

Programação de Aplicações Corporativas

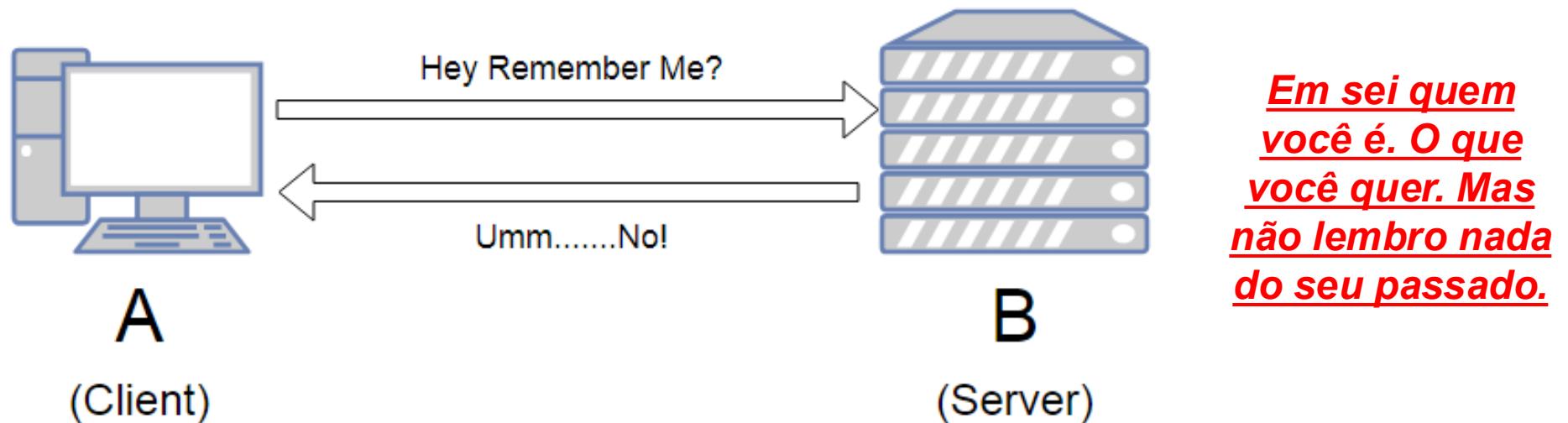
Arquitetura MVC sem a utilização
de Frameworks

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 e 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.



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

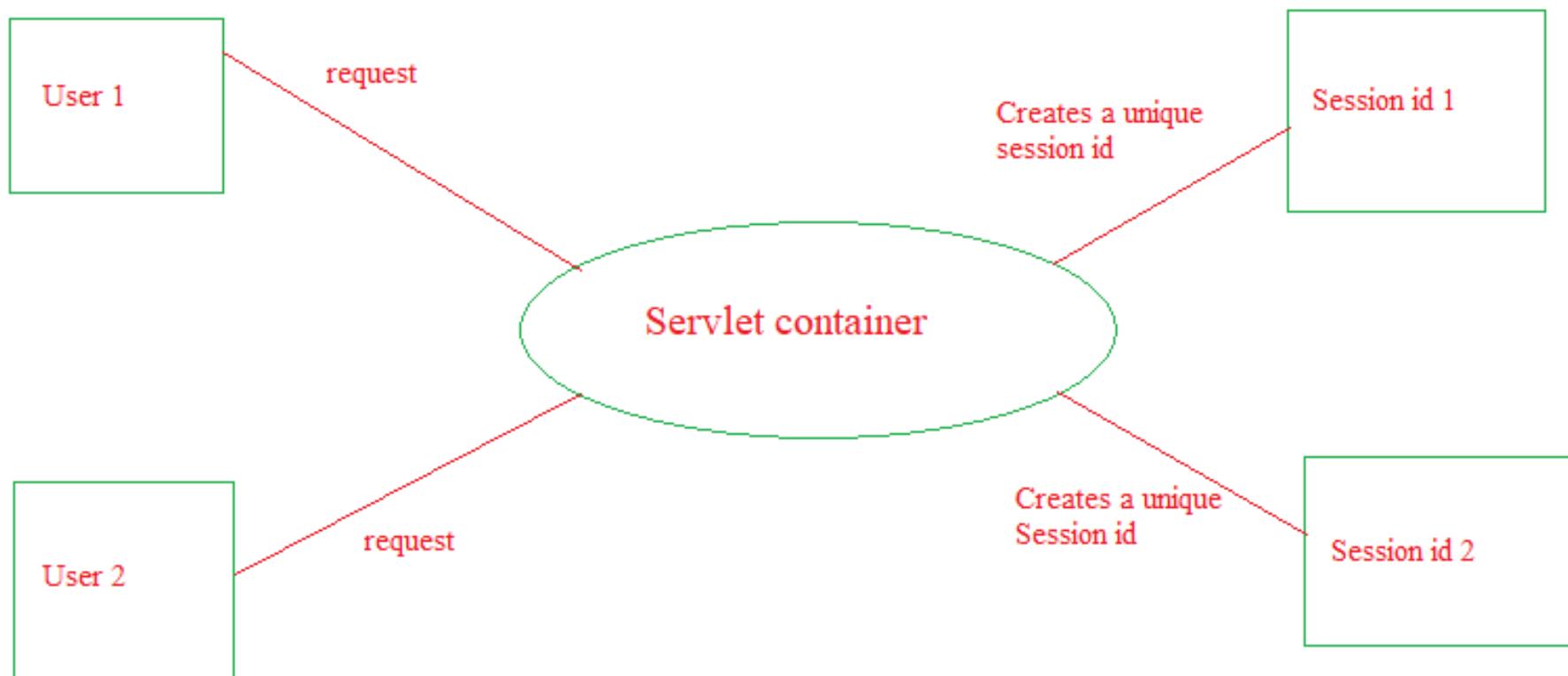
Conceitos

- 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



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

