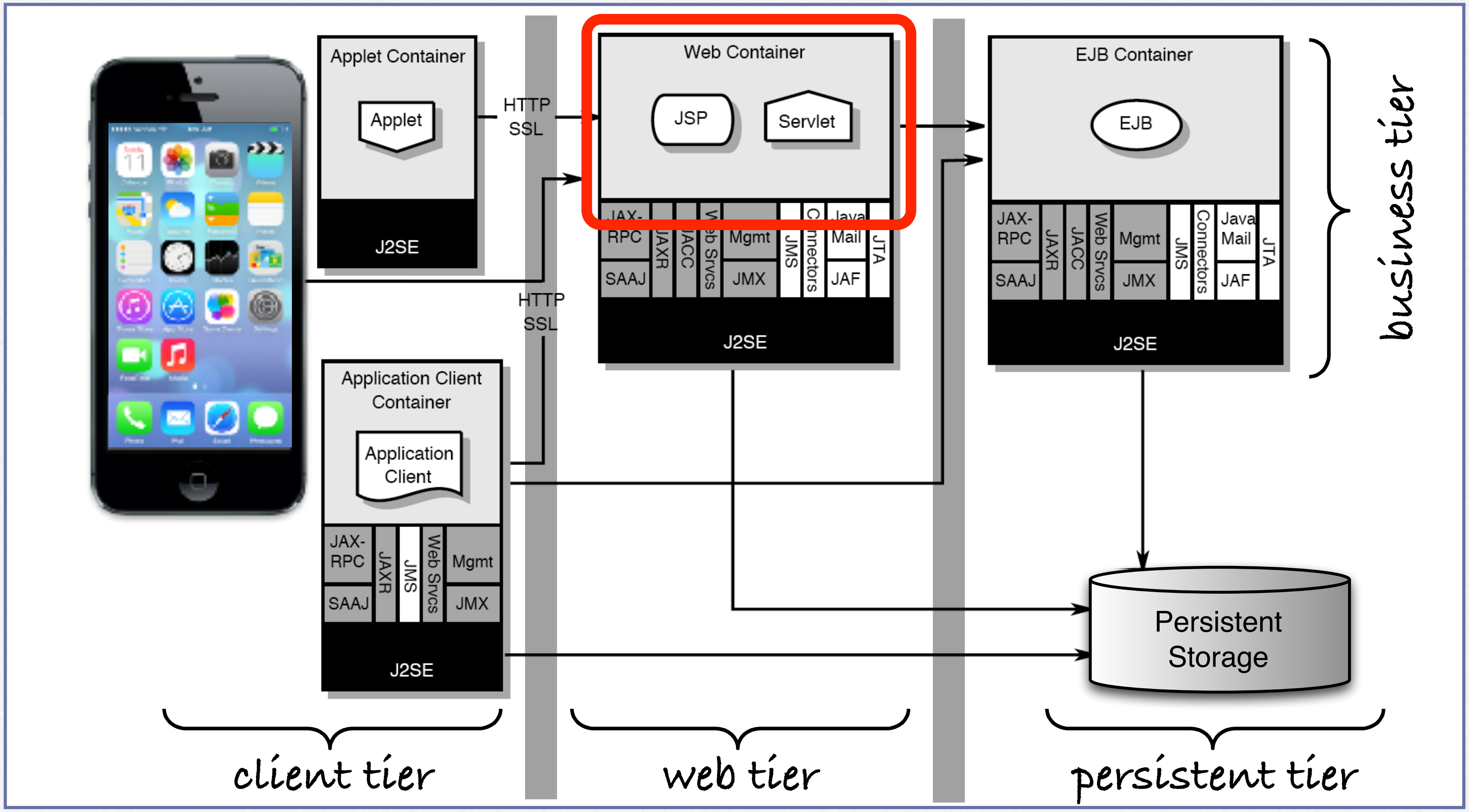
Web Tier
(Servlets & JSPs)

