Java Web Tiers: Servlet & JSP

IF3038 Pemrograman Internet

Semester 2/2009-2010

Arsitektur Java Enterprise

Aplikasi Enterprise umumnya memiliki arsitektur layer, dimana setiap layer memiliki tanggungjawab berbeda

Arsitektur Java Enterprise

Presentation layer: penanganan tampilan dan interaksi dengan user

business layer: eksekusi business logic

database layer: storage

Arsitektur Java Enterprise

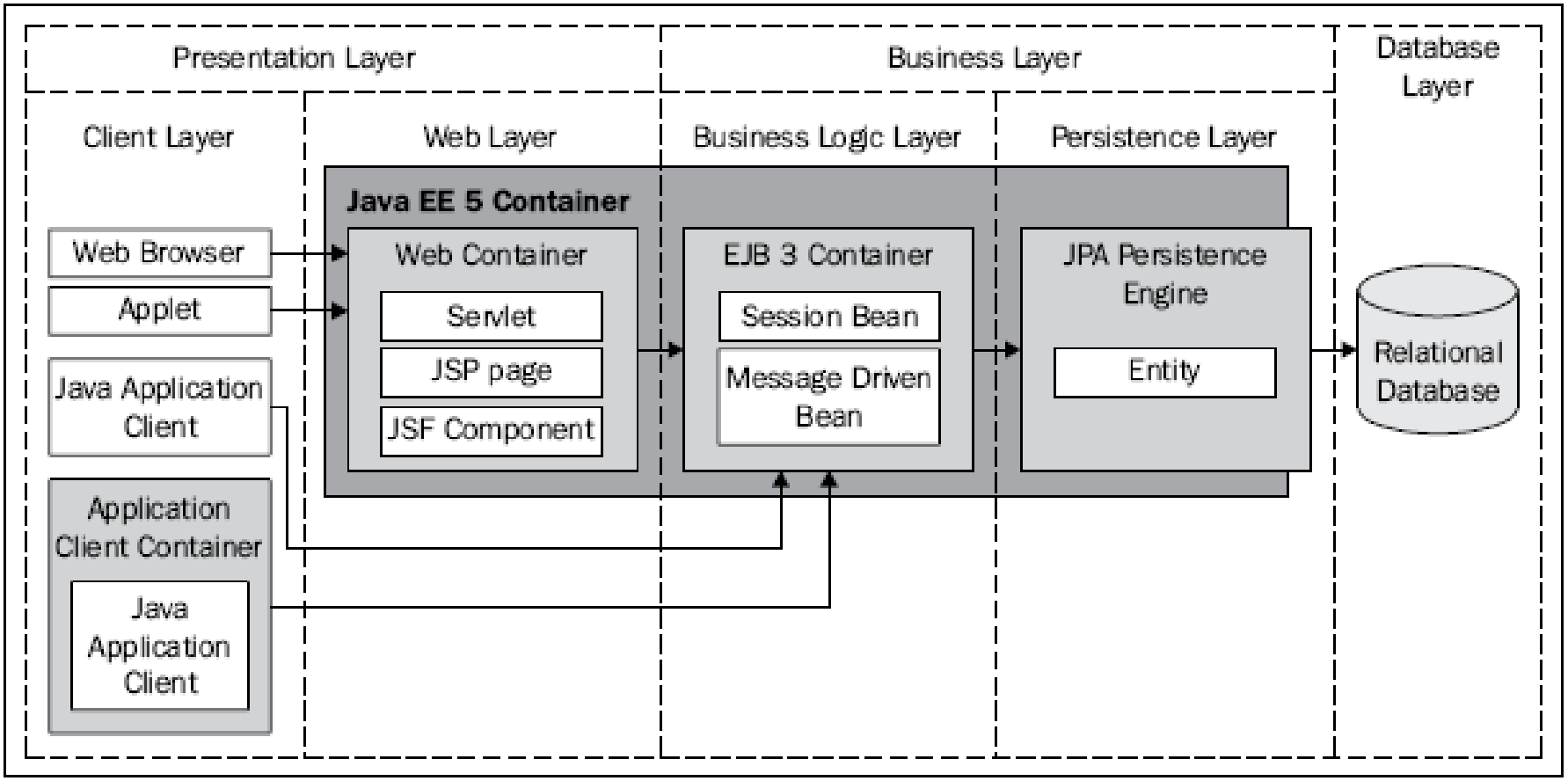
Mengapa layering:

- menyederhanakan problem, agar tidak terlalu kompleks.
  - Masing-masing layer memiliki tanggung jawab berbeda
  - masing-masing layer dapat didevelop oleh developer yang berbeda
- fleksibel
  - presentation dan business layer dapat ditangani dengan menggunakan produk software berbeda dan berada pada komputer yang berbeda

Arsitektur Java Enterprise

- Presentation layer: HTML, Javascript, JSP, JSF, Servlets
- Business layer: EJB (Enterprise JavaBeans)
  - session beans, message-driven beans
  - persistence sublayer: entity, persistence provider
- Database layer: relational database
  - e.g.: mySQL, Oracle

Arsitektur Java Enterprise



Java Enterprise App

Sebuah aplikasi Java Enterprise dikemas dalam format ear.

