

# Introduction

This course “Software Architecture with Java” will discuss the following topics:

Java servlets

Java Server Pages (JSP's)

Java Beans

JDBC, connections to RDBMS and SQL

XML and XML translations (XSLT)

Enterprise Java Beans (EJB's)

Struts and Java Server Faces (JSF)

Web Services

These tools will then be used to construct webapplications according to different models.

Literature:

Steelman & Murach,  
Murach's Java Servlets and JSP, 2nd ed.  
Mike Murach & Associates Inc, 2008

or

Mukhar, Zelenak,  
Beginning Java EE 5  
APress, 2005

Software:

Java SE, Java Standard Edition, version 1.5.x, 1.6.x  
Tomcat, [tomcat.apache.org](http://tomcat.apache.org), version 5.5.x, 6.x  
JSTL, [jakarta.apache.org/taglibs](http://jakarta.apache.org/taglibs), version 1.1  
MySQL DB-server, [www.mysql.com](http://www.mysql.com), version 4.1 or 5  
MySQL J/Connector, JDBC driver 5.1

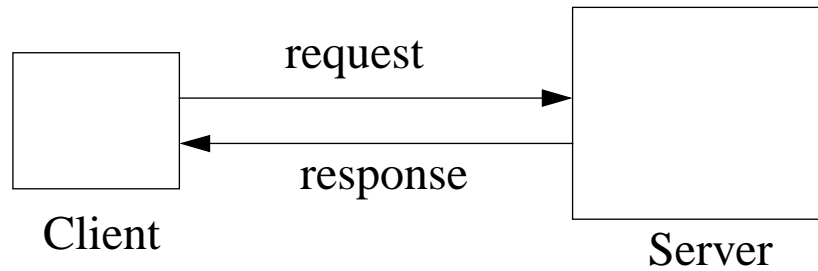
All of this is freeware and can be downloaded.

What is a web application?

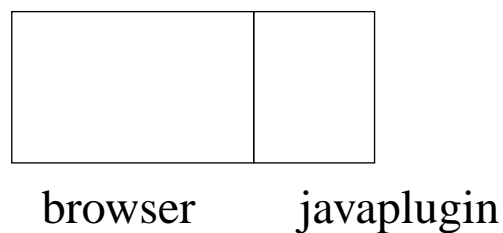
It is a set of web pages that are generated in response to a user request.

Examples:

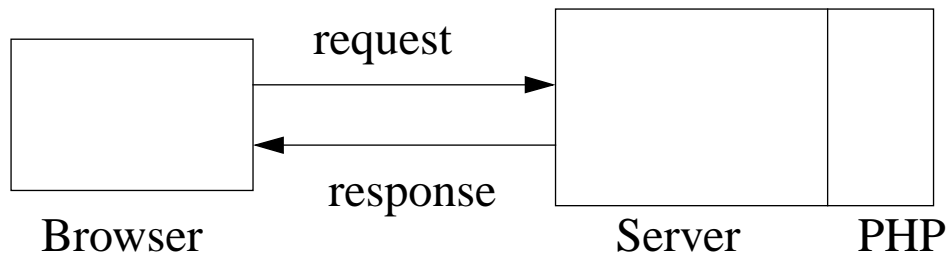
A browser that requests HTML-pages from a server. In the simplest case, these pages are static. The communication uses the HTTP protocol.



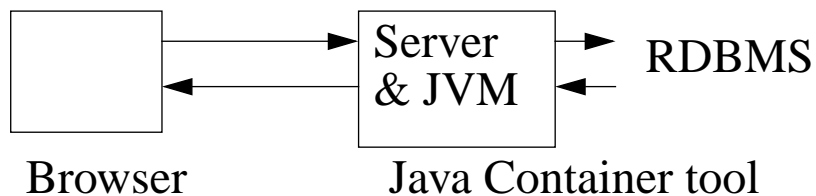
A browser that executes one or more Java applets. This is all done inside the browser (the client), therefore this is known as a “fat client”.



A browser that requests dynamic pages from a server. These can be generated by PHP or some similar system. This kind of system is called a “thin client” since the processing is done in the web server.



A browser that requests dynamic pages from a Java enabled server. The server somehow forwards these requests to a tool that generates and returns html-pages. There might be a database system involved in generating the answer.