Benoît Garbinato
distributed object programming lab

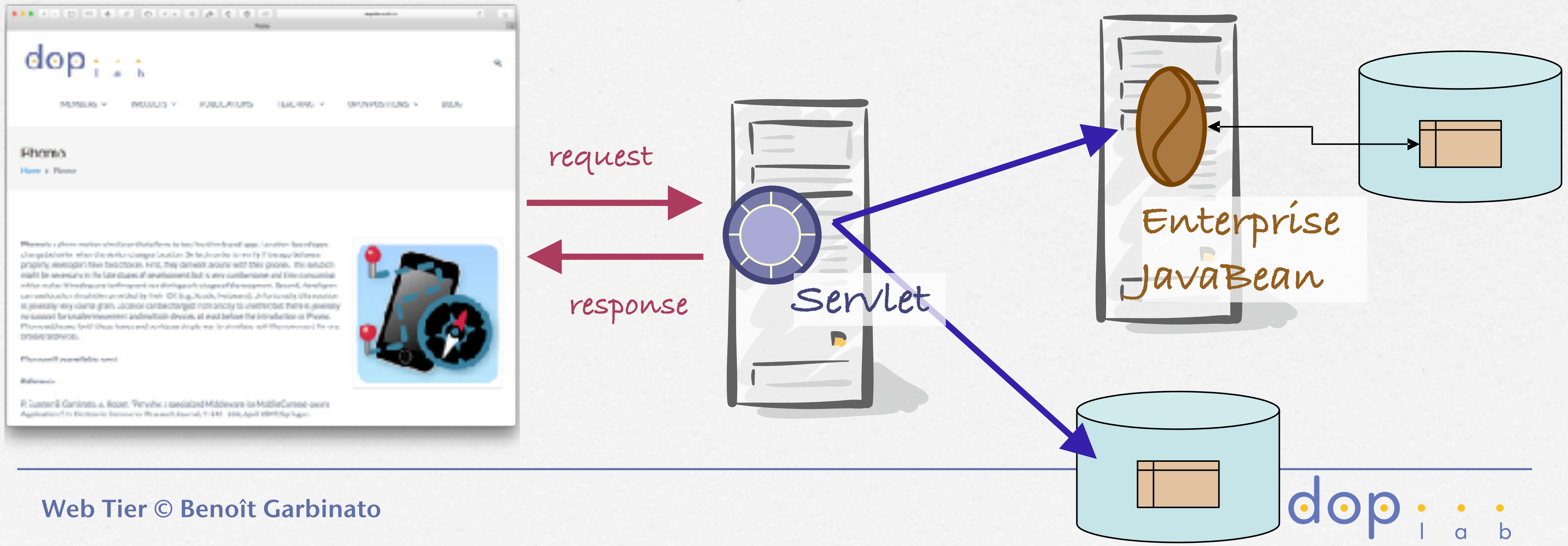
Background



Applet Container
Applet
J2SE
HTTP
SSL
Web Container
JSP
Servlet
JAX-RPC
JAXR
JACC
Web Srvcs
Mgmt
JMS
Connectors
Java Mail
JTA
SAAJ
JMX
JAF
J2SE
EJB Container
EJB
JAX-RPC
JAXR
JACC
Web Srvcs
Mgmt
JMS
Connectors
Java Mail
JTA
SAAJ
JMX
JAF
J2SE
business tier
HTTP
SSL
Application Client Container
Application Client
JAX-RPC
JAXR
JMS
Web Srvcs
Mgmt
SAAJ
JMX
J2SE
Persistent Storage
client tier
web tier
persistent tier

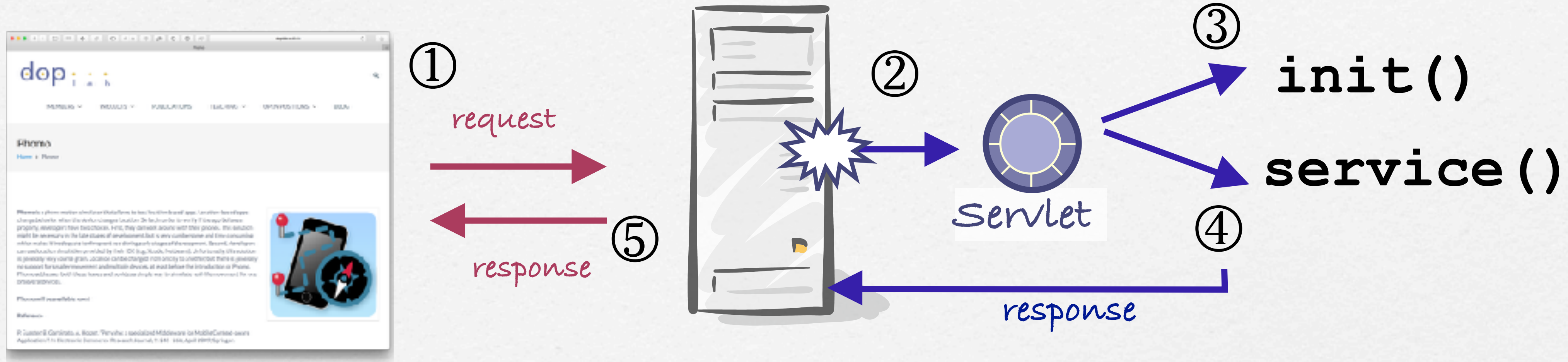
What are servlets?

