Servlets: Redirecionamento e Sessões

Laboratório de Programação (5COP011)

Prof. Bruno Bogaz Zarpelão

Departamento de Computação - 2016

Objetivos

- Conceitos adicionais de servlets:
  - Redirecionamento.
  - Sessões.

Redirecionamento

- Quando trabalhamos com servlets, podemos redirecionar o request recebido por um servlet para outros recursos do servidor Web.

Redirecionamento

- Por exemplo:
  - quando programamos um servlet, pode ser necessário montar toda a resposta HTML no próprio servlet, o que é trabalhoso e sujeito a erros.
  - usando redirecionamento, podemos usar o servlet só para realizar tarefas de lógica de negócios, e depois despachamos o request para uma página JSP, por exemplo, que será mais adequada para construir uma resposta.

Redirecionamento

- Para realizar o redirecionamento, os seguintes comandos são mais comuns:

```
RequestDispatcher view =
        request.getRequestDispatcher("adicionar-resposta.jsp");

view.forward(request, response);
```

OU

```
response.sendRedirect("http://www.uol.com.br");
```

Redirecionamento

- Quando usamos o método *forward()*, podemos precisar passar, juntamente com o encaminhamento do request, o valor de algum atributo que será utilizado na construção da resposta. Nesse caso, utilizamos o método:

```
request.setAttribute(“nomeDoAtributo", valorDoAtributo);
```

Redirecionamento

- Para recuperar o atributo, devemos utilizar o seguinte método:

```
ClasseDoAtributo umNome =
        request.getAttribute(“nomeDoAtributo");
```

Sessões

- O protocolo HTTP não prevê a persistência de estado entre diferentes transações.
- Cada transação no HTTP é executada de maneira totalmente independente, mesmo que envolva os mesmos pares (cliente-servidor).
- Dessa forma, assim que o servidor Web responde uma requisição, ele “esquece” todos os dados referentes a aquela requisição.

