Java Enterprise Edition

JACQUES ANDRE AUGUSTIN

- Java Certified Programmer
- > 15 years IT experience
- DevOps Advisor

…That's all you need to know ☺

Organization

- All course materials (PDF, exercices, sample code,..) shared on the dedicated space on Campus ("*516S7 - UE71IL - ST2JEE - Java Enterprise Edition*")
- Messages broadcasted through Campus
- Please be on time! 15 mins break
- Lecture - Lecture - Pract - Lecture - Pract - Lecture - Pract - Pract - PRJ - PRJ

- Evaluation
  - Project (*groups of 5 max*) ……. (at least) 50%
  - 1 final exam *(in english)*…… 50%

- Contact : jee@jacquesaugustin.com

Course references & some advice

- ✓ https://docs.oracle.com/javaee/7/index.html
- ✓ All other references (links, resources,..) are given either while seeing the topic or simply live during class
- ✓ Google is your best friend
- ✓ Take notes! I have removed a lot of details from the main course document so please don't rely on it only for help.
- ✓ *Before I forget.. The course pdf will be shared on Campus only after the corresponding lectures!*

Agenda

1. Distributed systems
   - Overview – A bit of history – MVC
2. Java Enterprise Edition Overview:
   - Mission – Components - APIs – Tools
3. Java Server Pages and Servlets
4. Java Server Faces
5. Enterprise Java Beans
6. Persistence
7. Web services

Distributed Systems

What Is A Distributed System?

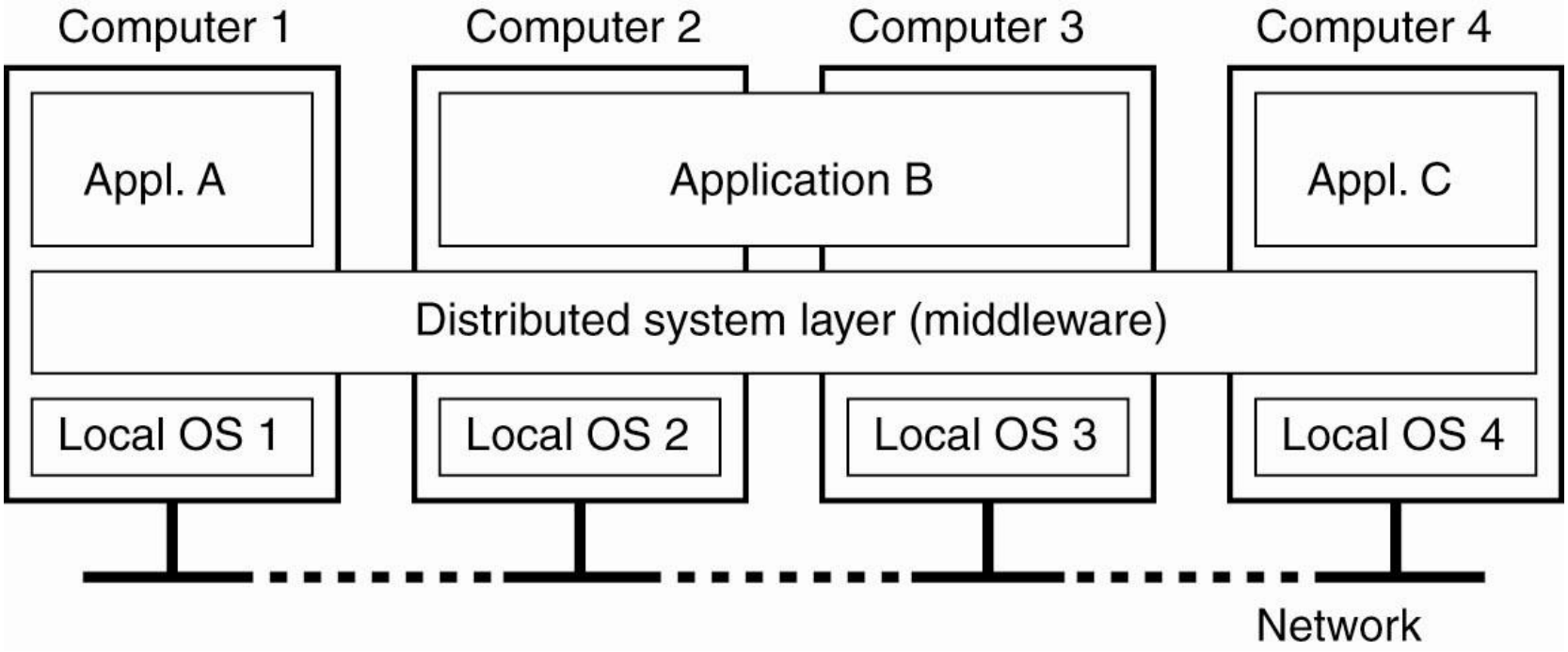
“A collection of independent computers that appears to its users as a single coherent system.”

- **Features:**
  - No shared memory – message-based communication
  - Each runs its own local OS
  - Heterogeneity
- **Ideal: to present a single-system image:**
  - The distributed system “looks like” a single computer rather than a collection of separate computers.

Distributed System Characteristics

- **To present a single-system image:**
  - Hide internal organization, communication details
  - Provide uniform interface
- **Easily expandable**
  - Adding new computers is hidden from users
- **Continuous availability**
  - Failures in one component can be covered by other components
- **Supported by middleware**

Definition of a Distributed System



And now.. The limitations

See previous slide..

Introducing : Layers!

