JavaServer Pages

Konrad Kurdej
Karol Strzelecki

Podejścia do projektowania web aplikacji za pomocą Javy

Serwlety Java

Serwlety Java to technologia wchodząca w skład Java EE (dawniej J2EE). Serwlet Java jest programem Java umiejscowionym po stronie serwera HTTP, służącym do automatycznego generowania dokumentów (np. HTML) stanowiących odpowiedzi na żądania HTTP. Serwlety Java wymagają dedykowanego środowiska uruchomieniowego nazywanego serwerem aplikacji Java EE (odpowiada on m.in. komunikację z bazami danych).

Serwlety Java mogą odwoływać się do kodu logiki biznesowej zaimplementowanego w formie obiektów Java Beans, Enterprise Java Beans, itp.

JavaBeans

JavaBeans - komponenty programowe dla środowiska Java opracowane w latach 90. przez firmę Sun Microsystems, będące niezależnymi modułami wywoływanymi i uruchamianymi w programach Java. W praktyce są to klasy napisane w określonej konwencji.

Specyfikacja JavaBeans firmy Sun definiuje je jako: "wielokrotnie używalne komponenty, które mogą być manipulowane wizualnie przez narzędzia do tworzenia oprogramowania".

Serwlet Java

Kod serwletów przypomina pisanie w PHP z tym że cały dokument to jest skrypt który wypisuje wszystkie części strony za pomocą np echo. Jest to o tyle nie praktyczne, że wszelkiego rodzaju narzędzia do wizualnej edycji stron nie dadzą się zastosować.

Tej wady pozbawione jest JSP które pozwala (podobnie jak PHP) na wstawianie dynamicznych fragmentów w kod HTML.

Serwlety: HTML wewnątrz Javy
JSP: Java wewnątrz HTMLa

Serwlet Java

Serwlet musi:

1. dziedziczyć po klasie HttpServlet
2. obsługiwać jedną z metod:
   - public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
   - public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

opcjonalnie (wywoływane tylko raz!):

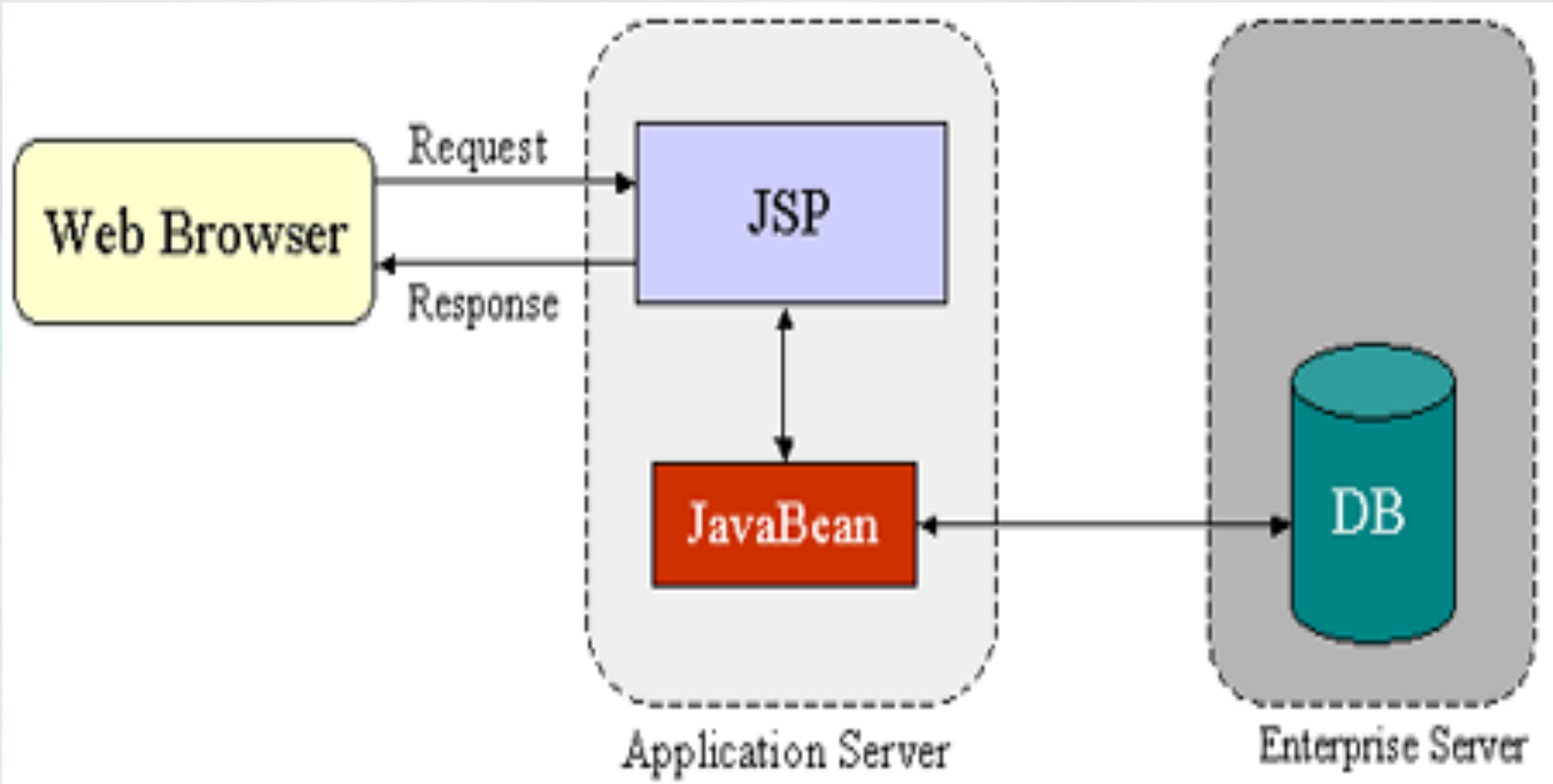
- public void init(ServletConfig config) throws ServletException
- public void destroy()