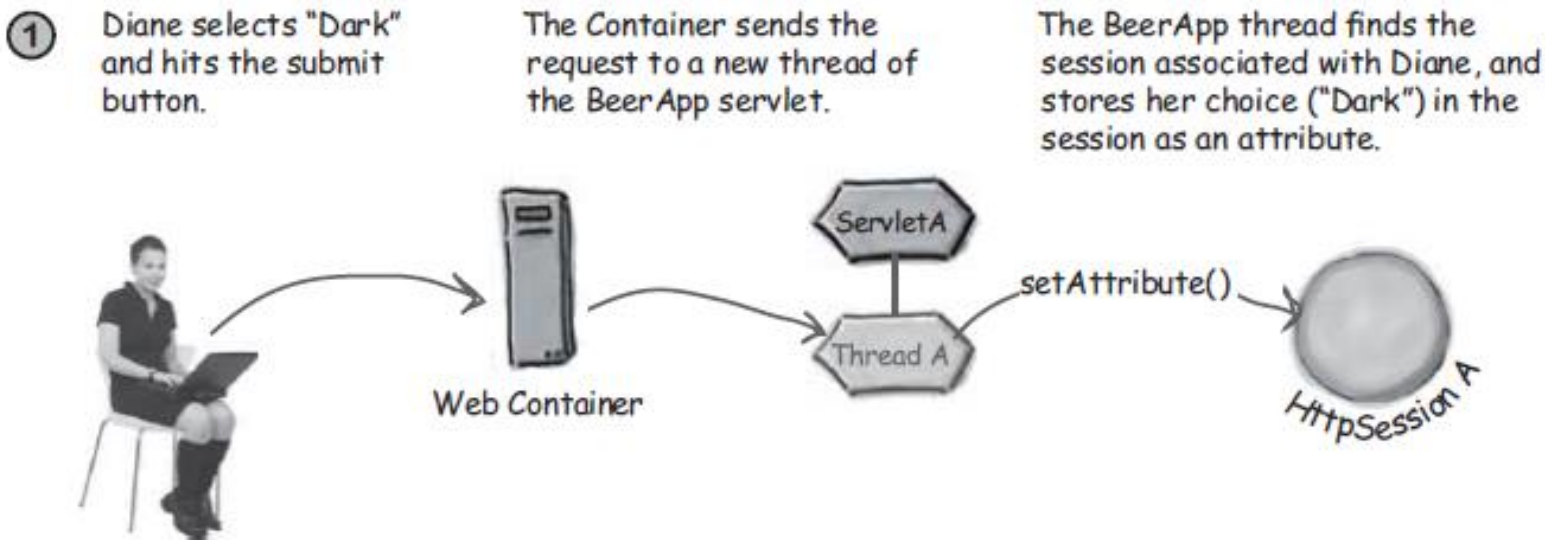
Sessões

- Como podemos fazer para guardar informações de um cliente entre diferentes transações?
- Exemplos clássicos:
  - carrinho de compras em sites de *e-commerce*.
  - produtos mais buscados por um determinado usuário.
  - consultas recentes realizadas.
  - o que mais?

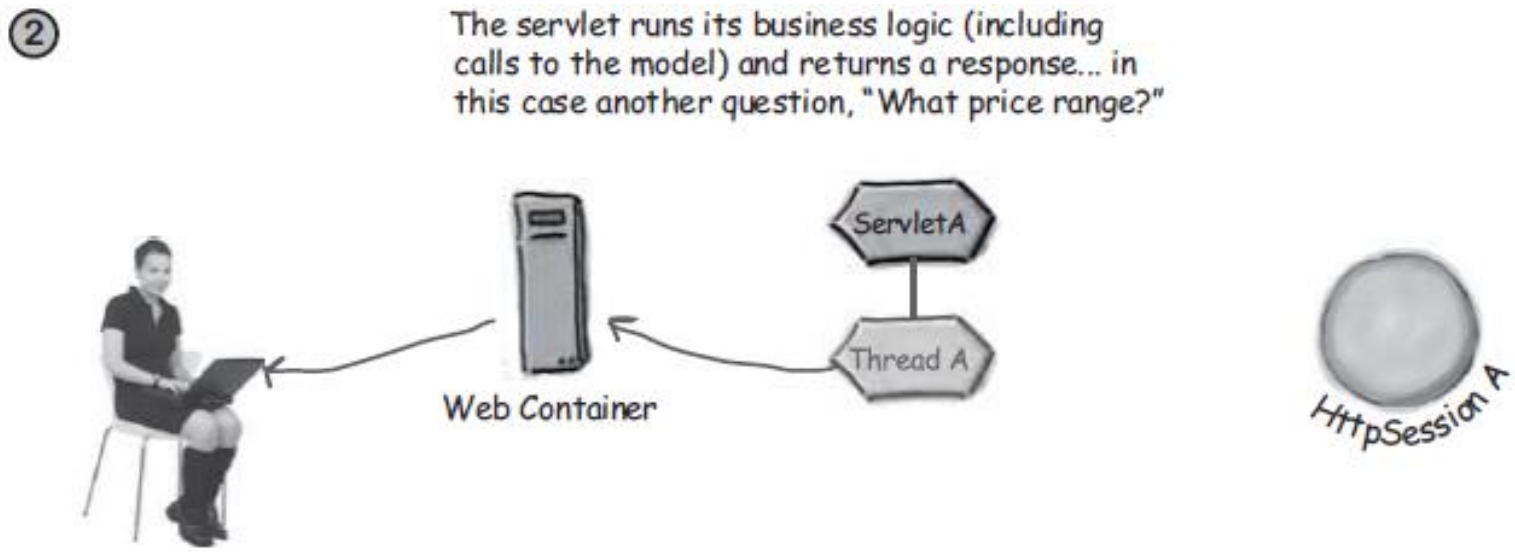
Sessões

- Enfim, como podemos armazenar dados relacionados a operações do cliente em diferentes transações?
  - Uma resposta é banco de dados.
  - A outra é HttpSession.

