Technologie pro web a multimedia

12. přednáška

Dynamický web, serverové technologie

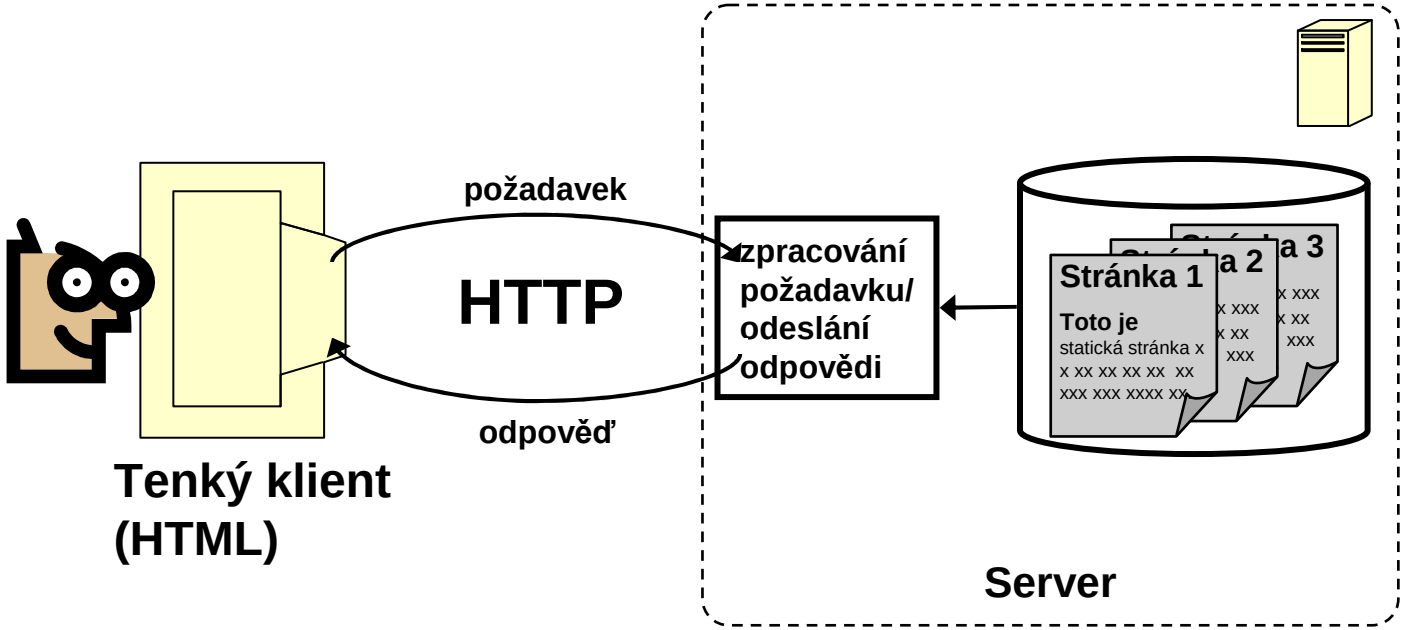
Martin Klíma, Miroslav Bureš

Computer Graphics Group

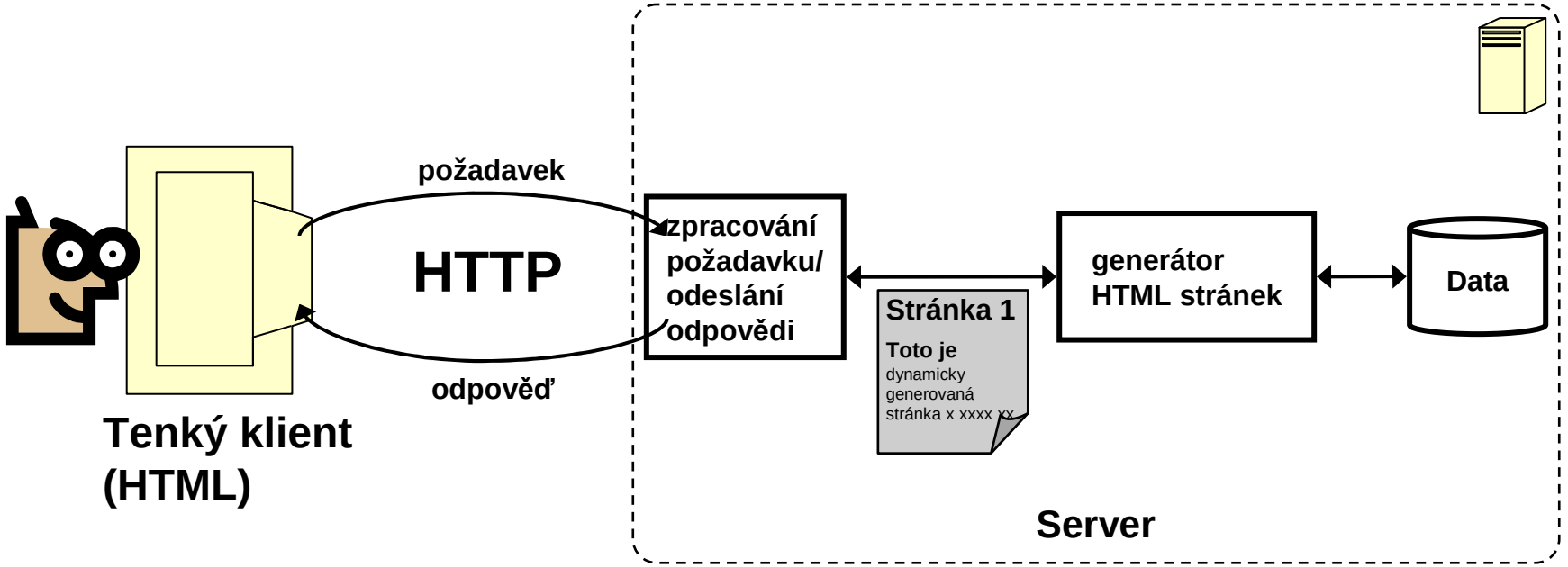
Obsah

- Web server
  - instalace
  - konfigurace
  - rozšíření
  - CGI
  - SSI
  - PHP
- Aplikační servery
  - J2EE
  - .NET

