X33EJA – Enterprise Java

Petr Šlechta
Sun Microsystems
petr.slechta@sun.com

Petr Aubrecht
CA (Computer Associates)
petr.aubrecht@ca.com

X33EJA (2+2) – Cvičení

- Formou samostatné práce na projektu
  - témata budou zadána 5. týden semestru
- Konzultace přes e-mail nebo po přednášce
- Klasifikovaný zápočet
  - na základě předvedení projektu a diskuse nad ním

X33EJA (2+2) – Přednášky

1. Úvod, organizace přednášek a cvičení
2. Přehled technologií J2EE, jejich společný kontext a vzájemné souvislosti, architektura J2EE systémů
6. Javové technologie pro webové aplikace, servlety, Java Server Pages (JSP)
3. Objektově-relační mapování, entity beans, Java Persistence API (JPA).
4. Session beans, transakční model
5. Aparát pro zasílání zpráv, message-driven beans, Java Message Service (JMS)
7. Přehled hlavních webových frameworků, Java server Faces (JSF), vybrané knihovny pro podporu JSF

X33EJA (2+2) – Přednášky

8. Web services (XML, SOAP, WSDL)
9. Související podpůrné javové technologie (JNDI, JTA, JTS, JCA)
10. Demonstrace návrhu a implementace jednoduché aplikace
11. Deplyment aplikací, konfigurační management J2EE systémů
12. Výkonové aspekty J2EE systémů, load balancing
13. Některé implementační techniky (proxy, connection pool, bean pool)
14. Současné trendy ve vývoji J2EE aplikací

Úvod do Java Enterprise technologií

Java

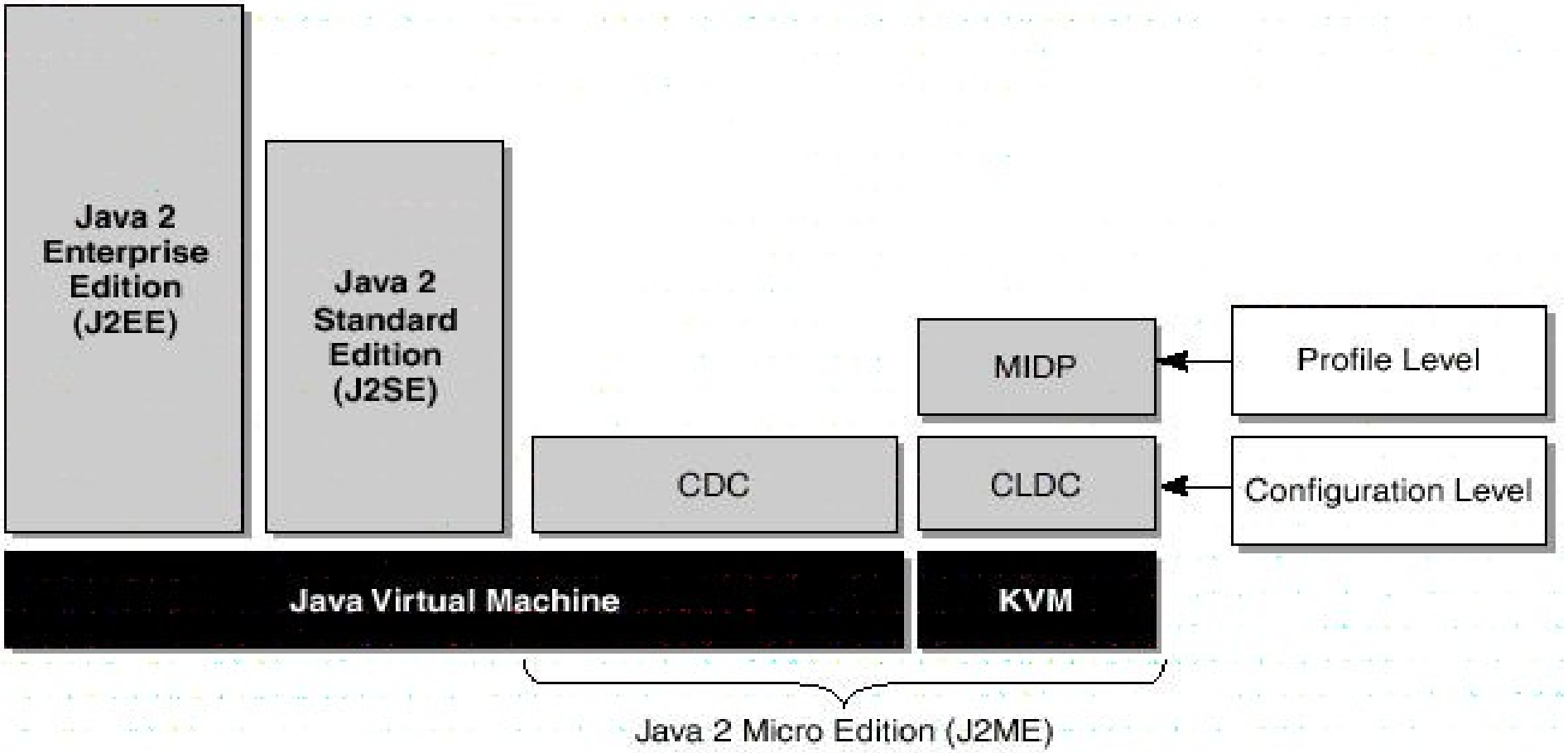
- Programovací jazyk
  - Jednoduchost, navržen s ohledem na malá zařízení
  - Přenositelnost (byte code)
  - Veřejné specifikace (JCP)
  - Implementace a podpora více firmami
    - Sun, IBM, ...
- Průřez historií
  - 1995 verze 1.0, 1998 Java 2 (J2SE 1.2, J2ME, J2EE)
  - 2006 GPLv2, 2007 free and open-source (problém s Java 2D)