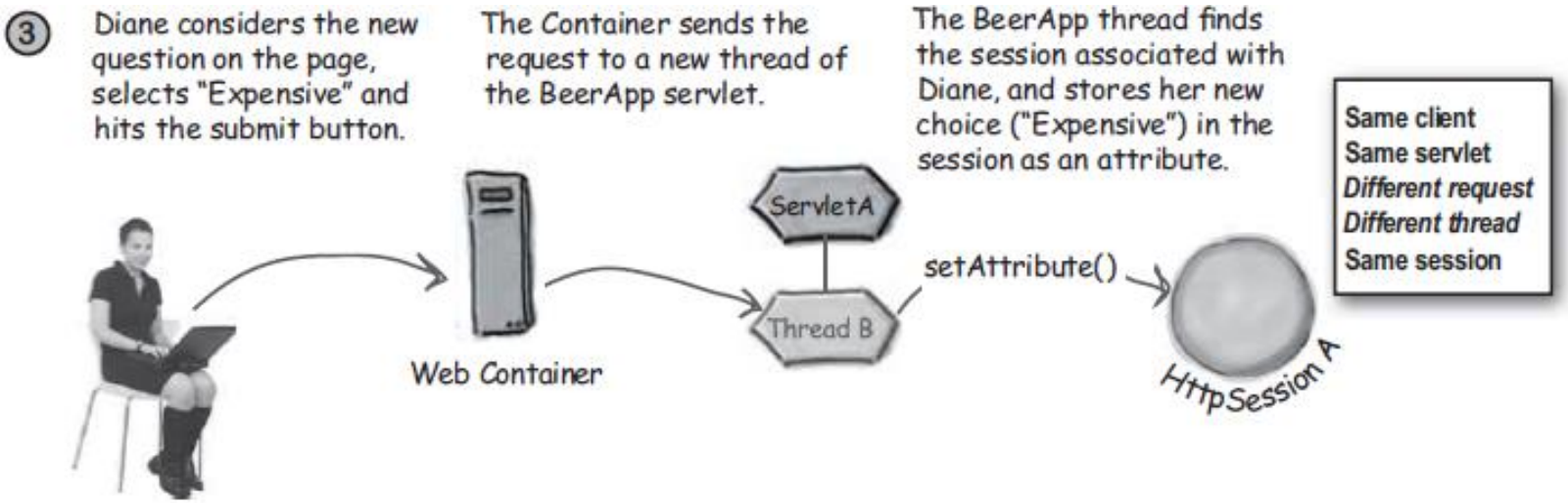
Sessões


1
Diane selects "Dark" and hits the submit button.
The Container sends the request to a new thread of the BeerApp servlet.
The BeerApp thread finds the session associated with Diane, and stores her choice ("Dark") in the session as an attribute.
ServletA
Thread A
setAttribute()
Web Container
HttpSession A