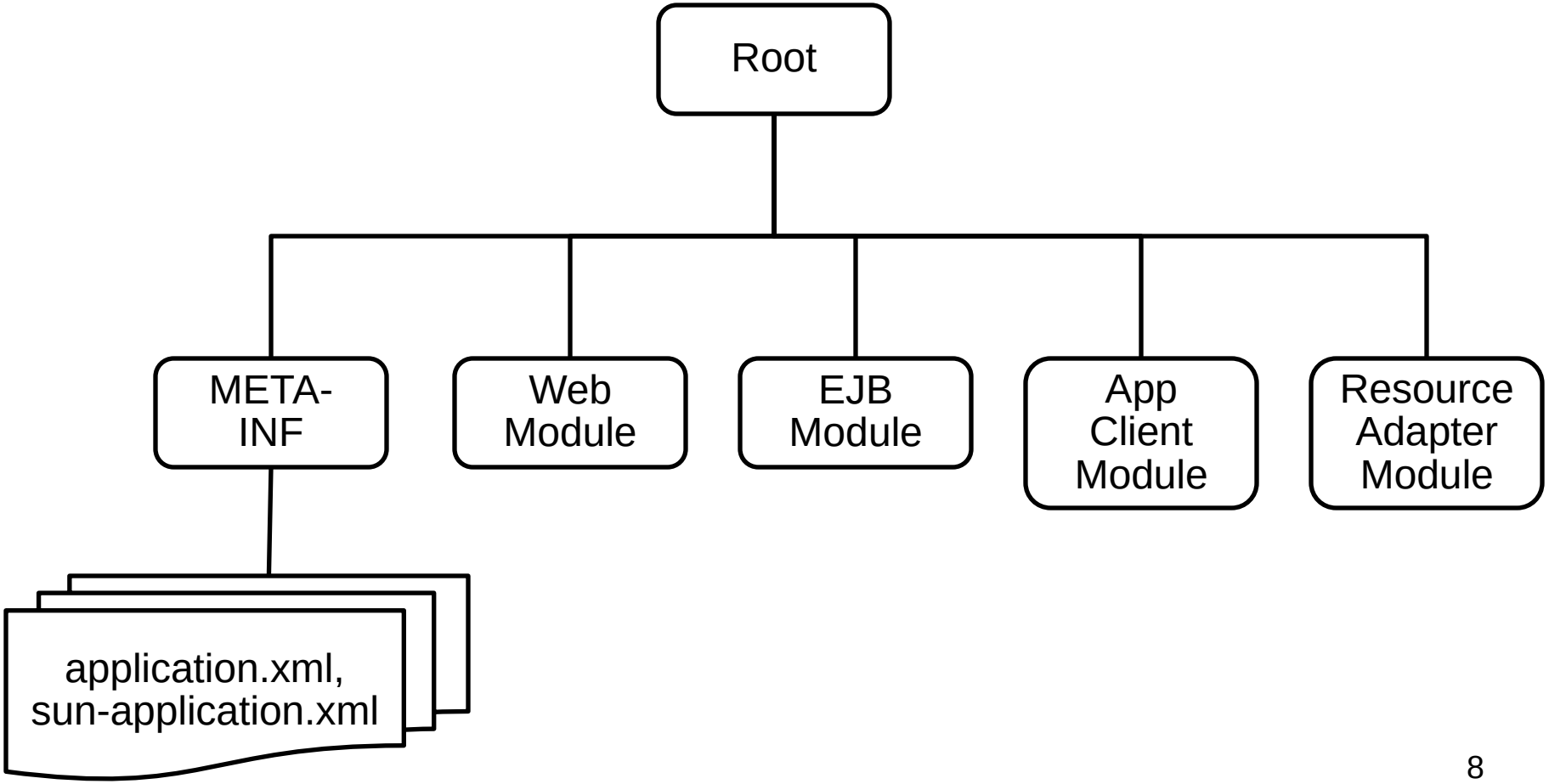
ear terdiri atas 1 atau lebih modul, dan deskriptor aplikasi

modul dapat berupa:

- Web module, EJB module, App. Client module, Resource Adapter module
- setiap modul memiliki deskriptor masing-masing

Note: meski servlet dan JSP dapat langsung diletakkan pada lokasi default untuk halaman web dan servlet, sebaiknya setiap aplikasi dikemas menjadi sebuah modul (web/ear)

Struktur ear



deskriptor

deskriptor: informasi tentang modul/aplikasi yang terdapat pada sebuah modul/aplikasi

deskriptor ada 2 macam:

- standar Java EE: application.xml, ejb-jar.xml, web.xml
- spesifik implementasi: sun-application.xml, sun-ejb-jar.xml

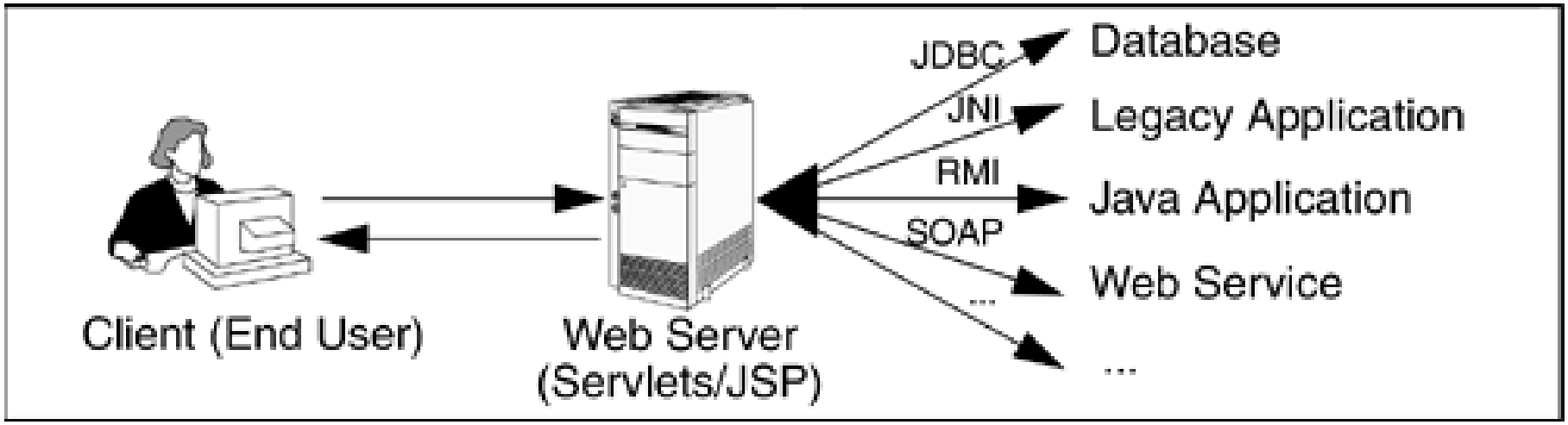
deskriptor

- file konfigurasi (format xml)
  - ear: application.xml (sun-application.xml)
    - berisi deskripsi modul yang ada dalam sebuah aplikasi (tidak diperlukan)
  - ejb: ejb-jar.xml (sun-ejb-jar.xml)
    - deskripsi enterprise beans & konfigurasinya (dapat menggunakan anotasi pada EJB 3)
  - web: web.xml (sun-web.xml)
    - konfigurasi web app, context parameter
    - servlet, servlet mapping, filter, filter mapping
    - security constraint

