DESENVOLVIMENTO DE APLICAÇÕES PARA INTERNET
Parte VII

Prof. Msc. Felipe Diniz Dallilo

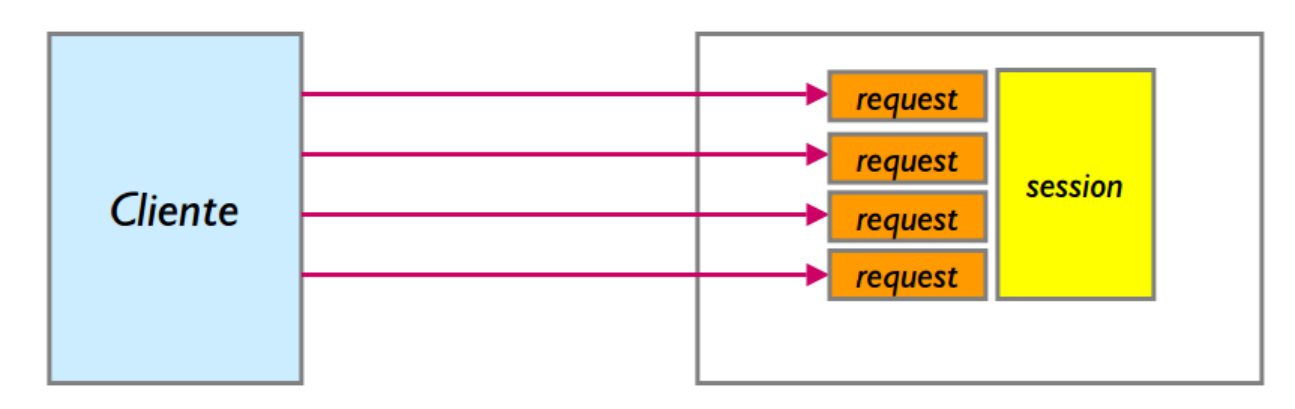
UNIARA ONLINE

Escopos web

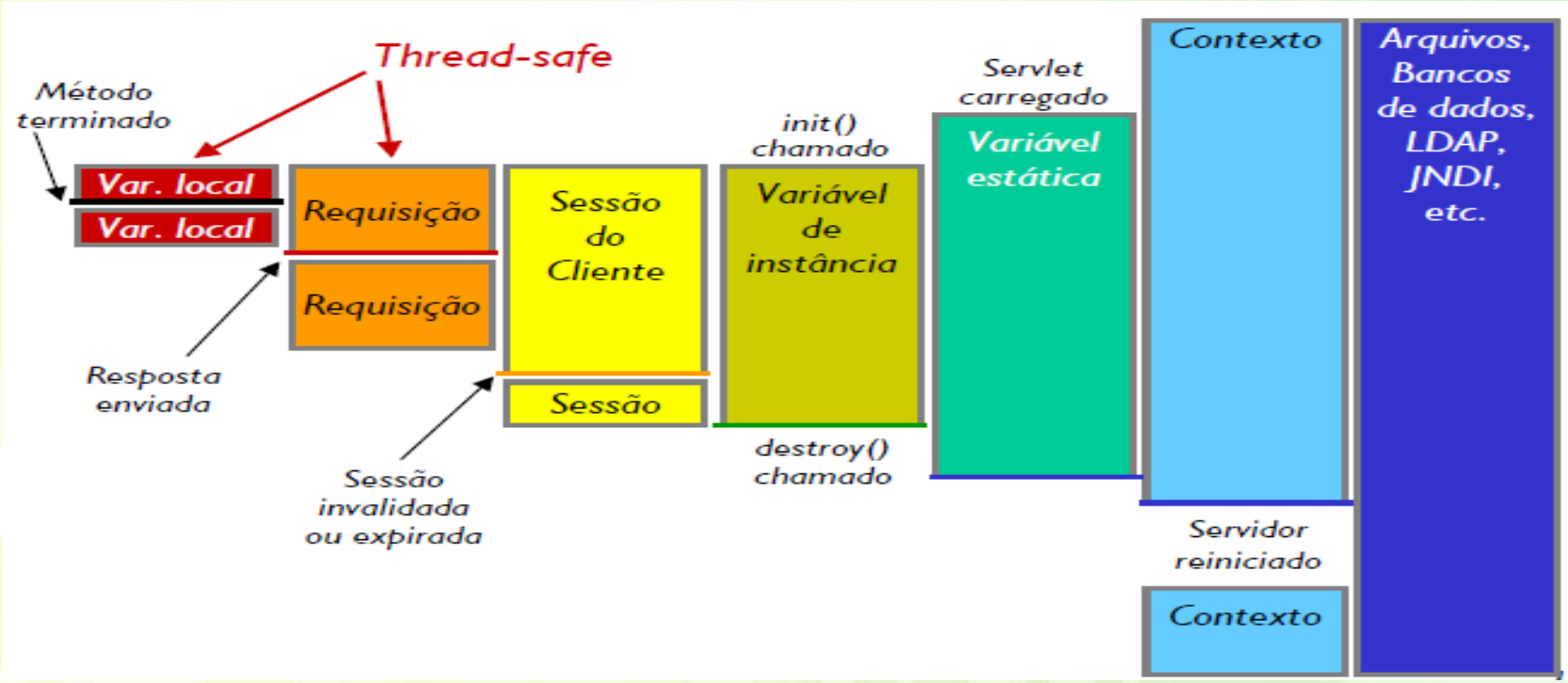
- Existem 4 tipos de escopo web, sendo eles:
- **Requisição -** Como o próprio nome diz (e massivamente utilizado na disciplina) dura apenas o tempo de uma requisição e ao término da mesma todos os dados "setados" no objeto que representa o escopo são perdidos.
- **Sessão -** Quando um usuário acessar o sistema web, ele estabelece com o servidor uma sessão. Os dados setados no objeto que representa este escopo existem desde o instante inicial, quando o usuário acessa a aplicação, até que essa expire por inatividade ou ao fechar o navegador, seja voluntariamente ou finalizada pela aplicação.
- **Aplicação -** Os objetos vivem desde a inicialização do servidor de aplicação até que ele seja finalizado.
- **Page -** Aplica somente ao ciclo de vida das JSP's da sua aplicação. Nele os objetos são definidos em e para cada página e existem apenas para elas mesmas.