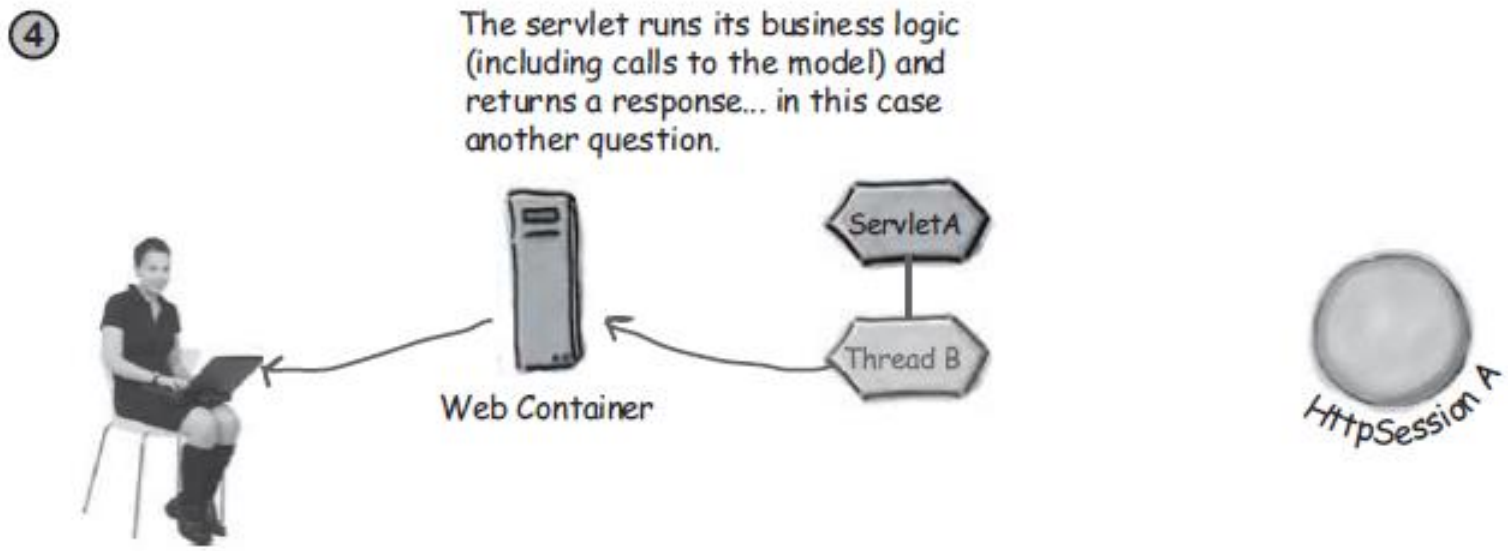
Sessões


②
The servlet runs its business logic (including calls to the model) and returns a response... in this case another question, "What price range?"
ServletA
Thread A
Web Container
HttpSession A

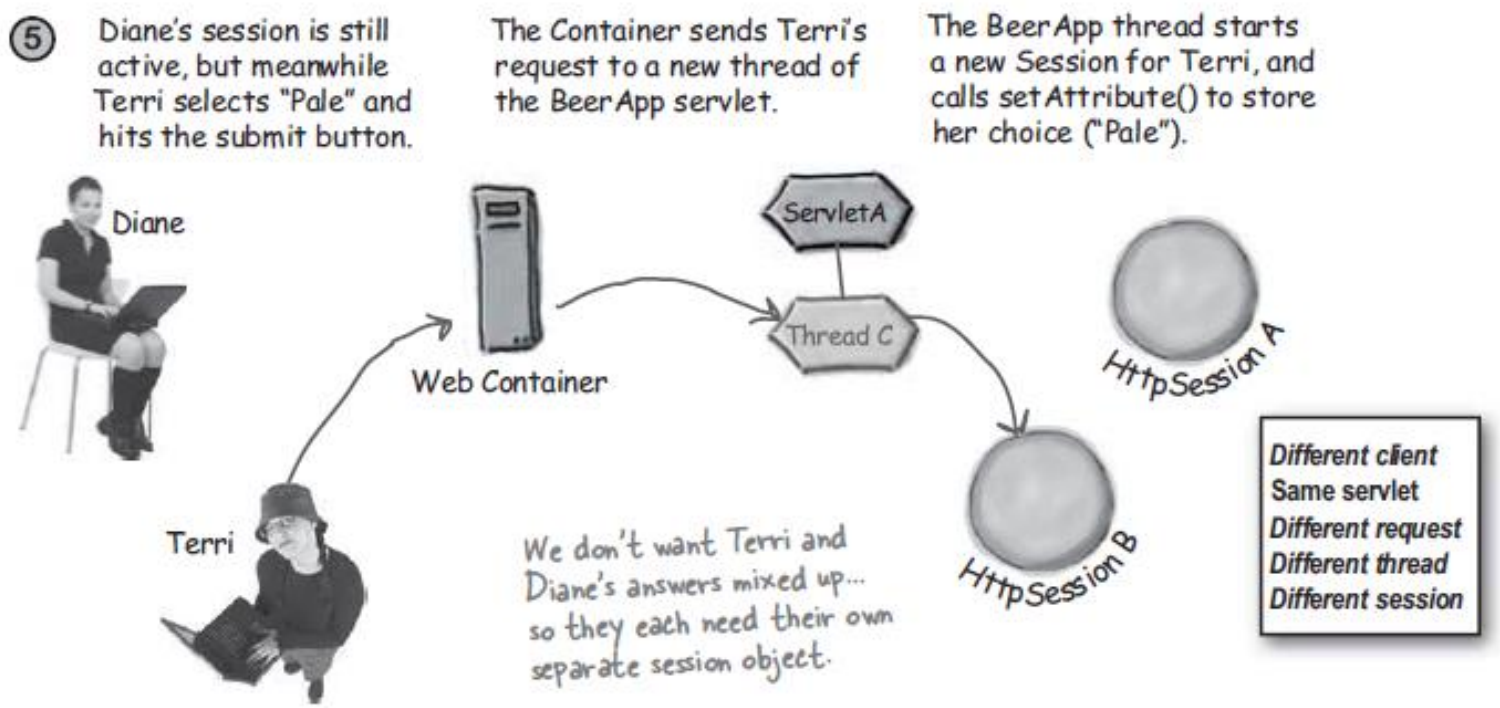
Sessões


3
Diane considers the new question on the page, selects "Expensive" and hits the submit button.
The Container sends the request to a new thread of the BeerApp servlet.
The BeerApp thread finds the session associated with Diane, and stores her new choice ("Expensive") in the session as an attribute.
ServletA
Thread B
Web Container
setAttribute()
HttpSession A
Same client
Same servlet
Different request
Different thread
Same session