// Pendurar 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();
```

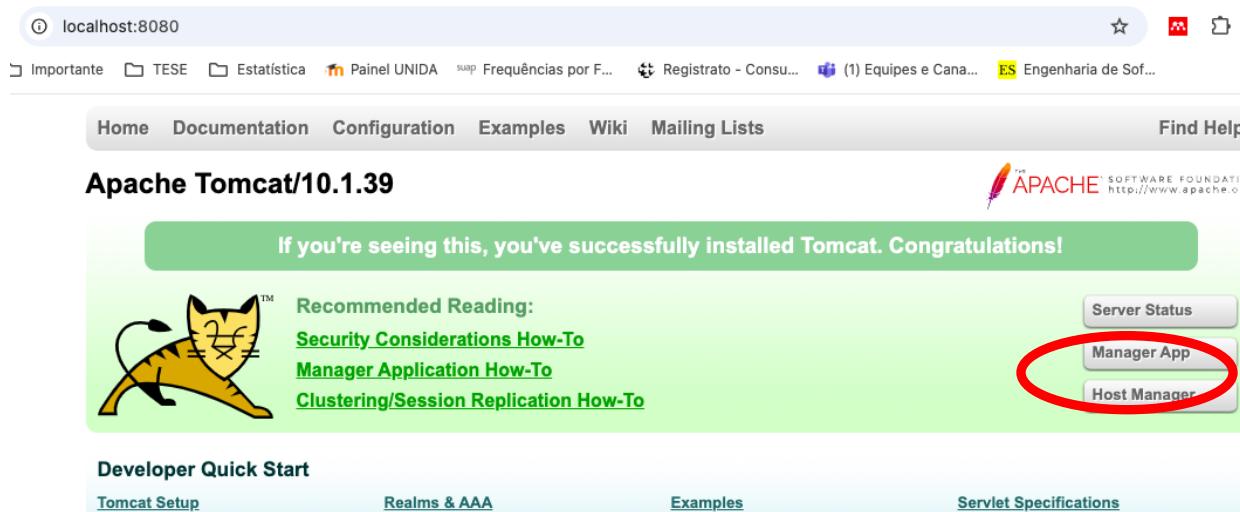
```
// Atribuir “pendurar” o objeto cliente na session
```

```
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"/>
```

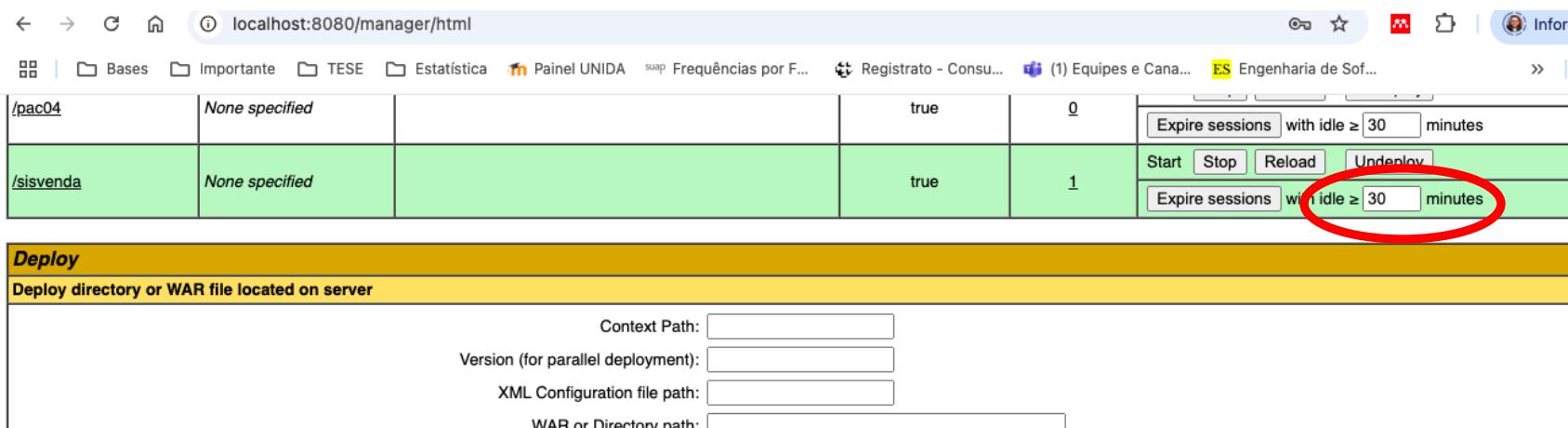


The screenshot shows the Apache Tomcat 10.1.39 homepage at localhost:8080. A green banner at the top says "If you're seeing this, you've successfully installed Tomcat. Congratulations!". Below it, there's a cartoon cat icon and a "Recommended Reading" section with links to Security Considerations How-To, Manager Application How-To, and Clustering/Session Replication How-To. On the right side, there are three buttons: "Server Status", "Manager App" (which is circled in red), and "Host Manager". At the bottom, there's a "Developer Quick Start" menu with links to Tomcat Setup, Realms & AAA, Examples, and Servlet Specifications.