We are to discuss the last kind of systems.

Why do we need a Container tool?

An application may result in several request/response pairs. Since HTTP is stateless (has no memory), we need something that can create resources, remember and identify connections and keep resources alive.

An example of a webcontainer is Jakarta Tomcat. It is not a complete Java EE container, it can therefore be called a servlet container.

There is a standard framework definition for Javabased webcontainers. This is known as Java EE, The Java Enterprise Edition. Servlets and JSP's are part of the Java EE specification. We will use version 2.5 of the Servlet spec. and 2.1 of the JSP specification.

Tomcat implements part of the Java EE specification but not all of it.

Note: Java EE was former known under the name  
J2EE

What is a servlet?

A servlet is a Java class that is derived from the `HTTPServlet` class.

It is executed on behalf of the webserver when requested from a user. The servlet receives the request, and produces HTML code that is returned via the response mechanism. Since the servlet is a program it can generate a dynamic response based on different conditions.

Since it is pure Java it is transportable and follows an official standard. It is multithreaded and only one instantiation is needed.

A simple servlet can look like

```
package serv;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class Serv extends HttpServlet {

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<H1> hello </H1>");
    }

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

This will receive a request and generate an HTML header-line as response to the browser.

To compile this you need some jar-files that are not part of the Java SE, but they are included in tomcat and in Java EE implementations.

To run it you need Tomcat or some other container. Tomcat can execute the servlet directly on request from the browser, but the normal use of a servlet is as part of a deployed application.

A deployed application is a complete system with different components that are kept together with a description file.

What is a JSP?

Java Server Pages consists of HTML-code, Java code embedded in a script language (NOT Javascript!) and other JSP tags.

The purpose of a JSP is to generate HTML code.

Technically a JSP is compiled into a servlet by the web-container.

Since it is Java it is transportable, but you do not need to compile it, therefore it is easier to work with.

JSP is extendable, i. e. you can introduce new tags that extends the functionality of a JSP.

An example of a JSP is:

```
<HTML>
<!-- Copyright (c) 1999 The Apache Software
      Foundation. All rights reserved.-->

<HEAD>
  <TITLE>
    Calendar: A JSP APPLICATION
  </TITLE>
</HEAD>

<BODY BGCOLOR="white">

<jsp:useBean id="table" scope="session" class="cal.TableBean" />

<% String time = request.getParameter ("time"); %>

<FONT SIZE=5> Please add the following event:<br>
  <h3> Date <%= table.getDate() %><br> Time <%= time %> </h3>
</FONT>

<FORM METHOD=POST ACTION=cal1.jsp><br>
<br> <INPUT NAME="date" TYPE=HIDDEN VALUE="current">
<br> <INPUT NAME="time" TYPE=HIDDEN
      VALUE=<%= time %>
<br>
<h2>
  Description of the event <INPUT
    NAME="description" TYPE=TEXT SIZE=20>
</h2>
<BR> <INPUT TYPE=SUBMIT VALUE="submit">
</FORM>
</BODY>
</HTML>
```

What is a Bean?

A Bean is an ordinary Java class that adheres to the Bean rules.

Beans can be used for anything, but in a webapplication they are often used to store and manipulate data.

An example:

```
package test;

public class TestBean {
    String name;
    Integer age;

    public TestBean() { }

    public String getName() {
        return name;
    }

    public void setName(String newName) {
        name = newName;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer newAge) {
        age = newAge;
    }}
```

What is JDBC, RDBMS and SQL?

JDBC (Java Data Base Connectivity) is a piece of Java software that allows Java classes to connect to a database and to perform queries and updates using SQL-statements. Each RDBMS (Relational Data Base Managing System) requires a vendor specific JDBC driver.

SQL (Standard Query Language) is a language that allows you to maintain your database. You can store and retrieve data.

What is XML?

XML (eXtensible Markup Language) is a markup language that is mainly used to store structured data. It is somewhat similar to HTML but there is no predefined tags. You can use any tags, but you must yourself assign meaning to the tags.

The benefit of XML is that it can be translated into other formats, i. e. HTML, PDF etc. Therefore it can be used as a general, transportable format.

XML can be translated into HTML by XSLT, eXtensible Style Language Translations.

An example:

```
<?xml version="1.0" encoding="ISO8859-1" ?>  
<?xml-stylesheet type="text/xsl"  
                href="employees.xsl"?>
```