Escopo web

- Até o momento, utilizamos o contexto de requisição como único escopo possível para uma aplicação Web, no entanto, como o HTTP não mantém estado da sua sessão, as aplicações Web precisam cuidar de mantê-lo quando necessário.



Escopo web


Thread-safe
Método terminado
Var. local
Var. local
Requisição
Requisição
Resposta enviada
Sessão do Cliente
Sessão
Sessão invalidada ou expirada
init() chamado
Variável de instância
destroy() chamado
Servlet carregado
Variável estática
Contexto
Servidor reiniciado
Contexto
Arquivos, Bancos de dados, LDAP, JNDI, etc.

Escopos web

- **escopo de página -** No escopo de página existe objetos que só estão disponíveis no momento da tradução da página de JSP para HTML (Variáveis scriptlets por exemplo).

  escopo de requisição - O escopo de requisição existe em todo o ciclo de vida da requisição (criação até a resposta).

  escopo de sessão - Cada usuário que acessa o servidor pode manter um estado de comunicação finito com o servidor onde seus objetos definidos são visíveis apenas ao mesmo.

  escopo de aplicação - Todos os usuários possuem acesso ao mesmo conjunto de objetos.

Request

Para alterar um valor na requisição é utilizada a instrução abaixo:

request.setAttribute("chave", "valor");

Para recuperar os valores:

Object valorRequest = request.getAttribute("chave", "valor");

*Chave é String e valor é Object

Requisição

- Estou em um site de compras e clico duas vezes no botão finalizar compras, como o servidor deveria se comportar?

Idempotente

Uma requisição idempotente possui sempre a mesma característica e sempre o mesmo tipo de retorno! Independente de quantas vezes é solicitado, a resposta é a mesma e normalmente não há nenhuma alteração no servidor!

O protocolo HTTP define o GET como idempotente!

Não Idempotente

- Ao contrário do GET, o protocolo HTTP implementa que o POST não é idempotente, em outras palavras, sua finalidade é de atualização de dados no servidor e portanto, trata as requisições de forma diferente caso haja duas requisições em sequencia pelo mesmo usuário!

Session

Para alterar um valor na sessão é utilizada a instrução abaixo:

request.getSession().setAttribute(“chave", "valor");

Para recuperar os valores

Object valorSession = request.getSession().getAttribute(“chave“);

*Chave é String e valor é Object

HTTPSession

O objeto session possui alguns métodos que devem ser destacados:

- IsNew() -- Se é uma nova session
- RemoveAtributte("chave"); -- Remover um atributo pela chave
- Invalidate() -- Destruir Session
- SetMaxInactiveInterval(int) -- Define o tempo máximo de Intervalo inativo (default no web.xml) (Também há get)
- GetLastAccessedTime() -- último acesso da session
- GetCreationTime() -- Horário de criação.

