

# Java Web–technológiák

## Bevezetés

# Áttekintés

- ▶ Model–View–Controller (MVC) elv
- ▶ J2EE
- ▶ Java alapú Web–alkalmazások

# MVC Model-View-Controller

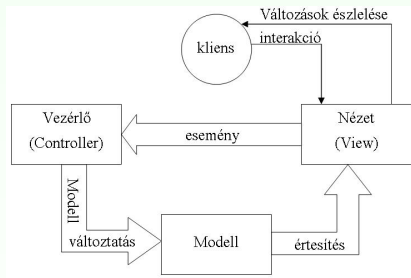
## eredete:

- ▶ kezdetben a SmallTalk OO programzási nyelvhez lett kifejlesztve
- ▶ eredetileg desktop alkalmazásoknál használták

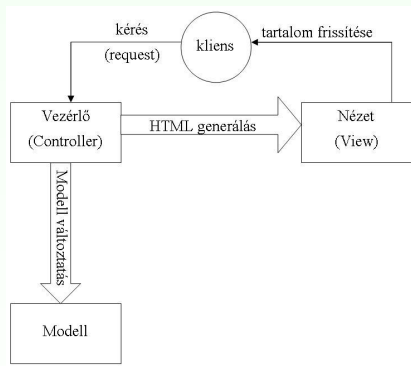
## lényege szétválasztani a következő részeket:

- ▶ üzleti adatokat tartalmazó objektumok, illetve a rajtuk végezhető műveletek (**model**)
  - ▶ megjelenítés (**view**)
  - ▶ vezérlő logika (**controller**)
- 
- ▶ jelenleg a legtöbb web-keretrendszer is a Model-View-Controller (MVC) mintát követi

# MVC desktop illetve webalkalmazás esetén



MVC – desktop alkalmazás esetén



MVC – Web-alkalmazás esetén

# MVC + Java web-technológiák

## MVC Java web-technológiák esetén:

- ▶ **Modell:** tipikusan JavaBean-ek alkotják + a rajtuk végzett műveletek (üzleti logika kód)
- ▶ **Nézet:** tipikusan Java Server Page-ek (JSP) képviselik (de más megjelenítési technológiák is felhasználhatók, pl. Velocity, FreeMarker, JasperReports, stb.)
- ▶ **Vezérlés:** tipikusan egy servlet, amely
  - ▶ fogadja a kérés (request) paramétereit
  - ▶ átalakítja a paramétereit a megfelelő típusra
  - ▶ ellenőrzi a paraméterek helyességét
  - ▶ meghívja a megfelelő üzleti logika metódusokat
  - ▶ az előbbi lépések eredményei alapján továbbít a megfelelő nézethez

## Az MVC elv működése Java web-alkalmazások esetén a következőképpen vázolható:

1. A kliens (web böngésző) kérést intéz a web szerverhez (GET vagy POST)
2. A webszerver a kérést (egy konfigurációs állomány alapján) a vezérlésért felelős servlethez irányítja
3. A vezérlő szervlet (diszpécser) a kérés URL alapján meghívja a megfelelő parancs-objektumot
4. A parancsobjektum
  - ▶ meghívja az üzleti logikát (itt történik pl. kommunikáció az adatbázissal vagy web-szolgáltatással), majd
  - ▶ átírányít a megfelelő nézetre (pl. JSP).
5. A nézet (pl. JSP) megjeleníti az üzleti objektumokat

# J2EE

- ▶ a Java alapú Web–alkalmazások tipikus alkotóelemei:
  - ▶ servlet (vezérlés)
  - ▶ JSP (nézet)
- ▶ ezek a komponensek a J2EE specifikáció szerves részei
  
- ▶ a továbbiakban áttekintést nyújtunk a J2EE technológiákról

# J2EE

## J2EE alkalmazás elkészítésének fázisai:

- ▶ tervezése (design)
- ▶ fejlesztése (development)
- ▶ összeállítása (**assembly**)
- ▶ telepítése (**deployment**)

## A Java2Enterprise platform az alábbiakat kínálja:

- ▶ alkalmazások *komponens alapú* fejlesztése – újrafelhasználható komponensek
- ▶ többretegű osztott alkalmazásmodell
- ▶ egységesített biztonsági modell
- ▶ XML alapú web-szolgáltatások (web service)
- ▶ platformfüggetlenség
- ▶ szerverfüggetlenség