Web Module

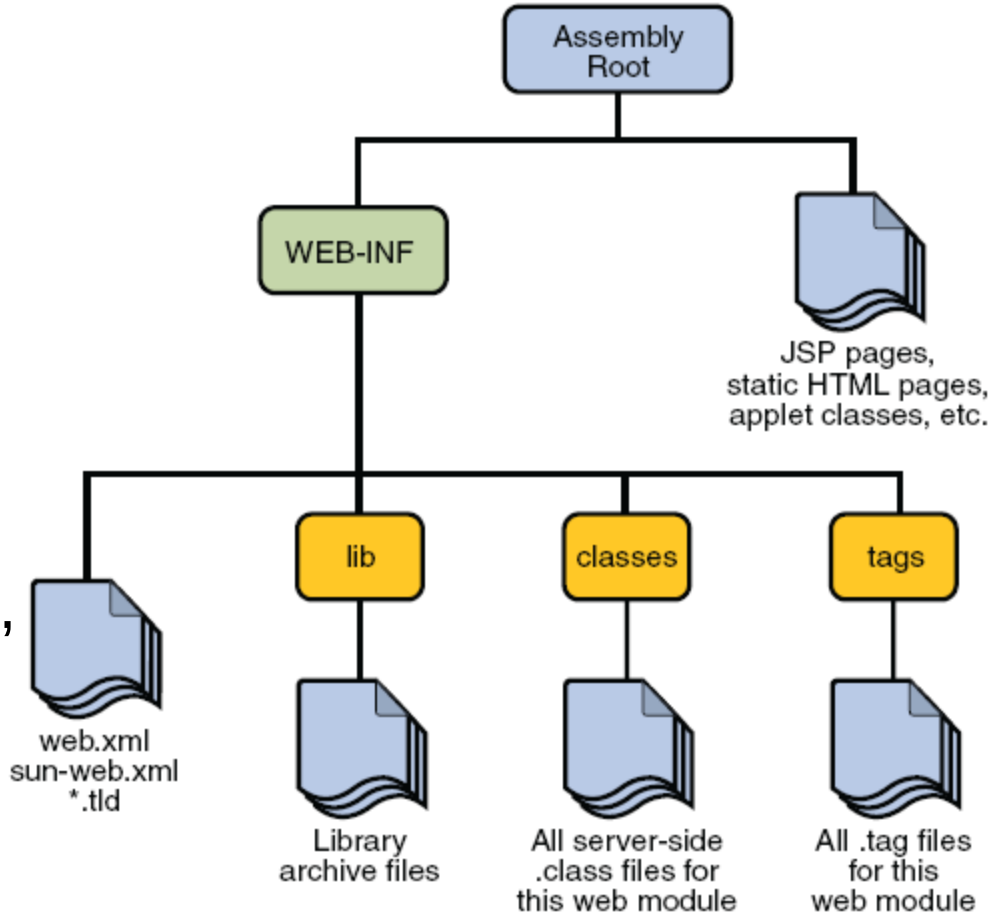
Modul yang digunakan untuk komponen Servlet, JSP, HTML, Javascript, JSF

Interaksi web



Struktur modul komponen web

- modul komponen web dikemas dalam sebuah file war
- WEB-INF tidak dapat diakses langsung user, berisi:
  - java classes
  - java lib
  - file konfigurasi (web.xml, sun-web.xml)



Struktur modul komponen web

sebuah modul web (war) dapat di-deploy langsung ke server, atau dikemas di dalam modul aplikasi (ear)

resource pada modul web diakses dengan menggunakan prefix, yang disebut sebagai context-root

konfigurasi context-root dilakukan pada sun-web.xml

file konfigurasi: web.xml

- konfigurasi umum
  - context parameter, listener
- konfigurasi servlet
  - servlet name, uri mapping, parameter
- konfigurasi filter
  - filter name, uri mapping
- konfigurasi page
  - welcome file, error page
- konfigurasi referensi
  - EJB, resource, message references
- konfigurasi security
  - security roles, security constraint

contoh

```
<?xml version="1.0" encoding="UTF-8"?>
"<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5"
xmlns:xsi=" http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" http://java.sun.com/xml/ns/ javaee
   http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">"
  <servlet>
    <servlet-name>SimpleServlet</servlet-name>
    <servlet-class>
      contoh.SimpleServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>SimpleServlet</servlet-name>
    <url-pattern>/simpleservlet</url-pattern>
  </servlet-mapping>
</web-app>
```