Przykładowy serwlet

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

public class HTTPServletTemplate extends HttpServlet {

public void doGet (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {

res.setContentType("text/html");
PrintWriter w = res.getWriter();
w.println("<html><head><title>Hello World</title></head>");
w.println("<body><h1>Hello World</h1></body></html>");

w.flush(); // Commits the response
w.close();
}

}
```

Dwa podejścia do projektowania Java web aplikacji

Model 1


Web Browser
Request
Response
JSP
JavaBean
DB
Application Server
Enterprise Server

Model 1

Świetnie się sprawdza w niewielkich projektach.

Ale...

Zbyt silne powiązanie logiki biznesowej z logiką prezentacji utrudnia rozwijanie i modyfikację projektu

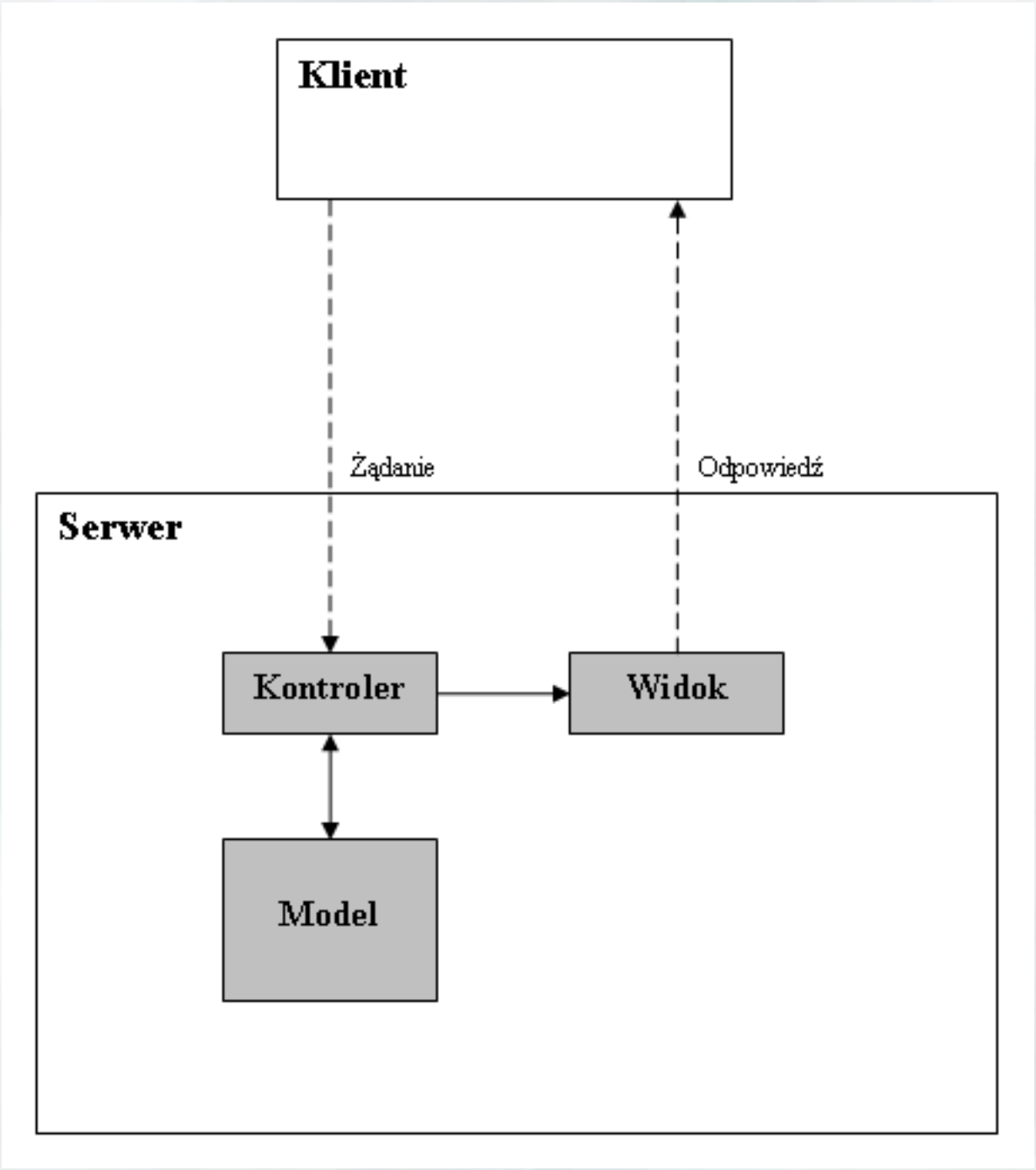
MVC - koszmar poprzedniego roku powraca :)

Poraz pierwszy opisał wzorzec Model-View-Controller w 1979 Trygve Reenskaug pracując nad Smalltalk-iem w Xerox PARC. MVP jest jego odmianą z początków lat 90..

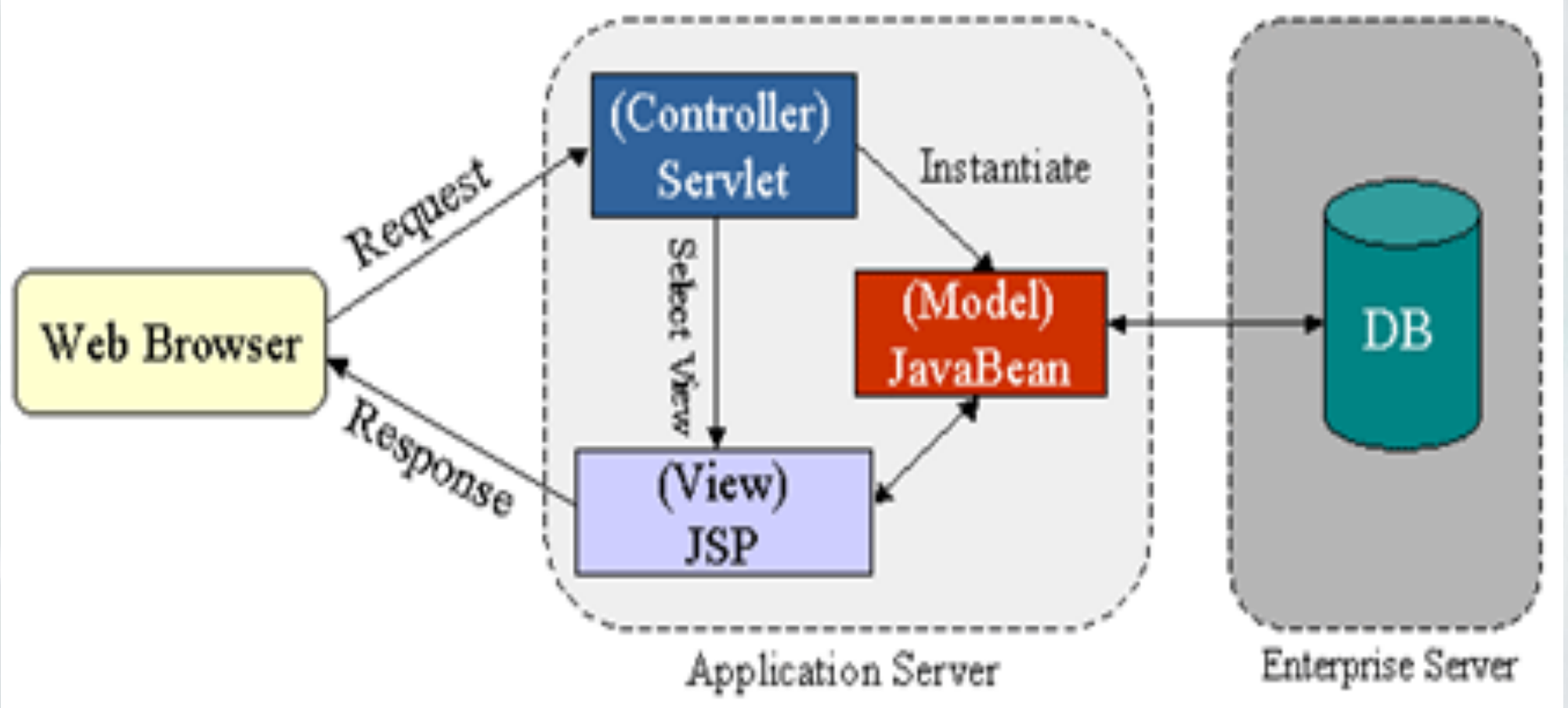
W listopadzie 2002 W3C uznało strukturę MVC za część ich architektury XForms dla wszystkich przyszłych web aplikacji. Zostanie to zintegrowane z specyfikacją XHTML 2.0.

Współcześnie rozdziela się MVC na części odpowiedzialne za:

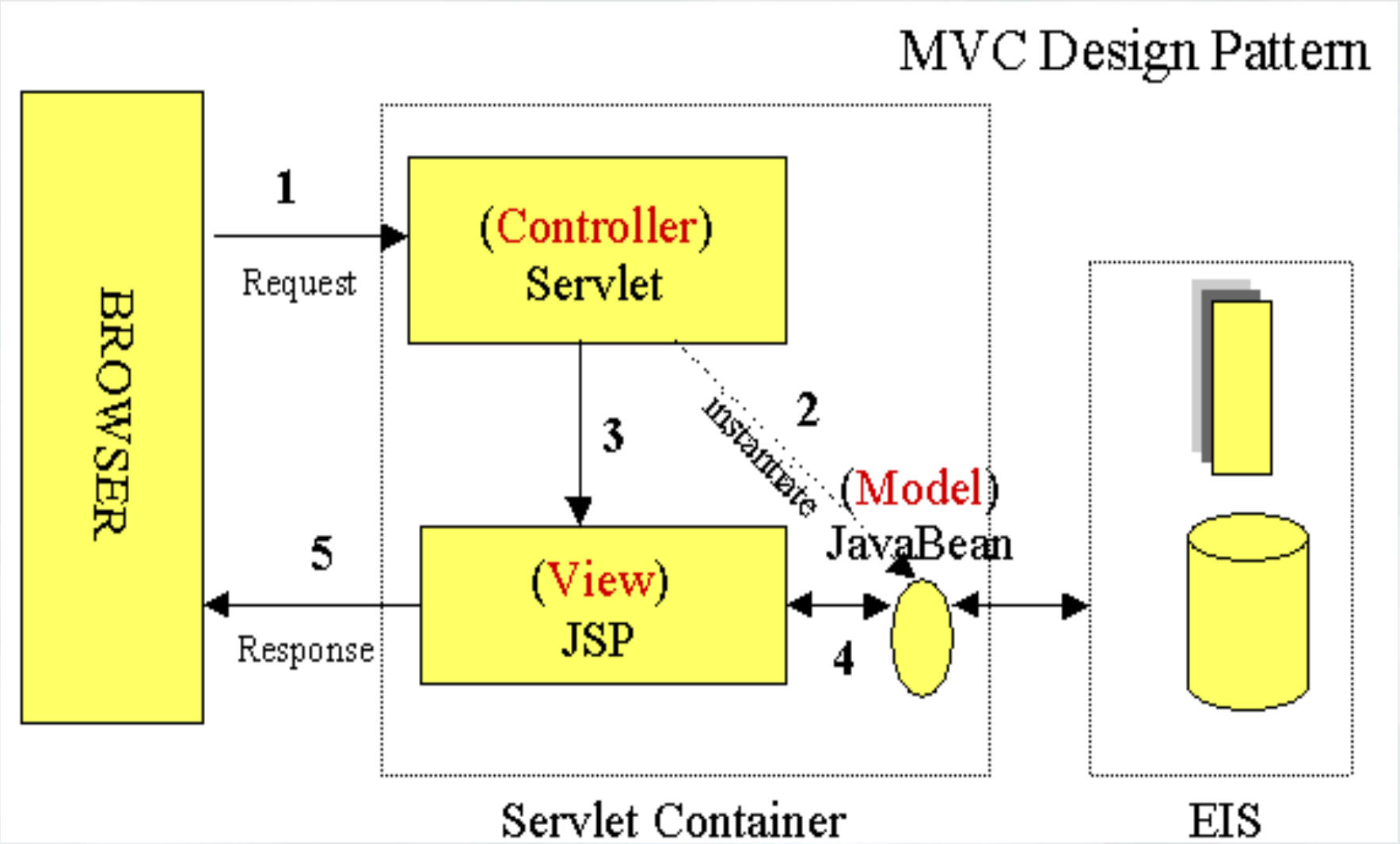
- model danych
- interfejs użytkownika
- logikę sterowania


Klient
Żądanie
Odpowiedź
Serwer
Kontroler
Widok
Model

Model 2


(Controller)
Servlet
Instantiate
Request
Select View
Web Browser
(Model)
JavaBean
DB
