JAVA technológiák - bevezető

Java

Java szigete - Indonézia

Tartalom

Java

Java:

- programozási nyelv
- futtatási környezet (runtime environment)

Java platformok:

- J2SE: asztali PC illetve szerver alakmazások számára
- J2EE: szerveroldali alkalmazásokra
- J2ME: mobil alkalmazások fejlesztésére (hurcolható, PDA).

előismeretek

feltételezett (...) Java-val kapcsolatos előismeretek:

- objektumorientáltság (az elemi típusok kivételével minden objektum),
- java szintaxis,
- referenciák,
- objektumok inicializálása,
- objektumok törlése (garbage collector),
- ???,
- kivételkezelés,
- objektumkollekciók,
- reflection, introspection,
- szálak stb.

előismeretek

feltételezett (...) Java-val kapcsolatos előismeretek:

- objektumorientáltság (az elemi típusok kivételével minden objektum),
- java szintaxis,
- referenciák,
- objektumok inicializálása,
- objektumok törlése (garbage collector),
- interfészek,
- kivételkezelés,
- objektumkollekciók,
- reflection, introspection,
- szálak stb.

J2EE

A Java2Enterprise platform az alábbiakat kínálja:

- alkalmazások *komponens alapú*
  - tervezése (design)
  - fejlesztése (development)
  - összeállítása (assembly)
  - telepítése (deployment)
- többrétegű osztott alkalmazásmodell
- újrafelhasználható komponensek
- egységesített biztonsági modell
- rugalmas tranzakció-vezérlés
- XML alapú web-szolgáltatások (web service)
- platformfüggetlenség
- szerverfüggetlenség

Többrétegű alkalmazásmodell

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

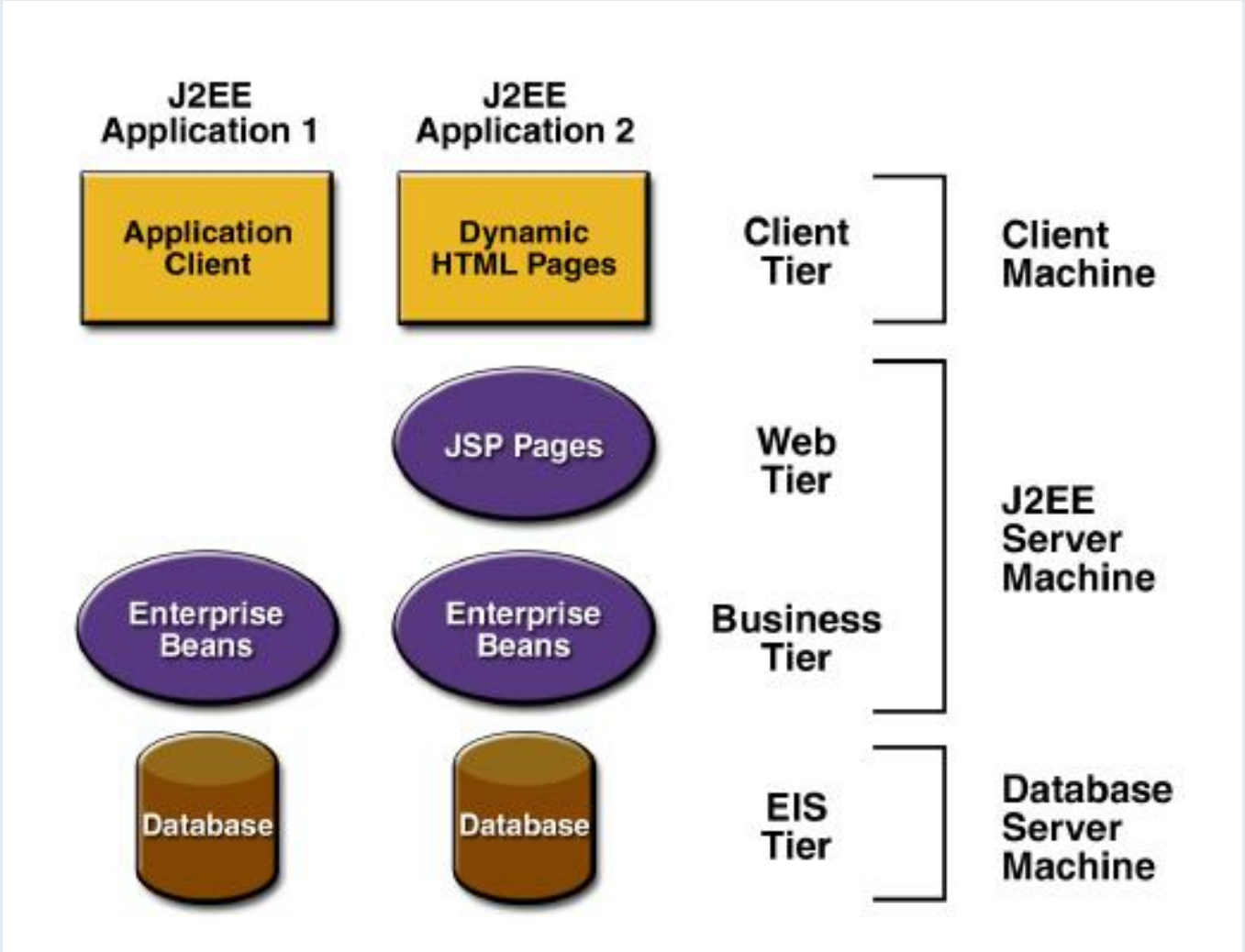
A következő rétegeket különböztetjük meg:

