Programação de Aplicações Corporativas

Arquitetura MVC sem a utilização de Frameworks

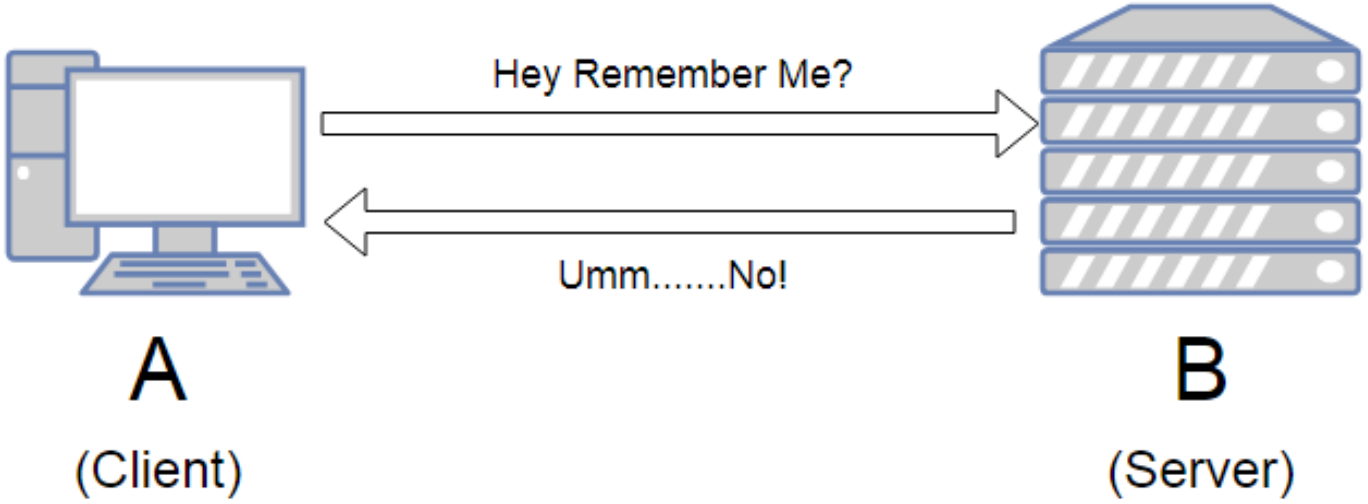
Cookie, Session e Filtro

Hash em Senha e Token ID

HTTP - Comunicação sem estado “**stateless**”

O protocolo HTTP não guarda o estado anterior de comunicação entre o cliente o servidor. Somente a solicitação do momento.

Cada requisição que é feita pelo navegador é independente. Assim que o navegador fecha a conexão *TCP*, toda informação é perdida.



Em sei quem você é. O que você quer. Mas não lembro nada do seu passado.

Como salvar informações sobre login, opções de navegação, preferência do usuário, cliques, visualizações de produtos, etc???

Conceitos

- Um **Cookie** é um pequeno fragmento de dados que um servidor envia para o navegador do usuário.

- Uma variável de **Sessão** é um recurso que permite armazenar dados específicos para um usuário no cache do servidor.

- **Filtro** em Java é uma classe que é executada toda vez que um usuário acessa um URI (URL).