Sessões


4
The servlet runs its business logic (including calls to the model) and returns a response... in this case another question.
ServletA
Thread B
Web Container
HttpSession A

Sessões

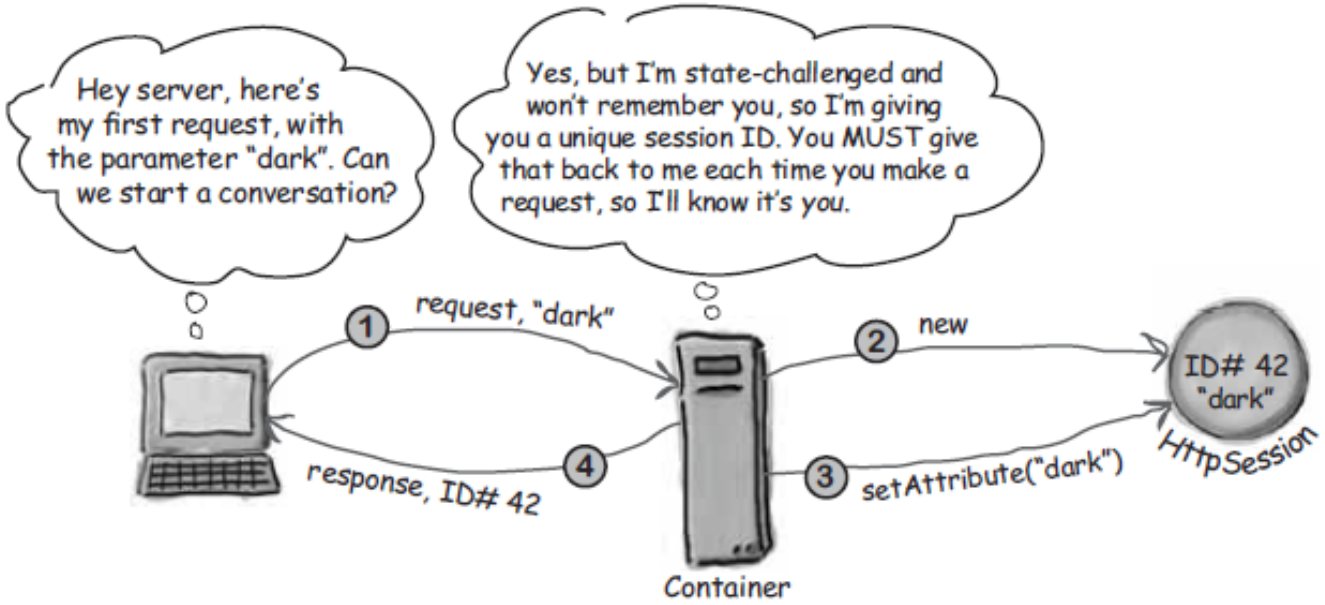

5
Diane's session is still active, but meanwhile Terri selects "Pale" and hits the submit button.
The Container sends Terri's request to a new thread of the BeerApp servlet.
The BeerApp thread starts a new Session for Terri, and calls setAttribute() to store her choice ("Pale").
Diane
Terri
Web Container
ServletA
Thread C
HttpSession A
HttpSession B
We don't want Terri and Diane's answers mixed up... so they each need their own separate session object.
Different client
Same servlet
Different request
Different thread
Different session