Java Editions

- JavaCard
- Java ME
  - CLDC (pagers, mobile phones), CDC
- Java SE
  - Java 1.x, J2SE 1.2 & 1.3 & 1.4, Java SE 5 & 6 & (7)
- Java EE
  - J2EE 1.2 & 1.3 & 1.4, Java EE 5 & (6)
- Jazyk stejný napříč edicemi
- Edice se liší hlavně knihovnami a podporovanými technologiemi

Java Editions

Figure 1–1 The various Java editions

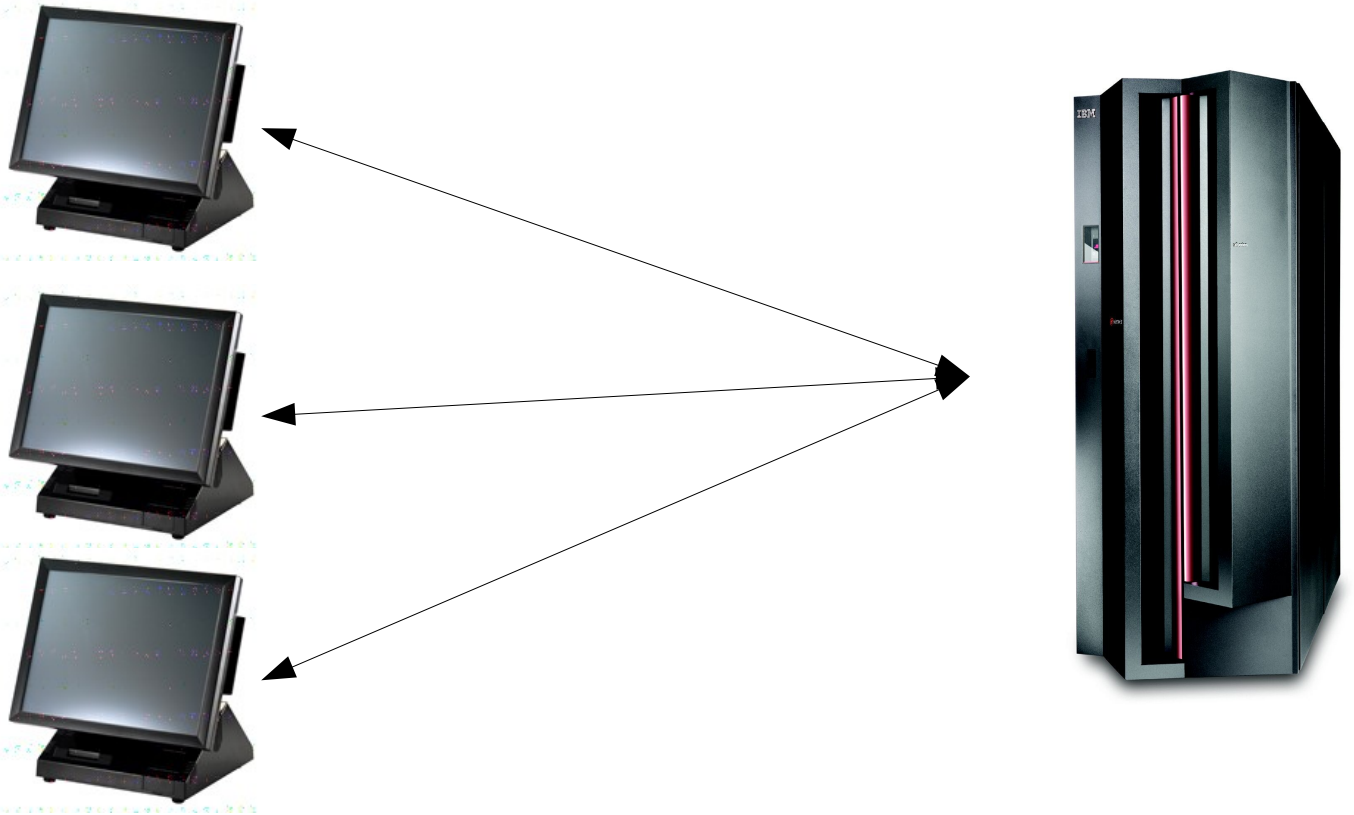


Enterprise Applications

- Velké objemy dat – databáze
- Paralelní přístup mnoha uživatelů
- Client – server architektura
  - Mainframes
  - 2 vrstvy
  - 3 vrstvy
  - ...

