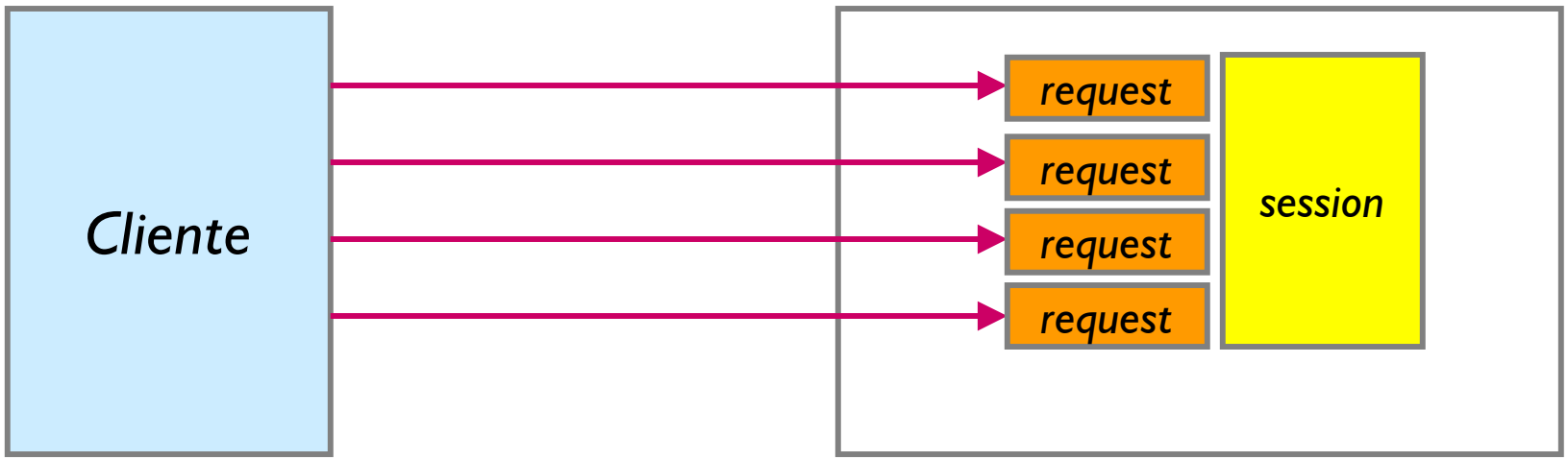
4 Sessões

Helder da Rocha (helder@acm.org)
www.argonavis.com.br

Sessões

- *Como o HTTP não mantém estado de sessão, são as aplicações Web que precisam cuidar de mantê-lo quando necessário*
- *Sessões representam um cliente*
  - *A sessão é única para cada cliente e persiste através de várias requisições*



Sessões

- Sessões são representados por objetos *HttpSession* e são obtidas a partir de uma requisição
- Dois métodos podem ser usados

```
HttpSession session = request.getSession(false);
```

  - Se a sessão não existir, retorna null, caso contrário retorna sessão.

```
HttpSession session = request.getSession();
```

  - Retorna a sessão ou cria uma nova. Mesmo que *getSession(true)*
- Para saber se uma sessão é nova, use o método *isNew()*

```
if (session.isNew()) {
  myObject = new BusinessObject();
} else {
  myObject = (BusinessObject) session.getAttribute("obj");
}
```

- *getSession()* deve ser chamado antes de *getOutputStream()**
  - Sessões podem ser implementadas com cookies, e cookies são definidos no cabeçalho HTTP (que é montado antes do texto)

*ou qualquer método que obtenha o stream de saída, como getWriter()

Compartilhamento de objetos na sessão

- *Dois métodos*
  - `setAttribute("nome", objeto);`
  - `Object getAttribute("nome");`

  permitem o compartilhamento de objetos na sessão. Ex:

  Requisição 1

```
String[] vetor = {"um", "dois", "tres"};
HttpSession session = request.getSession();
session.setAttribute("dados", vetor);
```