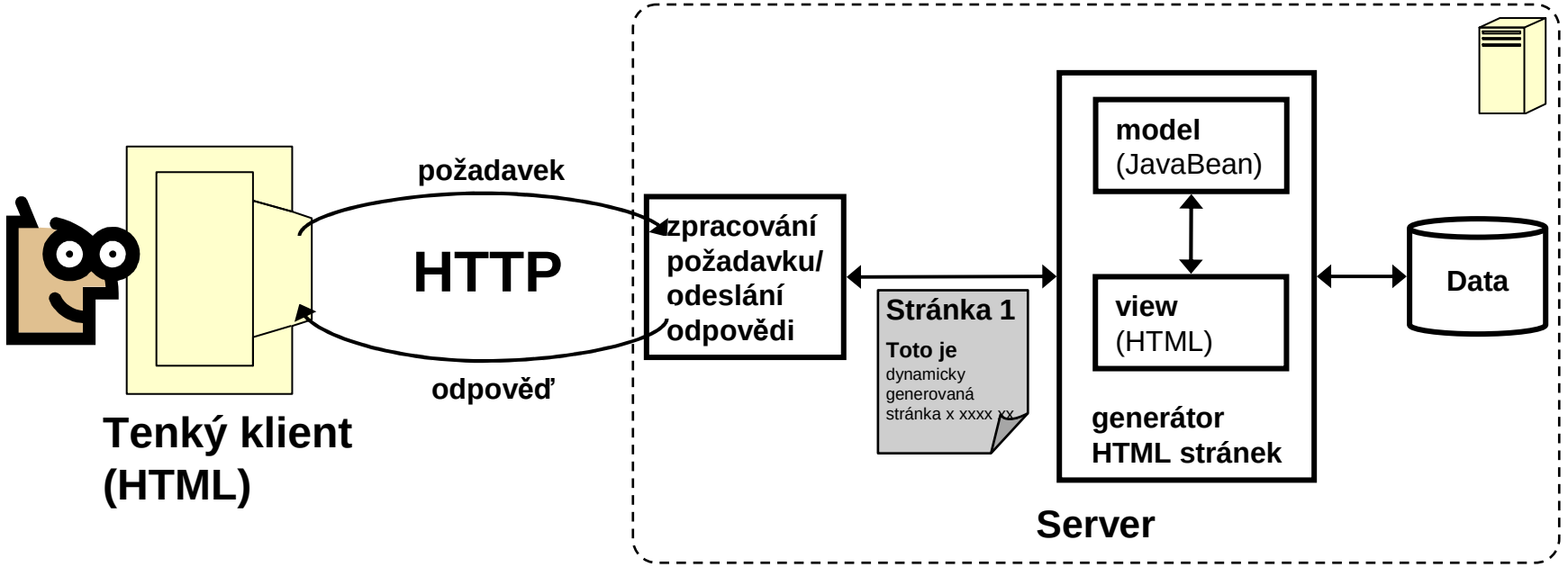
Architektura web aplikace: statický web



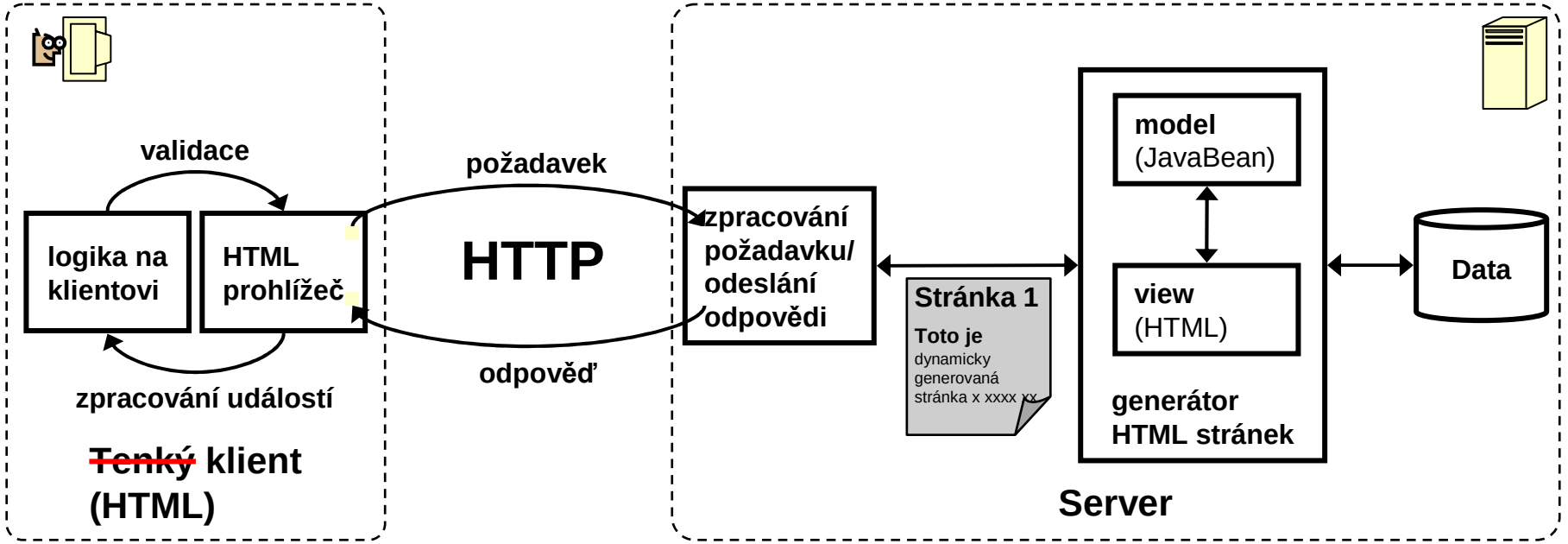
Architektura web aplikace: dynamický web