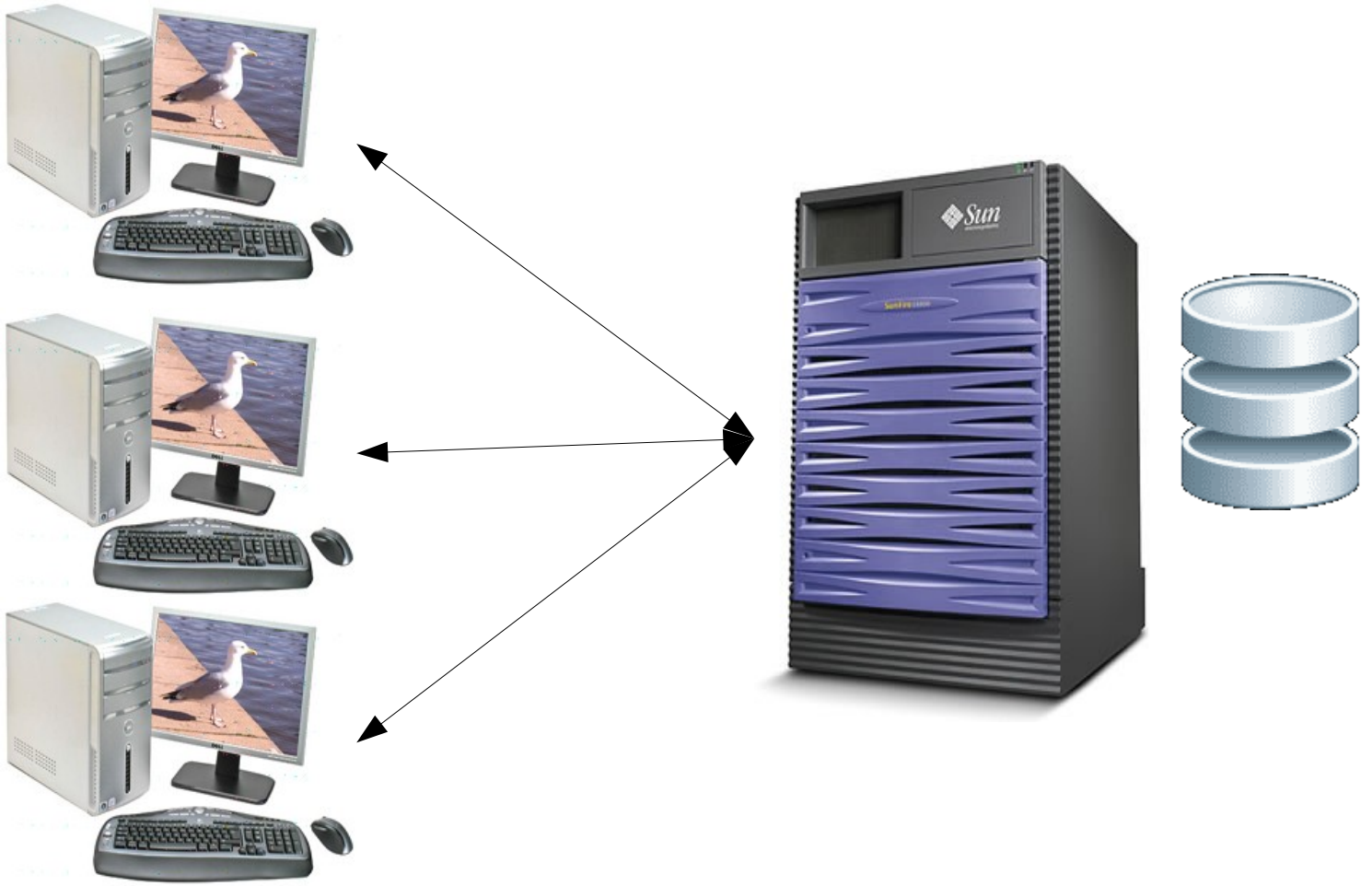
Mainframe

- Centralizovaný model
  - Jednoduché terminály připojené k hlavnímu počítači
  - Jednoduchá správa, velké datové přenosy



2 vrstvy

- Klienti připojeni přímo k databáze
  - Interaktivní, složitá správa, velké datové přenosy



3 vrstvy

- Databázový server, aplikační server(y) a klienti
  - Klient tenký nebo tlustý

3 vrstvy

- Databáze (popř. legacy systems)
  - Uložení dat
- Aplikační server
  - Business logika
- Klient
  - Visualizace výsledků, zadávání dat
  - Tenký: webový prohlížeč (bez speciální instalace)
  - Tlustý: aplikace (větší interaktivita, kontola dat)
- Jeden z prvních třívrstvých systémů byl SAP

Java EE

- Java EE je soubor technologií integrovaný v Java EE aplikačním serveru
  - Specifikace, více implementací
- Open-source implementace
  - GlassFish (Sun), JBoss (Red Hat), Apache Geronimo, JonAS
- Komerční implementace
  - Sun Java AS, IBM WebSphere, BEA WebLogic
- Kritéria
  - Open source, certifikace, škálovatelnost, konfigurace, ...