  Requisição 2

```
HttpSession session = request.getSession();
String[] dados = (String[])session.getAttribute("dados");
```

- *Como a sessão pode persistir além do tempo de uma requisição, é possível que a persistência de alguns objetos não sejam desejáveis*
  - *Use* `removeAttribute("nome")` *para remover objetos da sessão*

Gerência de sessões

- Não há como saber que cliente não precisa mais da sessão
  - Pode-se definir um timeout em minutos para a duração de uma sessão desde a última requisição do cliente

    `setMaxInactiveInterval(int)` define novo valor para timeout

    `int getMaxInactiveInterval()` recupera valor de timeout
  - Timeout default pode ser definido no web.xml para todas as sessões
  - Outros métodos úteis: getLastAccessedTime() e getCreationTime()
- Para destruir uma sessão use

  ```
  session.invalidate();
  ```
- Eventos de ligação e ativação de uma sessão podem ser controlados com implementações das interfaces HttpSessionBindingListener e HttpSessionActivationListener
  - Consulte a documentação. A abordagem dessas interfaces não faz parte do escopo deste curso

Timeout default no web.xml

- O elemento <session-config> permite definir a configuração da sessão
  - Deve aparecer depois dos elementos <servlet-mapping>
- O trecho abaixo redefine o tempo de duração default da sessão em 15 minutos para todas as sessões

```
<session-config>
    <session-timeout>15</session-timeout>
</session-config>
```

- Uma sessão específica pode ter uma duração diferente se especificar usando *setMaxInactiveInterval()*

Sessão à prova de clientes

- A sessão é implementada com cookies se o cliente suportá-los
  - Caso o cliente não suporte cookies, o servidor precisa usar outro meio de manter a sessão
- Solução: sempre que uma página contiver uma URL para outra página da aplicação, a URL deve estar dentro do método encodeURL() de HttpServletResponse

```
out.print("<a href='" +
                response.encodeURL("caixa.jsp") + "'>");
```

- Se cliente suportar cookies, URL passa inalterada (o identificador da sessão será guardado em um cookie)
- Se cliente não suportar cookies, o identificador será passado como parâmetro da requisição

  ex: http://localhost:8080/servlet/Teste;jsessionid=A424JX08S99

Captura de eventos de atributos

- É possível saber quando um atributo foi adicionado a uma sessão usando HttpSessionAttributeListener e HttpSessionBindingEvent
- Métodos a implementar do Listener
  - attributeAdded(ServletContextAttributeEvent e)
  - attributeRemoved(ServletContextAttributeEvent e)
  - attributeReplaced(ServletContextAttributeEvent)
- HttpSessionBindingEvent possui três métodos para recuperar sessão e nome e valor dos atributos
  - String getName()
  - String getValue()
  - HttpSession getSession()
- É preciso registrar o listener no web.xml

Captura de eventos do ciclo de vida

- Pode-se saber quando uma sessão foi criada, invalidada ou expirada usando HttpSessionListener:
  - Métodos sessionCreated() e sessionDestroyed()
- Para saber quando uma sessão existente foi ativada ou está para ser passivada usa-se HttpSessionActivationListener:
  - Métodos sessionDidActivate() e sessionWillPassivate()
- Para controlar quando objetos são associados a uma sessão e quando deixam a sessão (por qualquer razão) deve-se implementar um HttpSessionBindingListener
  - Métodos valueBound() e valueUnbound()
- Cada listener tem um evento correspondente, que é recebido em cada método. Para maiores detalhes, consulte a documentação e exemplos no Tomcat
  - Maiores detalhes sobre este assunto fogem ao escopo deste curso