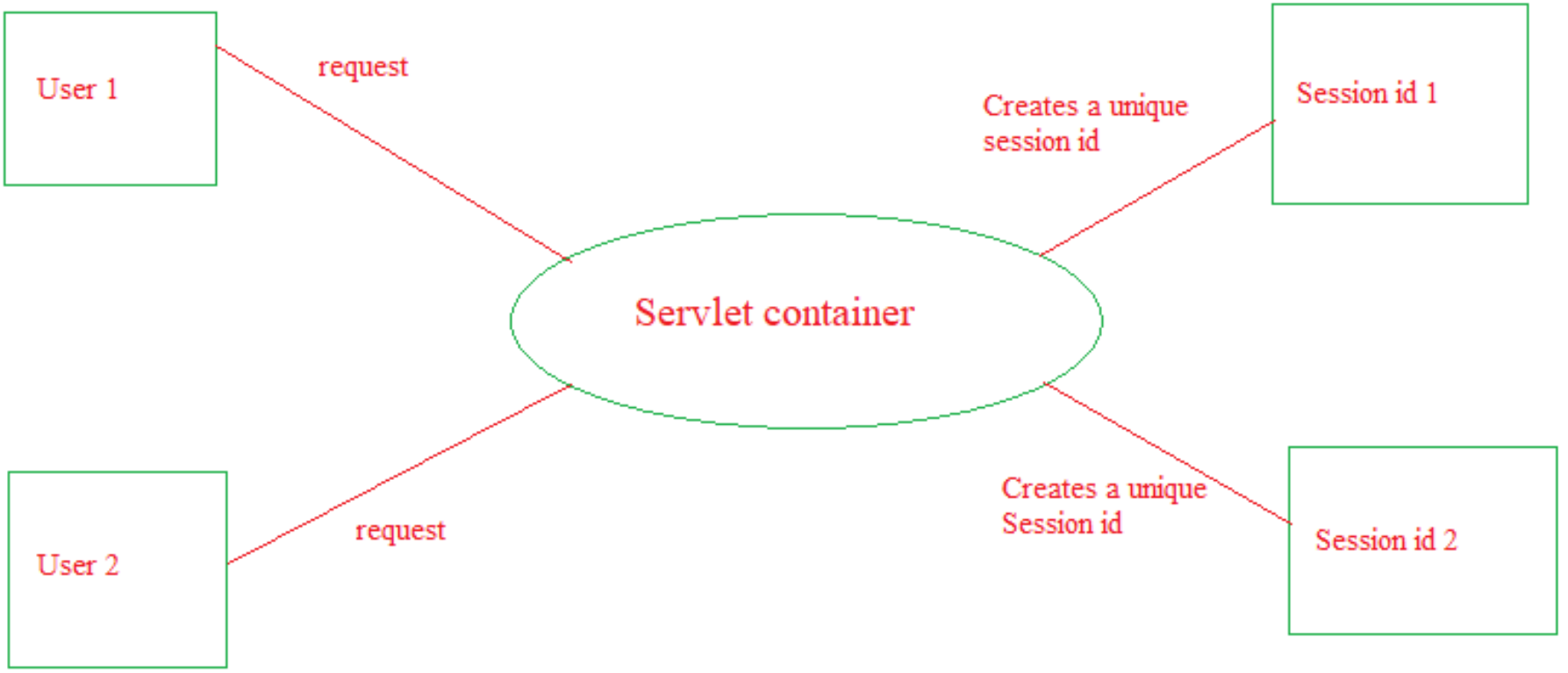
- **Hash:** utilizaremos a classe BCrypt para aplicar um algoritmo de hashing para salvar a senha do usuário.

- **Token** é uma sequência de caracteres única (geralmente aleatória ou criptografada) usada para representar a identidade de um usuário ou sessão.

Sessão

- Interface jakarta.servlet.http.**HttpSession.**
- Um objeto ou uma variável pode ser armazenado temporariamente em uma sessão.
- Quando um computador realiza uma comunicação com o servidor, uma sessão é estabelecida.
- Ela representa a estrutura lógica que contém as regras e os elementos de comunicação entre os equipamentos.
- Durante a importação (deploy) do projeto para o servidor WEB (TomCat) é estabelecido qual o limite máximo da sessão.
- Caso o usuário feche o navegador, as informações continuaram na session, até o final do “time out”.

Sessão


User 1
request
Creates a unique
session id
Session id 1
Servlet container
User 2
request
Creates a unique
Session id
Session id 2

Sessão - Exemplo

```
// Obter ou criar a sessão
HttpSession session = request.getSession();

// “Pendurar” um objeto na sessão
session.setAttribute("cliente", cliente);

// Pegar um objeto da Session
Cliente cliente = (Cliente) session.getAttribute("cliente");
```

Sessão - Exemplo

```
// Penduar um nome em uma session
String nome = “Maria”’
session.setAttribute(”nome", nome);

// Pegar o id de uma session
    session.getId()

// Encerrar uma session
    session.invalidate();
```