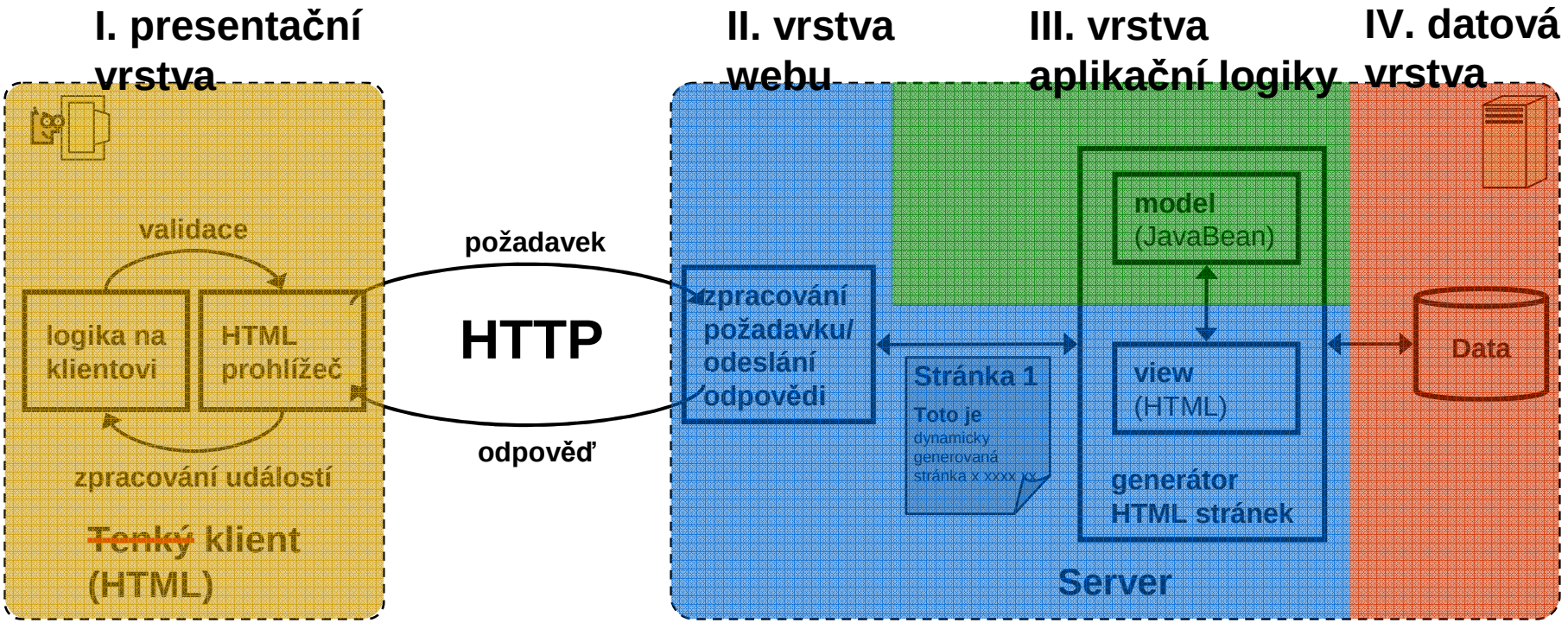
Architektura web aplikace: dynamický web



Architektura web aplikace: dynamický web



Architektura web aplikace: dynamický web



Web server

- Celá řada komerčních i nekomerčních
- Jednoduchý server není nic složitého
  - embeded servery v mnoha zařízeních (access point, router, web kamera,….)

Stručný seznam (zdroj wikipedia.cz)

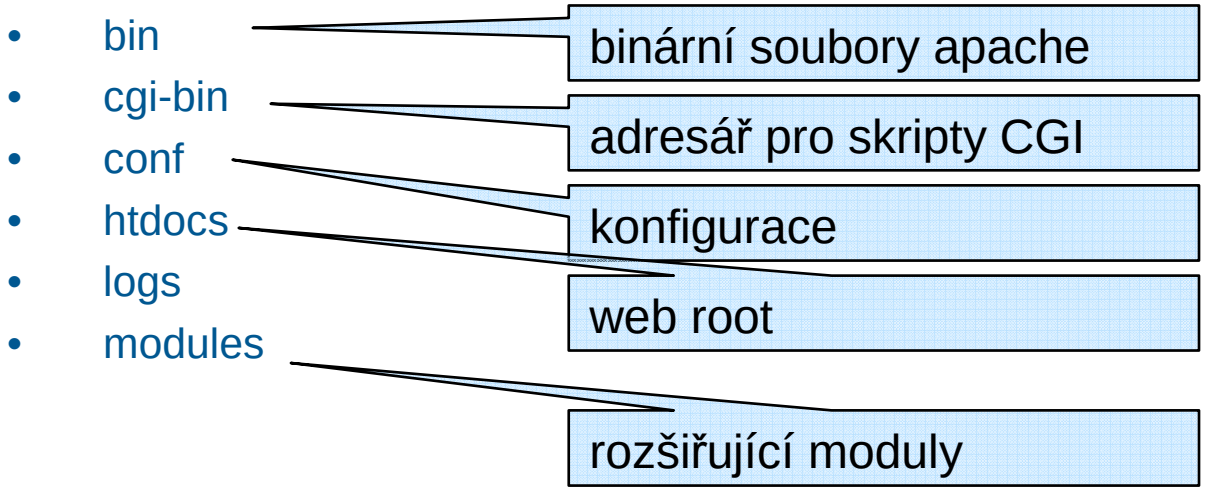
- Abyss Web Server
- Apache HTTP Server
- BadBlue
- Boa
- Caudium
- Covalent Enterprise Ready Server
- Fnord
- IBM HTTP Server
- Internet Information Services
- Light HTTP Server
- mathopd
- NaviServer
- NCSA HTTPd
- Oracle HTTP Server,
- PinkNet Web Server
- Roxen
- Small http server
- Sun Java System Web
- thttpd
- TinyWeb
- Xitami
- Zeus Web Server

Nejběžnější web servery

- Apache
- IIS
- Tomcat

Instalace a konfigurace Apache

1. Instalace
   - stáhnout z http://httpd.apache.org/ nebo nějaký balíček předkonfigurovaného apache, např. WinLamp http://sourceforge.net/projects/winlamp/
2. Konfigurace
   - hlavní adresáře vytvořené po instalaci (WinLamp)



Konfigurační soubor httpd.conf

- obsahuje globální nastavení web serveru
- obsahuje nastavení virtuálních web serverů
  - založených na různých IP adresách
  - založených na stejné IP adrese, rozlišuje se podle jména

Důležité direktivy

```
# kde je adresář serveru
ServerRoot "C:/Apache2"

# kde je obsah webu
DocumentRoot "C:/Apache2/htdocs"

# jak se server jmenuje (mělo by být stejné jako záznam v DNS)
ServerName localhost:80

# naslouchající port
Listen 80

# nahrání rozšíření PHP
LoadModule php5_module "C:/Apache2/modules/php/php5apache2.dll"

# přiřazení souborů *.php k aplikaci php
AddType application/x-httpd-php .php
```


Definice více virtual serverů

- většinou mám k dispozici jeden počítač s jednou IP adresou
- chci na něm mít více web serverů, např. pro vývoj