For most applications today, data is being manipulated and the application contains some kind of user interface which generates commands to create this manipulation.

It has been recognized that the data is logically independent from how it is displayed to the user.

This implies a natural “layered” model architecture that separates the actual logic from the “view” (display)

MVC Architecture

Model

This usually represents the data or the actual programming logic – this is what the program actually does.

Contains classes and methods that modify data or states

View

Renders data for the user to see.

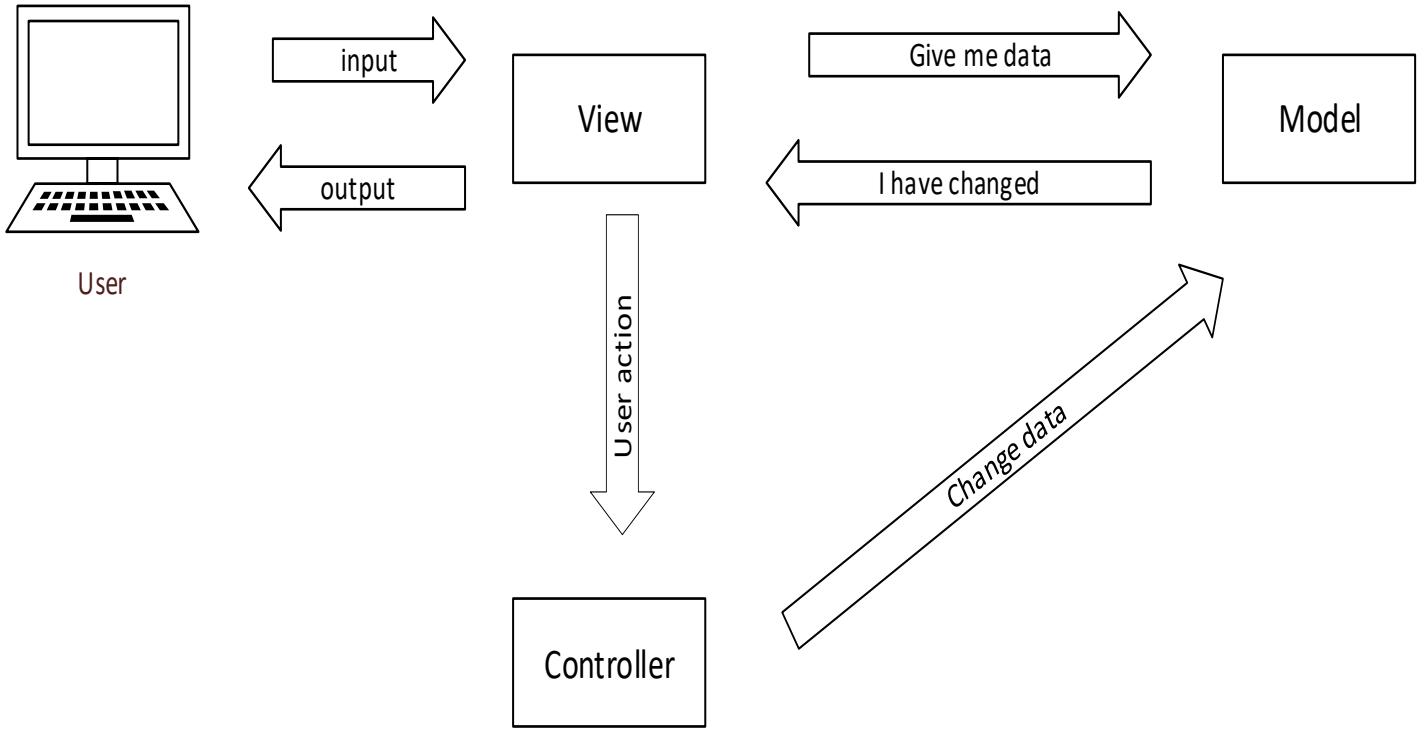
Accepts input from the user to initiate changes in the model

When the model changes, View must be updated.

Controller

Translates user actions (button clicks, etc.) into operations on the Model

MVC Architecture (Extended)



A bit of history

- **RPC**
- **CORBA / DCOM**
- **RMI**
- **And…**

Java EE Overview

- Application programming interface
- Contract
- Specifies how software components should interact with each other

- **Java EE is an application software platform from Oracle based on the Java programming language**.