Aplikační Server

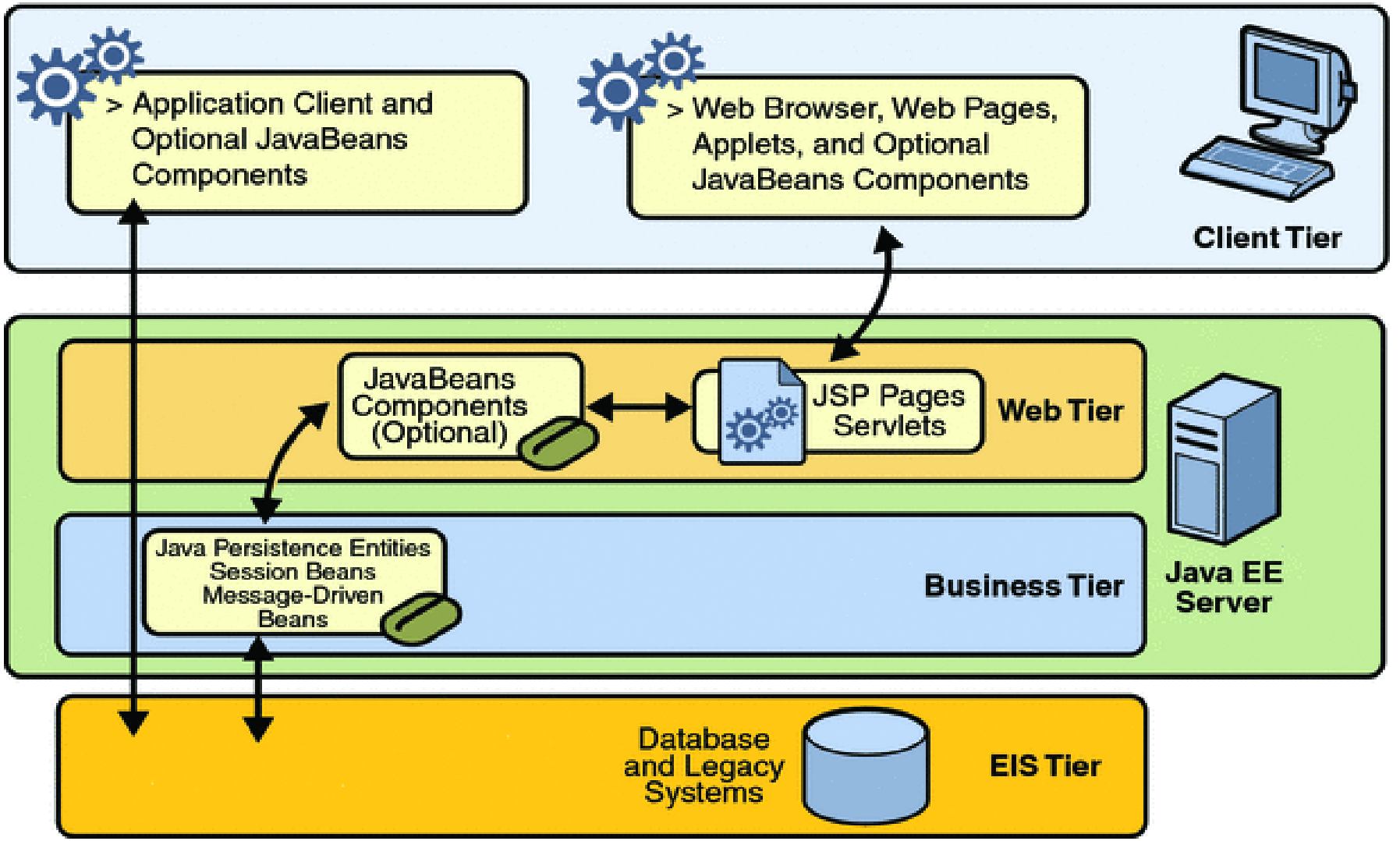
- Integrované Java EE API a technologie
- Unifikovaný management a deployment enterprise aplikací
- “Framework” zjednodušující tvorbu serverových aplikací (deklarativně či programově)
  - bezpečnost (security), transkační zpracování, sdílení zdrojů (pooling), synchronizovaný přístup ke komponentám, podpora persistence, distribuovatelné aplikace (local/remote access), škálovatelnost (jedna konfigurace, load balancing)

Aplikační Server – ukázka

- Administrativní konzola, spuštění
  - CLI
    - asadmin: start-appserv, stop-appserv, autodeploy
  - NetBeans IDE
    - registrace, start, stop, ...
- Deployment aplikace
  - HelloWorld.war

Struktura Aplikačního Serveru

- Dva základní kontejnery: web & business



Servlety

Cíl: Vytvořit jednoduchou web aplikaci

- Tenký klient (webový prohlížeč)
- AS: pouze web kontejner
  - Servlety
  - JSP stránky
- Bez business logiky a persistence
- Viz též
  - JSP
  - Web Frameworks

Dokumentace

- Java EE 5 API
  - http://java.sun.com/javaee/5/docs/api/
- NetBeans již tuto dokumentaci obsahují
  - http://www.netbeans.org/