- They dynamically generate responses to requests sent by web clients
- They are container-managed components

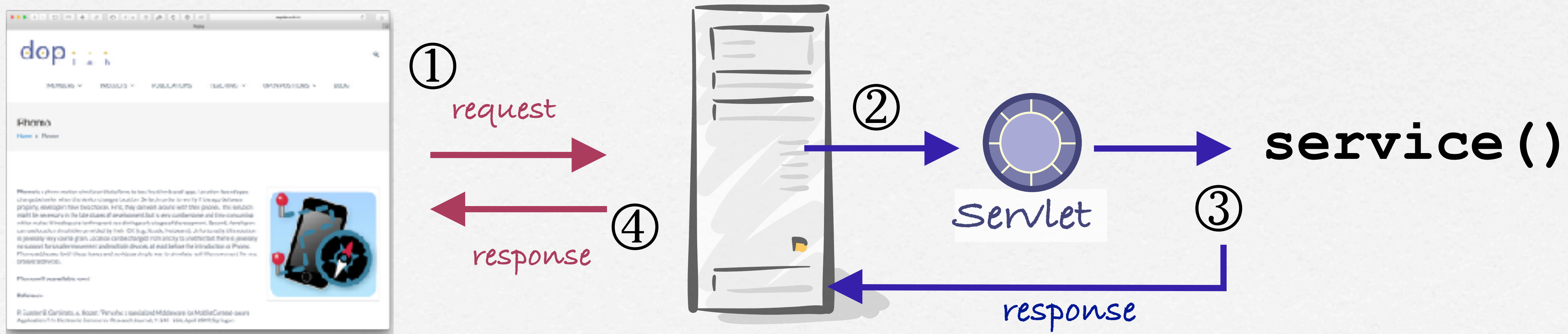


Servlet lifecyle

upon first request...



upon subsequent requests...