file konfigurasi: sun-web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD
   Application Server 9.0 Servlet 2.5//EN"
   "http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-
   0.dtd">
<sun-web-app error-url="">
  <context-root>/POFrontEnd</context-root>
  <class-loader delegate="true"/>
</sun-web-app>
```

Platform pengembangan Web

Glassfish: mendukung seluruh teknologi Java EE

Apache Tomcat: standalone/apache http plugin

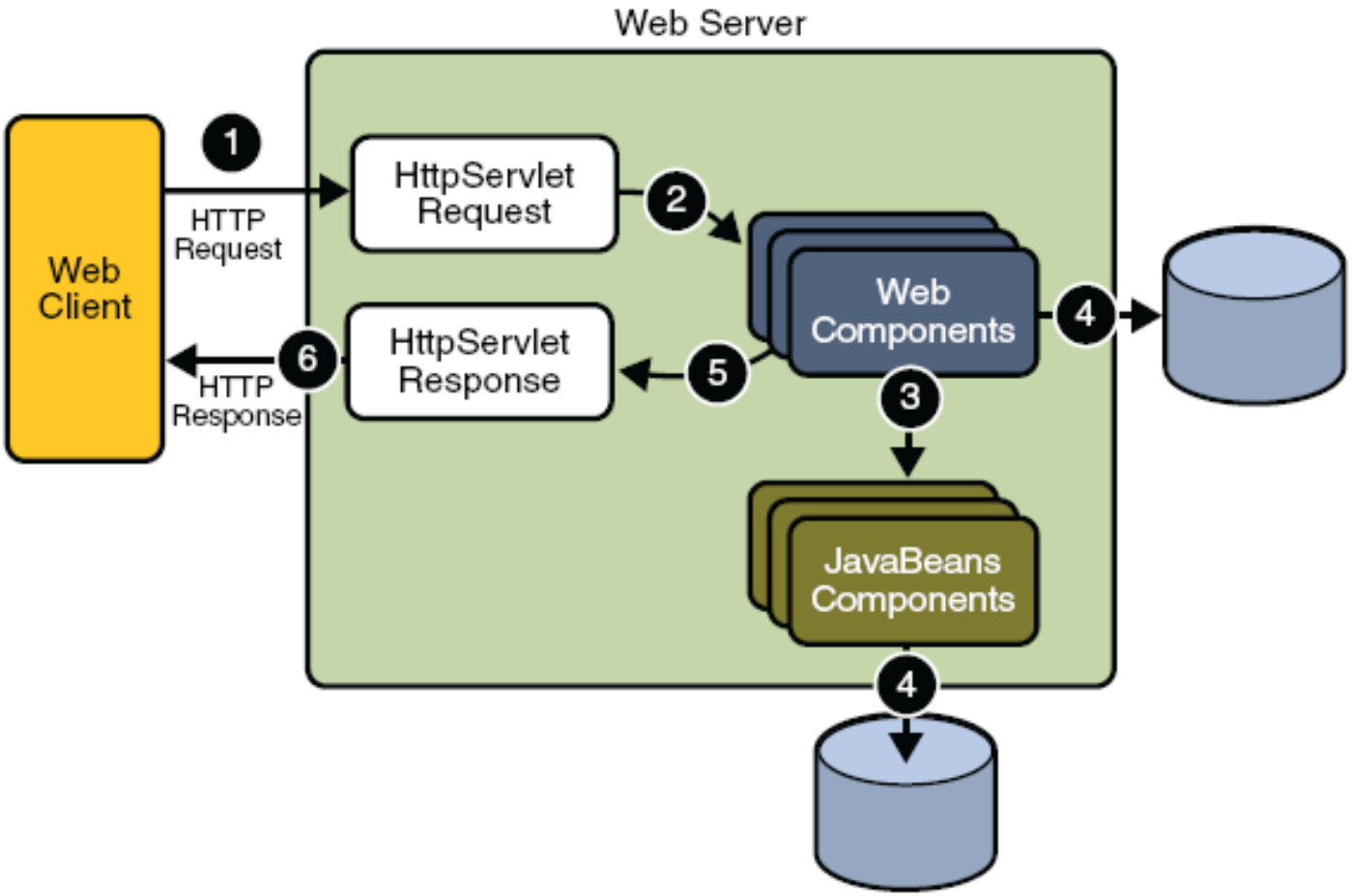
Jetty: open source standalone java web server

Servlet

Servlet adalah program Java yang berjalan pada web server:

- menerima request dari client
- membangkitkan kode HTML untuk dikirim ke client

Aplikasi Web



Contoh Servlet Sederhana

```
package contoh;
import ...

public class SimpleServlet extends HttpServlet {
  protected void doGet(HttpServletRequest request,
   HttpServletResponse response) {
    try {
      response.setContentType("text/html");
      PrintWriter printWriter = response.getWriter();
      printWriter.println("<h2>");
      printWriter.println("Hello");
      printWriter.println("</h2>");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Struktur servlet

Servlet diturunkan dari HttpServlet,

- override method doGet() dan/atau doPost
- doGet() dan doPost memiliki 2 argumen:
  - HttpServletRequest: informasi tentang request, e.g. : request URI, form data, HTTP request header, etc.
  - HttpServletResponse: output data, HTTP status code, objek PrintWriter untuk menuliskan HTML code

konfigurasi servlet

File servlet class diletakkan di /WEB-INF/classes

konfigurasi web.xml pada /WEB-INF (lihat contoh pada slide sebelumnya)

Servlet lifecycle

Servlet dibuat saat ada request pertama

multiple request akan memanggil servlet yang sama menggunakan thread yang berbeda

inisialisasi servlet dilakukan pada method init()

pemrosesan request dilakukan oleh method service() -> doPost, doGet, etc.