Sessão - Exemplo

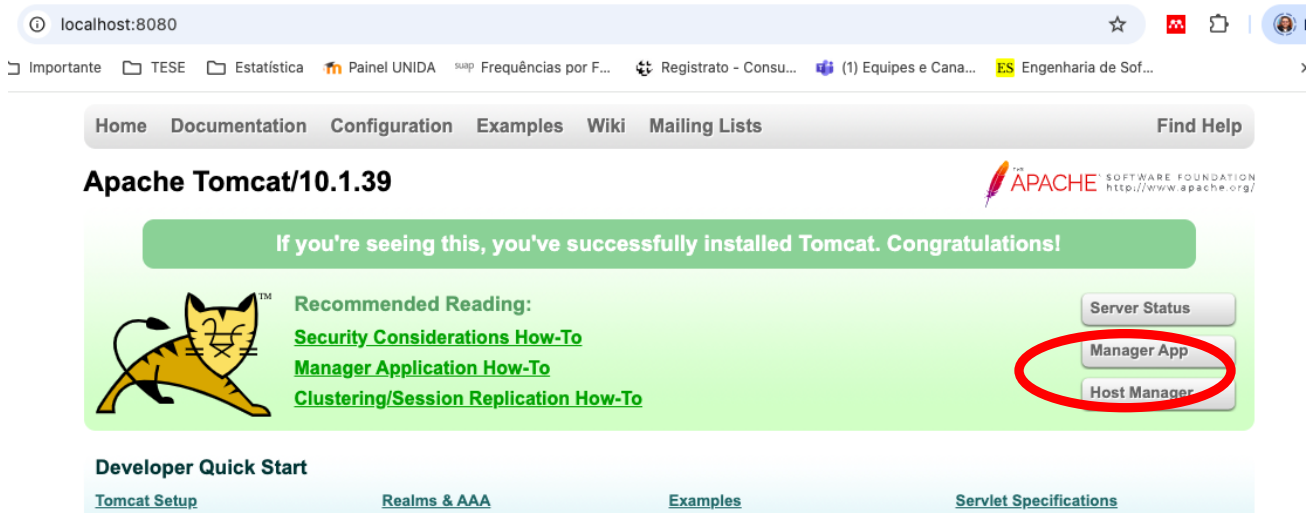
- O exemplo representa a instanciação do objeto cliente, a definição e a atribuição de um elemento a sessão.

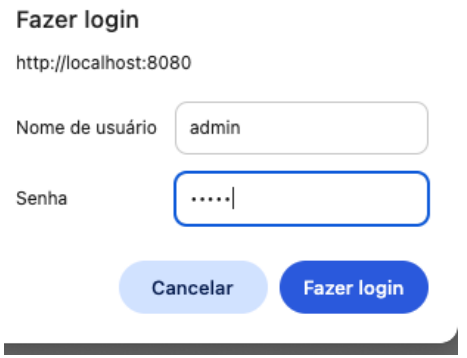
```
// Buscar o cliente, por meio do login e senha
    ClienteDao cDao = new ClienteDao();
    Cliente cliente = null;
    cliente = cDao.buscarUm(senha, login);

// Definir e pegar a session
    HttpSession session;
    session = request.getSession();

// Atribuir "pendurar" o objeto cliente na session
    session.setAttribute("cliente", cliente);
```