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

A J2EE alkalmazásokat általában három rétegűnek tekintik, mert három helyre vannak szétosztva:

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

Többrétegű alkalmazások



Többrétegű alkalmazások [J2EE Tutorial]

Web kliensek

Két részből állnak:

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

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)

Appletek

A szervertől kapott web oldal tartalmazhat egy beágyazott **applet**et.

applet:

egy Javaban íródott kisebb kliens-alkalmazás, amely a böngészőbe telepített JVM-ben fut le

A (szerver-oldali) web komponensek használata előnyösebb az appletekkel szemben:

- nincs szükség Java plugin-re
- egy tisztább és modulárisabb alkalmazástervezést biztosítanak (az alkalmazáslogika és a weboldal megjelenítési formája szétválasztható)

Alkalmazáskliensek

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 (lásd: Mr. Bean... :))

A szerver és kliens rétegek tartalmazhatnak **JavaBeans** komponenseket.

Az alábbiakban játszanak szerepet:

- alkalmazáskliensek vagy appletek és a J2EE komponensek közötti adatcserében
- a J2EE komponensek és az adatbázisréteg közötti adatcserében

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.

JavaBean-ek

A szerver és kliens rétegek tartalmazhatnak **JavaBeans** komponenseket.

Az alábbiakban játszanak szerepet:

- alkalmazáskliensek vagy appletek és a J2EE komponensek közötti adatcserében
- a J2EE komponensek és az adatbázisréteg közötti adatcserében

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:

- **Appletek**: kliensoldalon futó komponensek
- **Szervletek** 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

Szervletek:

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

JSP-k:

szöveg-alapú dokumentumok, amelyek a háttérben ugyancsak szervletként 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 használati esetet –use case– implementál) **EJB**-k által van megvalósítva.

$$\text{kliens program} \rightleftarrows \text{EJB (feldolgoz)} \leftrightarrows \text{EIS}$$

Háromféle EJB:

- **session bean**: ideiglenes (transient) kapcsolat egy klienssel. A klienstől kapott feladat elvégzése után, a session bean és a hozzá tartozó adat megszűnik.
- **entity bean**: egy adatbázistábla egy bejegyzésének (record) felel meg. A kliens program befejezése vagy akár a szerver leállítása után a háttérszolgáltatások biztosítják, hogy az entity bean le lesz mentve.
- **message-driven (üzenetvezérelt) bean**: összekapcsolja a session bean jellemzőit egy JMS (Java Message Service) alapú üzenet hallgatóval (message listener), hogy aszinkron módon kapjon JMS üzeneteket.