Fazer login
<http://localhost:8080>

Nome de usuário

Senha



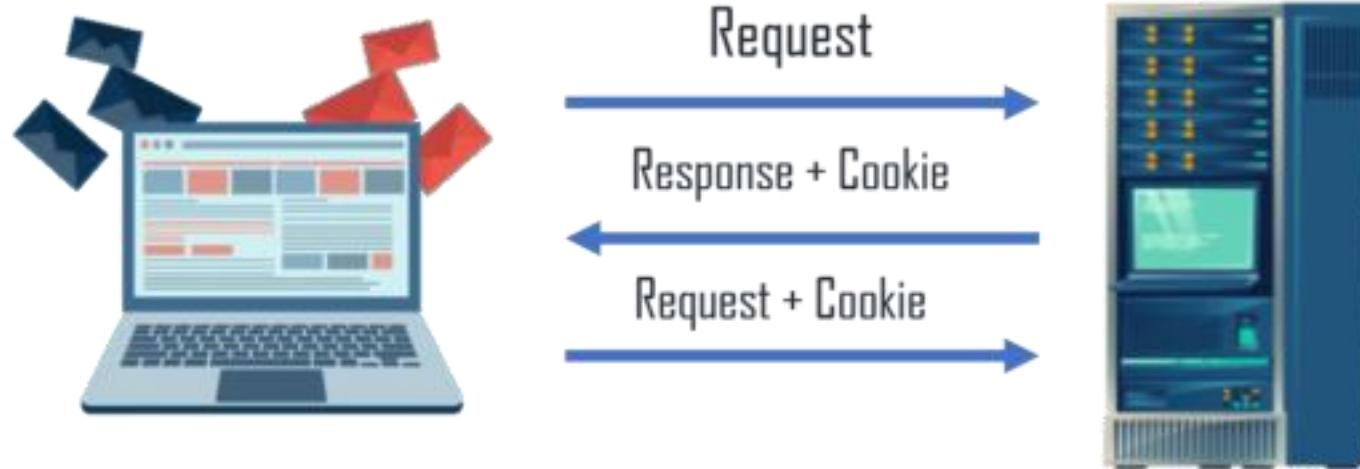
The screenshot shows the Tomcat Manager App interface at localhost:8080/manager/html. It displays a table of deployed applications:

| Context Path | Document Root | Deployed On | Status | Restart Count | Action Buttons |
|--------------|----------------|-------------|--------|---------------|--|
| /pac04 | None specified | | true | 0 | <input type="button" value="Expire sessions with idle ≥ 30 minutes"/> |
| /sisvenda | None specified | | true | 1 | <input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions with idle ≥ 30 minutes"/> |

Below the table, there's a "Deploy" section with fields for Context Path, Version (for parallel deployment), XML Configuration file path, and WAR or Directory path. The "Expire sessions with idle ≥ 30 minutes" button for the /sisvenda application is circled in red.