- name virtual host poslouchají nas stejné IP adrese a rozlišují obsluhu podle hlavičky

  Host: www.host1.com:80
- tato hlavička existuje od HTTP verze 1.1
- posílají jí všechny moderní prohlížeče

Definice name virtual host v apache

```
# zapni name virtual hosty, na všech ip adresách a portu 80
NameVirtualHost *:80

# definice virtual hostu
<VirtualHost *:80>
    # root tohoto webu
    DocumentRoot c:/www_root/test
    # jmeno, ze systému DNS
    ServerName test
    # alternativní jméno, pokud více záznamů ukazuje na stejný počítač
    ServerAlias testovaci_server
    # vytvořím si extra logovací soubory pro tento server
    ErrorLog logs/test_log
    CustomLog logs/test-log common
</VirtualHost>
```

Záznam v DNS

- …musí provést správce DNS
- pro potřeby vývoje stačí záznam v souboru hosts v operačním systému

- Windows:

  c:\windows\system32\drivers\etc\hosts
- Linux:

  /etc/hosts

Záznam v hosts

```
127.0.0.1       localhost

127.0.0.1       test testovaci_server
```

CGI technologie

Common Gateway Interface

- rozšíření serveru standardizovaným způsobem
- možnost dynamicky generovat odpověď
- standardizované rozhraní
- prakticky jakýkoli jazyk
- z důvodu bezpečnosti se omezuje na vybrané adresáře
- implementace možná v:
  - C/C++
  - Fortran
  - PERL
  - TCL
  - Unix shell
  - Visual Basic
  - AppleScript

Parametry předávané CGI skriptu

- veškeré parametry jsou předávány jako proměnné prostředí
- není rozdíl mezi během skriptu volaného webovým serverem a samostatně spuštěným
- web server naplní proměnné

Pingování na počítač tazatele

```
#!/bin/sh -f

echo "Content-type: text/html"

echo ""

echo "<HTML>"

echo "<H1>Ping info to host $REMOTE_HOST:</h1>"

echo "<DIV>"

ping -c 10 $REMOTE_HOST

echo "</DIV>"

echo "</HTML>"

exit 0
```

Nevýhody CGI

- Skript (program) se musí spustit při každém dotazu
  - alokování pamětí
  - uvolňování paměti
  - nemožnost komunikace mezi procesy
  - nemožnost sdílení paměti

Výhody

  - pád procesu neohrozí server
  - jednoduchost
  - mnoho různých programovacích jazyků
  - standard

Fast CGI

- Snaha o zrychlení odezvy
- Proces, ve kterém běží FastCGI je trvalý, je znovu použit při novém dotazu
- Zpětně kompatibilní s CGI
- Umožňuje běh i vzdálené FastCGI aplikace

Rozšíření Web Serveru pomocí interního rozhraní

- ISAPI - Internet Server API
  - definované firmou Microsoft a je založeno na volání funkcí z Dynamic Link Library (DLL)
  - Moduly běží (na rozdíl například od CGI) ve stejném kontextu a adresovém prostoru jako webový server
  - Data mezi "jádrem" serveru a ISAPI aplikací se předávají snadno a rychle pomocí ukazatelů, modul může jednoduše zjišťovat podrobnosti o serveru
  - Chyba v modulu může způsobit pád celého WWW serveru. Jako programovací jazyk lze použít C nebo C++

SSI = Server Side Includes

- Jednoduchý jazyk pro dynamické efekty v HTML stránkách

- Konfigurace apache:

```
Options +Includes
AddType text/html .shtml
AddHandler server-parsed .shtml
```

- Soubory .shtml

  do souborů se vkládají html komentáře, které jsou serverem interpretovány

př: <!--#echo var="DATE_LOCAL" -->

výstup: Tuesday, 29-May-2007 19:30:27 Střední Evropa (běžný čas)

PHP

- Personal Home Pages
  - jazyk speciálně jen pro generování web obsahu
  - syntakticky něco mezi C a Perl (spíš obojí než mezi)
  - interpretovaný jazyk => pomalý
  - ve své podstatě procedurální, v nové verzi objektový
  - velmi populární
  - proměnné prostředí se předávají ve romě polí

    $_GET, $_POST, $_REQUEST, $_SERVER, $_COOKIE, $_ENV,...
  - Existuje ve verzi CGI i ISAPI

PHP

- Soubory *.php jsou registrovány u web serveru a jsou předány modulu php
- Soubor php obsahuje HTML (XHTML) stránku s vloženými kusy PHP kódu

- Varianta GCI i ISAPI

PHP ukázka

PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Test datumu</title>
    </head>
    <body>
          Nahrani stranky <?php print date("d.m.Y h:i:s"); ?>
    </body>
</html>
```

Výsledek

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Test datumu</title>
    </head>
    <body>
          Nahrani stranky 26.05.2007 02:46:43    </body>
</html>
```

PHP – výhody, nevýhody

Výhody

- Rychlý vývoj malých aplikací
- Jednoduchý jazyk, rychlé učení
- Nenáročnost
- Velká komunita
- Mnoho rozšíření
- Běží prakticky všude
- Perfektní na malé projekty

Nevýhody

- Výkon
- Netypový jazyk
- Ve starších verzích problematická objektovost (ve verzi PHP5 již lepší)
- Load balancing = problém
- Nehodí se na velké projekty
- Striktní 3 vrstvá architektura

J2EE

Distribuce javy:

- J2SE
  - Standard Edition, pro potřeby běžných desktop aplikací
- J2ME
  - Micro Edition, pro potřeby mobilních zařízení
- J2EE
  - Enterprice Edition, pro web-based aplikace

Aplikační server

- Aplikace běží bez ohledu na http request
- Komplexnější architektura
- Hodí se na velké projekty
- Pro malé projekty je příliš komplikovaná
- Umožňuje load balancing

Nejznámější aplikační servery

OpenSource J2EE (Java)

- GlassFish
- JBoss
- Tomcat (jen servlet container)

Enterprise J2EE (Java)