ami a többrétegű 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 platfom-független J2EE architektúra a többrétegű 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 telepítni (deploy)

az összeállítás:

különböző konténer beállításokat feltételez –

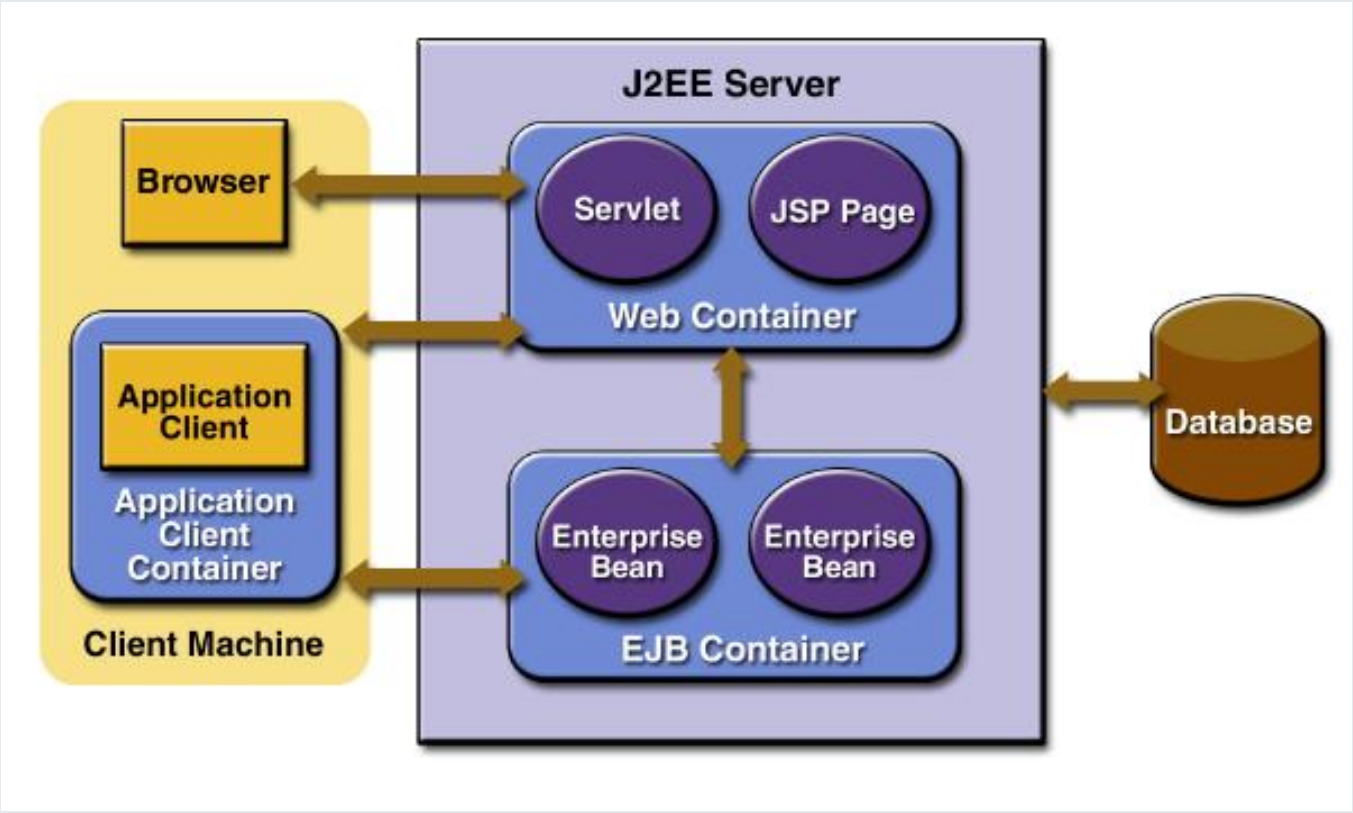
- külön-külön minden egyes komponens számára
- az egész alkalmazásra