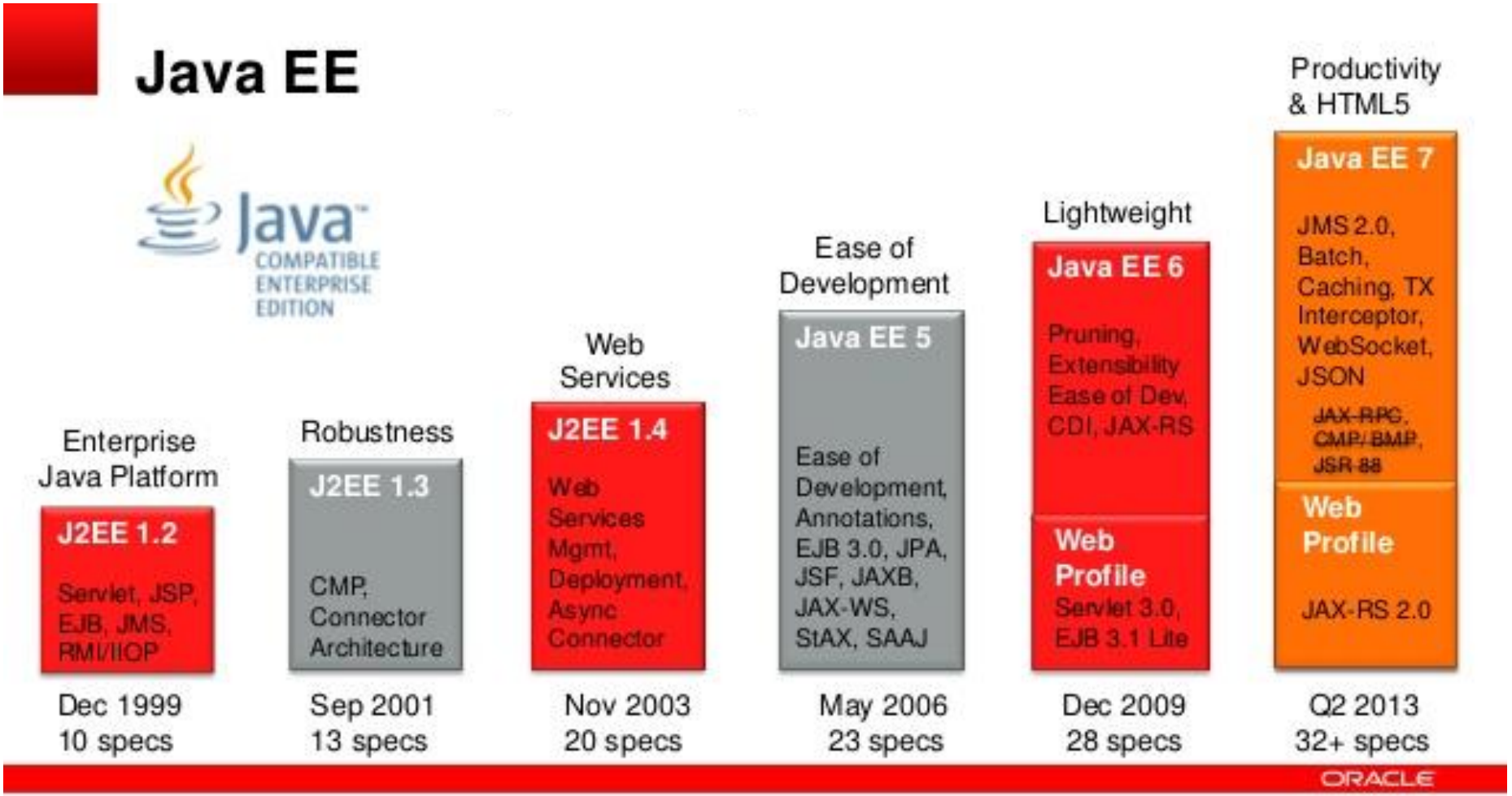
- Originally developed by Sun (which Oracle acquired in 2010) Java EE services are performed in the middle tier between the user's machine and the enterprise's databases and legacy information systems.

- Java EE comprises :
  - a set of specifications (standard),
  - a reference implementation
  - a set of testing suites.