- IBM: WebSphere
- BEA: WebLogic

Enterprise

- Microsoft: MS .NET

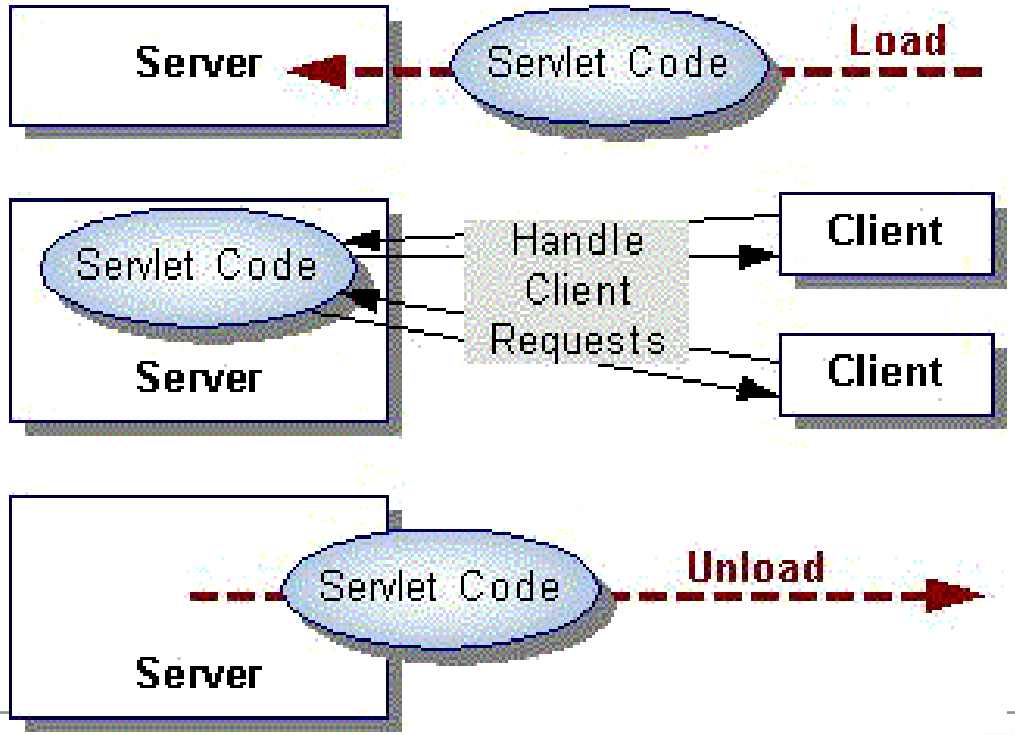
J2EE

- Vybrané komponenty:
  - Servlet
  - JSP
  - Enterprise Java Beans (EJB)
  - JavaMail API
  - Java API for XML processing
  - JDBC, …

Servlet

- Java program, který obsluhuje HTTP dotazy
- Žije uvnitř servlet containeru, např. Tomcat