Sessões

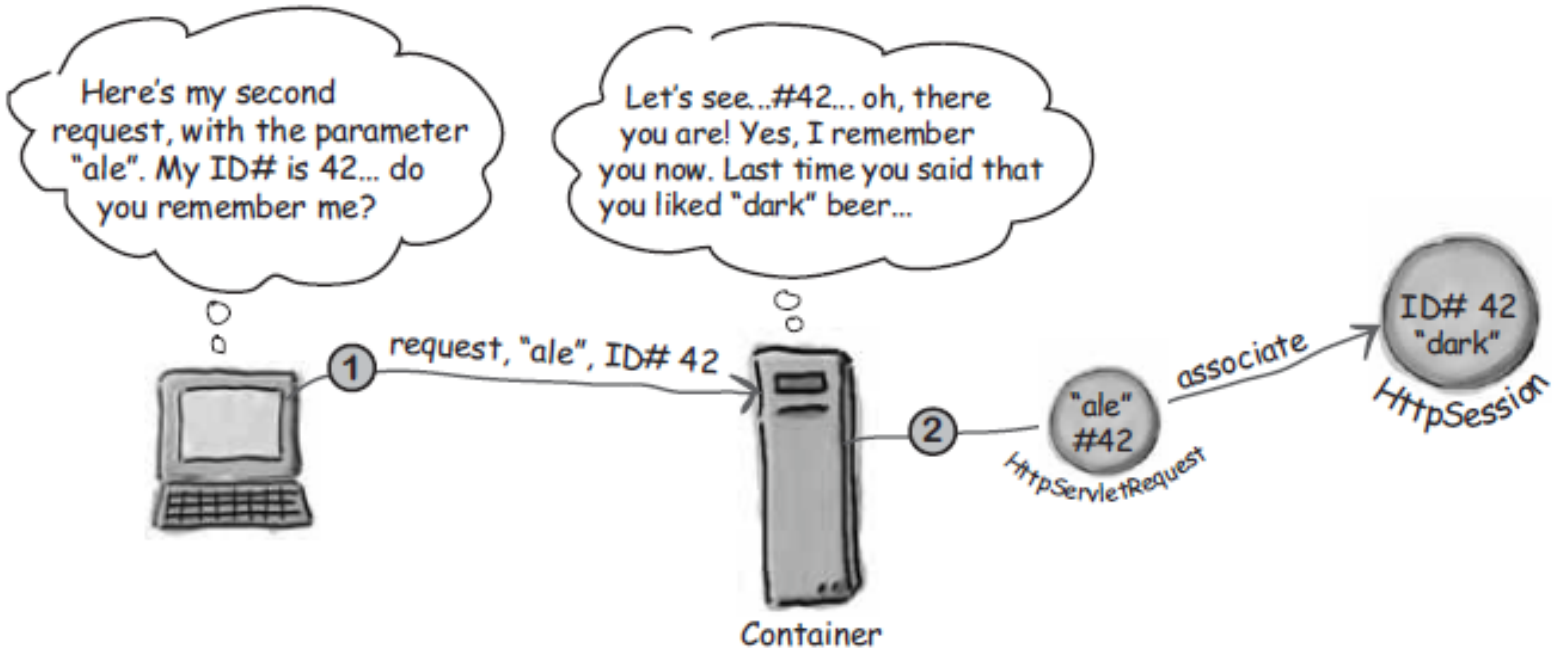
- Dúvida: como o servidor sabe qual é o HttpSession referente a cada cliente?

Sessões

- Para cada sessão, é definido um *session id.*



Sessões


Here's my second request, with the parameter "ale". My ID# is 42... do you remember me?
Let's see...#42... oh, there you are! Yes, I remember you now. Last time you said that you liked "dark" beer...
1
request, "ale", ID# 42
2
"ale"
#42
HttpServletRequest
associate
ID# 42
"dark"
HttpSession
Container

Sessões

- O cliente guarda o *session id* em *cookies.*
- O container Web faz toda a gestão dos *cookies* e *session ids* para o programador.

Sessões

- Para acessar a sessão:

```
//cria uma nova sessão caso não exista uma
HttpSession session = request.getSession();
```

- Para persistir um objeto em uma sessão:

```
HttpSession session = request.getSession();
session.setAttribute(“nomeObjeto”, objeto);
```

Sessões

- Para recuperar, em outro lugar do código, um objeto persistido em uma sessão:

```
HttpSession session = request.getSession();
Object objeto = session.getAttribute(“nomeObjeto”);
```

Sessões

- Para testar se a sessão é nova:

```
HttpSession session = request.getSession();
if session.isNew(){
  ...
}
```

Sessões

- Se é necessário utilizar uma sessão já criada, sem a possibilidade de criar uma nova.

```
//passando “false”, não é criada uma sessão nova caso ela
//não exista
HttpSession session = request.getSession(false);
if session != null{
 ...
}
```

Exercício

- Fazer carrinho de compras para a aplicação de cadastro de produtos.