Response
(View)
JSP
Application Server
Enterprise Server

Model 2


MVC Design Pattern
BROWSER
1
Request
(Controller)
Servlet
2
instantiate
3
(Model)
JavaBean
5
Response
(View)
JSP
4
Servlet Container
EIS

Model 2

1. Przeglądarka wysyła żądanie. Aplikacja jest tak skonfigurowana, że każde żądanie jest kierowane do serwletu – kontrolera.
2. Serwlet – kontroler analizuje żądanie i tworzy wymagane przez żądany widok obiekty klas zewnętrznych. Interakcja kontrolera z modelem może pociągać za sobą interakcję z bazą danych.
3. Serwlet przekazuje sterowanie do odpowiedniego widoku - strony JSP.
4. JSP pobiera dane z obiektów modelu przygotowanych przez kontroler. Obiekty te mogą udostępniać dane pobrane z bazy danych.
5. JSP generuje wynikowy dokument HTML przesyłany do przeglądarki.

Model 2

- Stosowany jest przy średniego i dużego rozmiaru projektach.
- Łatwiej rozwijać, zmieniać interfejs użytkownika itp.

Apache Struts

- open-sourcowy framework* dla aplikacji webowych Java EE
- wspiera tworzenie zgodnie z MVC (głównie widoku i kontrolera

* framework - struktura wspomagająca tworzenie aplikacji. Często udostępnia też dodatkowe aplikacje.

Model 2

Ogromną zaletą tego podejścia jest wykożystanie mocnych stron każdej z technologi i uniknięcie ich słabości. W JSP niewygodne jest pisanie bardziej skomplikowanych programów, ale za to łatwo jest generować kod strony. Z kolei w serwletach jest na odwrót.

Servlet container - kontener serwletów

Konieczne do uruchomienia serwletów jest funkcjonowanie serwera WWW z zaimplementowanym kontenerem serwletów.