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)

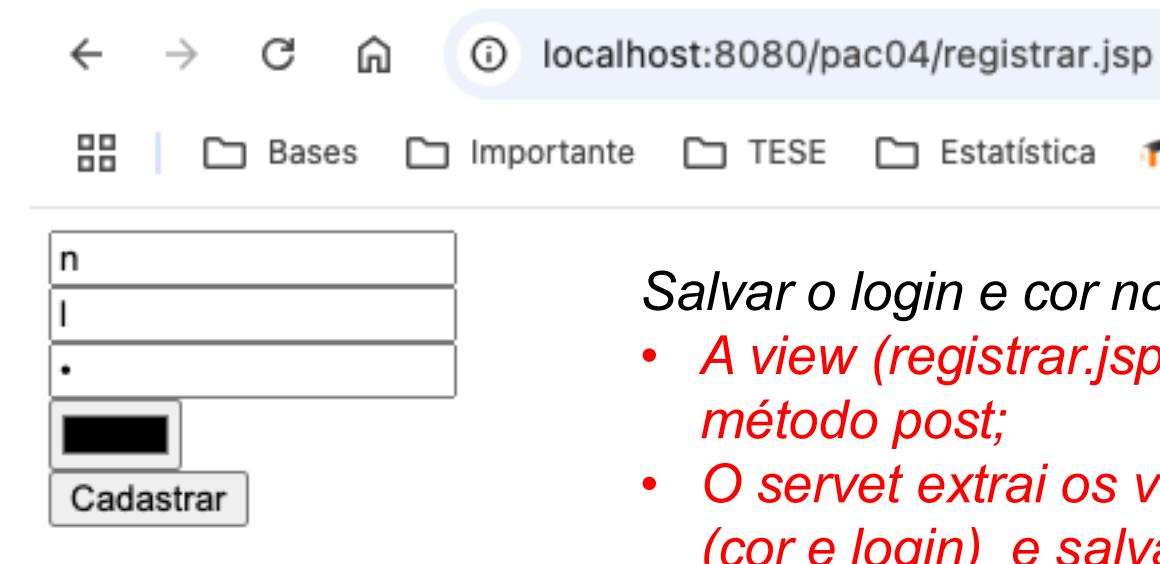
- //Listar o nome e o valor

```
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>");>%>
```

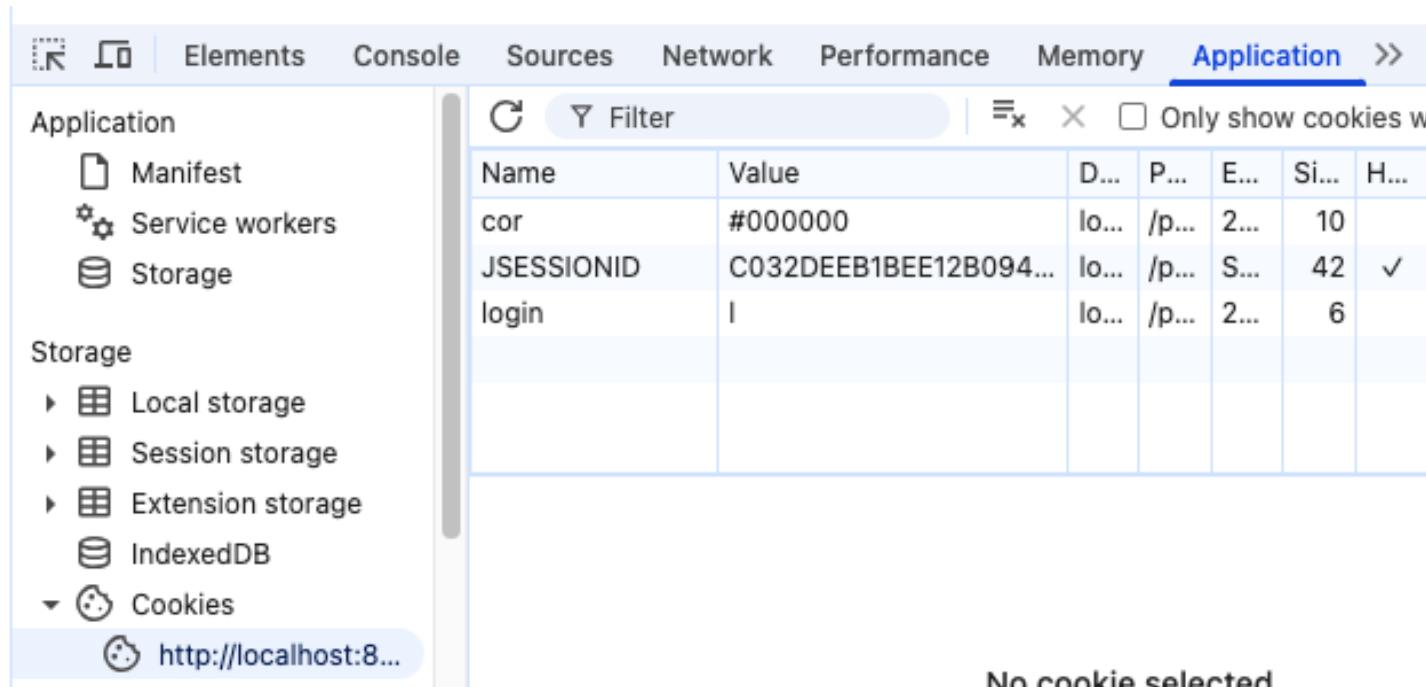
Exemplo de utilização de cookie



A screenshot of a web browser window. The address bar shows the URL `localhost:8080/pac04/registrar.jsp`. Below the address bar is a navigation bar with icons for back, forward, search, and home. To the right of the address bar is a folder icon with the number '1'. The main content area displays a registration form with three input fields for 'n', 'i', and '•', followed by a blacked-out image placeholder, and a 'Cadastrar' button.