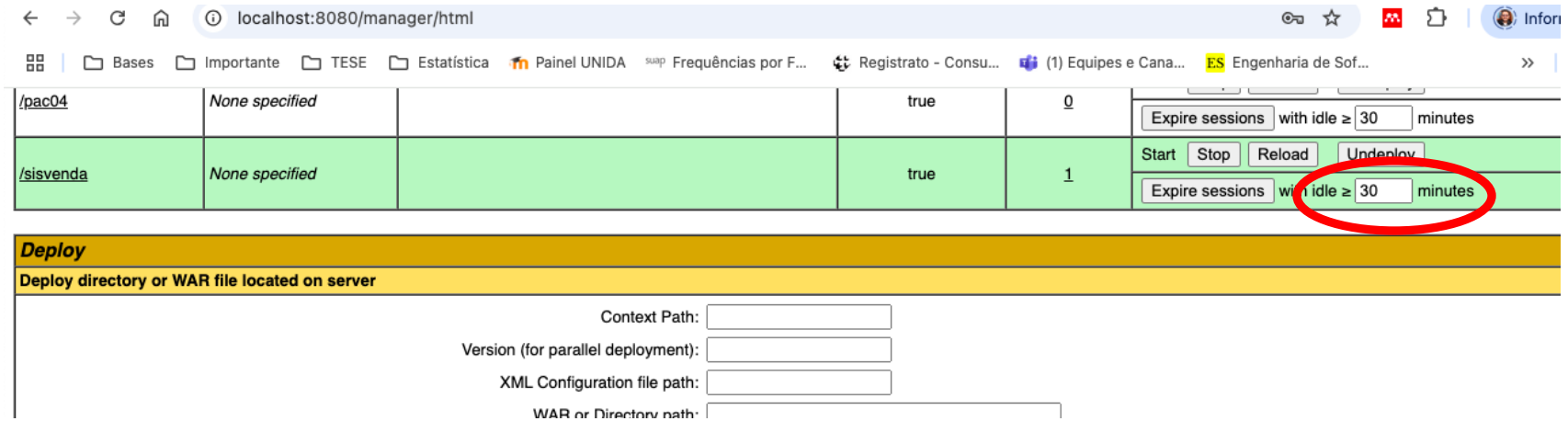
Tempo – Sessão - TomCat

Configurando acesso tomcat\conf\tomcat-users.xml

```
<role rolename="manager-gui"/>
<role rolename="admin"/>
<user username="admin" password="admin" roles="admin,manager-gui"/>
```

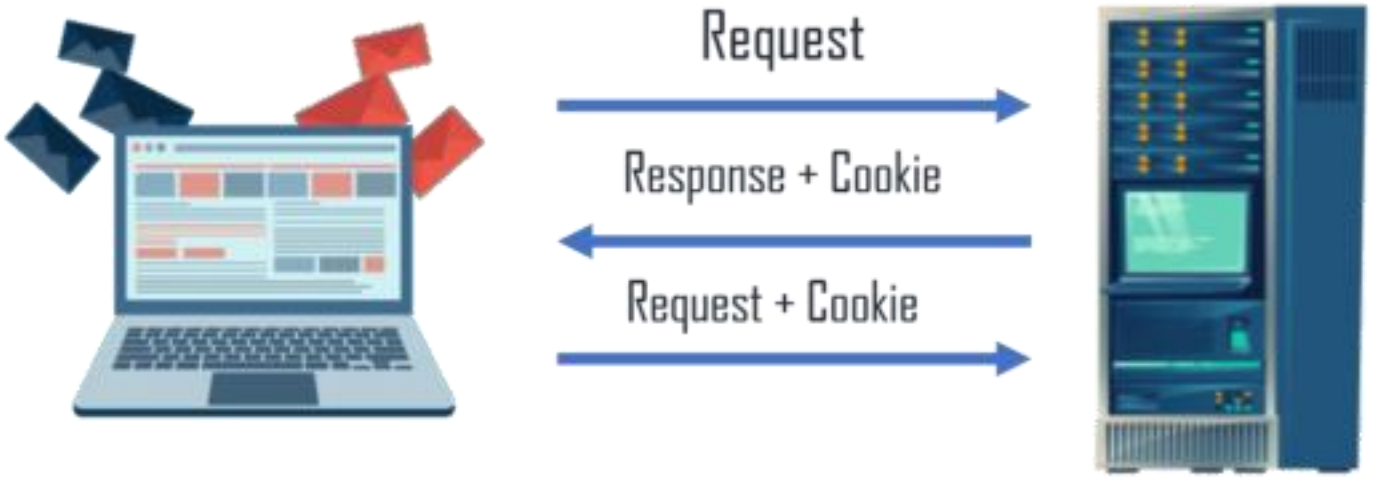






Cookies

- Classe: jakarta.servlet.http.Cookie;
- Um cookie HTTP (um cookie web ou cookie de navegador) é um pequeno fragmento de dados que um servidor envia para o navegador do usuário.
- Um cookie normalmente é um par de strings guardado no cliente.



Cookies (JSP/Servlet)

```
//Criar um cookie (Nome único, valor)
 Cookie cookie1 = new Cookie("nome",cliente.getNome());
 Cookie cookie2 = new Cookie("cor", "verde");

// Duração do cookie
 // Segundos, minutos, horas, dias, meses
 cookie1.setMaxAge(60*60*24*31*12);
 cookie2.setMaxAge(60*1);

 // Adicionar na máquina local
 response.addCookie(cookie1);
 response.addCookie(cookie2);
```

Cookies (JSP/Servlet)