```
<employees>  
  <employee empid="1">  
    <name>Fredrik Ålund</name>  
    <department depid="3">Services</department>  
  </employee>  
  
  <employee empid="2">  
    <name>Helena Larsson</name>  
    <department depid="3"> Services </department>  
  </employee>  
</employees>
```

Case sensitive, more strict than HTML. You must use closing tags for ALL elements.

## What is an Enterprise Java Bean?

An EJB is a server-side component architecture that is used to build enterprise-class distributed component applications in Java. It allows you to write scalable, reliable and secure applications without writing your own frameworks.

## What is Struts and JSF

Struts and JSF are Java- and MVC-based frameworks used to develop webapplications. They provide basic structures that can be configured to suit your needs.

What is Web services?

A Web service is a software application, accessible on the Web through a URL, that is accessed by clients using XML-based protocols, such as Simple Object Access Protocol (SOAP) sent over accepted Internet protocols, such as HTTP. Clients access a Web service application through its interfaces and bindings, which are defined using XML artifacts, such as a Web Services Definition Language (WSDL) file.

How do you combine these into an application?

One way to organize this is the MVC model. This means Model, View and Control.

Model is the datahandling and business logic

View is the presentation, i. e. the HTML pages

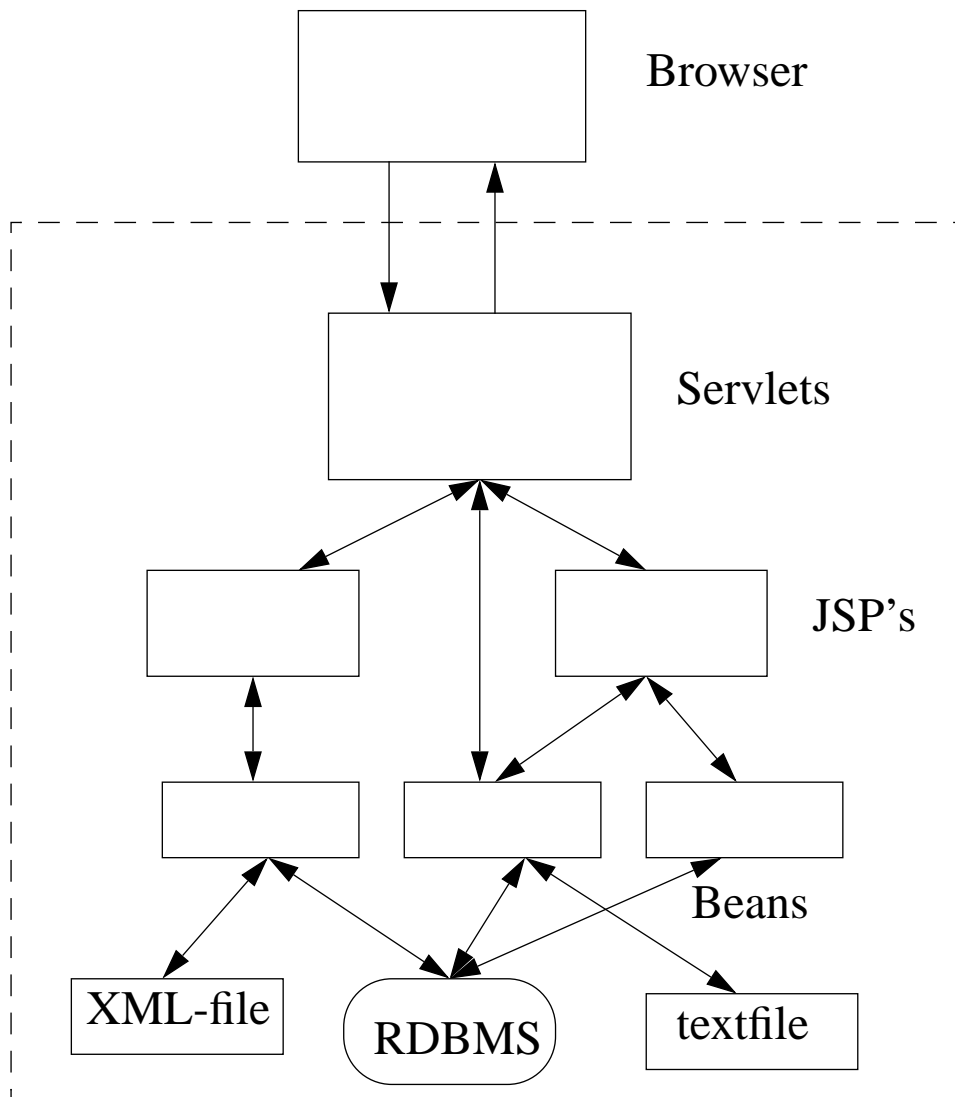
Control is the scheduler, the one that distributes the work.

XML-files are used to describe the application.

Beans or EJB's are good to obtain and maintain data. They can connect to data bases to get/store data. Since they are pure Java you have the full power of the language with all datatypes.

JSP's are best used to produce HTML. In addition you can embed JAVA in a JSP. However this makes them more difficult to read and to maintain because you mix a number of languages. Therefore they are most often used to produce presentation, i. e. HTML, of data produced by some beans.

Servlets are used as control. They receive the requests from the client, distributes the work to JSP's and beans and return the response. Usually you avoid generating HTML in the servlet because too much of that makes the servlet cluttered with strange HTML-strings hard to maintain.