Handling form data

Form data dikirim dengan 2 cara:

- HTTP Get menggunakan URI parameter
- HTTP Post menggunakan parameter value pada header

Pada Servlet, form data dibaca menggunakan HttpServletRequest.getParameter()

Handling form

menampilkan form

memproses data

menampilkan ulang form jika ada masalah

Menampilkan form

Form dapat dibuat menggunakan HTML atau dibangkitkan oleh servlet

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html> <head>
<title>Input Data</title>
</head>
<body>
<form method="post" action="formhandlerservlet">
<table cellpadding="0" cellspacing="0" border="0">
<tr> <td>Masukkan teks:</td>
<td> <input type="text" name="enteredValue" /> </td> </tr>
<tr> <td></td>
<td><input type="submit" value="Submit"></td> </tr>
</table> </form> </body>
</html>
```

Memroses data

```
package contoh;
import ...
public class FormHandlerServlet extends HttpServlet {
  protected void doPost(HttpServletRequest request,
    HttpServletResponse response) {
    String enteredValue;
    enteredValue = request.getParameter("enteredValue");
    response.setContentType("text/html");
    PrintWriter printWriter;
    try  {
      printWriter = response.getWriter();
      printWriter.print("<p>Teks: ");
      printWriter.print(enteredValue);
      printWriter.print("</p>");
    } catch (IOException e) {
      e.printStackTrace();
    } }}
```

Handling output

Output dapat dibangkitkan langsung dari servlet.

Cara yg lebih baik: menggunakan RequestDispatcher dan JSP untuk output (dibahas di akhir sesi)

Session tracking & data sharing

Servlet menyediakan data sharing pada level:

- Aplikasi: javax.servlet.ServletContext
- Session: javax.servlet.HttpSession
- Request: javax.servlet.ServletRequest
- Page: javax.servlet.jsp.JspContext (diakses dari JSP)

Session tracking & data sharing

Contoh level aplikasi:

```
ServletContext ctx = getServletContext();
Data data = new Data();
ctx.setAttribute("data", data);
```

Contoh level session:

```
protected doPost(HttpServletRequest request,
  HttpServletResponse) {
  HttpSession session = request.getSession();
  Data data = new Data();
  session.setAttribute("data", data);
```

Session tracking & data sharing

Contoh level request:

```
protected doPost(HttpServletRequest request,
  HttpServletResponse) {
  Data data = new Data();
  request.setAttribute("data", data);
```

Session tracking & Data sharing

HTTP protokol stateless

penanganan session menggunakan:

- cookies
- URL rewriting
- hidden form fields

Servlet menyediakan mekanisme HttpSession

- menggunakan cookies dengan nama JSESSIONID

HttpSession

Akses menggunakan HttpRequest.getSession() atau HttpRequest.getSession(false).

getSession() akan otomatis membuat sesi baru jika tidak ada

getSession(false) akan mengembalikan null jika tidak ada sesi

HttpSession

- akses value dengan
  - session.getAttribute(key)
  - session.setAttribute(key, object)
- sesi dibersihkan dengan perintah
  - session.removeAttribute(key)
    - menghapus sebuah atribut
  - session.invalidate()
    - menghapus sesi pada aplikasi ini
  - session.logout()
    - menghapus seluruh sesi pada semua aplikasi

JSP

HTML Page yang dapat berisi kode Java

- Perubahan kode tidak memerlukan kompilasi ulang
- Lebih natural dalam menuliskan kode HTML

Contoh JSP sederhana

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="java.util.Date" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tanggal dan Jam Server</title>
</head>
<body>
<p>Tanggal dan jam pada Server: <% out.print(new Date()); %>
</p>
</body>
</html>
```

JSP

Problem dengan servlet

- tidak natural dalam menuliskan kode HTML
- perubahan HTML membutuhkan kompilasi
- tidak dapat menggunakan tools/editor HTML
- hanya dapat diedit oleh Java programmer

JSP syntax

HTML standard

JSP comment

```
<%-- komentar      --%>
```

JSP expression

```
<%= Java value %>
```

JSP scriptlet

```
<%  Java statements %>
```

JSP declaration

```
<%! Field definition %>
<%! Method definition %>
```

JSP directive

```
<%@ directive att="val" %>
```

JSP syntax

JSP Action

<jsp:blah>…</jsp:blah>

JSP Expression Language Element

${ EL expression }

Custom tag

<prefix:tagname> body </prefix:tagname>

interaksi JSP - Java

menyisipkan kode Java langsung pada JSP

membuat kelas utility dan memanggil kode Java dari JSP

menggunakan beans

menggunakan MVC architecture

menggunakan JSP expression language

menggunakan custom tags

interaksi JSP - Java

kode Java dapat langsung dituliskan ke JSP, menggunakan

- <%= … %> (JSP expression)
- <% … %> (JSP scriptlet)
- <%! … %> (JSP declaration)