- Životní cyklus



Jednoduchý servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
   {
    public void doGet(HttpServletRequest request, HttpServletResponse
   response)                throws IOException, ServletException
      {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<html>");
      out.println("<body>");
      out.println("<head>");
      out.println("<title>Hello World!</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h1>Hello World!</h1>");
      out.println("</body>");
      out.println("</html>");
      }
    }
```

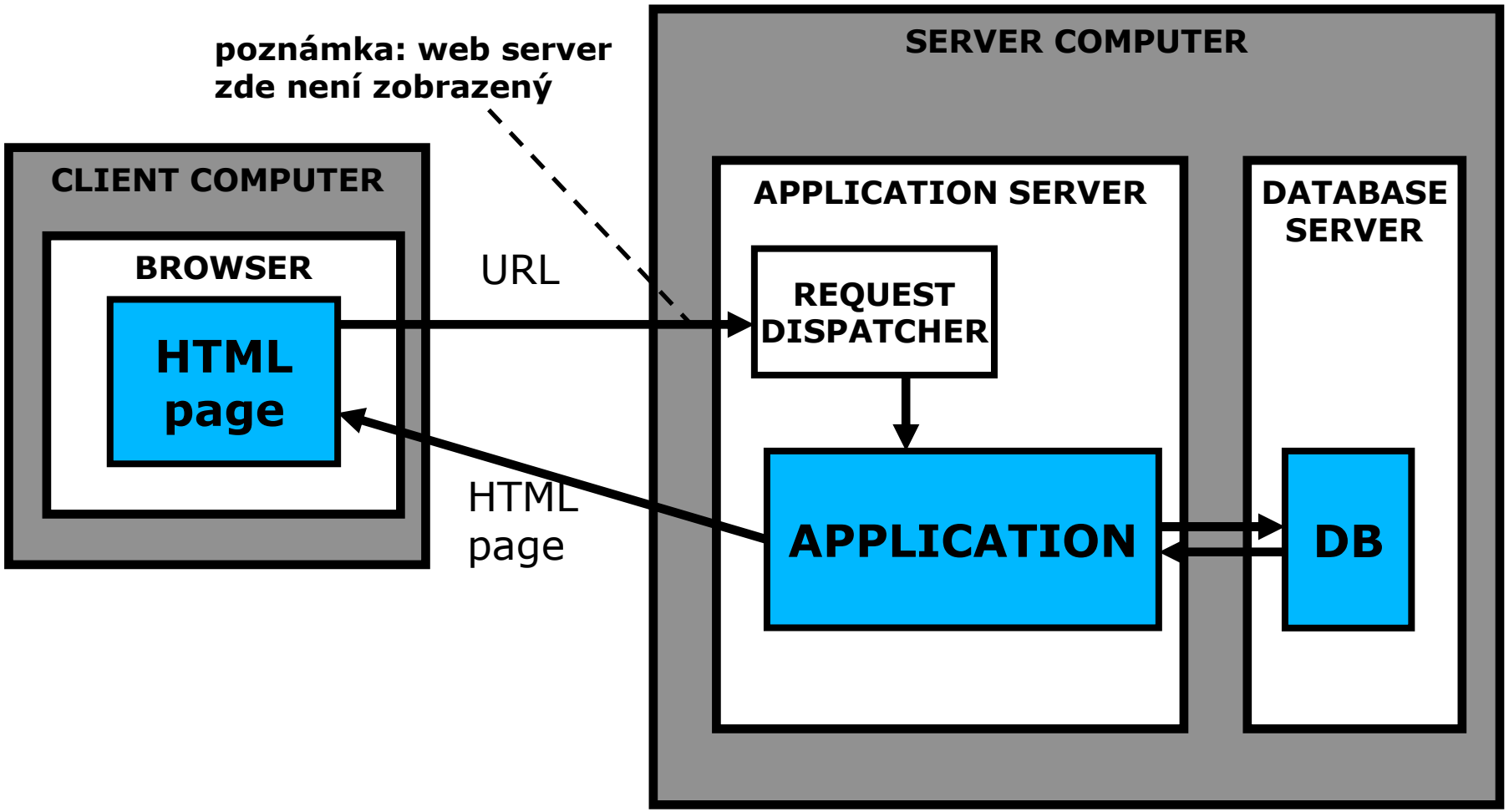
JSP = Java Server Pages

- Syntaxe podobná PHP, ale v Javě
- Logika ale stejná jako Servlet
- JSP stránka je při prvním zavolání přeložena na servlet
- Knihovny značek
- Vkládání aplikační logiky ve formě Beans
  - Beans jsou objekty

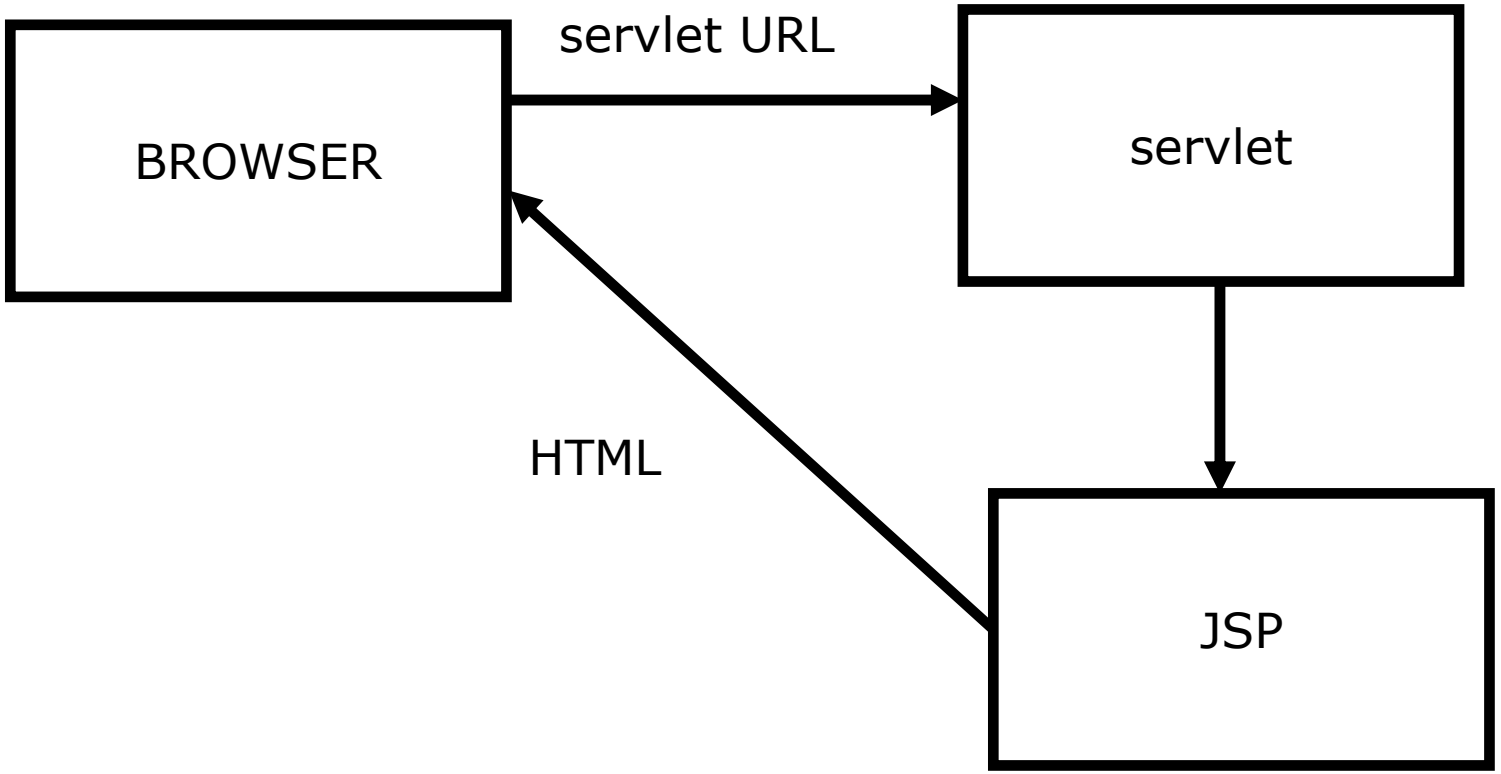
JSP ukázka

