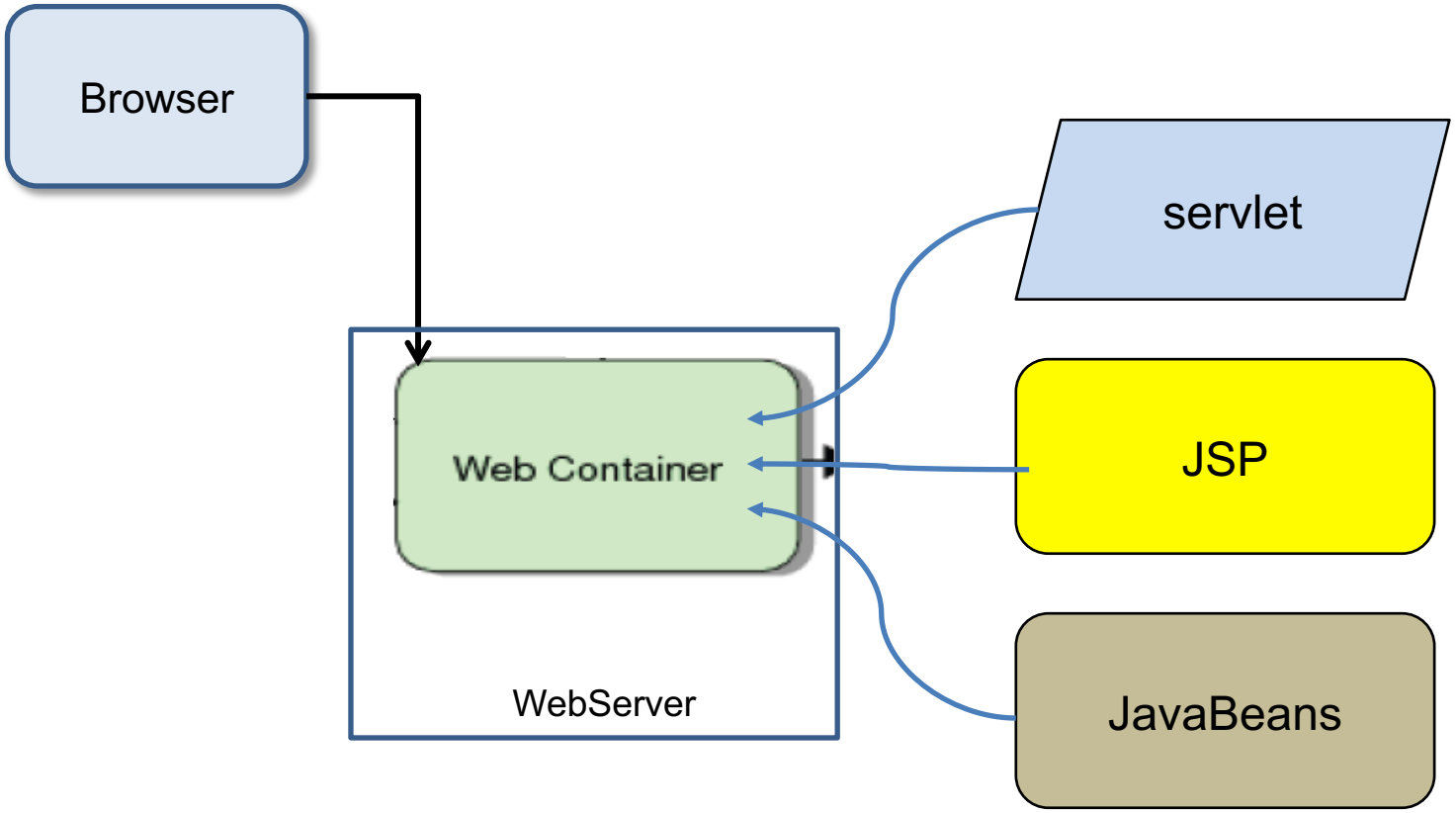
JSP, JavaBeans, CDI, Multithreading

Modul 5 - Java EE Web-Application Components



Servlet

```
@WebServlet(urlPatterns = {"/DemoServlet"})
public class DemoServlet extends HttpServlet {

protected void processRequest(HttpServletRequest
                request, HttpServletResponse response)
                  throws ServletException, IOException {
   response.setContentType("text/html;charset=UTF-8");
   try (PrintWriter out = response.getWriter()) {

            out.println("<!DOCTYPE html>");
            out.println("<html>");

            // Do Calculation here ...

            out.println("</html>");
        }
    }
```

JavaBean

Modul 5
JavaBeans – What are JavaBeans?

??????????

??????????

What is a JavaBean exactly?

A simple plain java class with:

- All properties private (use getters/setters)
- A public no-argument constructor
- Implements Serializable

```
public class MyJavaBean {
    int count = 0;

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }

    public void incCount() throws InterruptedException {
        this.count = this.count + 1000;
        Thread.sleep(2000);
        this.count = this.count - 999;
    }

}
```

```
public class MyJavaBean implements java.io.Serializable {

    private int count = 0;

    public MyJavaBean() {}

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }

    public void incCount() throws InterruptedException {
        this.count = this.count + 1000;
        Thread.sleep(2000);
        this.count = this.count - 999;
    }
}
```