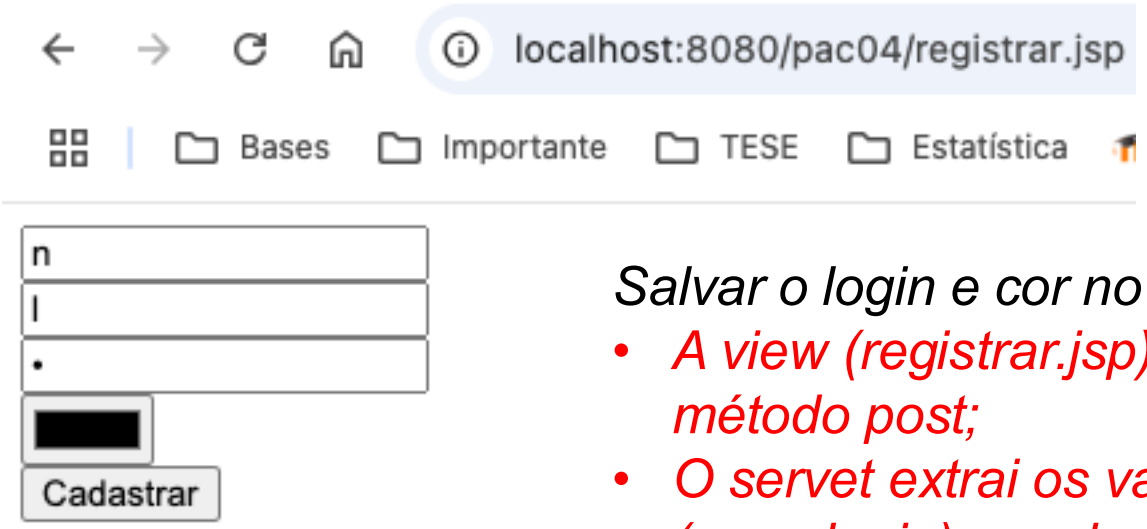
- //Listar o nome e o valor

```
response.setContentType("text html; charset =UTF 8");
Cookie cookies[] = request.getCookies();
if ( cookies != null )
        for ( Cookie c : cookies ) {
            response.getWriter().println(c.getName() + " - "
               +  c.getValue());
}
```

Cookies – Exemplo - Listar - JSP

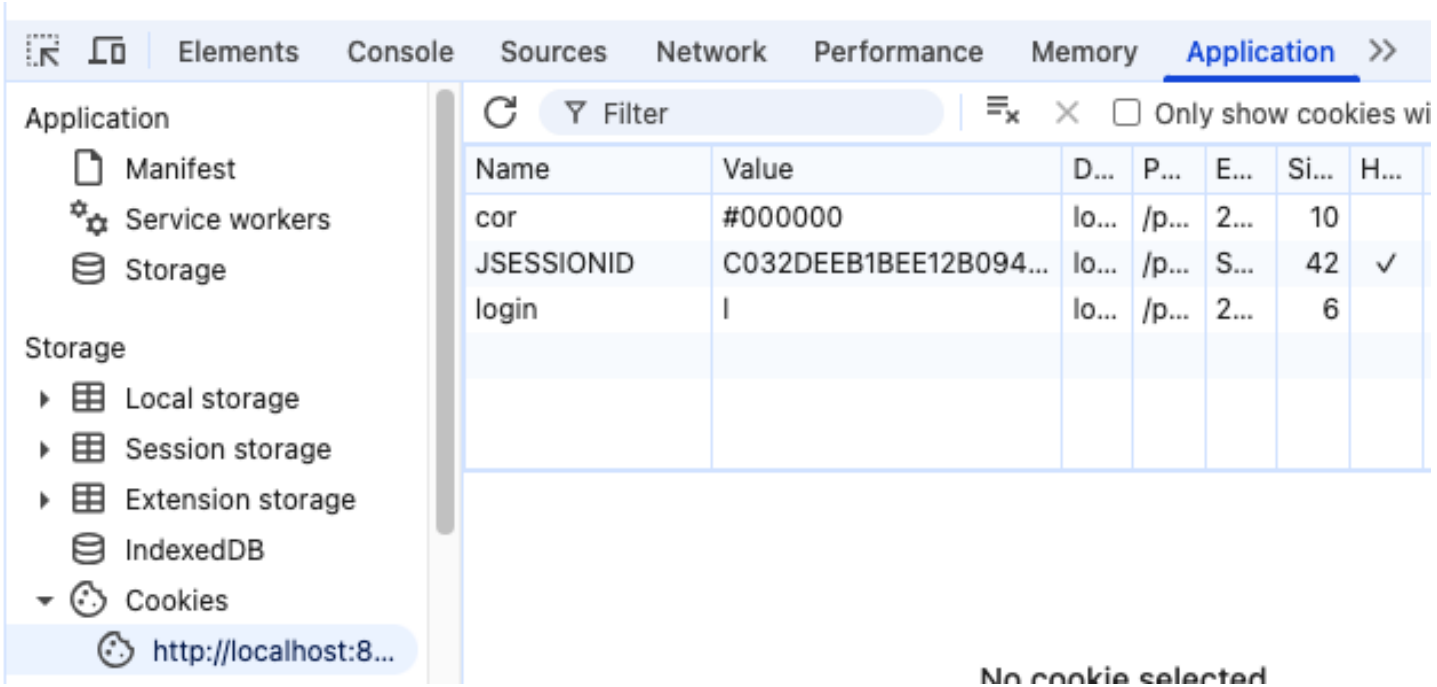
```
<body>
<% Cookie cookies[] = request.getCookies();
String nome = null;
String cor = null;
if (cookies != null){
        for (Cookie c : cookies) {
                if (c.getName().equals("nome"))
                        nome = c.getValue();
                if (c.getName().equals("cor"))
                        cor = c.getValue();
        }
}%>
<% out.print("<p> <font color= '" + cor + "'> "
        + nome + "</font>");
%>
```