```
<HTML>
 <HEAD>
  <TITLE>Hello World</TITLE>
 </HEAD>
 <BODY>
  <H1>Hello World</H1>
  Dnes je: <%= new java.util.Date().toString() %>
 </BODY>
</HTML>
```

Schéma aplikace v aplikačním serveru



Propojení JSP a servletu ... schéma



Různé základní architektury nad J2EE

- Pomocí servletů a JSP je možné vytvořit několik různých modelů aplikace

Před příchodem JSP:

- Servlet, který generuje HTML kód - skládá HTML jako textový výstup
- Zastaralé, stále však na něm existují produkční aplikace

Model 1:

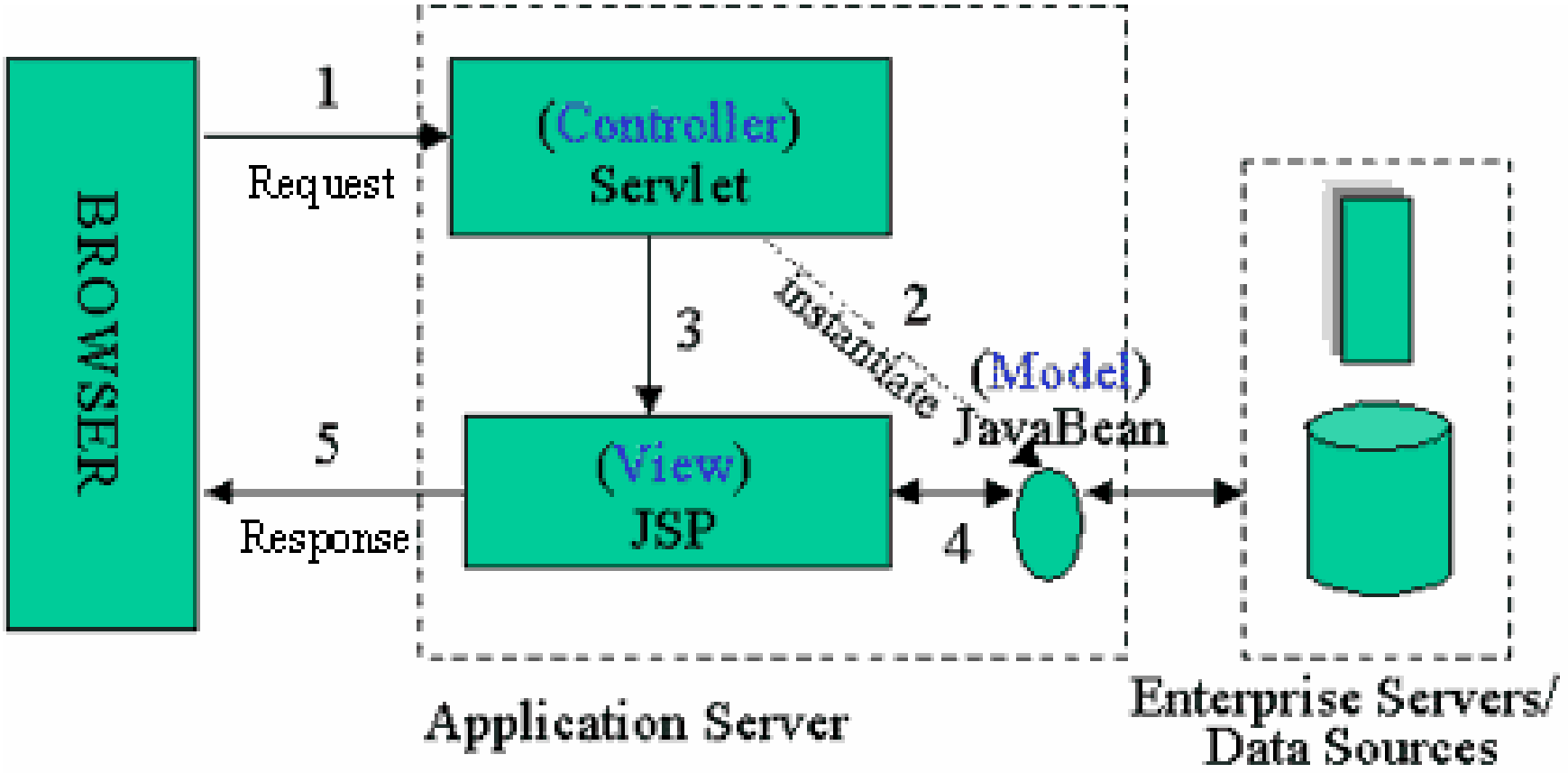
- JSP, ze kterého jsou volána EJB nebo přímo kód pro přístup do databáze
- Tok obrazovek je zajištěn přímo voláním JSP.
- V principu velmi podobný model, jako PHP nebo ASP
- Zastaralý, opouští se

Různé základní architektury nad J2EE

Model 2 - MVC:

- Kombinace Servletu a JSP. Servlet zajišťuje perzistentní kód.
- Tok obrazovek je určován v servletu.
  - **Model** ... vlastní aplikační logika, volání zajišťuje servlet
  - **View**... samotná JSP na která přesměrovává Controller (sled obrazovek který do prohlížeče produkuje aplikace)
  - **Controller** ... logika toku stránek, zajišťuje servlet
- Podstatnou výhodou je možnost načtení dat z databáze do paměti aplikačního serveru – každá JSP stránka nemusí pokaždé přistupovat k databázi

Model 2 - MVC


BROWSER
1
Request
(Controller)
Servlet
2
instantiate
3
(Model)
JavaBean
5
Response
(View)
JSP
4
Application Server
Enterprise Servers/
Data Sources

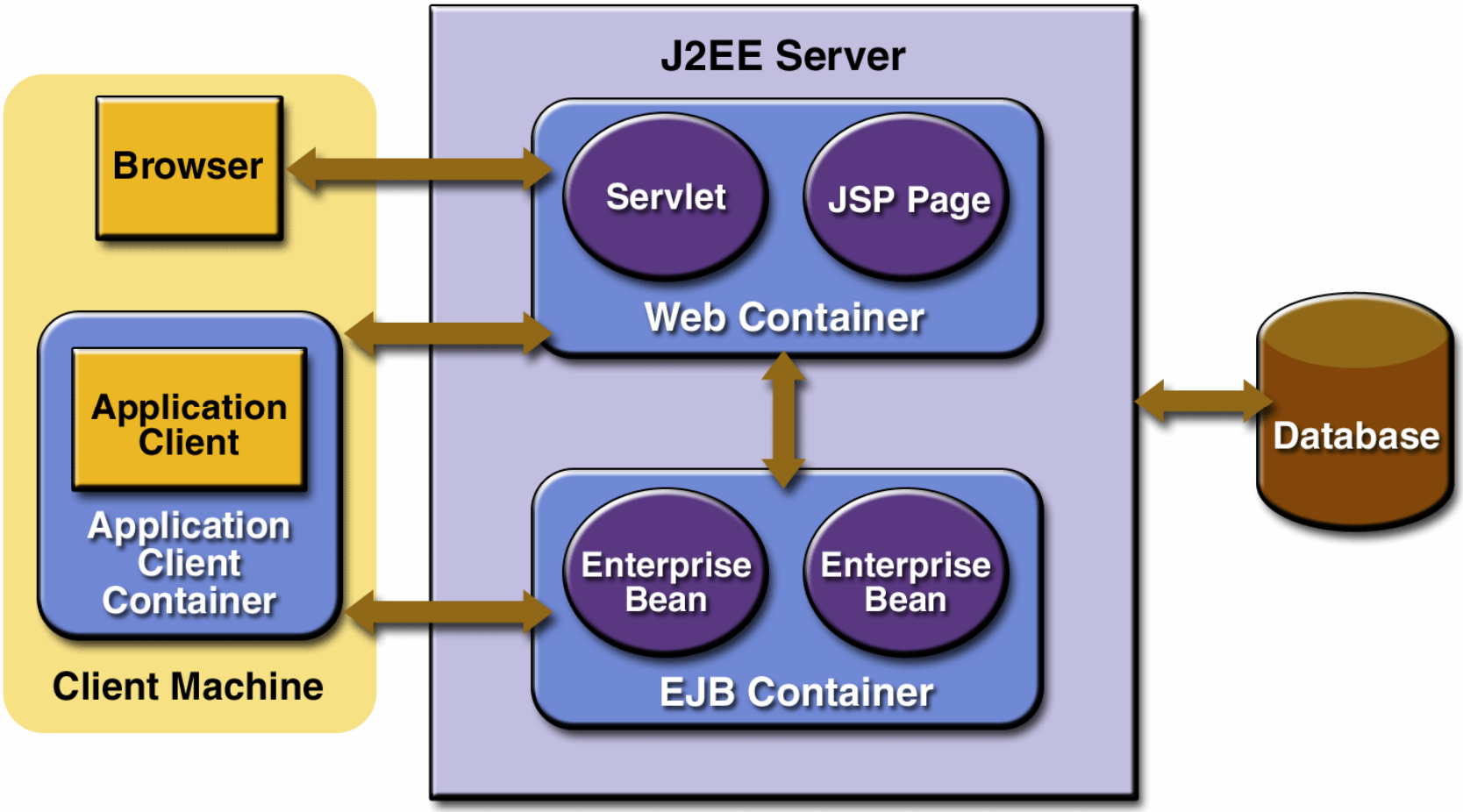
EJB = Enterprise Java Bean

- Jsou to komponenty, které běží v "Containeru"
- Containery poskytují služby:
  - Load/Initialize
  - Transakce
  - Perzistence
  - Komunikace s EJB klienty

EJB – typy beans

- Entity Bean
  - Odpovídá nějakému objektu v DB, typicky řádku
  - Zajišťuje perzistenci objektu
  - Zajišťuje abstraktní vrstvu pro práci s danou entitou
- Session Bean
  - Dělá nějakou práci = zajišťuje aplikační logiku
  - 2 typy:
  - Stateful Session Bean
    - Váže se ke konkrétnímu klientovi a pamatuje si svůj stav
  - Stateless
    - Nic si nepamatuje, jenom vykoná práci na základě požadavku

J2EE Aplikace – z pohledu EJB



http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Oveview3.html

Scaling

- Rozmístění aplikace na několik fyzických serverů (strojů) nebo procesorů ... cluster (svazek).

Horizontal scaling

- distribuce na 2 fyzické stroje
- vyšší výkon
- zajištění proti havárii stroje

Vertical scaling

- distribuce na víc procesorů v rámci jednoho stroje
- vyšší výkon

- Děkuji za pozornost

- **Další materiály**

Java Servlet

- Servlet běží jako perzistentní kód na aplikačním serveru.
- Jedná se o třídu v Javě podle API `javax.servlet.` Kód servletu dědí z abstraktní třídy HTTPServlet. Výsledný kód který se spouští ne severu je .class (bytecode).
- Jako request se nevolá přímo PHP nebo ASP skript, ale „abstraktní“ url, které je pak namapováno na konkrétní servlet.
- Toto namapování se provádí v konfiguračním souboru (pro Tomcat web.xml).

Příklad kódu HTML stránky která volá servlet