Serializable

field

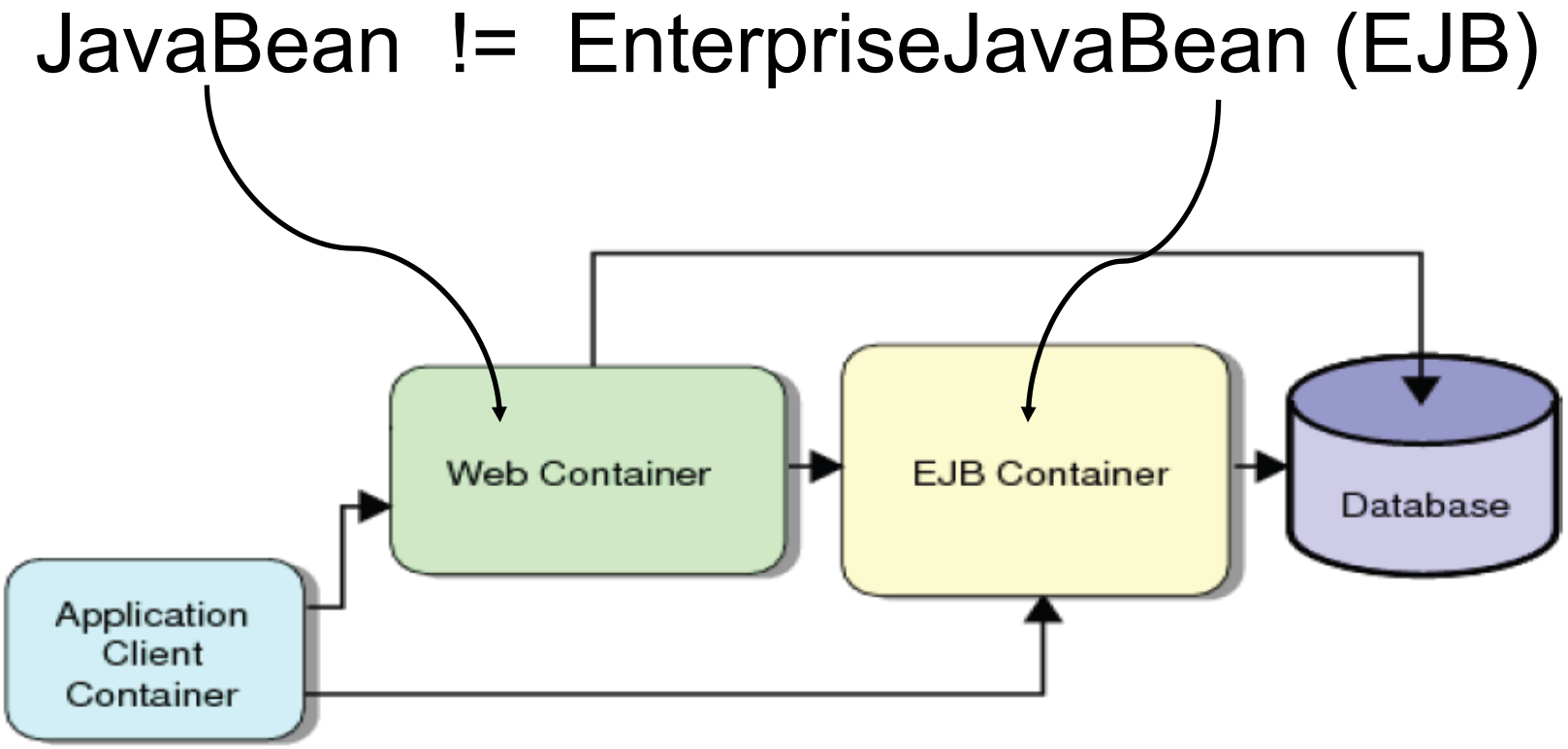
Empty Constructor

getter

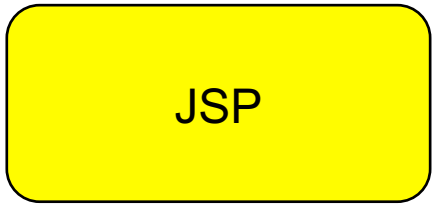
setter

Modul 5
JavaBeans

```
@WebServlet(name = "MultiThreadServlet", urlPatterns = {"/MultiThreadServlet"})
public class MultiThreadServlet extends HttpServlet {

    private MyJavaBean myBean = new MyJavaBean();

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html> <h1>");

            myBean.incCount();
            out.println(" count: " + myBean.getCount());

            out.println("</h1> </html>");
        } catch (InterruptedException ex) {
            Logger.getLogger(MultiThreadServlet.class.getName()).log(Level.SEVERE, null, ex);
        } finally {
            out.close();
        }
    }
```


JavaBean != EnterpriseJavaBean (EJB)
Web Container
EJB Container
Database
Application Client Container

JSP



Modul 5
JSP vs servlet

```
@WebServlet(urlPatterns = {"/DemoServlet"})
public class DemoServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");

            out.println("</html>");
        }
    }
```

SERVELT

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <%= new Date() %>
    </body>
</html>
```

JSP

Modul 5
JSP – Main Idea

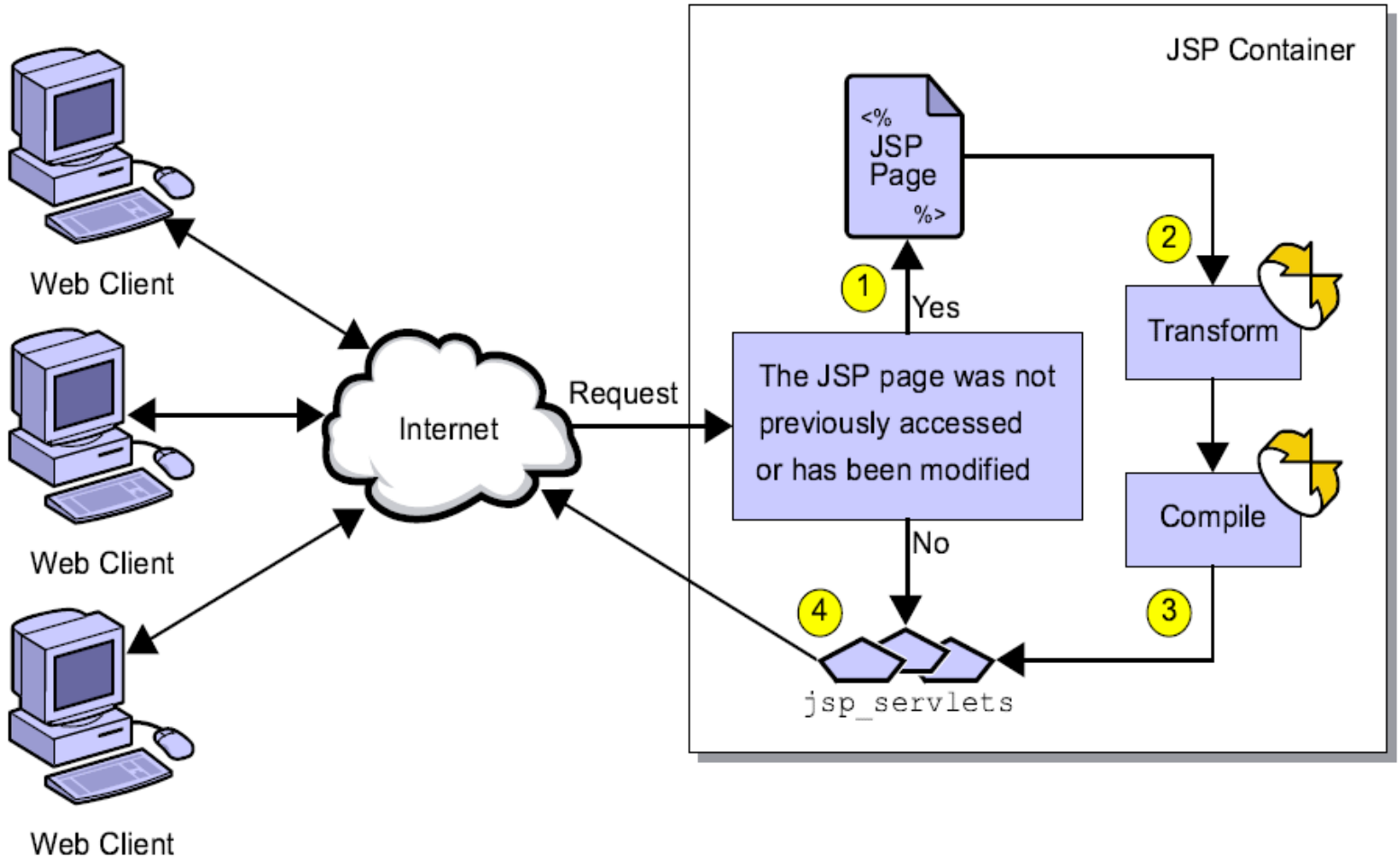
Layout in HTML