Sebaiknya, sedikit mungkin menuliskan kode Java pada JSP

Contoh JSP expression

<%= … %>

Waktu saat ini <%= new java.util.Date() %>

Variabel terdefinisi pada JSP:

- request (HttpServletRequest)
- response (HttpServletResponse)
- session (HttpSession)
- out (Writer)
- application (ServletContext)

Contoh JSP Expression

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="java.util.Date" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-
   8">
<title>Tanggal dan Jam Server</title>
</head>
<body>
<p>Tanggal dan jam pada Server: <%= new Date() %>
</p>
</body>
</html>
```

Contoh JSP expression

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
   <HTML>
   <HEAD> <TITLE>JSP Expressions</TITLE> </HEAD>
   <BODY>
   <H2>JSP Expressions</H2>
   <UL>
   <LI>Current time: <%= new java.util.Date() %>
   <LI>Server: <%= application.getServerInfo() %>
   <LI>Session ID: <%= session.getId() %>
   <LI>The <CODE>testParam</CODE> form parameter: <%=
   request.getParameter("testParam") %> </UL>
   </BODY>
   </HTML>
```

Contoh JSP scriptlet

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML> <HEAD> <TITLE>Color Testing</TITLE> </HEAD>
<% String bgColor = request.getParameter("bgColor");
   if ((bgColor == null) || (bgColor.trim().equals(""))) {
     bgColor = "WHITE";
   }
%>
<BODY BGCOLOR="<%= bgColor %>">
  <H2 ALIGN="CENTER">Testing a Background of "<%= bgColor %>"
  </H2>
</BODY></HTML>
```

Contoh JSP declaration

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML> <HEAD> <TITLE>JSP Declarations</TITLE> </HEAD>
<BODY>
<H1>JSP Declarations</H1>
<%! private int accessCount = 0; %>
<H2>Accesses to page since server reboot: <%= ++accessCount %>
</H2>
</BODY></HTML>
```

JSP Directive

directive: konfigurasi yang mempengaruhi hasil dari sebuah servlet/halaman JSP

ada 3 jenis directive:

- page directive
- include directive
- taglib directive

JSP Directive

page directive

Attribute	Description	Valid Values	Default Value
`autoFlush`	Determines whether the output buffer should be flushed automatically when it is full.	`true` or `false`	`true`
`buffer`	The output buffer size in kilobytes.	`N`kb where `N` is an integer number. `"none"` is also a valid value.	`8`kb
`contentType`	Determines the page's HTTP response, MIME type, and character encoding.	Any valid MIME type and character encoding combination	`text/html; charset= ISO-8859-1`
`errorPage`	Indicates to what page to navigate when the JSP throws an exception.	Any valid relative URL to another JSP.	N/A
`extends`	Indicates the class this JSP extends.	The fully qualified name for the JSP's parent class.	N/A

page directive

Attribute	Description	Valid Values	Default Value
`import`	Imports one or more classes to be used in scriptlets.	A fully qualified name of a class to import, or the full package name + ".*" to import all necessary classes from the package (e.g., `<%@ page import java.util.* %>`).	N/A
`info`	The value for this attribute is incorporated into the compiled JSP. It can later be retrieved by calling the page's `getServletInfo()` method.	Any string.	N/A
`isErrorPage`	Determines if this page is an error page.	`true` or `false`	`false`
`isThreadSafe`	Determines whether the page is thread safe.	`true` or `false`	`true`
`language`	Determines the scripting language used in scriptlets, declarations, and expressions in the JSP page.	Any scripting language that can execute in the Java Virtual Machine (groovy, jruby, etc.).	`java`
`pageEncoding`	Determines the page encoding, for example, "UTF-8".	Any valid page encoding.	N/A
`session`	Determines whether the page has access to the HTTP session.	`true` or `false`	`True`

Contoh

```
<%@ page import="java.util.*" %>
<%@ page contentType="application/vnd.ms-
    excel" %>
```

JSP directive

include directive

- jsp:include
- include directive

Contoh

```
<jsp:include page="/templates/footer.jsp" />
<%@ include file="snippet.jsp" %>
```

Menggunakan JavaBeans

- JavaBeans adalah kelas Java biasa yang memiliki setter & getter untuk setiap atributnya
  - memiliki default constructor
  - modifier private untuk setiap atribut
  - memiliki setter getter untuk setiap atribut
- JavaBeans dapat diakses langsung dari JSP menggunakan <jsp:useBean> tag dan <jsp:setProperty>, <jsp:getProperty>

Contoh

```
package contoh;
public class CustomerBean
{
  public CustomerBean() {  }
  String firstName;
  String lastName;
  public String getFirstName() {  return firstName; }
  public void setFirstName(String firstName) {
    this.firstName = firstName; }
  public String getLastName() { return lastName; }
  public void setLastName(String lastName) {
    this.lastName = lastName; }
}
```