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.*



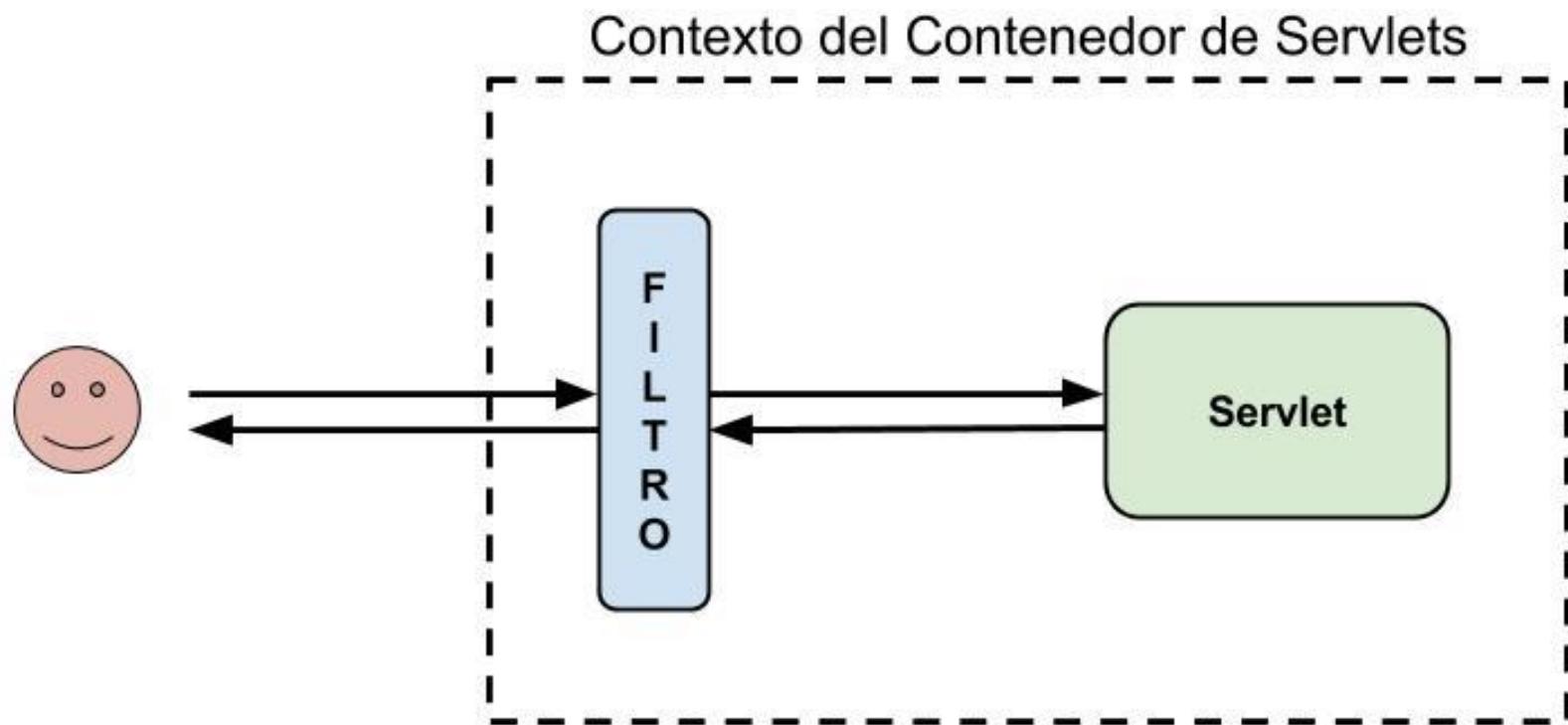
A screenshot of the Chrome DevTools Application tab. The left sidebar shows sections for Application (Manifest, Service workers, Storage), Storage (Local storage, Session storage, Extension storage, IndexedDB, Cookies), and a network entry for `http://localhost:8080/pac04/registrar.jsp`. The main area is titled 'Application' and contains a table of stored cookies. The table has columns for Name, Value, and various metadata like Last access, Path, and Expiry. Three cookies are listed: 'cor' with value '#000000', 'JSESSIONID' with value 'C032DEEB1BEE12B094...', and 'login' with value 'I'. A message at the bottom right says 'No cookie selected'.

| Name | Value | D... | P... | E... | Si... | H... |
|------------|-----------------------|-------|-------|------|-------|------|
| cor | #000000 | lo... | /p... | 2... | 10 | |
| JSESSIONID | C032DEEB1BEE12B094... | lo... | /p... | S... | 42 | ✓ |
| login | I | lo... | /p... | 2... | 6 | |

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 está 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
```

```
}
```