A J2EE szerver ezen beállítások alapján különböző szolgáltatásokat biztosít:

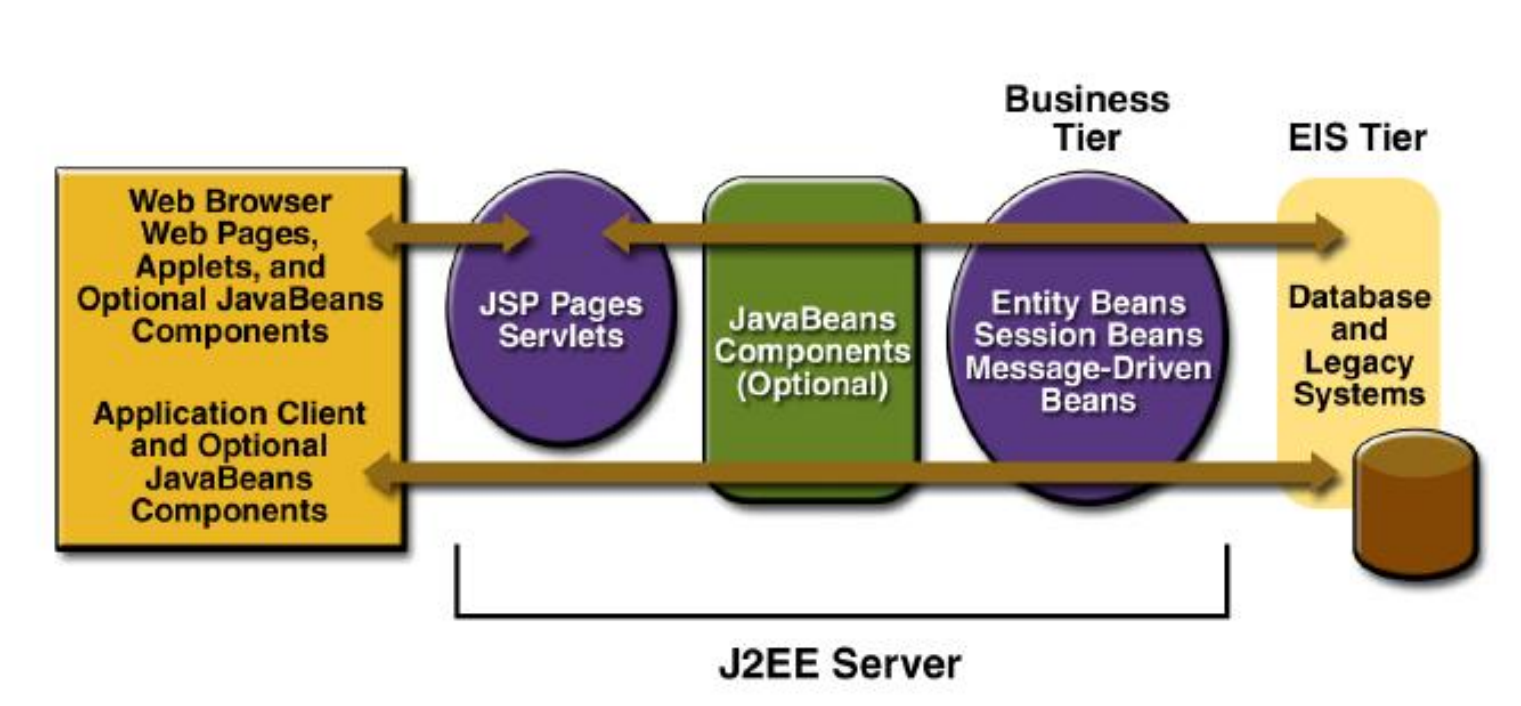
- biztonsági szolgáltatások
- tranzakciókezelés,
- JNDI,
- távoli csatlakozások (EJB-k és kliensek között, mintha ugyanabban a JVM-ben futnának)