Servlet

- Objekt spravovaný (managed) web kontejnerem
- Obecný servlet (javax.servlet.*)
- HTTP servlet (javax.servlet.http.*)
- Odstíněn od některých detailů komunikace (listenery, cache) a HTTP protokolu (bezestavový)
- Na základě vstupních informací (HttpServletRequest) generuje výstupní informace (HttpServletResponse, HTML stránku)

Mapování servletů

- Konfigurační data pro servlety (a ostatní komponenty web aplikace) jsou uloženy v soubory web.xml (descriptor)
- Mapování servletů na URL
  - Aplikace sama má kontext
    - http://localhost:8080/**HelloWorld**
  - V rámci web aplikace je nutno určit, které dotazy budou zpracovány daným servletem
    - http://localhost:8080/HelloWorld/**SayHello**

WAR soubor

- Celá web aplikace je zabalena do WAR (Web Archive) souboru
- WAR = JAR (Java ARchive) s určitou strukturou (některé položky jsou stejné jako v JAR souboru)
  - JAR = ZIP s určitou struktorou
- WAR může obsahovat Java kód, JSP, HTML, resources, konfigurační informace (descriptors), ...

Ukázka – vstup a výstup

- Servlet provede
  - Zpracování vstupních parametrů
  - Zápis do logu
  - Generování výstupní stránky
- Ukázka ladění na úrovni HTTP protokolu

Lifecycle servletu

- AS má obvykle více threadů zpracovávajících požadavky klientů
  - Viz konfigurace AS
- Lifecycle: init(), service(), service(), ..., service(), destroy()
- Kód servletu není synchronizován (!)
  - Metoda service() může být prováděna několika thready najednou
  - Nutno psát reentrantní kód (pouze s lokálními proměnnými) a synchronizovat všechny přístupy ke sdíleným zdrojům

Uchování kontextu

- HTTP je bezestavový protokol
- Kontext lze uchovat na klientu či na serveru
  - Klient: cookies
  - Server: sessions
- Různé rozsahy (scopes)
  - Application (javax.servlet.ServletContext)
  - Session (javax.servlet.http.HttpSession)
  - Page (javax.servlet.jsp.PageContext)

Ukázka – cookies

- Využití API
- HTTP ladění

Konfigurace servletu

- web.xml může obsahovat konfigurační data pro web aplikaci (app scope)
  - Java EE web aplikace by neměly být závislé na externích zdrojích (soubory, atd.)
  - Měly by být nezávislé na OS a AS