```
<html>
  <head>
    <title>Pokus se servletem...</title>
  </head>

  <body>
    <form action="MujServlet" method="post">
      Zadejte text:<br>
      <textarea name="txtvstup" rows="5"
            cols="50"></textarea><br>
      <input type="submit" value="Odeslat">
    </form>
  </body>
</html>
```

Namapování servletu na URL

Příklad namapování servletu „com.servlety.TestServlet.java“ na URL začínající na „/Pokusy/“ přes virtuální jméno „MujServlet“.

```
<web-app>
  <servlet>
    <servlet-name>MujServlet</servlet-name>
      <servlet-class>
         com.servlety.TestServlet
      </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MujServlet</servlet-name>
    <url-pattern>/Pokusy/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Kód servletu „com.servlety.TestServlet.java“:

```
package com.servlety;

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

public class TestServlet extends HttpServlet {

  // vyvolá se na základě POST požadavku (náš formulář)
  public void doPost(
                  HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
```

Kód servletu „com.servlety.TestServlet.java“:

```
 //text z formuláře
    String text = request.getParameter(“txtvstup”);
    PrintWriter out = response.getWriter();

    out.print(“<!DOCTYPE HTML PUBLIC”);
    out.println(“\"-//W3C//DTD HTML 4.0
       Transitional//EN\">”);
    out.println(“<html><head><title>Počet
       znaků</title></head>”);
    out.println(“<body><h1>Počet znaků:” +
       text.length());
    out.println(“</h1></body></html>”);
  }
}
```

Co se stane na serveru:

- V prohlížečí máme HTML stránku, která se vrátila po URL začínajícím na „/Pokus/“ (viz výpis 1).
- Na ní je textové pole a tlačítko „Odeslat“. Pokud zmáčkneme toto tlačítko, na server se pošle požadavek.
- Ten se přes konfiguraci (viz výpis 2) namapuje na servlet „com.servlety.TestServlet.java“ (viz výpis 3).
- Servlet se spustí. Pomocí „request.getParameter(“txtvstup”)“ získáme formulářová data ze stránky.
- Výstupní stream Javy poté nastavíme tak, abychom zapisovali do objektu „response“.
- Textový výstup ze servletu tak vytvoří novou HTML stránku která se odešle klientovi do prohlížeče.

Základní objekty a metody servletu

Základní metody Servletu:

- service() ... default, programátor jí může předefinovat, rozpozná, jaký požadavek přijde a na jejich základě volá:
- doGet() ... obsluhuje požadavek GET
- doPost() ... obsluhuje požadavek POST
- ...

V parametrech metod se předají objekty:

- request ... obsahuje parametry, hlavičku... z URL
- response ... slouží pro zápis výstupu ze servletu – odezvy na request

Java Server Pages (JSP)

- Novější technologie než servlet
- JSP je HTML do kterého jsou vložené skriptlety (značky pro vkládání kódu do HTML) <% %> a <%= %>. Kód v skripletu je Java.
- JSP se při prvním požadavku na jeho spuštění kompiluje na servlet (.class), ten pak produkuje HTML kód.
- Základní objekty v JSP – request a response ... globální v rámci JSP. Jejich použití je stejné jako u servletu.

Jednoduchý příklad JSP stránky

Počítání znaků v řetězci, který JSP stránky dostane v parametru „txt“:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<html>
   <head><title>Počet znaků</title></head>
   <body>
     <% String delkaStr = request.getParameter(“txt”); %>
      <h1>Počet znaků: <%=delkaStr.length()%></h1>
    </body>
</html>
```