Konténer típusok

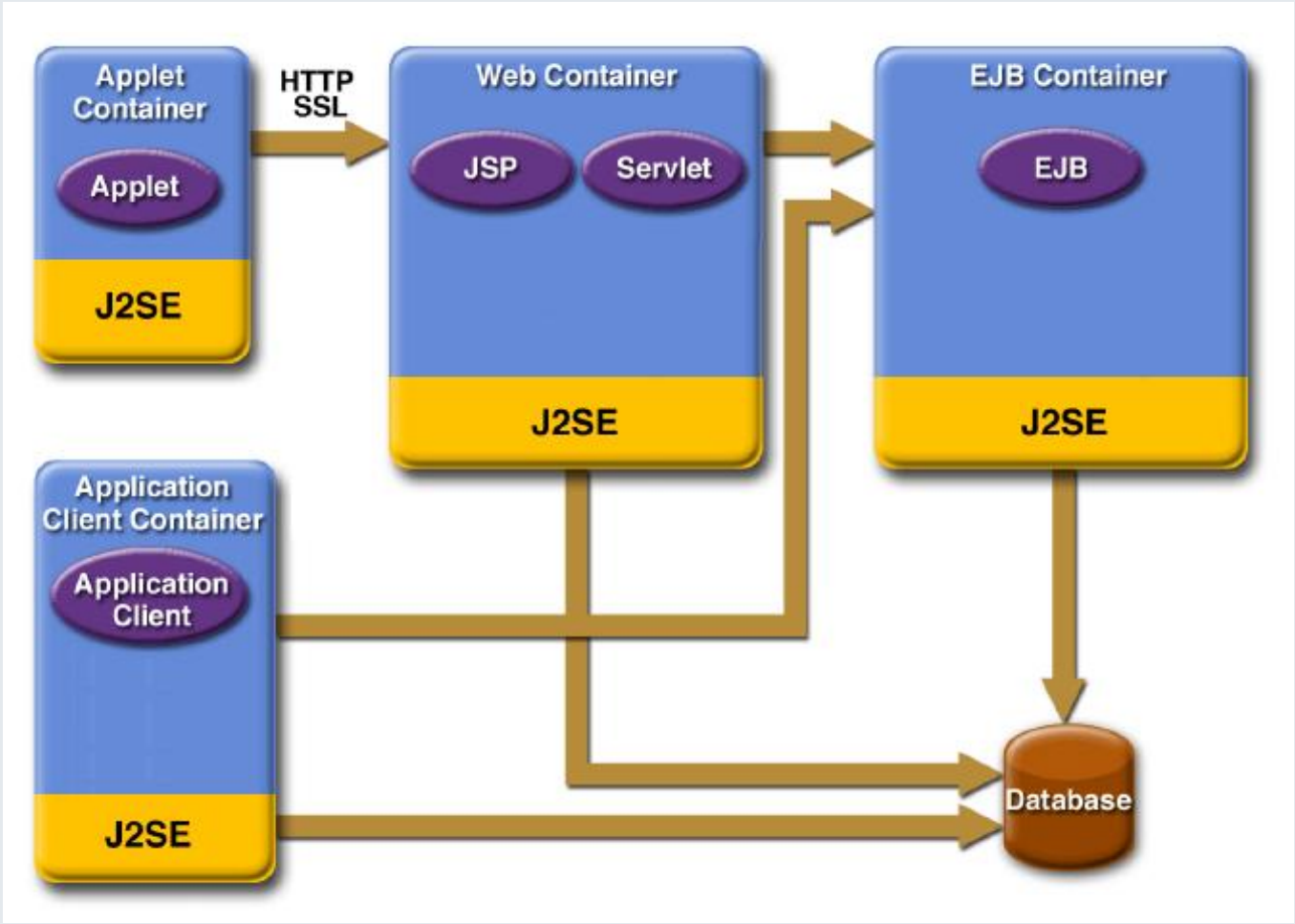
egy J2EE szerver EJB és Web konténereket biztosít:



J2EE szerver és konténerek [J2EE Tutorial]



Kapcsolatok I. [J2EE Tutorial]



Kapcsolatok II. [J2EE Tutorial]

Java2EE API-k

EJB technológia:

Az EJB egy metódusokat és mezőket tartalmazó kódrész, amely az üzleti logika egy bizonyos részét implementálja.
Ez egy építőkocka, ami magában vagy más EJB-kel együttműködve az üzleti logikát végzi el a J2EE szerveren.

Szervlet technológia:

A kérés-válasz (request-response) alapú alkalmazásmodellen alapulnak.
Bármilyen típusú kérést kiszolgálnak, de tipikusan webszervereken használják (HTTP protokol)

JSP (Java ServerPages) technológia:

Szervlet kód beágyazása szöveges dokumentumba.
A dokumentum tehát kétféle típusú szöveget tartalmaz:

- statikus szöveg (HTML, XML)
- JSP elemek – ezek a dinamikus tartalom létrehozására szolgálnak

JMS (Java Message Service)

Üzenetkezelő standard, amin keresztül a J2EE komponensek üzeneteket hoznak létre, küldenek, fogadnak és olvasnak.

kevésbé kötött, megbízható és aszinkron osztott kommunikációt tesz lehetővé

Java Transaction

Standard interfészt biztosít tranzakciók lebonyolítására.
A JTA API-t két vagy több egymáshoz kötődő adatbázisművelet elvégzésére használjuk, biztosítva, hogy vagy mindkettő lefusson (commit) vagy egyik sem (rollback)

JavaMail

E-mailek küldésére és fogadására használható.
Két részből áll:

- alkalmazás rétegű interfész, amit a komponensek e-mail küldésre használnak
- szolgáltató csomag (service provider) rész

JAX-RPC (Java API for XML-based RPC)

Egy SOAP és HTTP alapú standard, melyen keresztül a kliensek egy távoli szerveren levő XML alapú eljárásokat hívhatnak meg az Interneten keresztül.

JDBC

SQL hívásokat biztosít java-ból.
Két részből áll:

- alkalmazás rétegű interfész – amit a komponensek adatbázisműveletek elvégzésére használnak
- szolgáltató csomag (service provider) – ami a JDBC drivert teszi hozzáférhetővé

JAXP (Java API for XML Processing):

XML elemzők (parser): XML dokumentumok feldolgozása az alábbiakat felhasználva:

- DOM (Document Object Model) illetve SAX (Simple API for XML)
- XSLT (XML transzformációk)

Namespace támogatás: névkonfliktusok elkerülésére.
A JAXP **implementációfüggetlen** XML feldolgozást biztosít: lecserélhető az XML elemző (parser) vagy XSL processzor, a kliens kód módosítása nélkül.

Név szolgáltatás (Java Naming and Directory interface, JNDI)

Standard név- és katalógus-műveleteket (directory-operations) tesznek lehetővé, mint pl.:

- attribútumok társítása objektumokhoz
- objektumok keresése az attribútumok alapján

JNDI segítségével egy J2EE alkalmazás bármilyen típusú névvel ellátott Java objektuma

- eltárolható
- visszanyerhető

Az alkalmazásklienseknek, web komponenseknek valamint EJB-nek egy név alapú környezetet biztosít, mely lehetővé teszi, hogy egy komponens a forráskód módosítása nélkül állítható (customizable) legyen.
A JNDI többféle implementációfüggetlen név- és katalógus-művelet szolgáltatás elérését teszi lehetővé: LDAP, NDS, DNS és NIS

Web-alkalmazások

A webalkalmazások alapgondolata:

bizonyos webcímek mögött nem statikus tartalom van (pl. HTML), hanem a szerver a böngésző kérésére röptében dinamikus tartalmat hoz létre, és küld el a böngészőnek.