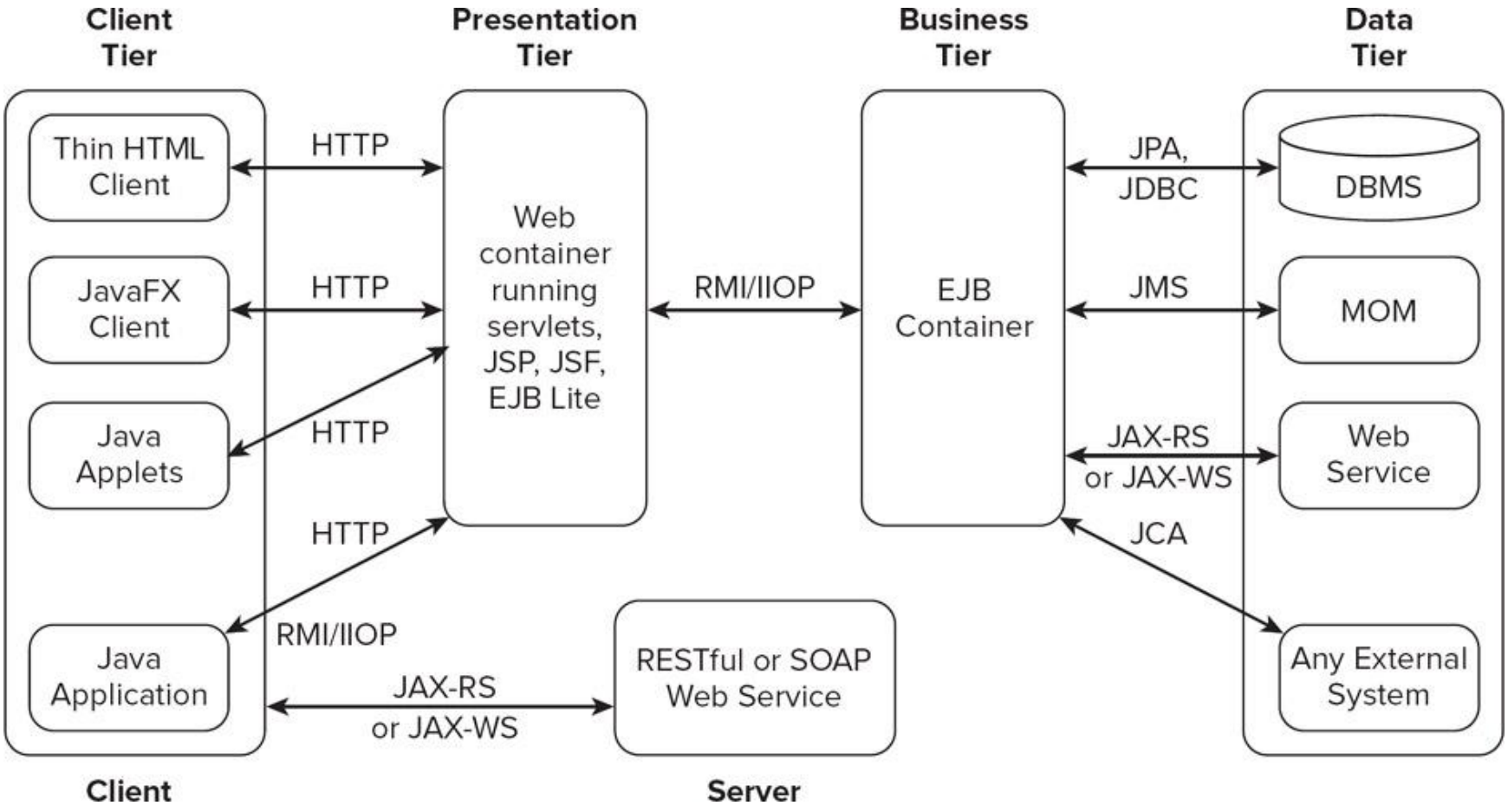
- Java EE 7 28.05.2013

- J2EE => Java EE

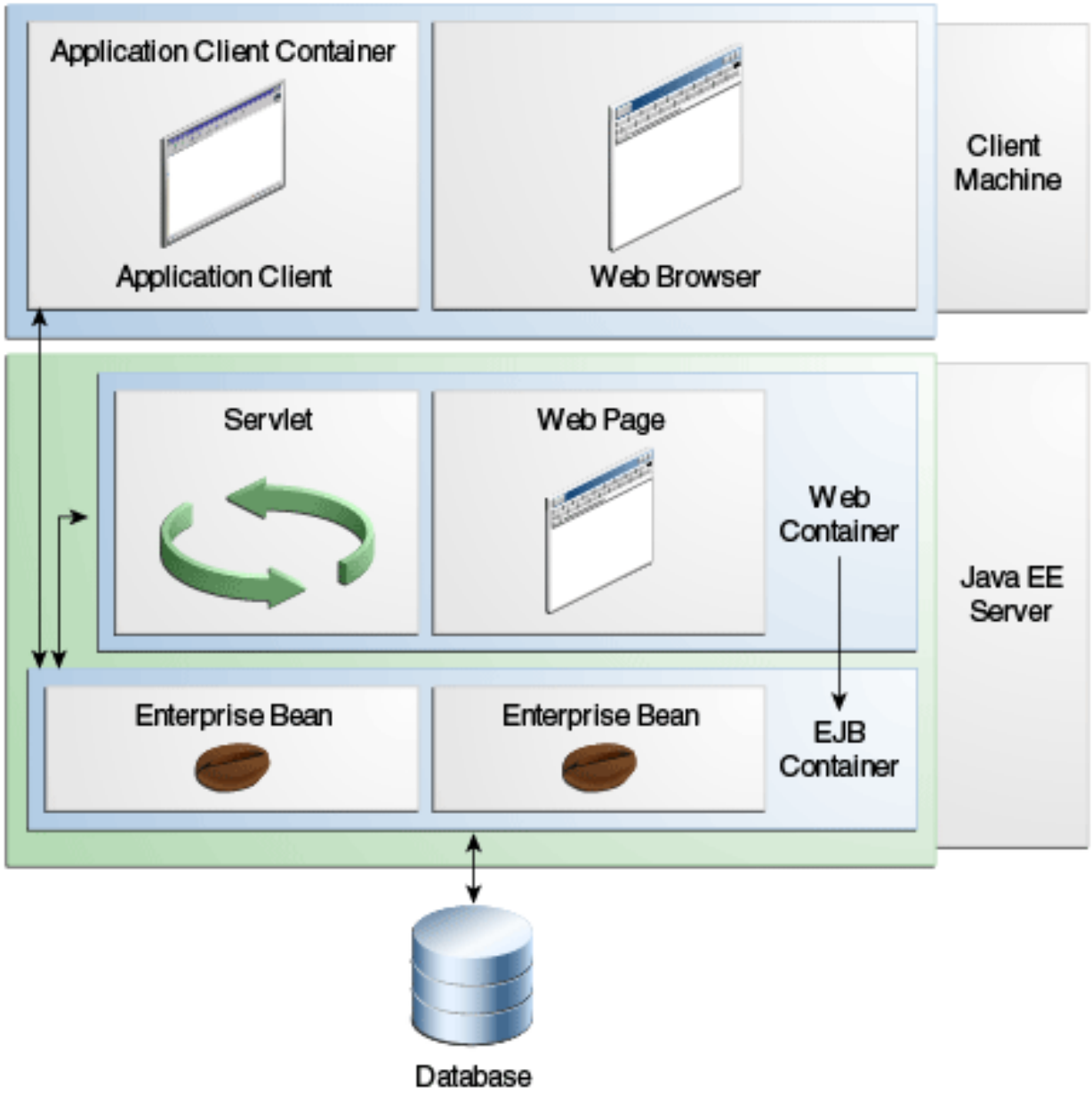
Java EE: Past & Present



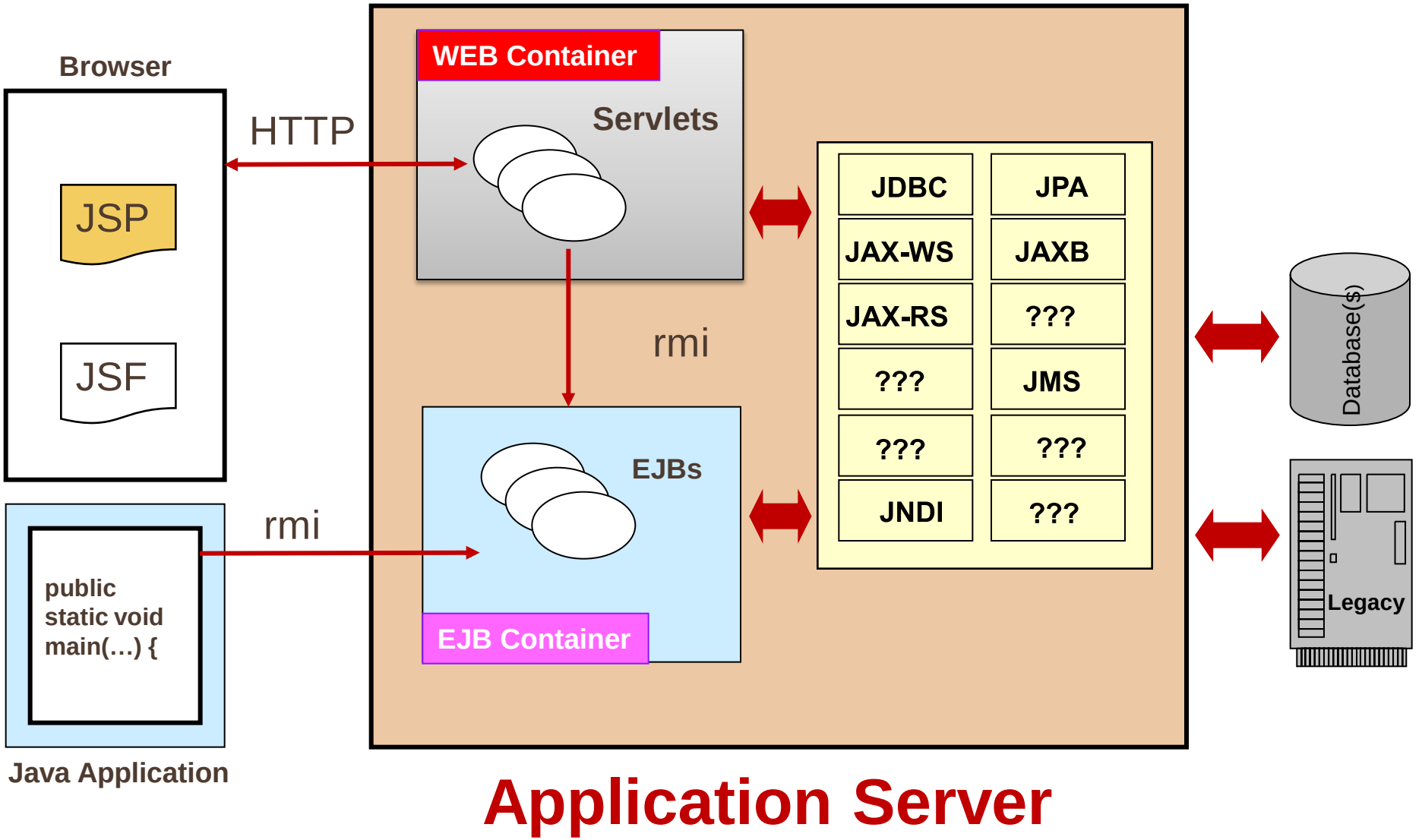
Java EE APIs – The big picture



Java EE – Simple Architectural Overview



Java EE APIs – The big picture

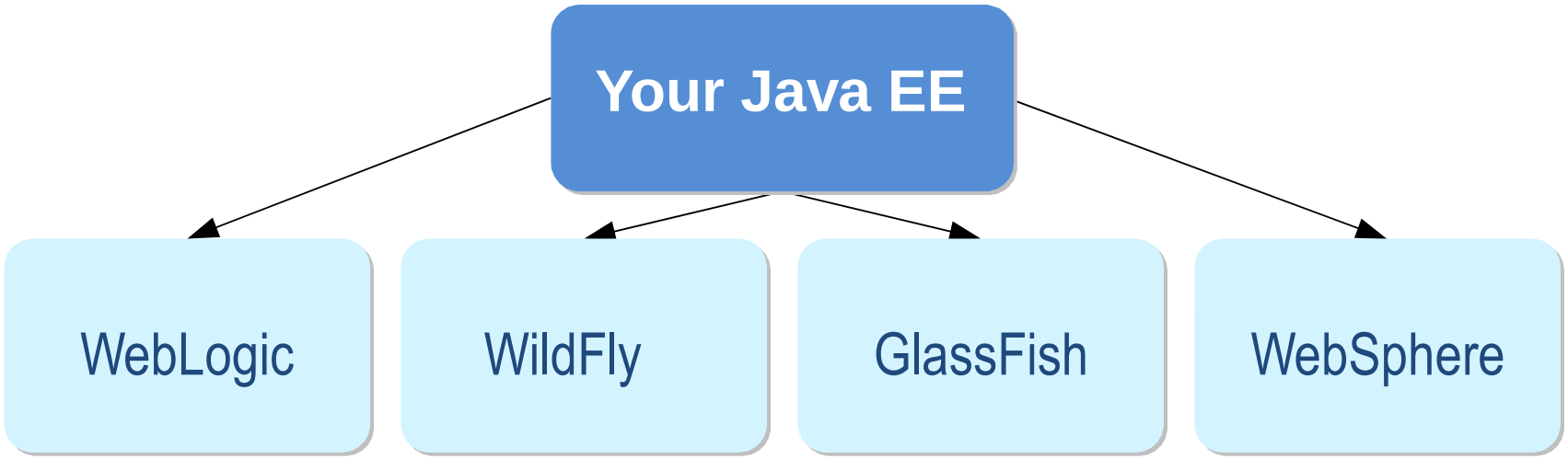


Focus on the Application Server

- The backbone of every Java EE System
  - Oracle
    - Glassfish Server 4
    - Weblogic Server 12.2.1
  - RedHat
    - WildFly
    - JBoss EAP 7
  - IBM
    - WebSphere Application Server Community Edition
    - WebSphere Application Server 8.5

Why a Java EE standard is important

- Standards are about choice
- If not satisfied with vendor, switch to another one
- New vendors can always arise
- Other middleware solutions available, only one standard
- Open source has great value but is not a standard



Frameworks