Contoh

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<jsp:useBean id="customer" class="contoh.CustomerBean" scope="page">
</jsp:useBean>
<jsp:setProperty name="customer" property="firstName" value="Albert"/>
<jsp:setProperty name="customer" property="lastName" value="Chan"/>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html> <head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JavaBean Properties</title> </head>
<body> <form>
<table cellpadding="0" cellspacing="0" border="0">
<tr> <td align="right">First Name: </td>
<td> <input type="text" name="firstName"
value='<jsp:getProperty name="customer" property="firstName"/>'>
</td> </tr>
```

Contoh

```
<tr> <td align="right">Last Name: </td>
<td> <input type="text" name="lastName" value='<jsp:getProperty
   name="customer" property="lastName"/>'>
</td>
</tr>
<tr>
<td></td>
<td><input type="submit" value="Submit"></td>
</tr>
</table>
</form>
</body>
</html>
```

Sharing beans

sharing beans antar page atau request dilakukan dengan menset property scope pada jsp:useBeans

- page
- request
- session
- application

contoh

```
<jsp:useBeans …   scope="session" />
```

Integrating JSP & Servlet

beans digunakan untuk merepresentasikan data

servlet digunakan untuk handle request

servlet mengisi data melalui beans

bean disimpan pada scope request, session, page atau application

forward request ke JSP

JSP ekstrak data dari bean, dan ditampilkan

Contoh

pada servlet

```
ValueObject value = new ValueObject(...);
request.setAttribute("key", value);
RequestDispatcher dispatcher =
  request.getRequestDispatcher("/WEB-
  INF/SomePage.jsp");
dispatcher.forward(request, response);
```

pada JSP

```
<jsp:useBean id="key" type="somePackage.ValueObject"
  scope="request" />
<jsp:getProperty name="key" property="someProperty" />
```

Menggunakan EL

```
public void doGet(HttpServletRequest request, HttpServletResponse
   response) throws ServletException, IOException {
    NameBean name = new NameBean("Marty", "Hall");
    CompanyBean company = new CompanyBean("coreservlets.com",
       "J2EE Training and Consulting");
    EmployeeBean employee = new EmployeeBean(name, company);
    request.setAttribute("employee", employee);
    RequestDispatcher dispatcher =
      request.getRequestDispatcher("/el/bean-properties.jsp");
    dispatcher.forward(request, response); }
```

Menggunakan EL

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
   <HTML> <HEAD><TITLE>Accessing Bean Properties</TITLE> </HEAD>
    <BODY>
    <TABLE BORDER=5 ALIGN="CENTER"> <TR><TH CLASS="TITLE">
    Accessing Bean Properties </TABLE>
    <P> <UL>
    <LI><B>First Name:</B> ${employee.name.firstName}
    <LI><B>Last Name:</B> ${employee.name.lastName}
    <LI><B>Company Name:</B> ${employee.company.companyName}
    <LI><B>Company Business:</B> ${employee.company.business}
   </UL>
   </BODY></HTML>
```

JSTL – JSP Standard Tag Library

- Pada JSP dapat menggunakan standard taglib yang disediakan oleh Java
- JSTL menyediakan kelompok tag library:
  - Core: http://java.sun.com/jsp/jstl/core
  - XML: http://java.sun.com/jsp/jstl/xml
  - Internationalization: http://java.sun.com/jsp/jstl/fmt
  - SQL: http://java.sun.com/jsp/jstl/sql
  - Functions: http://java.sun.com/jsp/jstl/functions

JSTL

Area	Subfunction	Prefix
Core	Variable support	c
	Flow control	
	URL management	
	Miscellaneous	
XML	Core	x
	Flow control	
	Transformation	
I18N	Locale	fmt
	Message formatting	
	Number and date formatting	
Database	SQL	sql
Functions	Collection length	fn
	String manipulation	

JSTL

untuk menggunakan JSTL, pada halaman JSP didefinisikan

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

uri adalah alamat uri untuk taglib tersebut (contoh core), dan prefix adalah prefix yang digunakan saat menggunakan taglib

JSTL

Contoh penggunaan

```
<c:if test="${sessionScope.cart.numberOfItems > 0}">
  <c:url var="url" value="/bookshowcart" >
    <c:param name="Clear" value="0" />
    <c:param name="Remove" value="0" />
  </c:url>

<p><a href="${url}">Show Cart</a>   
    <c:url var="url" value="/bookcashier" />
    <a href="${url}">Buy</a></p>
</c:if>
```