# Többrétegű alkalmazásmodell

**Rétegek (tier)** - szétválasztják az alkalmazást logikai egységekre (mindegyiknek külön feladata van)

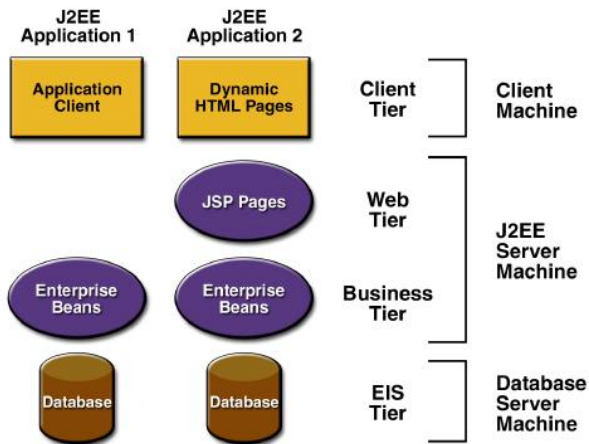
A következő rétegeket különböztetjük meg:

- ▶ **Kliens (ügyfél)-réteg:** A kliens gépen futó komponensek (pl. applet, swing alkalmazás)
- ▶ **Web-réteg:** a J2EE szerver web-konténerében futó komponensek (szervlet, jsp)
- ▶ **Üzleti logikai-réteg:** a J2EE szerver EJB konténerében futó komponensek (ejb)
- ▶ **Vállalati információs réteg** (Enterprise Information System (EIS)): EIS szerveren futó szoftver (pl. adatbázisszerver)

A J2EE alkalmazásokat általában három rétegűnek tekintik:

- ▶ kliens gép (felhasználói felületet biztosít), J2EE szerver (az üzleti logikát tartalmazza) és adatbázis szerver

# Többrétegű alkalmazások



Többrétegű alkalmazások [J2EE Tutorial]

# Web kliens

## a web böngésző:

- ▶ megmutatja a szerverről érkező oldalakat:
  - ▶ HTML, XML stb. alapú dinamikus weblapok – a web-rétegen futó komponensek generálják őket

## web kliensek – ún. könnyű (thin) kliensek

- ▶ nem kérdeznek le adatbázisokat, nem végeznek komplex üzleti műveleteket
- ▶ ezek a “nehéz” műveletek a J2EE szerveren hajtódnak végre (hasznosítva a szerveroldali technológiák gyorsaságát, biztonságát, megbízhatóságát)

# Alkalmazáskliensek (desktop kliens)

## Alkalmazáskliensek:

A kliens gépen futnak és egy komplexebb felhasználói felületet biztosítanak a html, xml alapú nyelvekhez képest

- ▶ leginkább jellemző egy Swing vagy AWT alapú grafikus felhasználói felület
- ▶ lehet akár parancssor alapú felület is

Ezek a kliensek már közvetlenebbül kapcsolódnak az alkalmazáslogikához és több funkcionalitást tartalmazhatnak.

# JavaBean-ek

## Az alábbiakban játszanak szerepet:

- ▶ a J2EE komponensek és az adatbázisréteg közötti adatcserében
- ▶ web-alkalmazás esetén a nézet a standard web-es névterekbe helyezett bean-eket mutatja meg

A JavaBean-ek nem részei a J2EE specifikációnak.

Az implementált JavaBean osztályok meg kell feleljenek a JavaBean specifikációnak:

- ▶ get/set metódusok a tulajdonságok lekérdezésére, beállítására,
- ▶ paraméter nélküli publikus konstruktor,
- ▶ szerializálható kell legyen, stb.

# J2EE komponensek

## J2EE komponens:

egy **önálló funkcionalitású** szoftverrész, amely a hozzá tartozó Java osztály- és erőforrásfájlokkal egy alkalmazás keretében van **telepítve**, és más komponensekkel **kommunikál**

## A következő J2EE komponensek vannak definiálva:

- ▶ **Servletek** és **JSP** (JavaServer Pages): web komponensek, melyek a J2EE szerver Web-konténerében (pl. Tomcat) futnak
- ▶ **EJB-k** (Enterprise JavaBeans): üzleti logika komponensek, melyek a J2EE szerver ejb-konténerében (pl. Weblogic, Websphere, JBoss) futnak

## néhány jellemző:

- ▶ a J2EE komponensek Java nyelven íródnak
- ▶ fordításuk bármilyen más java programéhoz hasonlóan történik

## egy J2EE komponens és egy standard Java osztály közti különbség:

- ▶ a J2EE komponensek meg kell feleljenek az illető típusú komponensre vonatkozó J2EE specifikációnak
- ▶ a komponenseket össze kell rakni egy J2EE alkalmazásba
- ▶ ezt követően a szerverre lesznek telepítve

# Web komponensek

## Servletek:

Java osztályok, amelyek dinamikusan dolgozzák fel a kérést (request) és építik fel a választ (response)

## JSP-k:

szöveg-alapú dokumentumok, amelyek a háttérben ugyancsak servletként futnak le, de egy természetesebb megközelítést biztosítanak a tartalom létrehozására



# Üzleti logika komponensek – Enterprise Java Bean-ek

Egy J2EE alkalmazásban az üzleti logika kódja (ami egy konkrét követelményt –use case– implementál) **EJB**-k által van megvalósítva.