Apache Tomcat

- kontener serwletów
- rozwijany przez Apache Software Foundation (ASF)
- serwlety Javy oraz JavaServerPages zaimplementowane zostały według specyfikacji Sun Microsystems
- czysty Javowy serwer stron (nie mylić z Apache web serwer)
- wydawany na licencji Apache License 2.0 (open source)
- ostatnia wersja 6.0.18 (obsługa specyfikacji 2.5 servletów i 2.1 JSP)
- wersje 5.5.x wspierają serwlety w wersji 2.4 i JSP 2.0
- Apache Ant powstał przy okazji przenoszenia Tomcat-a na otwartą licencję (odpowiednik make)

Apache Tomcat

Składa sie z trzech części:

- Catalina - kontener serwletów
- Coyote - komponent do obsługi połączeń HTTP (wspiera HTTP/1.1). Oczekuje na połączenia na odpowiednim porcie i przekazuje je do Tomcat Engine aby je obsłużył i wysyła odpowiedź
- Jasper - silnik JSP. Parsuje pliki JSP i aby potem skompilować je do kodu Javy jako serwlety (zajmuje się tym Catalina). W trakcie działania potrafi wykryć zmiany w plikach JSP i przekompliować je

Bardzo popularny. Dostępny chociażby przy instalowaniu pakietu NetBeans.

Jetty

- w 100% napisany w Javie serwer HTTP i kontener serwletów
- open-source - na licencji Apache 2.0 License
- wykorzystywany w takich projektach jak JBoss czy Geronimo (serwery aplikacji)
- autorzy skupiają się głównie na prostocie, wydajności i przenośności dzięki czemu nadaje się on także do systemów wbudowanych

JSP - JavaServer Pages

- standard opracowany przez Sun
- działa po stronie serwera
- kompilowane do serwletu