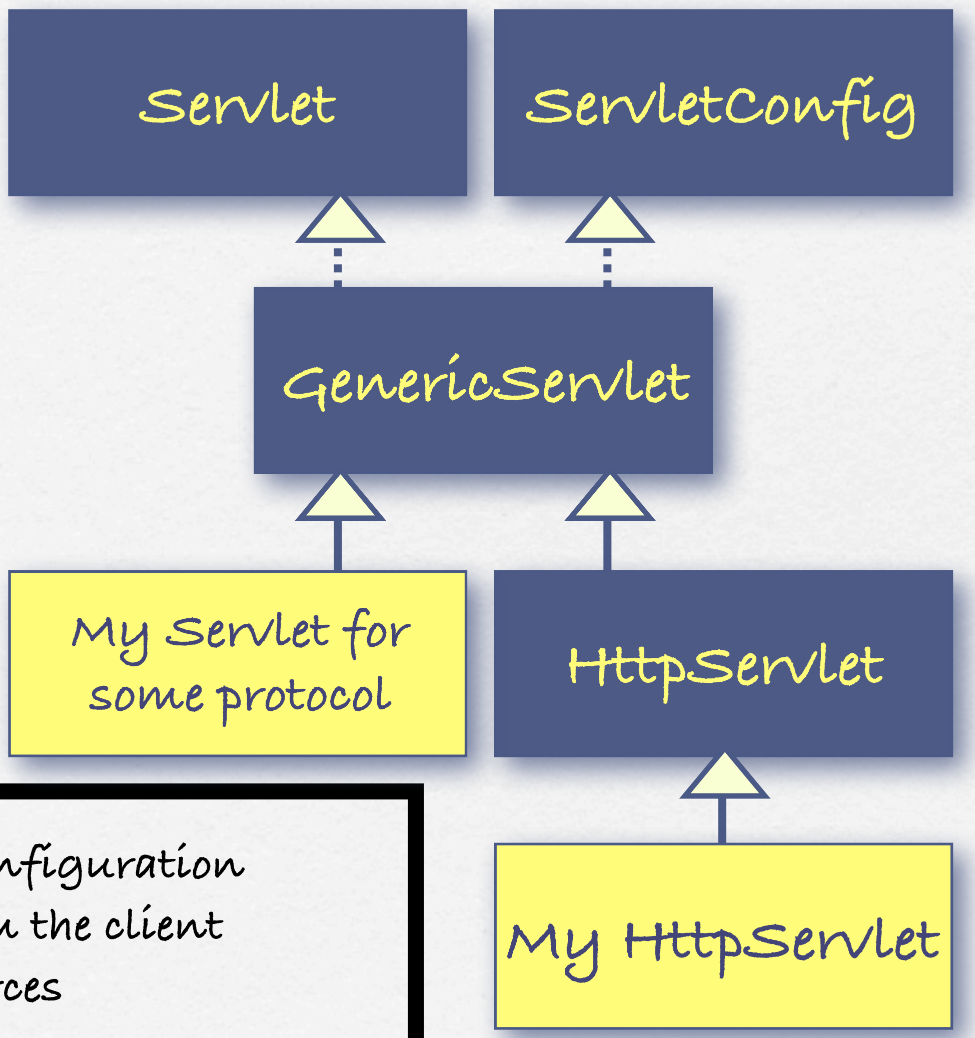
Servlet interfaces & hierarchy

Typicall callbacks

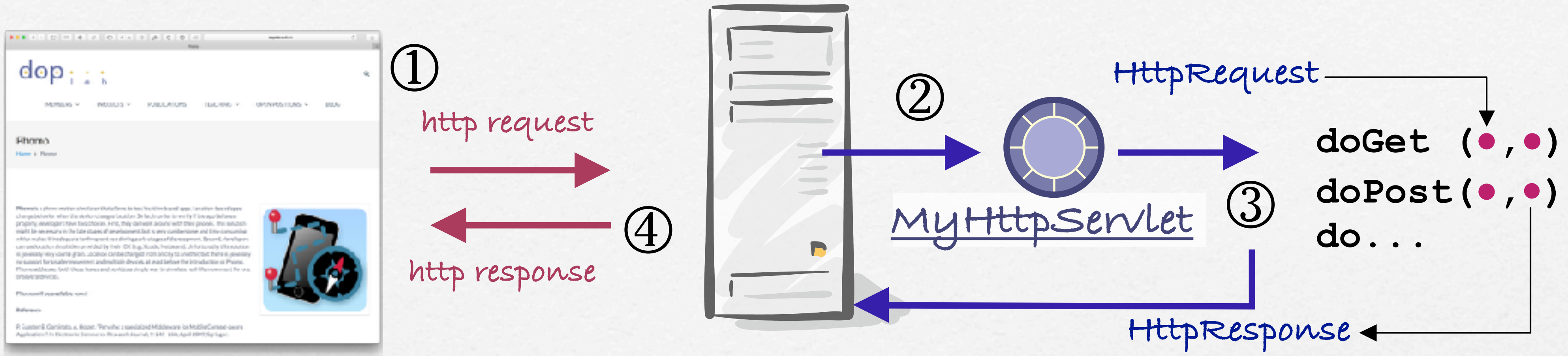
`doGet()`	request information from the server
`doPost()`	modify persistent data on the server
`doPut()`	analogous to sending a file via FTP
`doDelete()`	removes a Uniform Resource Identifier (URI) from the server

Lifecycle management

`init()`	performs one-time setup and configuration
`service()`	carries out a single request from the client
`destroy()`	destroys a servlet and its resources



What do you write?

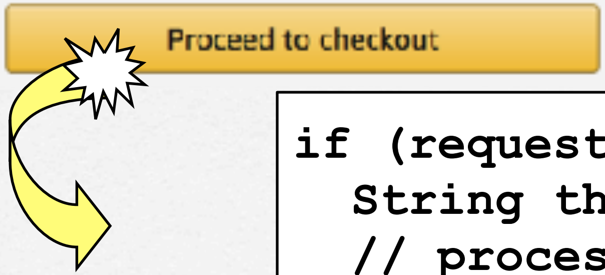


```
public class MyHttpServlet extends HttpServlet {
   ...
   protected void doGet(...) throws... {
      processRequest(request, response);
   }
   protected void doPost(...) throws... {
      processRequest(request, response);
   }
   protected void processRequest(...) throws...{
      // Do the actual work here
   }
   ...
}
```

How to retrieve parameters?

Parameters in HTML form:

```
<html>
  <form METHOD="POST">
    <input TYPE="hidden" NAME="CustomerID" VALUE="10020">
    ...
    <input TYPE="submit" NAME="Buy" VALUE="Buy">
    <input TYPE="submit" NAME="Refresh" VALUE="Refresh">
  </form>
</html>
```



Retrieving parameters in Servlet code:

```
if (request.getParameter("Buy") != null) {
  String thisCustomer = request.getParameter("CustomerID");
  // process buy request...
}
else if (request.getParameter("Refresh") != null) {
  // refresh the screen...
}
```

How to generate a response?

Printing data in servlet code:

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html><title>User Login</title>");
out.println("<h3>Please enter your username and password:");
out.println("<form method=\"POST\">");
  ...
out.println("<input type=\"submit\" name=\"LOGIN\" value=\"Login\">");
out.println("</form></html>");
out.close();
```

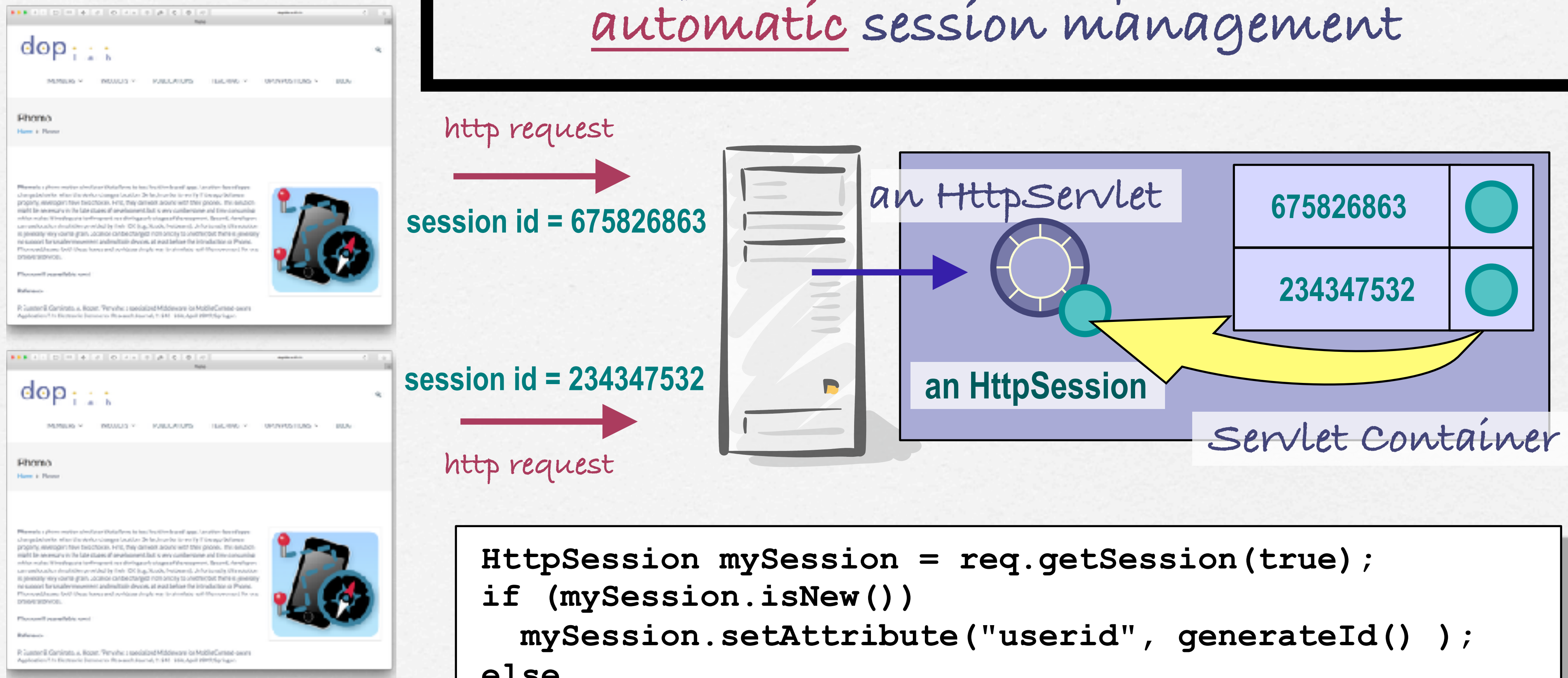
Generated HTML form:

```
<html> <title>User Login</title>
  <h3>Please enter your username and password:
  <form method="POST">
      ...
  <input type="submit" name="LOGIN" value="Login">
</form></html>
```

Session management (1)

HTTP is a stateless protocol but...

→ The Java Servlet API provides automatic session management



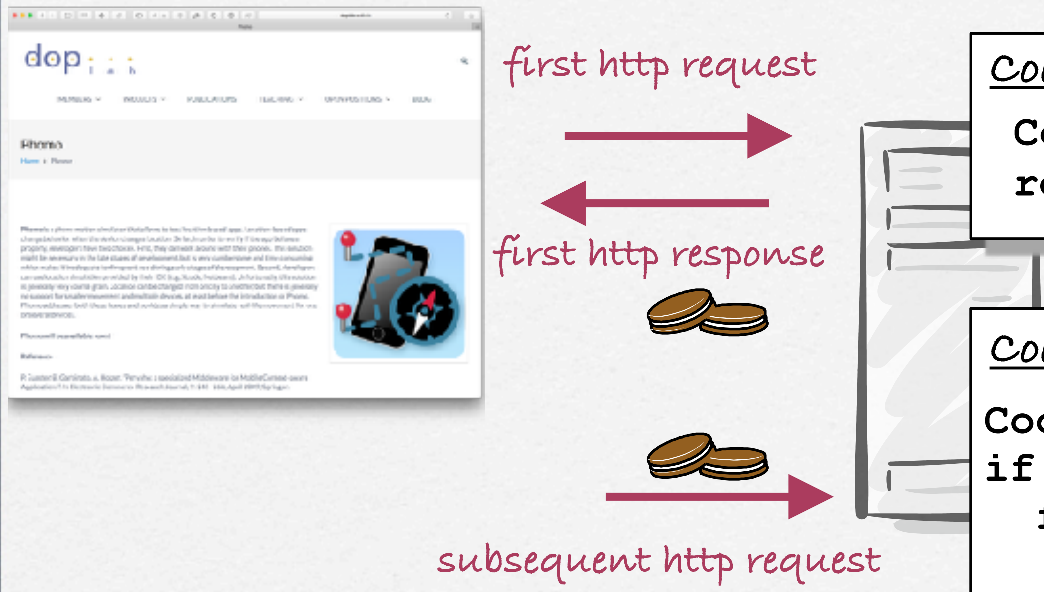
```
HttpSession mySession = req.getSession(true);
if (mySession.isNew())
  mySession.setAttribute("userid", generateId() );
else
  userid= (Integer) mySession.getAttribute("userid");
```

Session management (2)

- When using servlets, session management is done automatically for you, i.e., you do not manage session IDs yourself

- For doing this, the web container relies on one or more of the following techniques:
  - implicit cookies
  - URL rewriting
  - hidden fields

Managing cookies explicitly



<u>Cookie creation:</u>

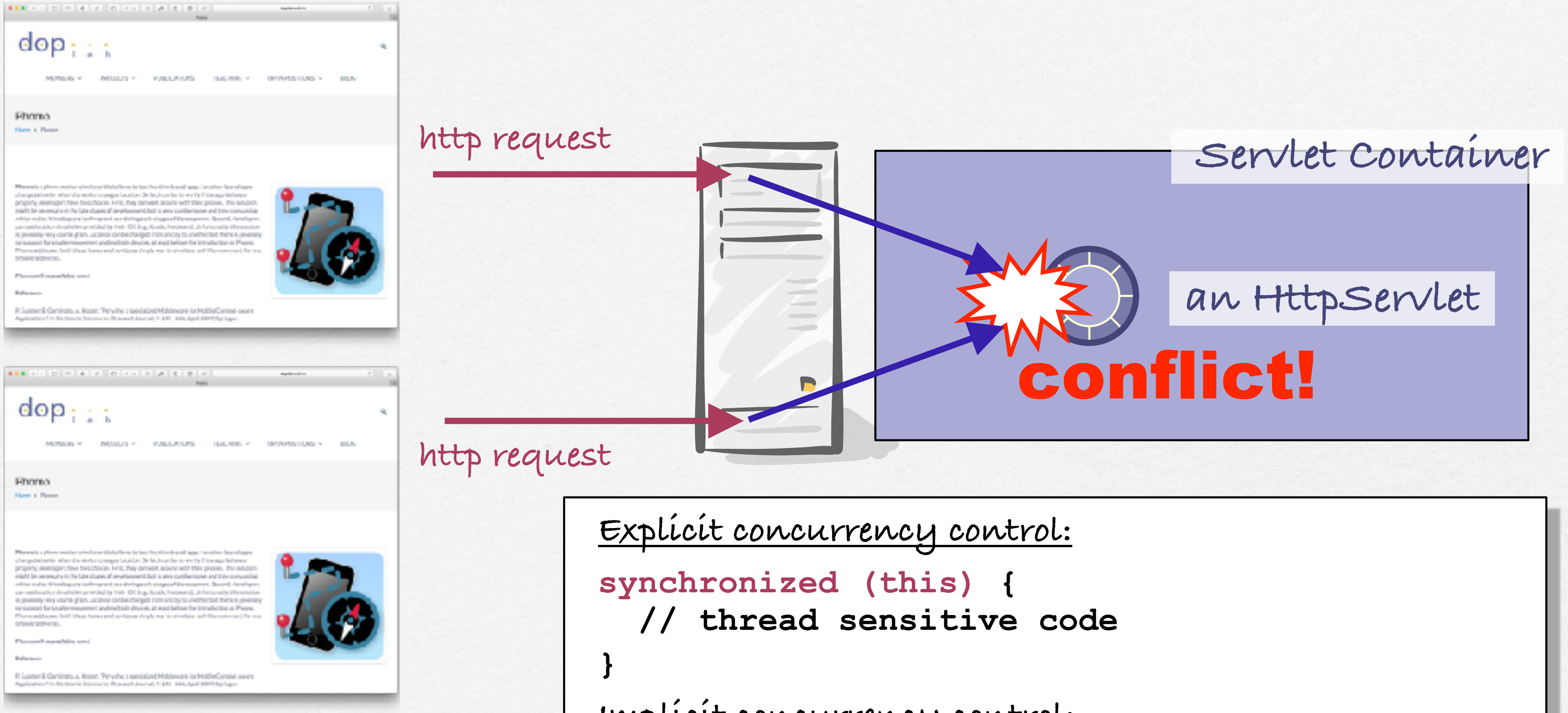
```
Cookie c = new Cookie("color","green");
response.addCookie(c);
```

<u>Cookie retrieval:</u>

```
Cookie[] cookies = request.getCookies();
if(cookies != null) {
  for (int x=0; x<cookies.length; x++){
    String name = cookies[x].getName();
    if (name.equals("color")) {
      String clr = cookies[x].getValue();
      break;
    }
  }
}
```

Browsers are expected to support at least 20 cookies for each web server, 300 cookies total, and may limit cookie size, e.g., to 4 KB each

How to manage concurrency?



Explicit concurrency control:

```
synchronized (this) {
  // thread sensitive code
}
```

Implicit concurrency control:

```
public class MyHttpServlet extends HttpServlet
      implements SingleThreadModel {
         // servlet code
}
```

What are JavaServer Pages?

- They dynamically generate responses to requests sent by web clients
- They are container-managed components

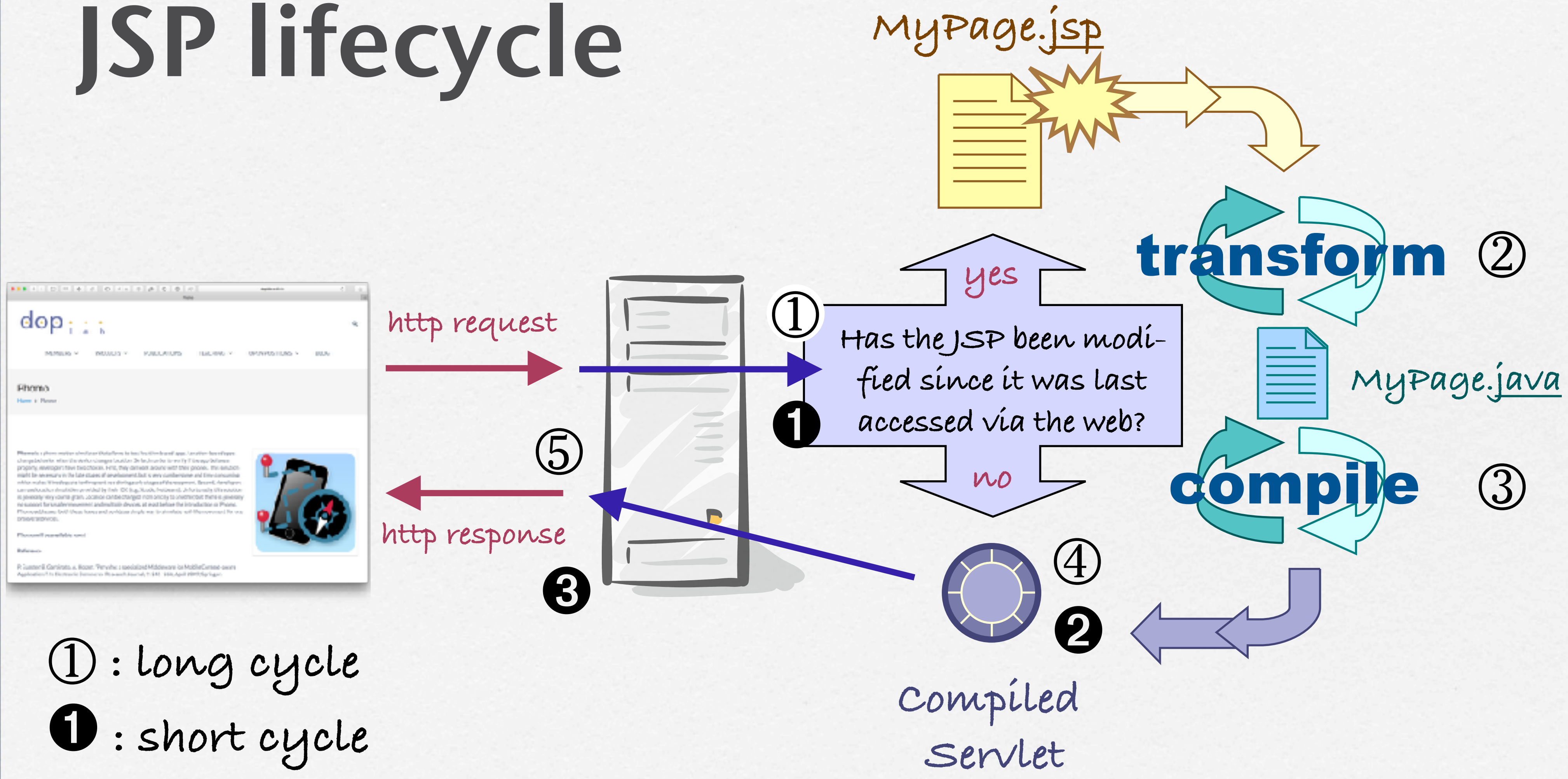
→ So how do they differ from servlets?

- They "reverse" the Java – HTML relationship
- They are a facility built on top of servlets
- They allows web authors & web developers to work jointly on the same page

Overview of JSP syntax