- Open source frameworks are a critical part of Enterprise Java development

- Java EE 6 Expert Group defined way for framework technologies to be integrated with the platform

  > Examples: Spring, Struts, GUICE

- Characteristics
  - Usually massive and complex enterprise scale information systems
  - Comprise different Java EE components and legacy systems
    - Distributed
    - Long lifetime
    - Modules built with common patterns
      - *Leads easy maintenance*
  - **High demand, high performance**
  - SOA and BPM solutions

Traditional Java EE Users

- Examples of use
  - Insurance companies
  - Online banks
  - Manufacturing industries
  - Public administration

Java EE is not only Java

- Many languages with seamless integration: you can share libraries, code, classes, etc.
  - Groovy
    - Object-oriented
    - Inspired by Java, Python, Ruby, Smalltalk
  - Scala
    - Functional, object-oriented
    - ”Cutting away Java’s syntactic overhead, adding power”
    - Inspired by Java, Scheme, Erlang, Haskell, Lisp
  - JRuby: implementation of Ruby on JVM
  - Jython: implementation of Python on JVM

"Trendy" Java EE users: LinkedIn

- Started with Java platform, using Java EE and extensions
  - Spring Framework
  - Grails
- Now utilizing also Scala and JRuby
  - Scala for back-end processing
  - JRuby for integration interfaces

"Trendy" Java EE users: Twitter

- Started with Ruby on Rails

- Now using Java and Scala in back-end processing
  - Why?
    - Scalability and Performance
    - SOA
    - Encapsulation (re-use, maintenance)

"Trendy" Java EE users: Others

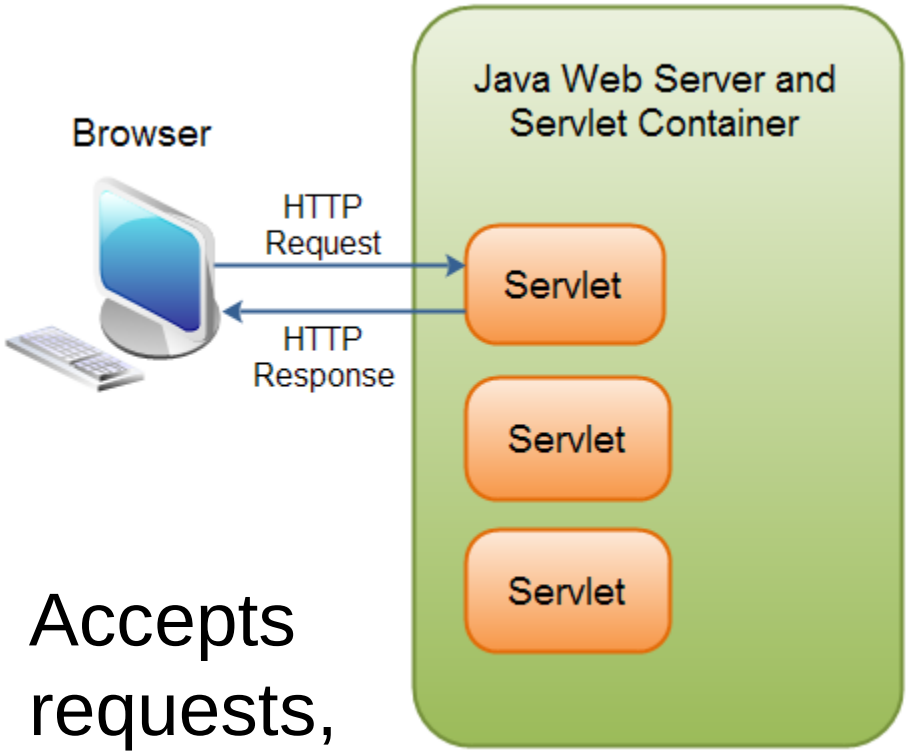
- Google, Amazon and many others use Java EE

- What about Facebook?
  - Writing PHP, which quickly lead to serious performance issues
  - Started compiling PHP to C++
  - Are now investigating using PHP on JVM