Application

Para alterar um valor na aplicação é utilizada a instrução abaixo:

getServletContext().setAttribute(“chave", "valor");

Para recuperar os valores no servlet:

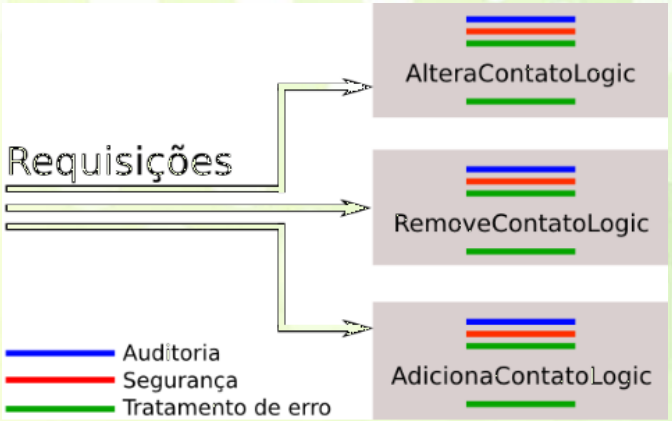
Object valorApplication = getServletContext().getAttribute(“chave“);

- Para recuperar os valores no JSP:
- **Object valorApplication = pageContext.getServletContext()**

*Chave é String e valor é Object

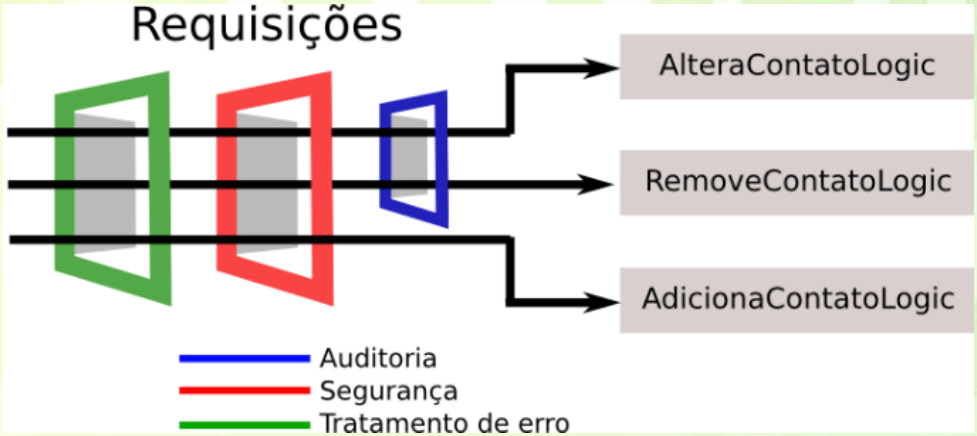
Filtros

- Na implementação de aplicações WEB, é necessária as implementações de Auditorias (Logging), Segurança e Tratamentos de Erros. Isto pode ser feito pontualmente em cada Servlet e JSP.