Escopo de objetos em servlets

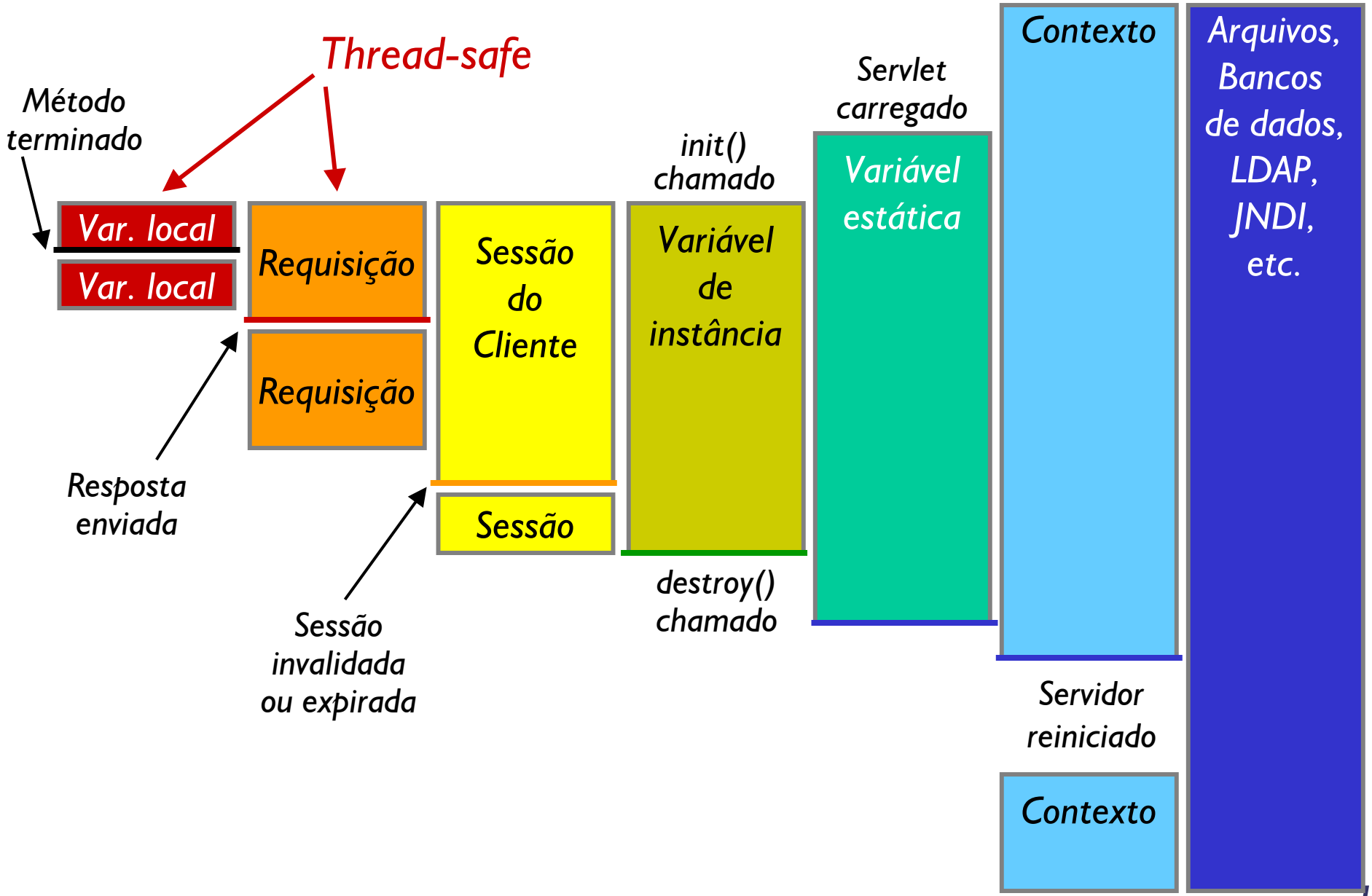
- Servlets podem compartilhar informações de várias maneiras
  - Usando meios persistentes (bancos de dados, arquivos, etc)
  - Usando objetos na memória por escopo (requisição, sessão, contexto)
  - Usando variáveis estáticas ou de instância
- Servlets oferecem três níveis diferentes de persistência na memória (ordem decrescente de duração)
  - Contexto da aplicação: vale enquanto aplicação estiver na memória (javax.servlet.ServletContext)
  - Sessão: dura uma sessão do cliente (javax.servlet.http.HttpSession)
  - Requisição: dura uma requisição (javax.servlet.ServletRequest)
- Para gravar dados em um objeto de persistência na memória

```
objeto.setAttribute("nome", dados);
```