- **Application software, that relies on web browser to render it**

- Building blocks in Java EE:
  - Web Container
  - Servlet
  - JSP or JSF

Web Container

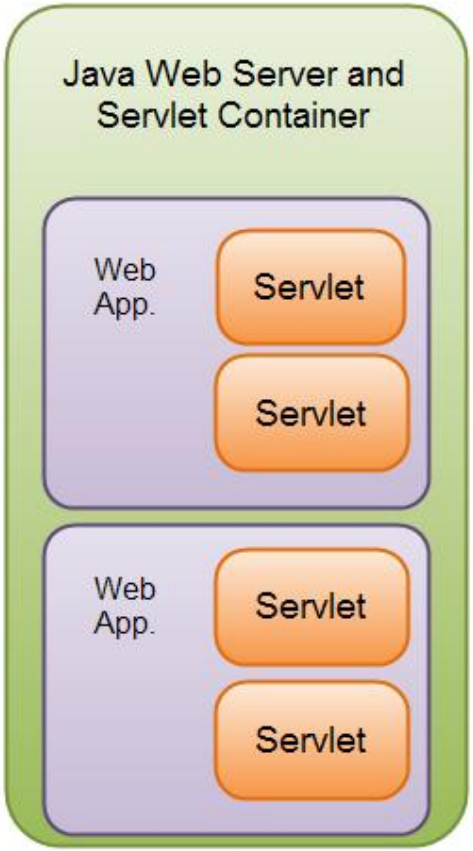


Manages component life cycles

Routes requests to applications

Accepts requests, sends responses

Web Containers



Multiple applications inside one container

Content shared live

Deployment descriptor : WEB.XML (1)

- Instructs the container how handle this application

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi=
    <servlet>
        <servlet-name>HelloWorld</servlet-name>
        <servlet-class>exemple.HelloWorld</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloWorld</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
        </welcome-file-list>
    </web-app>
```

WEB.XML (2)

- In Servlet API version 3.0 most components of web.xml are replaced by annotations that go directly to Java source code.

- **Before Servlet 3.0 web.xml**

```
<servlet>
   <servlet-name>hello</servlet-name>
   <servlet-
   class>example.HelloServlet</servle
   t-class>
</servlet>

<servlet-mapping>
   <servlet-name>hello</servlet-name>
   <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

- **In Servlet 3.0 via annotations**

```
@WebServlet("/hello")
public class HelloServlet
   extends HttpServlet {
...
```

Java Beans

POJO class :

- *private* Attributes
- *public* getters and setters
- Default constructor

Java Bean : example

```
public class User {
    private String login;
    private String pass;

    public String getLogin() {
        return login;    }

    public void setLogin(String login) {
        this.login = login;    }

    public String getPass() {
        return pass;    }

    public void setPass(String pass) {
        this.pass = pass;    }
  }
```

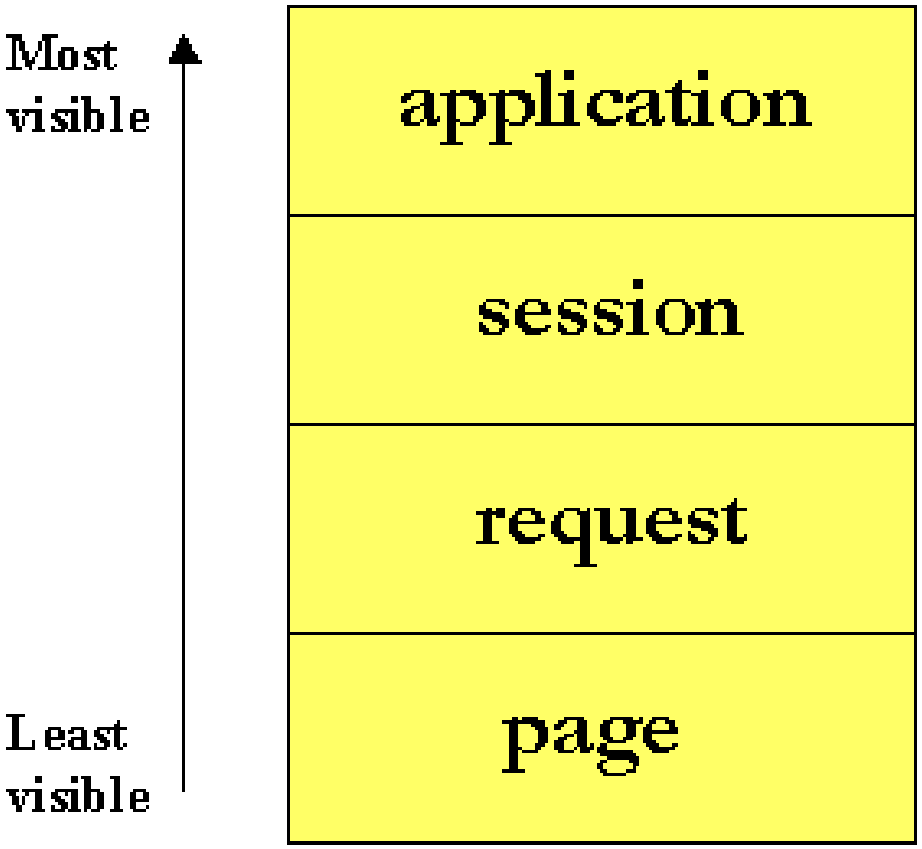
Servlets

- Java class :
  - processes requests (in)
  - returns responses (out)
- Managed by a web container
- 2 main methods :
  - doGet();
  - doPost();

Servlet example

```
public void doGet(HttpServletRequest
  request,
    HttpServletResponse response)
    throws IOException, ServletException
  {
        …
}
public void doPost(HttpServletRequest
  request,
    HttpServletResponse response)
    throws IOException, ServletException
  {
              …
```

Scopes (First look)



Objects accessible from pages that belong to the same application

Objects accessible from pages belonging to the same session as the one in which they were created

Objects accessible from pages processing the request where they were created

Objects accessible only within pages where they were created

Scopes

- ServletContext – web context, one per application/JVM
- Session – one per user sessioon
  - Usually a browser sessioon
- Request – scope of a specific request

HttpSession example

```
HttpSession session = req.getSession();
int visit;
if (session.isNew()) {
  visit = 0;
} else {
  visit = (Integer)
  session.getAttribute("visit");
}
session.setAttribute("visit", ++visit);
```

HttpServletRequest

- Contains request information

- Parameters:

```
String value = request.getParameter("name");
```

- Attributes:

```
request.setAttribute(“key", value);
request.getAttribute(“key”);
```

Servlets – should I write one?