```
01. <%@ page language="java" contentType="text/html;
charset=ISO-8859-2"
02. pageEncoding="ISO-8859-2"%>
03. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
04. "http://www.w3.org/TR/html4/loose.dtd">
05. <%! int k=5; %>
06. <html>
07. <head>
08. <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-2">
09. <title>Przykładowa strona JSP</title>
10. </head>
11. <body>
12. Aktualny czas: <%=java.util.Calendar.getInstance().
getTime()%>
13. <%
14. for (int i=0; i<k; ++i) {
15. %>
```

Podstawy

- umożliwia osadzanie kodu Javy w dokumencie html
- osadzony kod nazwiemy skryptletem

znaczniki:
<% instrukcje %> -> tak oznaczamy obszar, w którym ma zostać wykonany kod, np.:
<% for(;;); %> spowoduje, że trochę poczekamy na odpowiedź

<%= wyrażenie %> -> w ten sposób wklejamy wartość wyrażenia, np.:
<%= new java.util.Date() %> wypisze na ekranie aktualną datę

cd.

<%! instrukcje %>
    treść metod/pól, które znajdą się w servlecie wytowrzonym z danego pliku *.jsp

<%-- komentarz --%>
    bez komentarza

możliwe również komentarze w obrębie skryptletu

    <% //np. takie %>

html i jsp

kody html i jsp mogą być mieszane, zapis