+

```
…

<%= new java.util.Date() %>

…

<%= count++ %>

…
```

insert some code for dynamic values

Modul 5
JSPs are transformed into servlets



Modul 5
JSPs are transformed into servlets

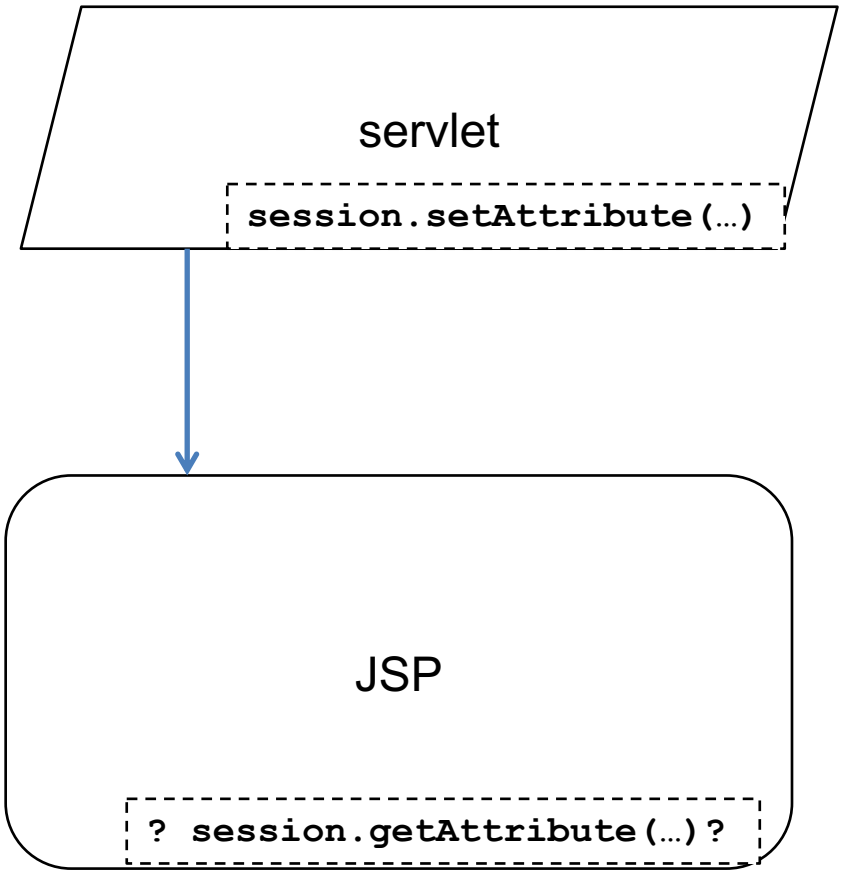
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head><title>First Example</title></head>
    <body>
        <h2>Hello World-JSP</h2>

        Your browser is: <%= request.getHeader("User-Agent")%><br>
        Current Time: <%= new java.util.Date() %><br>

    </body>
</html>
```

Quelle: http://docs.oracle.com/javaee/6/tutorial

Modul 5
JSPs are transformed into servlets

```
…
out.write("\n");
out.write("\n");
out.write("\n");
out.write("<!DOCTYPE html>\n");
out.write("<html>\n");
out.write("    <head><title>First Example</title></head>\n");
out.write("    <body>\n");
out.write("        <h2>Hello World-JSP</h2>\n");
out.write("\n");
out.write("        Your browser is: ");
out.print( request.getHeader("User-Agent"));
out.write("<br>        \n");
out.write("        Curretn Time: ");
out.print( new java.util.Date() );
out.write("<br>\n");
out.write("        \n");
out.write("    </body>\n");
out.write("</html>");
…
```

Modul 5
Processing Attributes From Servlets



Modul 5
Processing Attributes From Servlets

```
…
            HttpSession se = request.getSession();
            se.setAttribute("me", "It's me my friend!");
…
```
servlet

```
…
    <body>
        <h1>Hello World!</h1>

        <%= session.getAttribute("me")%>  <br>

        ${sessionScope.me} <br>

        ${me}

    </body>
…
```
JSP

Modul 5
Processing Objects From Servlets

The `jsp:useBean` Action

Syntax of `jsp:usebean`:

```
<jsp:useBean id="name" scope="scope" typeSpec />
```

Alternate syntax with initialization code:

```
<jsp:useBean id="name" scope="scope" typeSpec >
  <% ...initialization code... %>
</jsp:useBean>
```

`typeSpec` can be any of the following:

```
class="className"
class="className" type="typeName"

type="typeName"
```

Too Old

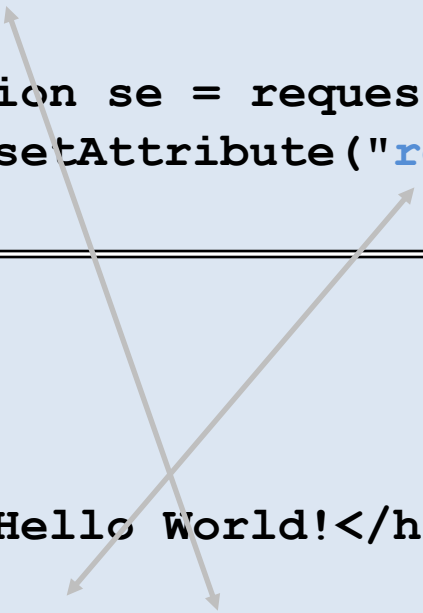
Modul 5
Processing Objects From Servlets

```
...
    MyObject myObj = new MyObject();
    myObj.name="Jan";

    HttpSession se = request.getSession();
    request.setAttribute("reqTag", myObj );
...
```

servlet

```
...

    <body>

        <h1>Hello World!</h1>

        ${ reqTag.name }

    </body>
...
```

JSP



Modul 5
Java Standard Tag Library (JSTL)

Functional Area	URI	Prefix
core	http://java.sun.com/jsp/jstl/core	c
XML processing	http://java.sun.com/jsp/jstl/xml	x
I18N capable formatting	http://java.sun.com/jsp/jstl/fmt	fmt
relational db access (SQL)	http://java.sun.com/jsp/jstl/sql	sql
Functions	http://java.sun.com/jsp/jstl/functions	fn

Modul 5
JSTL Example

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<c:forEach var="item" items="${requestScope.list}">
   <tr><td>${item.var1}</td><td>${item.var2}</td></tr>
</c:forEach>

<c:if test="${x < 3}" >

</c:if>

<c:choose>
   <c:when test="${requestScope.message == null}">

   </c:when>
   <c:otherwise>

   </c:otherwise>
</c:choose>
```

Modul 5
Expression Language

The Expression Language is an easy to use language that can be embedded in JSP pages instead of scriptlets (when used with tag libraries). Its syntax is similar to JavaScript™ and can be learned by non-programmers.

```
${ 3 + 2 }

${ param.address }

${ requestScope.customer.name }

${ not empty sessionScope.message }
```

Modul 5
Expression Language vs. scriptlets