- Writing HTML in Java is hideous

```
PrintWriter writer = resp.getWriter();
writer.println("<html><head><title>Hello</title><
  /head><body>");
writer.println("<p>Hello World!</p>");
writer.println("<p>Current time: " + new Date() +
  "</p>");
writer.println("</body></html>");
```

Java Server Pages

JSP to the rescue!

- JSP (Java Server Pages)
- Write HTML
  - Standard markup language
- Add dynamic scripting elements
- Add Java code

JSP example

- <Your Project>/WEB-INF/jsp/hello.jsp

```
<%@page import="java.util.Date"%>

<html>
  <head><title>Hello</title></head>
  <body>
      <p>Hello World!</p>
      <p>Current time: <%= new Date() %></p>
  </body>
</html>
```

Dynamic content

- Expression

```
<p>Current time: <%= new Date() %></p>
```

- Scriptlet

```
<p>Current time: <% out.println(new Date()); %></p>
```

- Declaration

```
<%!
  private Date currentDate(){
     return new Date();
  }
%>

<p>Current time: <%= currentDate() %></p>
```

Predefined variables

- request – HttpServletRequest
- response – HttpServletResponse
- out – Writer
- session – HttpSession
- application – ServletContext
- pageContext – PageContext

JSP actions

- jsp:include
  Includes a file at the time the page is requested
- jsp:forward
  Forwards the requester to a new page
- jsp:getProperty
  Inserts the property of a JavaBean into the output
  jsp:setProperty
  Sets the property of a JavaBean
- jsp:useBean
  Finds or instantiates a JavaBean

Expression Language (EL)

- Easy way to access JavaBeans in different scopes
- Total Sum: ${row.price * row.amount}

Basic Operators in EL

Operator	Description
.	Access a bean property or Map entry
[]	Access an array or List element
( )	Group a subexpression to change the evaluation order
+	Addition
-	Subtraction or negation of a value
*	Multiplication
/ or div	Division
% or mod	Modulo (remainder)
== or eq	Test for equality
!= or ne	Test for inequality
< or lt	Test for less than
> or gt	Test for greater than
<= or le	Test for less than or equal
>= or gt	Test for greater than or equal
&& or and	Test for logical AND
|| or or	Test for logical OR
! or not	Unary Boolean complement
empty	Test for empty variable values

Scopes (revisited)

```
<%
application.setAttribute("subject", "Web information systems");
session.setAttribute("topic", "Servlets");
request.setAttribute("lector", "Anti");
%>

Subject: ${subject}
Topic: ${topic}
Lector: ${lector}
```

Scopes

```
<%
application.setAttribute("subject", "Web information
   systems");
session.setAttribute("topic", "Servlets");
request.setAttribute("lector", "Anti");
pageContext.setAttribute("subject", "The new topic");
application.setAttribute("subject", "The newest topic");
%>

Subject: ${subject}
Topic: ${topic}
Lector: ${lector}
```

Scopes

```
<%
application.setAttribute("subject", "Web information
   systems");
session.setAttribute("topic", "Servlets");
request.setAttribute("lector", "Anti");
pageContext.setAttribute("subject", "The new topic");
application.setAttribute("subject", "The newest topic");
%>

Subject: ${subject}
Topic: ${topic}
Lector: ${lector}
```

JavaBeans & EL (1/2)

```
public class Person implements Serializable {

   private String name;

   public Person() {}

   public String getName() {
       return name;
   }

   public void setName(String name) {
       this.name = name;
   }
}
```

JavaBeans & EL (2/2)

```
Person person = new Person();
person.setName("Alan Turing");
request.setAttribute("person",
  person);

<p>Person: ${person.name}</p>
```

- Set of standard tools for JSP

```
<%
  List<String> lectors = Arrays.asList("Jack", "Jill", "Anti");
pageContext.setAttribute("lectors", lectors);
%>

<c:set var="guestLector" value="Anti" />

<c:forEach var="lector" items="${lectors}">
   Name: ${lector}
   <c:if test="${lector eq guestLector}">(guest)</c:if>
   <br />
</c:forEach>
```

So, life is perfect with JSPs ?
Not really..

- Writing Java in JSP is hideous

```
<p>Current time: <%= currentDate() %></p>
```

MVC to the rescue : Servlet controller / JSP view

```
protected void doGet(HttpServletRequest req, HttpServletResponse
   resp)       throws ServletException, IOException {

req.setAttribute("currentDate", new Date());

req.getRequestDispatcher("/WEB-INF/jsp/hello.jsp").forward(req,
   resp);
}
```

Servlet controller, JSP view

- WEB-INF/jsp/hello.jsp

```
<html>
...
  <body>
     <p>Current time: ${currentDate}</p>
  </body>
</html>
```