```
<% for(int i = 0; i < 10; i++){ %>
    <B>tratatata</B>
<% } >
```

zostanie prawdopodobnie przetłumaczony na kod servletu

```
for(int i = 0; i < 10; i++){
    out.println("<B>tratatatata</B>");
}
```

eliminuje to istotną niewygodę w używaniu serwletów

Obiekty niejawne

out

- obiekt typu javax.servlet.jsp.JspWriter
- użycie <% out.println("fiu fiu") %>
- zamiast <%= "fiu fiu" %>

request

- typu javax.servlet.http.HttpServletRequest
- przechowuje żądanie http
- możemy go odpytywać o pola formularza

request cd.

strona1.jsp

```
...
<form>
    <intput type="text" name="pole1" />
    <input type="submit" action="strona2.jsp" />
</form>
...
```

strona2.jsp

```
<% out.println(request.getParameter("pole1")); %>
```

wypisze treść pola

Inne metody request'a

zapytania nt. serwera
- getServerName();
- getProtocol();

nagłówki http request
- getContentType();
- getContentLength();

również tak (ale nie wszystkie):
- getHeader("Accept");

response

- typu javax.servlet.http.HttpServletResponse
- można dołączać nagłówki

```
reponse.setContentType("text/plain");
```

- można przekierować na inną stronę

```
//to pierwsze ustawia status
 response.setStatus(response.SC_MOVED_TEMPORARILY);
 response.setHeader("Location", "http://www.fiufiu.com");
```

reponse cd.

- można też wysłać strumień danych

```
response.setHeader("Content-Disposition", "attachment;
filename=\"" + "nazwa_pliku" + "\"");