```
<%@ page import="org.jsptutorial.examples.firstexample.domain.*,
                 java.util.*" %>
<%-- ... --%>
<%
if (pageContext.findAttribute("answers") != null &&
                                ((List)pageContext.findAttribute("answers")).size() != 0) {
   List<VocabAnswer> allAnswers = (List<VocabAnswer>)pageContext.findAttribute("answers");
%>
   <table id="resultsTable" cellpadding="0" cellspacing="0" border="0">
      <tr>
         <th><fmt:message key="showResultsPage.th.orig" /></th>
         <th><fmt:message key="showResultsPage.th.translation" /></th>
         <th><fmt:message key="showResultsPage.th.answerGiven" /></th>
      </tr>
<%
   for (VocabAnswer answer: allAnswers) {
%>
      <tr class="isCorrect_<%= Boolean.toString(answer.getIsCorrect()) %>">
         <td><%= answer.getOrig().getOrig() %></td>
         <td><%= answer.getOrig().getTranslation() %></td>
         <td><%= answer.getAnswerText() %></td>
      </tr>
<%
   }
%>
   </table>
<%
}
%>
```

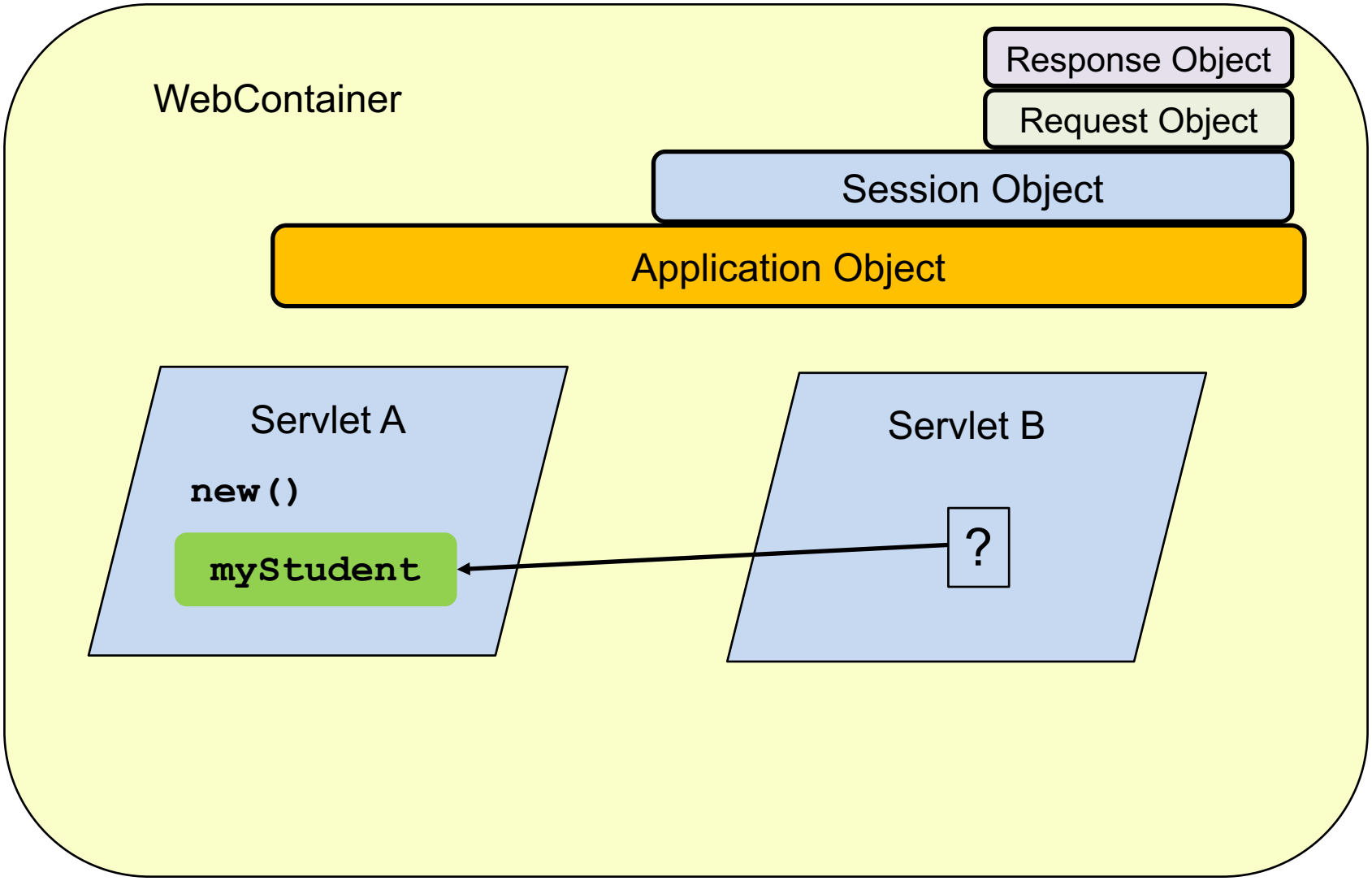
Quelle: http://www.jsptutorial.org/

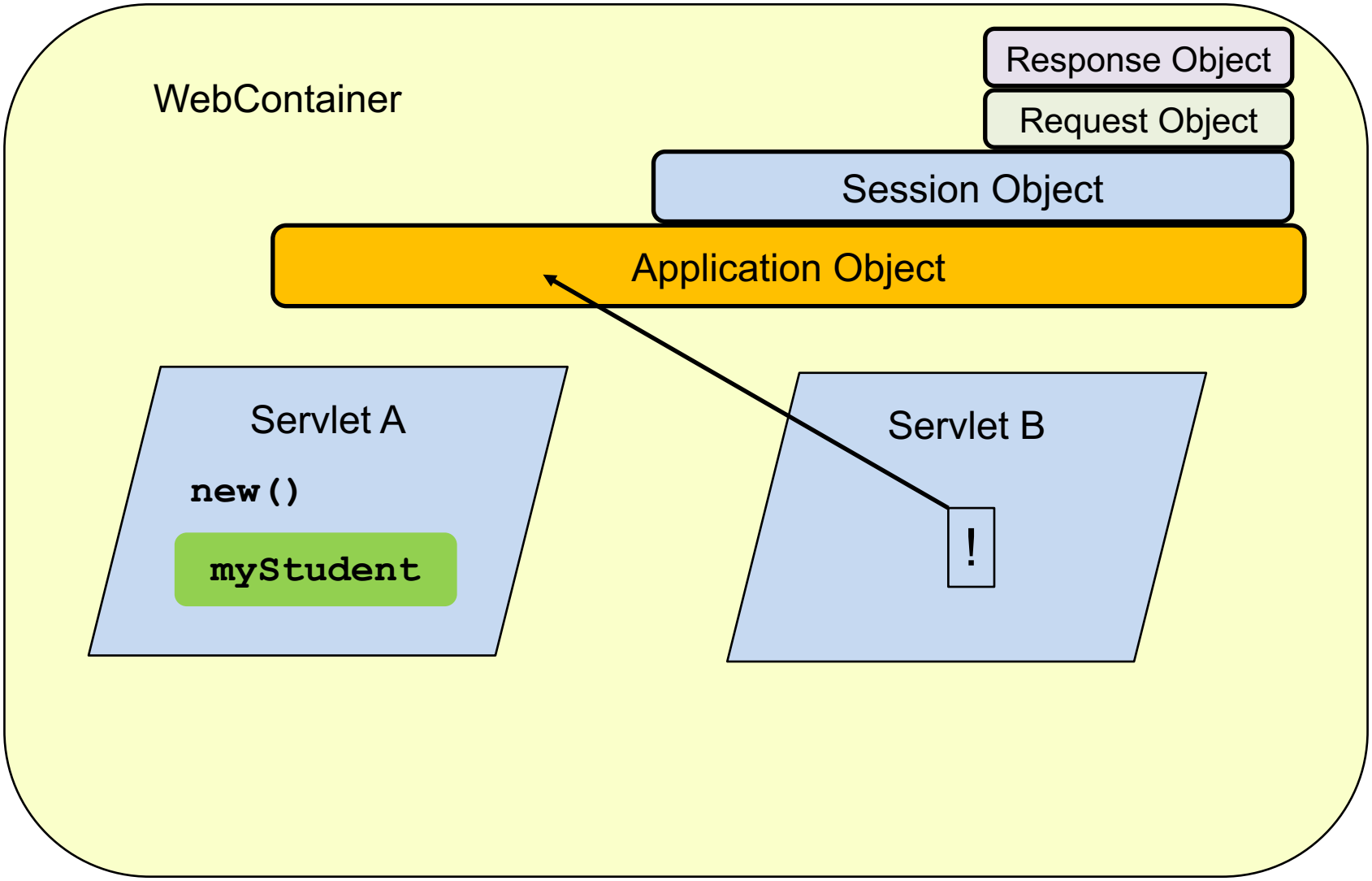
Modul 5