```
<%@ page import="java.util.*,ProductCatalog.CatalogItem" %>
<%! Iterator it=null; Vector cpi=null; CatalogItem ci=null;%>
<html><head><title>Shopping Catalog</title></head><body>
<jsp:useBean id="catalogPageData" scope="session" class="MVCApp.CatalogPageData" />
<table>
<%
cpi = catalogPageData.getCatalogPageItems();
if (cpi != null){
  it = itemList.iterator();
  while (it.hasNext()) {
    ci = (CatalogItem)it.next();
%>
  <tr>
    <td><%= ci.getTitle() %></td>
    <td><form method="get" action="../servlet/MVCApp.Controller">
          <input type="hidden" name="itemId" value="<%= ci.getId() %>">
          <input type="submit" name="addButton" value="Add Item To Cart">
          <input type="hidden" name="action" value="addItemToCart">&nbsp;&nbsp;
          <input type="text" size="3" name="quantity" value="1">
          </form></td>
  </tr>
<%}%>
</table>
```

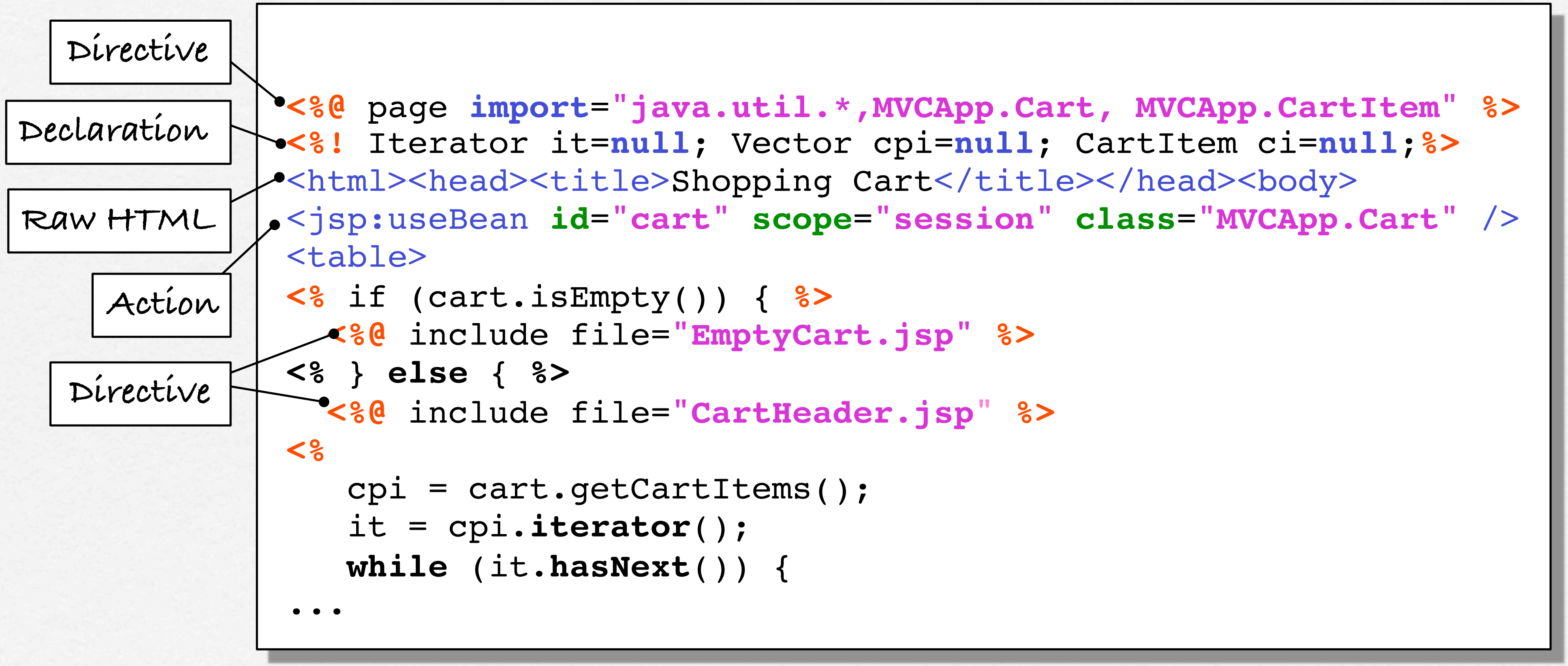
JSP lifecycle



① : long cycle

❶ : short cycle

Elements of a JSP (1)



```
<%@ page import="java.util.*,MVCApp.Cart, MVCApp.CartItem" %>
<%! Iterator it=null; Vector cpi=null; CartItem ci=null;%>
<html><head><title>Shopping Cart</title></head><body>
<jsp:useBean id="cart" scope="session" class="MVCApp.Cart" />
<table>
<% if (cart.isEmpty()) { %>
   <%@ include file="EmptyCart.jsp" %>
<% } else { %>
   <%@ include file="CartHeader.jsp" %>
<%
   cpi = cart.getCartItems();
   it = cpi.iterator();
   while (it.hasNext()) {
...
```

Elements of a JSP (2)

Scriptlet

Expression

Implicit object

```
...
<%@ include file="CartHeader.jsp" %>
<%
   cpi = cart.getCartItems();
   it = cpi.iterator();
   while (it.hasNext()) {
     ci = (CartItem)it.next();
%>
  <tr valign="top">
    <td><%= ci.getTitle() %></td>
    <td align="right"><%= ci.getFormattedUnitCost() %></td>
    <td align="right"><%= ci.getQuantity() %></td>
  </tr>
<% } // end while %>
</table>
<% } // end if %>
<% String action = request.getParameter("action"); %>
...
```

Implicit objects...

... are predefined variables,

... do not require declaration,

... include (among others):

`request`	request triggering the service
`response`	response to the request
`session`	object representing the client session

Custom tab libraries

- A custom tag library extends the set of tags a JSP container can interpret
- A custom tag library associates a tag prefix with a tag library

```
...
<%@ taglib uri="http://forte.webtags" prefix="table" %>
...
<table:showTable cells="4" object="cart.getItems() ">
  <table:Parameter parameter="tableHeader"
    value="Title;Artist;Price;Quantity" />
  <table:Parameter parameter="tableItems"
    value="getTitle();getArtist();getPrice();getQty()" />
</table:showTable>
...
```