## Háromféle EJB:

- ▶ **session bean**: az üzleti logikát implementálja
- ▶ **entity bean**: egy adatbázistábla egy bejegyzésének (record) felel meg. Biztosítja, hogy az adatok automatikusan le legyenek mentve az adatbázisba.
- ▶ **message-driven (üzenetvezérelt) bean**: üzenetküldést tesz lehetővé

ami a többretegű alkalmazások megírását bonyolulttá teszi. . .

komplex kód írására van szükség:

- ▶ a tranzakciók kezeléséhez,
- ▶ többszálú programozáshoz,
- ▶ hatékony erőforrás-tároláshoz (resource pooling),
- ▶ más alacsony szintű művelethez

A komponens alapú és platform-független J2EE architektúra a többretegű alkalmazások fejlesztését megkönnyíti, mivel:

- ▶ az üzleti logika újrafelhasználható komponensekbe van szervezve
- ▶ a fenti alacsony szintű műveleteket a J2EE szerverek biztosítják, ezáltal a fejlesztő a konkrét feladat (üzleti logika) megoldására koncentrálhat.

A web komponenseket vagy enterprise beaneket nem lehet egyből lefuttatni:

- ▶ össze kell állítani őket (assembly) egy J2EE modulba
- ▶ a konténerbe kell telepíteni (deploy)

### az összeállítás:

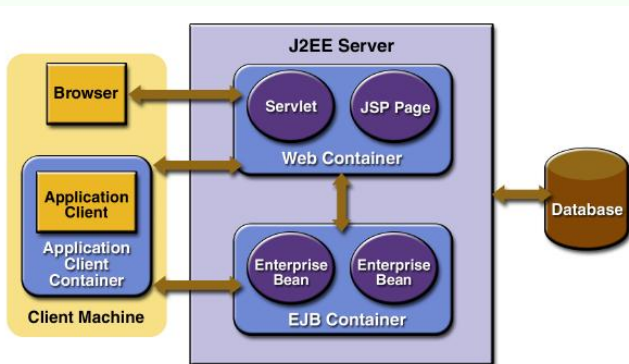
különböző konténer beállításokat feltételez – különböző konfigurációs állományokban

A J2EE szerver ezen beállítások alapján különböző szolgáltatásokat biztosít:

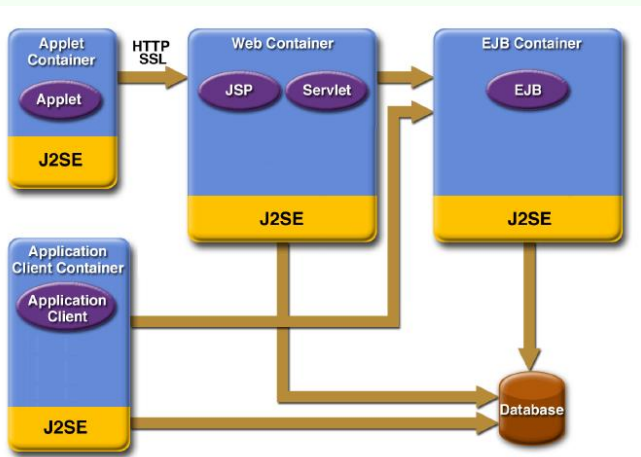
- ▶ biztonsági szolgáltatások
- ▶ tranzakciókezelés,
- ▶ JNDI,
- ▶ távoli hívások (EJB-k és kliensek között, mintha ugyanabban a JVM-ben futnának)

# Konténer típusok

egy J2EE szerver EJB és Web konténereket biztosít:



J2EE szerver és konténerek [J2EE Tutorial]



Kapcsolatok [J2EE Tutorial]

## A web-kliens és web-alkalmazás közti kapcsolat a következőképpen működik:

1. A kliens egy HTTP kérést küld a web-szervernek
2. A web-konténer (mely a Servlet és JSP technológiákat implementálja) a kérést egy HttpServletRequest objektummá alakítja.
3. Ezt az objektumot megkapja a web-komponens (servlet vagy JSP), amely JavaBean-ekkel vagy az adatbázissal együttműködve dinamikus tartalmat generál.
4. A web-komponens egy HttpServletResponse objektumot fog generálni vagy továbbadja (forward) a kérés objektumot egy másik web-komponensnek.
5. A web-szerver ezt az objektumot egy HTTP válaszá (response) alakítja és visszaküldi a kliensnek.