Expression Language vs. scriptlets

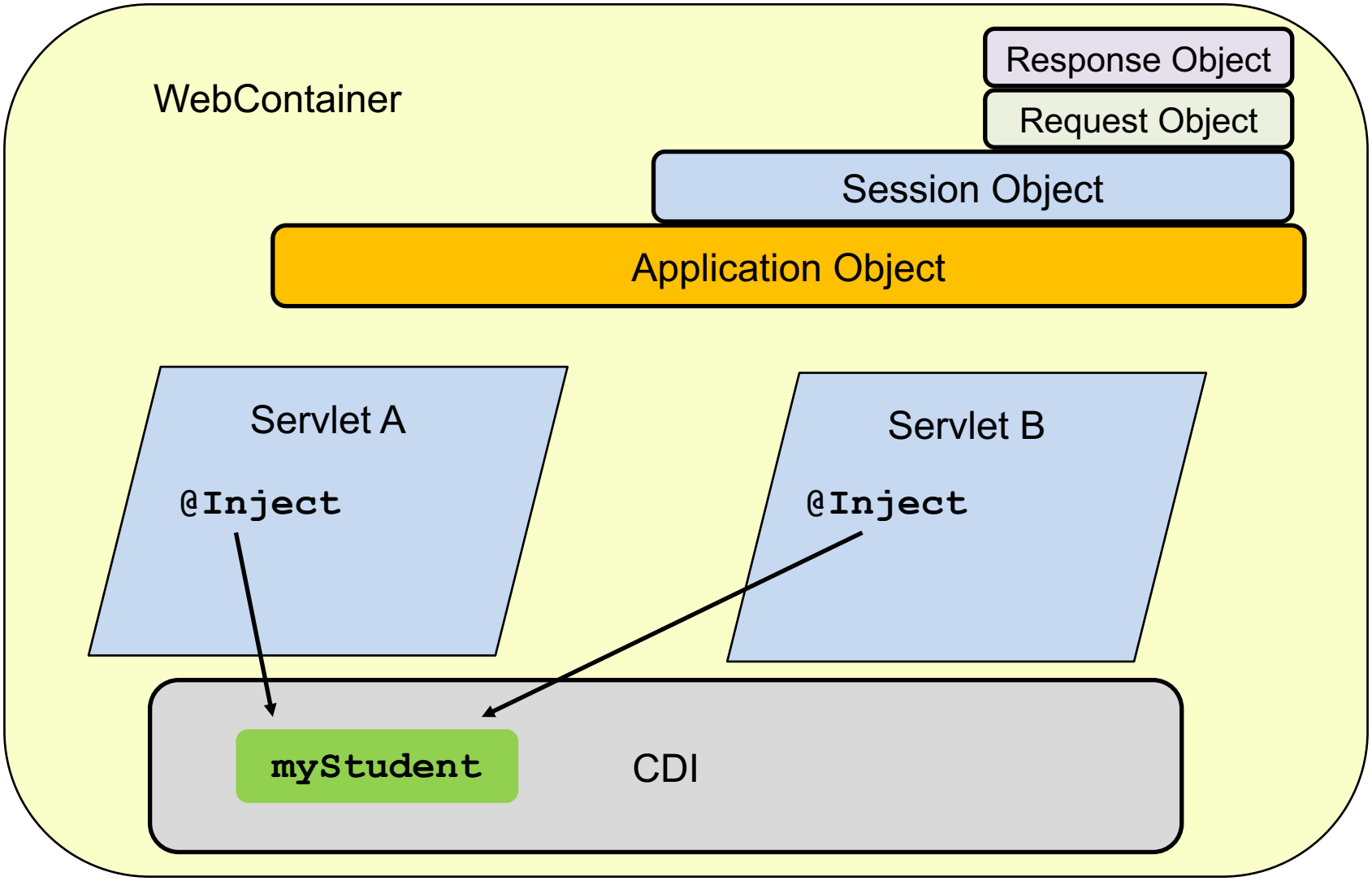
```
<c:if test="${not empty answers}">
   <table id="resultsTable" cellpadding="0" cellspacing="0" border="0">
      <tr>
         <th><fmt:message key="showResultsPage.th.orig" /></th>
         <th><fmt:message key="showResultsPage.th.translation" /></th>
         <th><fmt:message key="showResultsPage.th.answerGiven" /></th>
      </tr>
      <c:forEach items="${answers}" var="answer">
         <tr class="isCorrect_${answer.isCorrect}">
            <td>${answer.orig.orig}</td>
            <td>${answer.orig.translation}</td>
            <td>${answer.answerText}</td>
         </tr>
      </c:forEach>
   </table>
</c:if>
```

Quelle: http://www.jsptutorial.org/

CDI (Part 1)


WebContainer
Response Object
Request Object
Session Object
Application Object
Servlet A
new()
myStudent
Servlet B
?


WebContainer
Response Object
Request Object
Session Object
Application Object
Servlet A
new()
myStudent
Servlet B
!


WebContainer
Response Object
Request Object
Session Object
Application Object
Servlet A
@Inject
Servlet B
@Inject
myStudent
CDI