Exemplo de utilização de cookie



Salvar o login e cor no cookie:

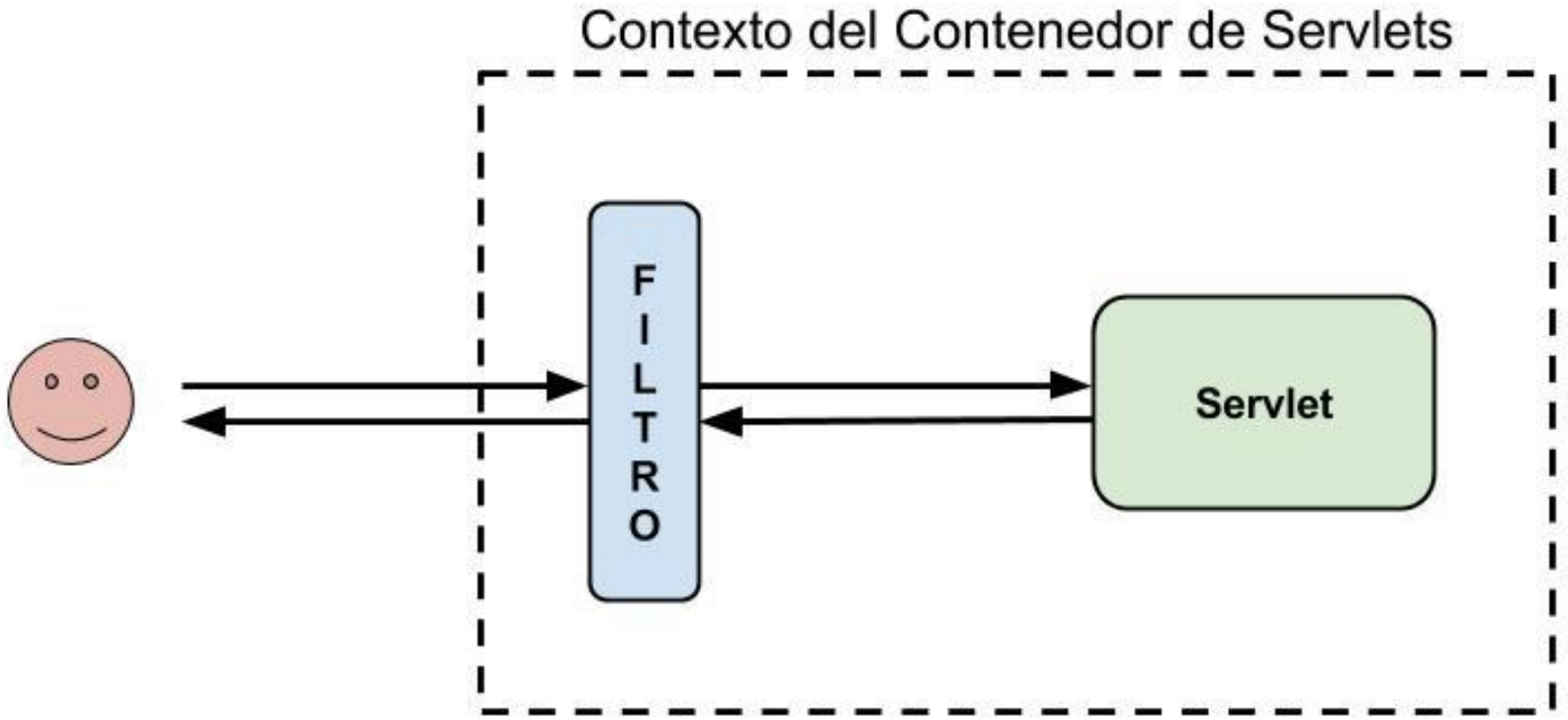
- *A view (registrar.jsp) chama /login, no método post;*
- *O servet extrai os valores do request (cor e login) e salva no cookie.*