## Servletek:

Java osztályok, melyek dinamikusan dolgozzák fel a kérést és hozzák létre a választ.

## JSP-k:

szöveg alapú dokumentumok, melyek szintén servletként futnak le, de a statikus tartalmat (HTML, XML) jóval egyszerűbben lehet létrehozni a segítségükkel.

- ▶ elvileg – a servletek és JSP-k felcserélhetőek
- ▶ gyakorlatilag – minkettőnek megvan az erőssége és a gyengéje

# összehasonlítás

## A servletek alkalmasabbak:

- ▶ kontrollerként egy web–alkalmazásnál (pl. kérestovábbítás)
- ▶ nem szöveg alapú adatok feldolgozására

## A JSP-k alkalmasabbak:

- ▶ szöveg alapú (HTML, SVG, WML, XML) oldalak létrehozására

A servlet és JSP technológiák bevezetése óta újabb Java technológiák fejlődtek ki, amelyek az előbbi kettőn alapulnak. Pl. JSF, JSP elemkönyvtárak (pl. JSTL)



## Egy web-alkalmazás a következő komponensekből áll:

- ▶ web-komponensek
- ▶ telepítés-leíró (deployment descriptor)
- ▶ statikus erőforrások (pl. képek, statikus HTML oldalak)
- ▶ Java osztályok
- ▶ jar programcsomagok (libraries)

## Egy web-alkalmazás létrehozása és futtatása a következő lépésekből áll:

- ▶ a web komponensek (servlet, JSP) illetve segédosztályok megírása
- ▶ a telepítés-leíró (deployment descriptor) létrehozása
- ▶ a komponens osztályok illetve az ezekből hivatkozott segédosztályok lefordítása, illetve a megfelelő struktúrájú web-alkalmazás összeállítása (build-tool segítségével:pl. Ant)
- ▶ [opcionálisan] összecsomagolni az alkalmazást egy telepíthető egységbe (deployable unit). (.war állomány)
- ▶ telepíteni az alkalmazást a web-konténerbe
- ▶ böngészőből meghívni a megfelelő URL-ket, melyek a web-alkalmazásra hivatkoznak

## Egy web modul sajátos struktúrával rendelkezik:

a gyökérben található a

- ▶ JSP oldalak
- ▶ statikus web erőforrások

szintén a gyökér tartalmaz egy WEB-INF nevű katalógust, amely a következőket tartalmazza:

- ▶ web.xml : a webalkalmazás telepítés-leírója (deployment descriptor)
- ▶ classes katalógus: szerver-oldali osztályok: servletek, segédosztályok és JavaBeans komponensek
- ▶ lib katalógus: jar csomagok, melyeket a szerver-oldali osztályok hívnak meg

Egy web modul telepíthető

- ▶ nem csomagolt formában
- ▶ csomagolt formában (.war állomány)

A .war állomány egy speciális struktúrájú .jar állomány.

Egy web-alkalmazás **szerverfüggetlen**: bármilyen web-konténerbe telepíthető, amelyik megfelel a Java Servlet/JSP specifikációnak.

# Hasznos web-címek

## Néhány hasznos web-cím:

- ▶ [www.java.sun.com](http://www.java.sun.com) – A Sun hivatalos, Java-val kapcsolatos weboldala
- ▶ [www.apache.org](http://www.apache.org) – Hasznos programcsomagok, keretrendszerek, API implementációk (pl. Tomcat)
- ▶ [www.eclipse.org](http://www.eclipse.org) – Az Eclipse fejlesztői környezet hivatalos honlapja
- ▶ [www.google.com](http://www.google.com) ... :)

# Fejlesztői környezet – pl.

## Az alábbi eszközöket fogjuk használni:

- ▶ JDK 1.5 (vagy 1.6) (Java Development Kit) – futtatási környezet és Java fejlesztői eszközök (pl. fordító)
- ▶ Eclipse – Java fejlesztői környezet
- ▶ Ant – build tool
- ▶ Tomcat 6 – webkonténer
- ▶ MySQL – adatbáziszerver
- ▶ tetszés szerinti böngésző