```
@Named                  // Objekt wird durch CDI erzeugt
@SessionScoped          // Bei jeder neuen Session
public class Student {

    private String name;
    private int matNum;

    // Setter & Getter-Methoden ...

}
```

Objekte sollen durch den CDI-Container erzeugt werden

```
@Named                  // Objekt wird durch CDI erzeugt
@RequestScoped          // Bei jeder neuen Request
public class Student {

    private String name;
    private int matNum;

    // Setter & Getter-Methoden ...

}
```

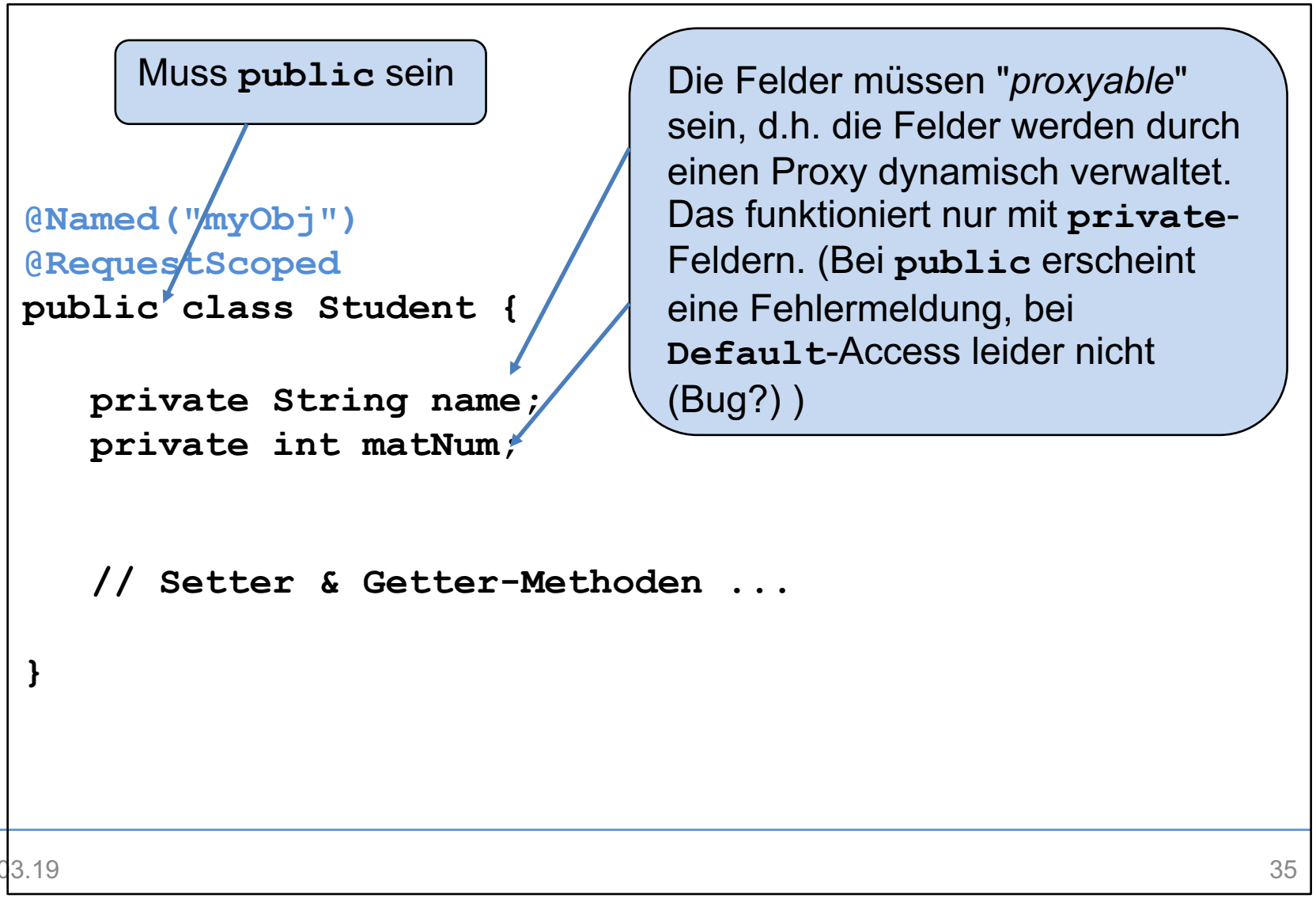
Objekte sollen durch den CDI-Container erzeugt werden

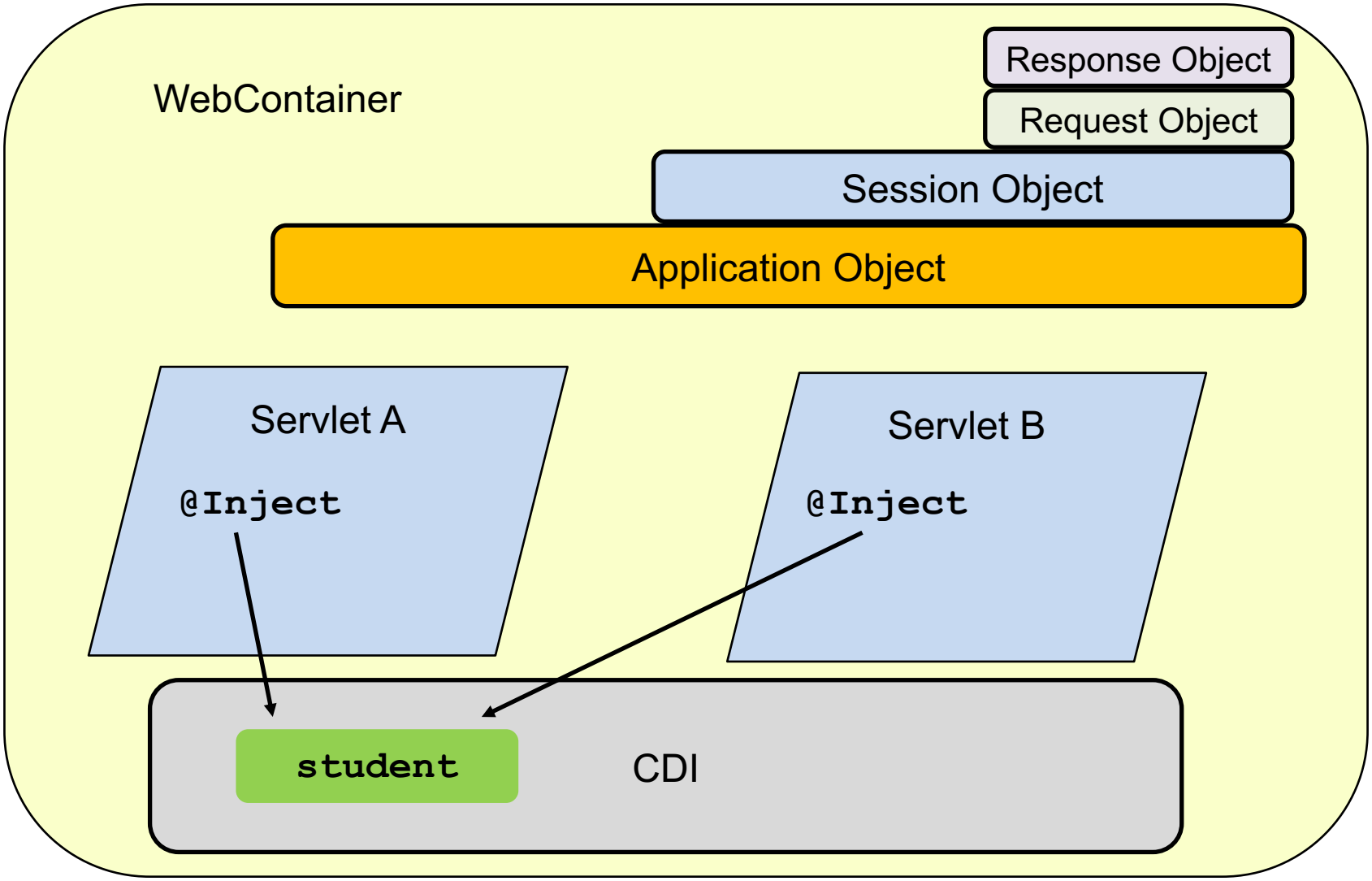
```
@Named                  // Objekt wird durch CDI erzeugt
@ApplicationScoped  // Einmal pro Anwendung
public class Student {

    private String name;
    private int matNum;

    // Setter & Getter-Methoden ...

}
```

Objekte sollen durch den CDI-Container erzeugt werden

```
@Named("myObj")  // Objekt in JSP mit ${myObj.XXX} referenzieren
@RequestScoped
public class Student {

    private String name;
    private int matNum;

    // Setter & Getter-Methoden ...

}
```

Objekte sollen durch den CDI-Container erzeugt werden


Muss public sein
Die Felder müssen "proxyable" sein, d.h. die Felder werden durch einen Proxy dynamisch verwaltet. Das funktioniert nur mit private-Feldern. (Bei public erscheint eine Fehlermeldung, bei Default-Access leider nicht (Bug?))
@Named("myObj")
@RequestScoped
public class Student {
private String name;
private int matNum;
// Setter & Getter-Methoden ...
}


WebContainer
Response Object
Request Object
Session Object
Application Object
Servlet A
@Inject
Servlet B
@Inject
student
CDI

Modul 5
CDI

```
@WebServlet(urlPatterns = {"/NewServlet"})
public class NewServlet extends HttpServlet {

    @Inject
    Student myStud;