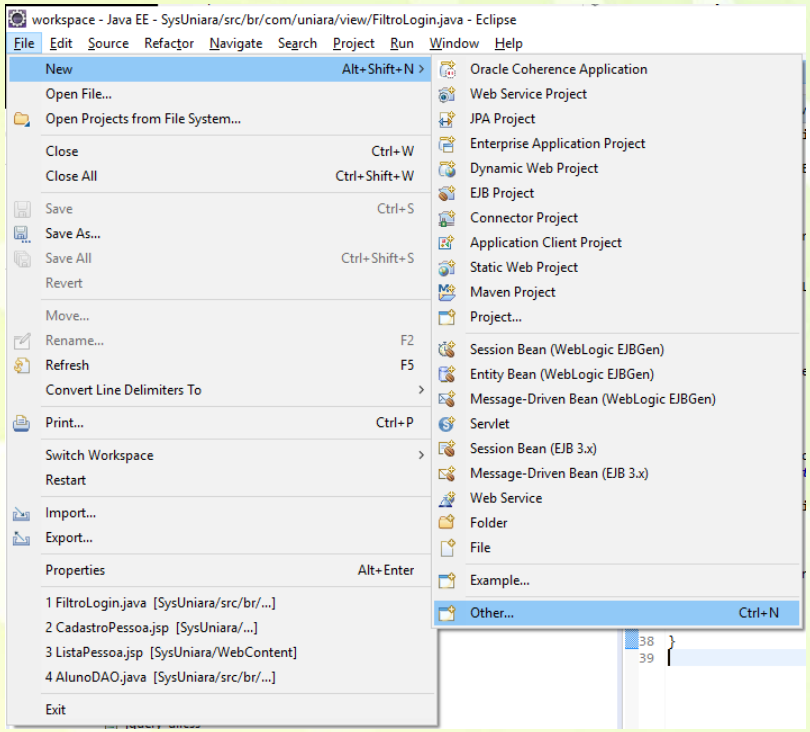
Filtros

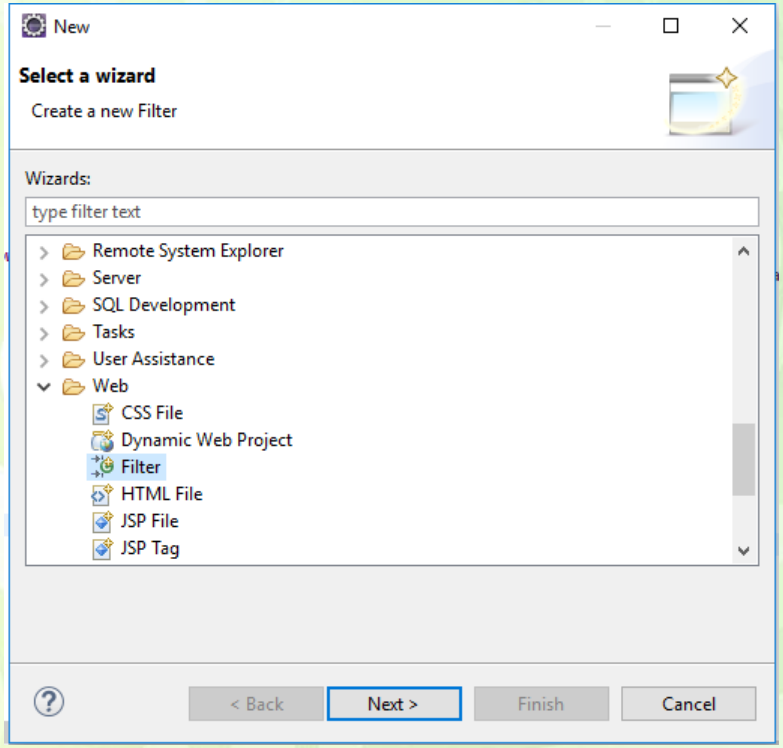
- Ou **filtros** podem ser criados para serem chamados antes das requisições, garantindo (se desejado) uma antecedência de tratativas antes dos Servlets e JSP.



Filtros

- Para criar um filtro:





Filtro

```
public class FiltroLogin implements Filter {


    public FiltroLogin() {

    }


    public void destroy() {

    }


    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletExcep
        System.out.println("Passou pelo Filtro");

        chain.doFilter(request, response);
    }


    public void init(FilterConfig fConfig) throws ServletException {

    }

}
```

Filtro

- Init – Executa algo quando o Container carrega.
- Destroy – Executa algo quando o Container finaliza.
- doInit – Método chamado para a interceptação do Filter.

Filtros

- O Mapeamento de Filtros na API de Servlets do Java EE 5 é feita no web.xml da aplicação:

```
<filter>
  <filter-name>Filtro</filter-name>
  <filter-class>Filtro</filter-class>
</filter>

<filter-mapping>
  <filter-name>Filtro</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Filtro

- Para o Mapeamento de Filtros na API de Servlets do Java EE 6 é utilizada a annotation @WebFilter contida no próprio Filter:

```
@WebFilter("/*")
public class FiltroLogin implements Filter {
```

- O parâmetro recebido informa o que será filtrado. Abaixo algumas possibilidades:
- @WebFilter("/Recurso") - Intercepta um Servlet ou Filter com o nome Recurso.
- @WebFilter("/*.jsp") - Intercepta qualquer JSP.
- @WebFilter("/*") - Intercepta tudo.
- @WebFilter(name = "Filtro", ulrPatterns = {"/oi", "/ola"}) - Nome para um filtro e url´s que interceptará
- @WebFilter(name = "MeuFiltro", servletNames = {"meuServlet", "outroServlet"}) - Nome do filtro e Servlets que irá interceptar.

Referências

- Deitel, Paul J.; Deitel, Harvey M. Java: como programar - 8ª edição. Editora Pearson. ISBN: 9788576055631
- Barnes, David.; Kölling, Michael. Programação Orientada a Objetos com Java: uma introdução prática usando o Blue J - 1ª edição. 2013. Editora Pearson. ISBN: 9788576050124
- Puga, Sandra; Rissetti, Gerson. Lógica de Programação e Estruturas de Dados: com aplicações em Java. 1ª edição. 2013. Editora Pearson. ISBN: 9788587918826
- Lemay, Laura; Colburn, Rafe; Tyler, Denise. Aprenda a Criar Páginas Web com HTML e XHTML em 21 Dias. Editora Pearson. 2013. 1°Edição. ISBN: 9788534614283
- Chak, Andrew. Como Criar Sites Persuasivos. Editora Pearson. 2012. 1°Edição. ISBN: 9788534615112
- Fábio Flatschart; Clécio Bachini; Cesar Cusin. Open Web Platform. Editora Pearson. 2013. 1°Edição. ISBN: 9788574526140