//a następnie piszemy na response.getOutputStream();
```

pageContext, session

pageContext ma odniesienia do request oraz response, można również pobrać resztę z wymienianych

session - umożliwia śledzenie sesji, udostępnia m.in. metody:

- String getId();
- boolean isNew();
- void invalidate();
- void putValue(String key, Object value); -> umożliwia zapamiętywanie jakichś wartości

Akcje w JSP

na stronie możemy umieścić coś takiego:

```
<jsp:nazwa_akcji  atrybut1="w1" />
```

są 2 wspólne atrybuty dla wszystkich akcji:

- id - identyfikator
- scope - zasięg ze zbioru {page, request, session, application}

<jsp:useBean>

wyszukuje lub tworzy egzemplarz JavaBean'a, czyli klasy z elementarnymi polami i metodami getNazwa oraz setNazwa

ma już trochę więcej atrybutów

```
<jsp:useBean id="fasolka" class="mojaPaczka.Fasolka" />
```

tworzy obiekt klasy mojaPaczka.Fasolka o nazwie fasolka.

inne atrybuty:

type - określa bazową klasę, która może być następnie rzutowana przez atrybut class

beanName - podobnie jak class, korzysta z Beans.instantiate();

<jsp:setProperty>

umożliwia automatyczną inicjalizację JavaBean'a

```
<jsp:useBean id="beanek" class="mojaPaczka.Beanek">
    <jsp:setProperty name="beanek" property="*" />
</jsp:useBean>
```

Taki kod pobiera wszytkie pola z formularza, dopasowuje ich nazwy do pól JavaBean'a i inicjalizuje automatycznie pola beanka.

Zamiast property="*" możemy użyć property="pole"

cd.

można też tak:

<jsp:setProperty name="beanek" property="poleBeanka" param="poleZFormularza" />

w ten sposób można postąpić w przypadku, gdy nie zgadzają się nazwy pól formularza z polami Bean'a

można też tak:

<jsp:setProperty name="beanek" property="poleBeanka" value="<% new String("szaleństwo") %>"

Takie postępowanie umożliwia oddzielenie warstwy modelu od warstwy prezentacji.

<%@ czyli znaczków cd.

dołącznie plików w czasie tłumaczenia:

```
<%@ include file="/menu.html" %>
<%@ include file="deklaracje.inc" %>
<body>
Zmienna zadeklarowana w innym pliku: <%= zmienna %>
</body>
```

<jsp:include page="strona.jsp" /> dołącza stronę w czasie żądania

znaczki cd.

<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>

w taki sposób można określać różne właściwości strony
a tak robimy import

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

Wtrącenie o ciasteczkach

W łatwy sposób można zarządzać ciasteczkami. Tworzy się je konstruktorem

```
public Cookie(String name, String value);
```

wysyła

```
response.addCookie(Cookie cookie);
```

a odbiera

```
Cookie[] request.getCookies();
```

Definiowanie własnych znaczników

Ponoć bardzo przydatny mechanizm, umożliwia grupowanie tworzenie własnych tagów jako kompozycji dostępnych.

Stworzenie nowego taga jest skomplikowane, użycia bardzo proste:

<%@ taglib uri="http://localhost/taglib/mojeTagi.jar" prefix="mojeTagi" %>

<mojeTagi:mojTag parametr1="wartosc1"> tekst </mojeTagi:mojTag>

Ćwiczenia:

http://java.sun.com/developer/onlineTraining/JSPIntro/exercises/Counter/index.html

http://java.sun.com/developer/onlineTraining/JSPIntro/exercises/Forms/index.html

Bilbiografia

http://en.wikipedia.org/wiki/JavaServer_Pages
http://en.wikipedia.org/wiki/Java_Servlet
http://pl.wikipedia.org/wiki/JSP_Model_2
http://java.sun.com/developer/technicalArticles/javaserverpages/servlets_jsp/
http://java.sun.com/developer/onlineTraining/JSPIntro/contents.html
http://osilek.mimuw.edu.pl/index.php?title=AWWW-1st3.6-w10.tresc-1.1-toc
"Java Servlet Programowanie", Jason Hunter i William Crawford
"JSP Tworzenie Stron WWW", Damon Hougland, Aaron Tavistock

Przykład:
http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html