    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
                           throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");

            myStud.setName("hhh");

            out.println("</html>");
        }
    }
```

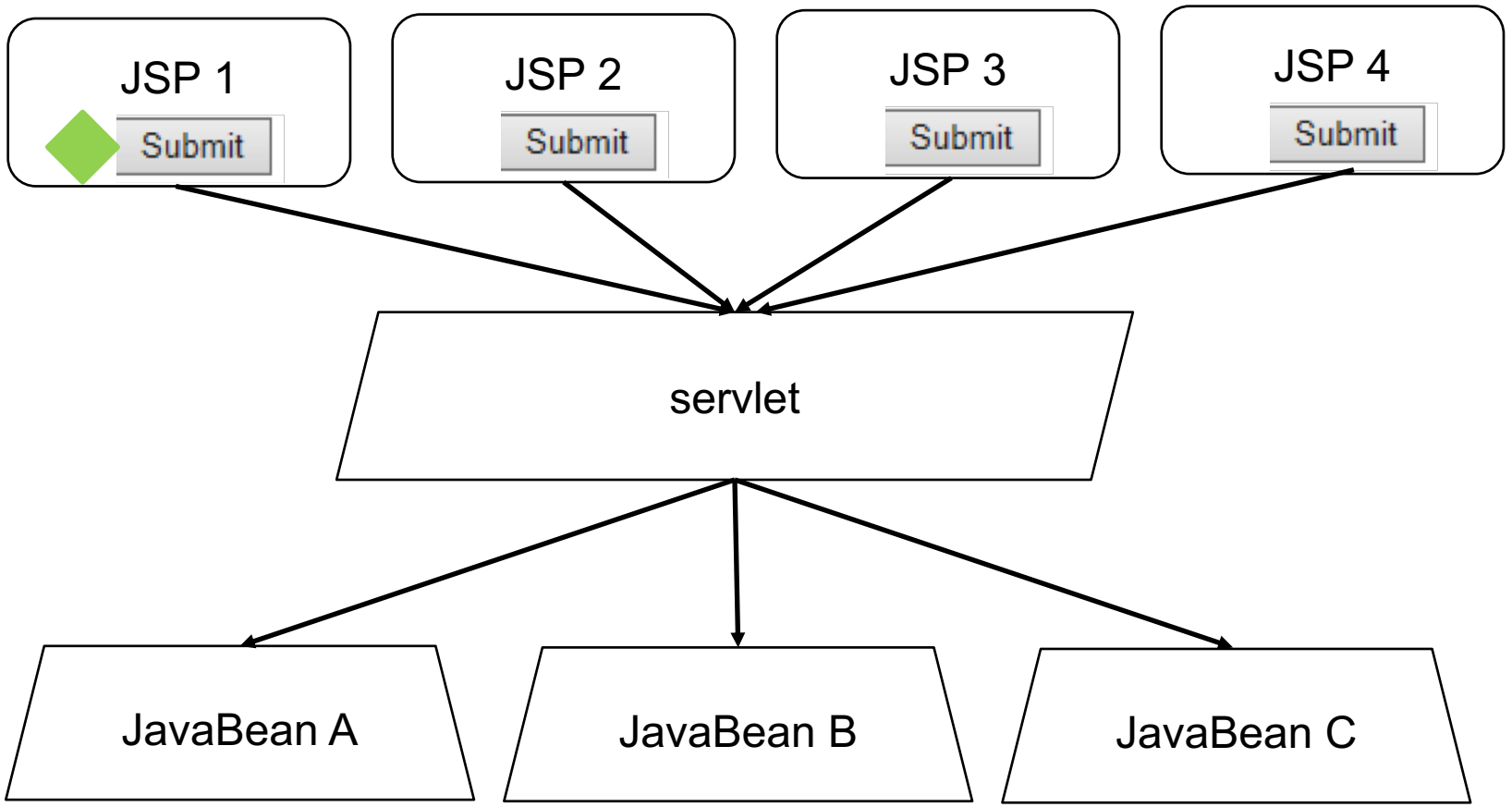
Kein **new()** mehr!

JSP

```
<body>
        <h1>Hello World!</h1>

        ${ student.myName }

        ${ student.myName="Müller" }

        ${ student.matNum }


</body>
```

MVC

Modul 5
MVC


JSP 1
Submit
JSP 2
Submit
JSP 3
Submit
JSP 4
Submit
servlet
JavaBean A
JavaBean B
JavaBean C

Modul 5
MVC – Identify the calling JSP page

```
<form action="/myServlet">

   First name:<br>
   <input type="text" name="firstname" value="Mickey">
   <br>

   Last name:<br>
   <input type="text" name="lastname" value="Mouse">
   <br><br>

   <input type="hidden" name="myPage" value="p1">

   <input type="submit" value="Submit">

</form>
```

Quelle: http://www.jsptutorial.org/

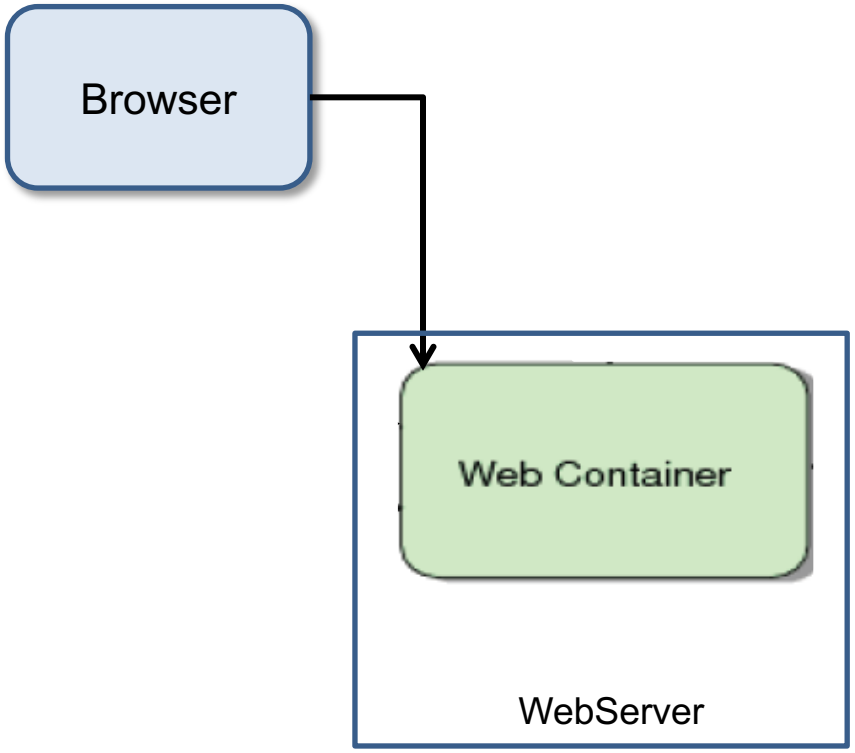
Modul 5
MVC – Identify the calling JSP page

```
String page = request.getParameter("myPage");

 switch (page) {
     case "p1":
        ...
        RequestDispatcher rd = request.getRequestDispatcher("welcome.jsp");
        rd.forward(request, response);
        break;
     case "p2":
          ...
          RequestDispatcher rd = request.getRequestDispatcher("error.jsp");
          rd.forward(request, response);
          break;
     case "p3":
          ...
          RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
          rd.forward(request, response);
          break;
     default:
          System.out.println("Fehlerhafte Eingabe!");
          break;
     }
```

Quelle: http://www.jsptutorial.org/

Multi-Threading

Modul 3 - Java EE Model
Java EE Web-Application (Web-Centric)



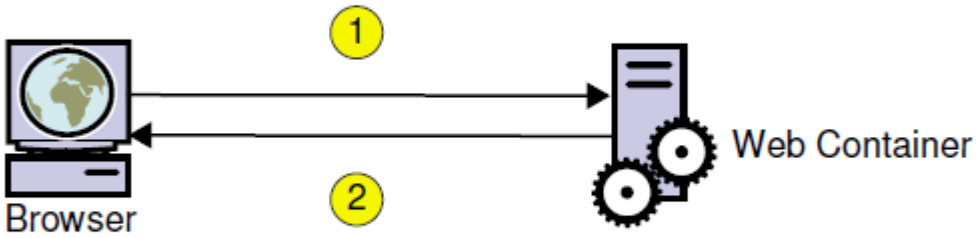
Modul 5
HTTP Request-Response Model

Klasse: `MyHTTPServlet extends …`
Methode:

```
processRequest(){
    …
    out.prinln("<html>");
    …
}
```

Implizite Objekte:

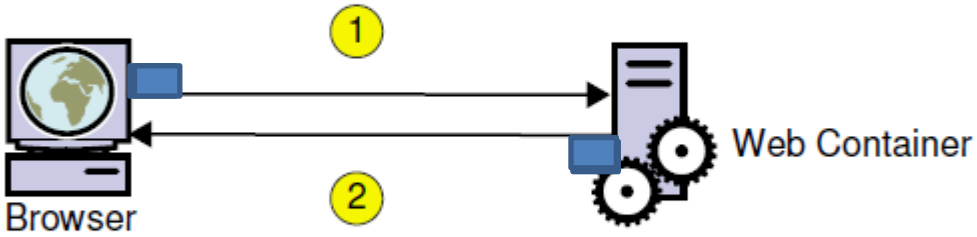
1. request
2. session
3. application
4. response



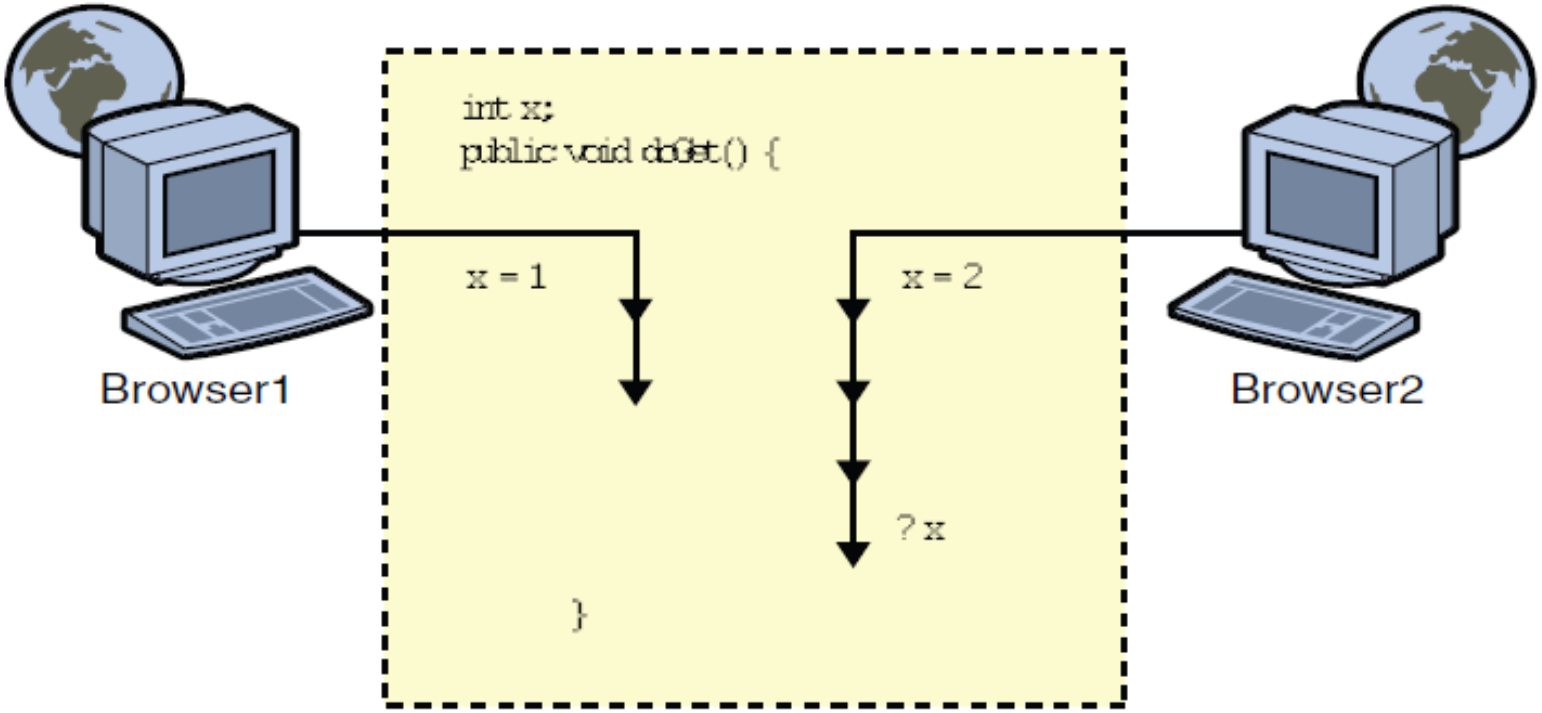
Modul 5
HTTP Request-Response Model

Klasse: `MyHTTPServlet extends …`
Methode:

```
processRequest(){
    …
    out.println("<html>");
    …
}
```



Modul 5
Web Component Thread Model



Because the web container enters the service method on multiple concurrent threads to support multiple simultaneous requests, the developer must ensure that web components are thread-safe:

- Use instance variables cautiously
- Use class variables *very* cautiously
- Provide access to external resources cautiously
- Use synchronization constructs to denote critical sections:

```
synchronized (this) {
// This section is only entered by one thread at a time
}
```

Modul 5
Implications for the Developer

```
@WebServlet(name = "NewServlet", urlPatterns = {"/NewServlet"})
public class NewServlet extends HttpServlet {

    private static int a = 0;
    private int b = 0;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        int c = 0;

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");

            out.println(" a: " + a++);
            out.println(" b: " + b++);
            out.println(" c: " + c++);

            out.println("</html>");
        } finally {
            out.close();
        }
    }
```

Modul 5
Implications for the Developer

Sie haben die drei Vabiablen **`a`**, **`b`**, und **`c`** gesehen. Was ist korrekt:

A.
```
a ist eine Instanzvariable,
b ist eine Instanzvariable,
c ist eine lokale Variable
```

B.
```
a ist eine Klassenvariable,
b ist eine Instanzvariable,
c ist eine lokale Variable
```

C.
```
a ist eine Instanzvariable,
b ist eine Klassenvariable,
c ist eine lokale Variable
```

D.
```
a ist eine Klassenvariable,
b ist eine Klassenvariable,
c ist eine lokale Variable
```

m.socrative.com
Room: 790303

Was wird das servlet nach dem ersten Reload ausgeben?

A. `a: 0, b: 0, c: 0`

B. `a: 1, b: 0, c: 0`

C. `a: 0, b: 1, c: 0`

D. `a: 1, b: 1, c: 0`

E. `a: 1, b: 0, c: 1`

F. `a: 1, b: 1, c: 1`

Modul 5
Implications for the Developer

```
@WebServlet(name = "NewServlet", urlPatterns = {"/NewServlet"})
public class NewServlet extends HttpServlet {

    private int count = 0;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");

            count = count + 1000;
            try {
                Thread.sleep(2000);
            } catch (InterruptedException ex) {
                Logger.getLogger(NewServlet.class.getName()).log(Level.SEVERE, null, ex);
            }
            count = count - 999;

            out.println(" count: " + count);

            out.println("</html>");
        } finally {
            out.close();
        }
    }
```

Modul 5
Implications for the Developer

```
@WebServlet(name = "NewServlet", urlPatterns = {"/NewServlet"})
public class NewServlet extends HttpServlet {

    private int count = 0;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");

            count = count + 1000;
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                Logger.getLogger(NewServlet.class.getName()).log(Level.SEVERE, null, ex);
            }
            count = count - 999;

            out.println(" count: " + count);

            out.println("</html>");
        } finally {
            out.close();
        }
    }
```

kritischer Abschnitt