- Para recuperar ou remover os dados

```
Object dados = objeto.getAttribute("nome");
objeto.removeAttribute("nome");
```

Escopo de objetos em servlets: resumo



Lidando com recursos compartilhados

- Há vários cenários de acesso concorrente
  - Componentes compartilhando sessão ou contexto
  - Threads acessando variáveis compartilhadas
- Servlets são automaticamente multithreaded
  - O container cria um thread na instância para cada requisição
  - É preciso sincronizar blocos críticos para evitar problemas decorrentes do acesso paralelo
- Exemplo: protegendo definição de atributo de contexto:

```
synchronized(this) {
    context.setAttribute("nome", objeto);
}
```

- Para situações onde multithreading é inaceitável, servlet deve implementar a interface SingleThreadModel (só um thread estará presente no método service() ao mesmo tempo)
  - Evite isto a todo custo: muito ineficiente!

Exercício

1. Criar uma aplicação Web usando os objetos de negócio
   - j550.cap04.Produto. Atributos (métodos get/set): int id, String nome, String preco
   - j550.cap04.Carrinho. Métodos: addProduto(Produto), removeProduto(id), Produto getProduto(id), Produto[] getProdutos()
   - Veja como usá-los na classe TestaProdutos.java
- a. Crie um servlet j550.cap04.web.AdminLojaServlet
  - LojaServlet recebe parâmetros para adicionar um produto e lista os produtos existentes como resposta
- b. Crie um servlet j550.cap04.web.ComprasServlet e classe
  - ComprasServlet lista todos os produtos disponíveis com um botão Adicionar ao lado de cada um. O botão deve adicionar o produto correspondente no objeto Carrinho.
  - A resposta deve mostrar cada item incluído com um botão Remover. Deve haver também botão Comprar Mais e Encerrar
  - O Carrinho deve persistir entre requisições

Cookies

- Como já podemos manipular sessões de maneira transparente com HttpSession, usamos cookies principalmente para definir preferências que irão durar além do tempo da sessão
  - Servidor irá criar cabeçalho que irá instruir o browser a criar um arquivo guardando as informações do cookie
- Para criar cookies que duram mais que uma sessão (cookies persistentes no disco do cliente) é preciso
  - Criar um novo objeto Cookie
  - Definir a duração do cookie com o método setMaxAge()
  - Definir outros métodos se necessário
  - Adicionar o cookie à resposta

Como usar Cookies

- *Exemplo de gravação: 1) definir um cookie que contenha o nome do usuário recebido como parâmetro na requisição*

```
String nome = request.getParameter("nome");
Cookie c = new Cookie("usuario", nome);
```

- *2) Definir a duração do cookie em segundos*

```
c.setMaxAge(1000 * 24 * 3600 * 60); // 60 dias
```

- *3) Adicionar o cookie à resposta*

```
response.addCookie(c);
```

- *Exemplo de leitura: 1) recuperar o cookie da requisição*

```
Cookie[] cookies = request.getCookies();
```

- *2) Extrair cookie para um objeto local*

```
for (int i = 0; i < cookies.length; i++) {
   if (cookies[i].getName().equals("nome") {
        usuario = cookies[i].getValue();
    }
}
```

Exercícios

- 2. Crie uma tela de entrada na loja LojaServlet com links para os servlets.
  - Ela deve requisitar um e-mail. Grave o e-mail como um Cookie com duração de 30 dias. "Lembre-se" do e-mail na próxima requisição e mostre-o no text-field

helder@acm.org

argonavis.com.br