Filtros

- Filtro é uma classe Java que permite a implementação de requisitos não funcionais, tais como: Log ou Permissão de Acesso.
- Pode-se definir os arquivos ou extensões para que quando forem executados ou visualizados, primeiramente a requisição passe por um filtro, para que determinada condição seja realizada.
- O método principal do Filtro é o doFilter, antes de qualquer de acesso a qualquer página, a classe filter, por meio do método doFilter é executado.
- Além de criar e programar o filtro, é necessário acessar o indicar os arquivos ou extensões que serão filtrados. Por exemplo, `@WebFilter("/*")`.

Filtros



Filtros / Exemplo - Login

```
public void doFilter(ServletRequest request,
ServletResponse response, FilterChain chain) throws
IOException, ServletException {

    HttpServletRequest  req = (HttpServletRequest) request;
    HttpServletResponse resp = (HttpServletResponse)
    response;
    // Url que o usuário esta tentando acessar
    String url = req.getRequestURI();
    // Pegar a session
    HttpSession session = req.getSession();
    // Pegar o Cliente
    Cliente cliente = (Cliente)
    session.getAttribute("cliente");
```

Filtros / Exemplo - Login

```
// O método chain.doFilter é único meio de sair do filtro
// O exemplo não permite a visualização do diretório adm...
if (cliente==null){
    if (url.startsWith("/sisvenda/paginas/adm/"))
        resp.sendRedirect("/sisvenda/index.jsp");
    else
        chain.doFilter(request, response);

}else{
    chain.doFilter(request, response);
}
…..
```

Token

- Um token é uma sequência de caracteres única (geralmente aleatória ou criptografada) usada para representar a identidade de um usuário ou sessão.

Exemplo de uso:

- Gerar um **token UUID** ao logar.
- Salvar esse token em **cookie ou session**.
- No filtro, liberar o acesso por meio do Token;

Exemplo:

tokenId = java.util.UUID.randomUUID().toString();

Token

ex.: Salvando um Token na session e no cookie (fins didáticos)

```
// Gerando um Token
tokenId = java.util.UUID.randomUUID().toString();

// Capturando a session e salvando o token
HttpSession session = request.getSession();
session.setAttribute("tokenId", tokenId);

// Salvando o token em um cookie
Cookie cookie1 = new Cookie("tokenId",tokenId);
cookie1.setMaxAge(60*60*24*31*12);
response.addCookie(cookie1);
```

Hashing (BCrypt)

- Uma senha, por exemplo, jamais deve ser armazenada em texto puro.
- Utilize uma função de hash criptográfico segura para criptografar os dados.

Exemplo de Uso:

- Pegar a senha em texto puro;
- Junta com o salt gerado (um **"salt"** é um valor aleatório único adicionado à senha antes de aplicar o hash)
- Aplica o algoritmo de hashing **BCrypt**
- Retorna uma **string longa que representa o hash da senha com o salt embutido**

Pom.xml

```
<dependency>
<groupId>org.mindrot</groupId>
<artifactId>jbcrypt</artifactId>
<version>0.4</version>
</dependency>
```

Hashing (BCrypt)

```
//Senha String puro
String senhaTxt = request.getParameter("senha");

// Criptografar a Senha
String senhaHash = BCrypt.hashpw(senhaTxt,
BCrypt.gensalt());

//Verificar senhas e gerar um token
 if (senhaTxt == BCrypt.checkpw(senhaTxt,
senhaHasch){
     tokenId = java.util.UUID.randomUUID().toString();
 }
 else{
 ... Senha inválida
}
```