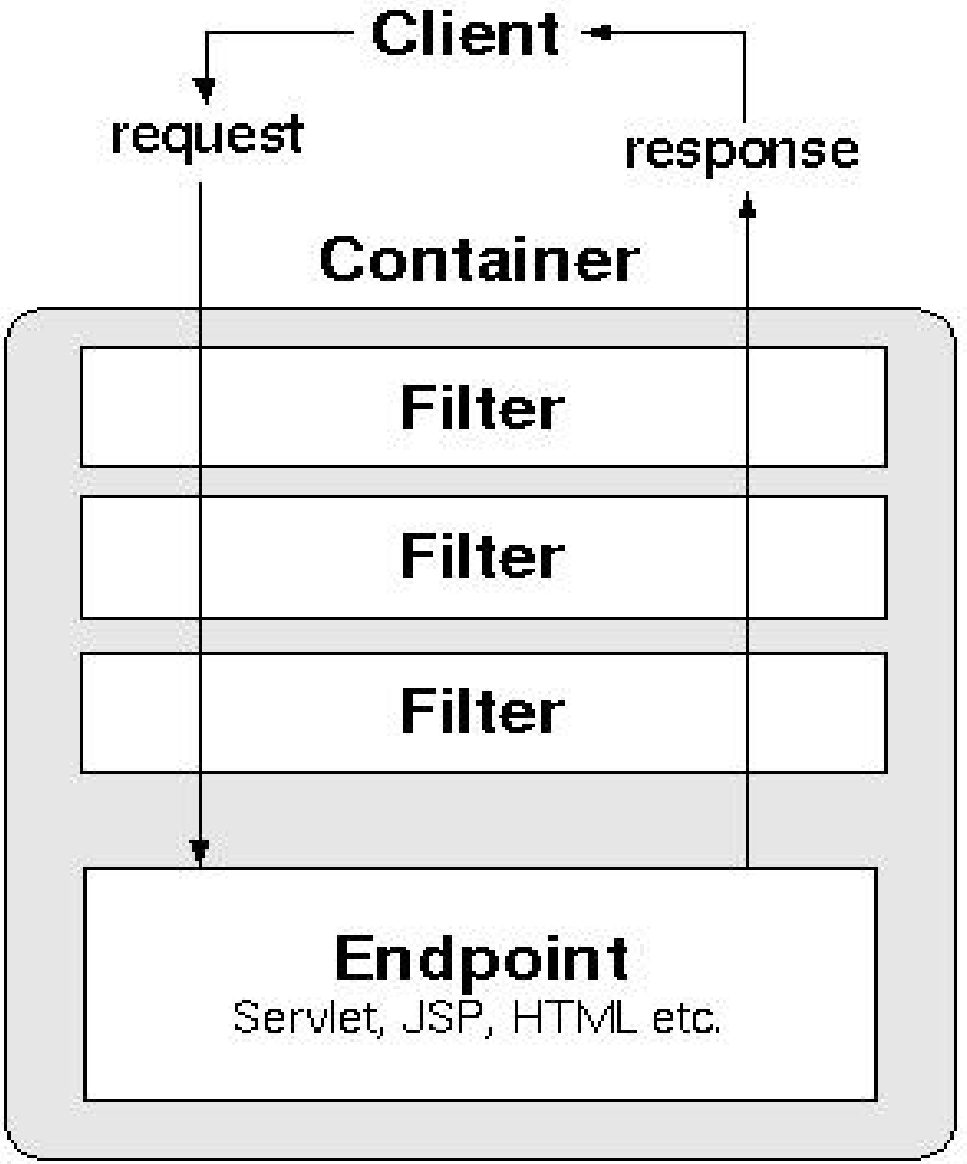
Ukázka

- Zpřístupnění souborů přes web
  - Různí uživatelé, různé sessions
  - Volba jak zakódovat parametry do URL (REST styl vs. klasický styl)
  - Konfigurace aplikace

Filtry (filters)

- Umožňují vstoupit mezi klienta a servlet a změnit data
  - Komprese, kódování obrázků, ...
  - Pozměňování výstupu servletu (logo, ...)
  - Bezpečnost (odmítnutí přístupu)
  - Integrace web aplikací (SSO)
  - ...
- Konfigurace filtru pomocí web.xml (záleží na pořadí položek)
- Řetězení filtrů do filtrovacího řetězce (filter chain)

Řetězení filtrů



Implementace filtru

- doFilter(...) metoda
- Návrat z metody = prázdná odpověď
- Lze přesměrovat klienta na jiný zdroj (redirect)
- Pomocí “zabalení” (wrapping) request a response objektů lze dosáhnout pozměnění vstupů či výstupů pro další filtr v řetězci (HttpServletRequestWrapper a HttpServletResponseWrapper)
- Ukázka: kontrola přístupu k informacím podle přihlášení uživatele

Přesměrování a vložení zdroje

- HttpServletResponse.sendRedirect(String location)
- Pomocí RequestDispatcher (metody forward a include):

```
RequestDispatcher dispatcher =
   session.getServletContext().getRequestDispatcher("/banner");

if (dispatcher != null)
   dispatcher.include(request, response);
```

Listeners

- Umožňují reagovat na změnu v lifecycle servletu či session
  - ServletContextListener
  - ServletContextAttributeListener
  - ServletRequestListener
  - ServletRequestAttributeListener
  - HttpSessionListener
  - HttpSessionActivationListener
  - HttpSessionAttributeListener
  - HttpSessionBindingListener

Ošetření chyb

- Web kontejner generuje standardní stránku v případě chyby
- Lze definovat chybovou stránku pro danou vyjímku
  - Ve web.xml, tag <error-page>

SingleThreadModel

- SingleThreadModel Interface pro servlety se stavem
  - lépe je použít synchronize blok
  - SingleThreadModel může způsobovat potíže s výkoností AS