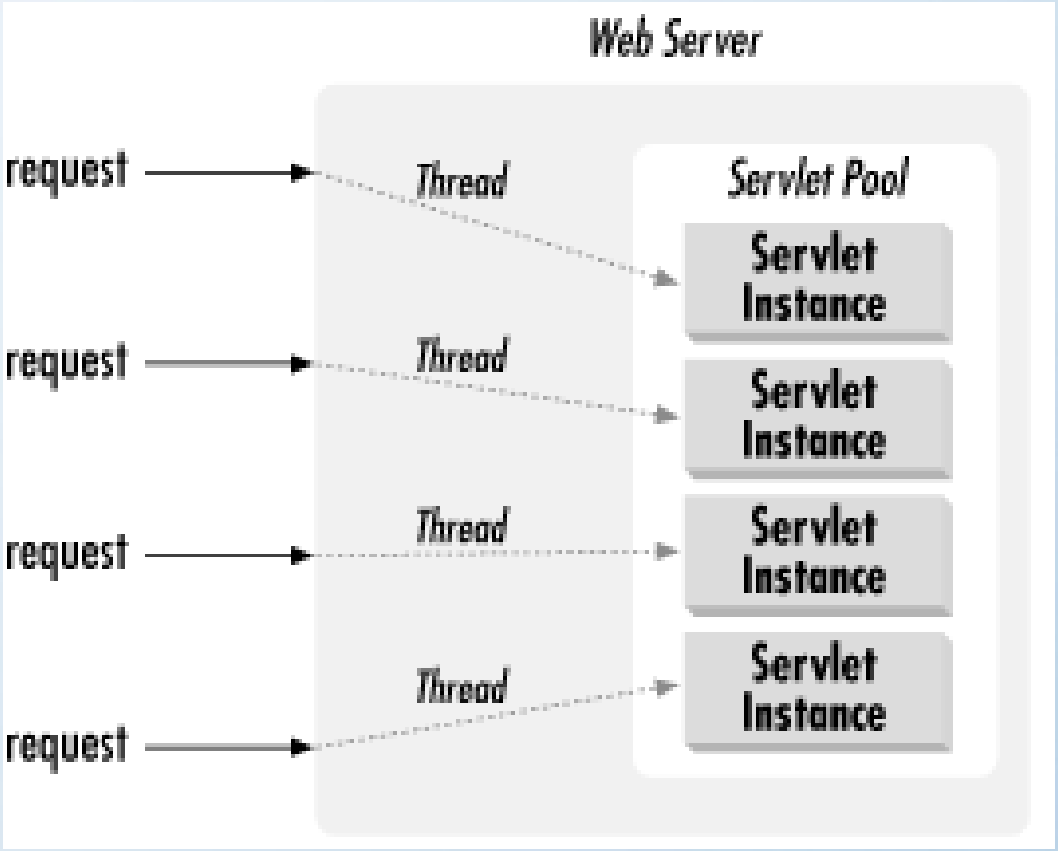


GenericServlet

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;

public class SimpleServlet extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>");
        out.println("Extending GenericServlet");
        out.println("</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<b>Extending GenericServlet makes your code simpler.</b>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Acesso concorrente ao servlet

- Vamos criar um servlet que:
  - Abre um arquivo;
  - Lê um inteiro.
  - Fecha o arquivo.
  - Incrementa.
  - Abre o arquivo.
  - Escreve o inteiro.
  - Fecha o arquivo.



SingleThreadedServlet

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

public class SingleThreadedServlet extends GenericServlet{
    private static final long serialVersionUID = 1L;
    public synchronized void  service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
        int counter = 0;
        try{
            FileReader fr = new FileReader("counter.txt");
            BufferedReader reader = new BufferedReader(fr);
            counter = Integer.parseInt(reader.readLine());
            reader.close();
        }catch (Exception e){
        }
        counter++;
        System.out.println(counter);
        try {
            Thread.sleep(6000);
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        }
        try{
            BufferedWriter write = new BufferedWriter(new FileWriter("counter.txt"));
            write.write(Integer.toString(counter));
            write.close();
        }catch (Exception e) {
        }
    }
}
```

Dúvidas

?