We have no special development tools available here but there are such tools:

Full Java EE tools:

*Oracle JDeveloper*. Complete environment, arbitrary database. Free for personal use. High quality. See [www.oracle.com](http://www.oracle.com)

*Borland Jbuilder Enterprise*. A 30-day trial version can be downloaded from [www.codegear.com](http://www.codegear.com)

*Eclipse*, Open Source but need plugins. [www.eclipse.org](http://www.eclipse.org)

*NetBeans*, [www.netbeans.org](http://www.netbeans.org) or  
[java.sun.com/javaee/downloads/index.jsp](http://java.sun.com/javaee/downloads/index.jsp)

Alternatives to tomcat:

*JBoss*, [www.jboss.org](http://www.jboss.org)

*Sun J2EE Reference Implementation*,  
[java.sun.com/javaee/index.jsp](http://java.sun.com/javaee/index.jsp)

## Some relevant books:

*Wutka.*

*Java Server Pages and Servlets.*

*Que, 2000.*

*Bond, Haywood, Law, Longshaw, Roxburgh.*

*Sams Teach Yourself J2EE with EJB, JSP, Servlets, JNDI, JDBC and XML  
in 21 days.*

*Sams, 2002.*

*Bergsten.*

*Java Server Pages, 3rd ed.*

*O'Reilly, 2003.*

*Hunter, Crawford.*

*Java Servlet Programming.*

*O'Reilly, 2001.*

*Zeid.*

*Mastering the Internet and HTML.*

*Prentice Hall, 2000.*

*Bradley.*

*The XML companion.*

*Addison-Wesley, 2002.*

*Tidwell.*

*XSLT, mastering XML transformations.*

*O'Reilly, 2001.*

*Roman, Sriganesh, Brose.*

*Masterings Enterprise Javabeans , 3'rd ed.*

*Wiley, 2005*

*Brose, Sriganesh, Silverman,*

*Masterings Enterprise Javabeans 30. 4'th ed.*

*Wiley, 2006*

*Monson-Haefel.*

*Enterprise Javabeans 4'th ed.*

*O'Reilly, 2004.*

*Alur, Crupi, Malks.*  
*Core J2EE Patterns, Best Practices and Design Strategies.*  
*Prentice Hall, 2003.*

*Husted, Dumolin, Franciscus, Winterfeldt.*  
*Struts in Action.*  
*Manning, 2003.*

*Ship.*  
*Tapestry in Action.*  
*Manning, 2004.*

*Bergsten.*  
*Java Server Faces.*  
*O'Reilly, 2004*

*Singh, Brydon, Murray, Ramachandran, Violleau, Stearns.*  
*Designing Web Services with the J2EE 1.4 Platform,*  
*JAX-RPC, SOAP, and XML Technologies.*  
*Addison-Wesley, 2004.*

*Chopra, Bakore, Eaves, Galbraith, Li, Wiggers.*  
*Professional Apache Tomcat 5.*  
*Wrox, 2004.*

*Bauer, King,*  
*Java Persistence with Hibernate,*  
*Manning, 2006*