Java platformon kétféle web-alkalmazást különböztetünk meg:

- megjelenítés alapú: a kérésre (request) választ (reponse) azaz interaktív web-oldalakat generál, melyek HTML, XML stb. tag-ekből valamint dinamikus tartalomból állnak.
- szolgaltatás alapú: egy webszolgáltatás két végpontját implementálja.

A megjelenítés alapú web-alkalmazások gyakran kliensei a szolgáltatás alapú web-alkalmazásoknak.

Web-komponensek:

- Java szervletek
- JSP oldalak
- web szolgáltatás végpontok (web service)

Web-komponensek:

- Java szervletek
- JSP oldalak
- web szolgáltatás végpontok (web service)

Mi a szervlet/JSP technológiával fogunk megismerkedni közelebbről.

Web-komponensek:

- Java szervletek
- JSP oldalak
- web szolgáltatás végpontok (web service)

Mi a szervlet/JSP technológiával fogunk megismerkedni közelebbről.

Más, hasonló jellegű technológiák, melyek szintén dinamikus tartalmat hoznak létre: CGI, ASP, PHP.

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

1. A kliens egy HTTP kérést küld a web-szervernek
2. A web-szerver (mely a Servlet és JSP technológiákat implementálja) a kérést egy HTTPServletRequest objektummá alakítja.
3. Ezt az objektumot megkapja a web-komponens, 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 válasszá (response) alakítja és visszaküldi a kliensnek.

Szervletek:

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

JSP-k:

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

- elvileg – a szervletek és JSP-k felcserélhetőek
- gyakorlatilag – minkettőnek megvan az erőssege és a gyengéje

összehasonlítás

A szervletek alkalmasabbak:

- szolgáltatás alapú web-alkalmazások létrehozására
- kontrollerként egy megjelenítés-alapú alkalmazásnál (pl. kéréstovábbítás)
- nem szöveg alapú adatok feldolgozására

A JSP-k alkalmasabbak:

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

A szervlet és JSP technológiák bevezetése óta újabb Java technológiák és keretrendszerek (framework) fejlődtek ki, amelyek az előbbi kettőn alapulnak. (Pl. JSTL, JSF)

Egy web-alkalmazás a következő komponensekből áll:

- web-komponensek
- telepítési leíró (deployment descriptor)
- statikus erőforrások (pl. képek)
- segédosztályok
- jar programcsomagok (libraries)

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

- a web komponensek (szervlet, JSP) illetve segédosztályok megírása
- a telepítési leíró (deployment descriptor) létrehozása
- a komponens osztályok illetve az ezekből hivatkozott segédosztályok lefordítása
- [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 találhatók a

- JSP oldalak
- kliens oldali osztályok (applet osztályok)
- statikus web erőforrások

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

- web.xml : a webakalmazás telepítő leírója (deployment descriptor)
- Tag leíró állományok (library descriptor files)
- `classes` katalógus: szerver-oldali osztályok: szervletek, segédosztályok és JavaBeans komponensek
- `tags` katalógus: tag-eket implementáló osztályok
- `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-modul **szerverfüggetlen**: bármilyen web-konténerbe telepíthető, amelyik megfelel a Java Servlet specifikációnak.

Dokumentáció

- Bruce Eckel: Thinking in Java - alap Java kézikönyv
- J2EE Tutorial

Hasznos web-címek:

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

Fejlesztői környezet

Az alábbi eszközöket fogjuk használni:

- JDK 1.5 (Java Developement Kit) – futtatási környezet és Java fejlesztői eszközök (pl. fordító)
- Eclipse – Java fejlesztői környezet
- Ant – build tool
- Tomcat 5.5 – webkonténer
- MySql – adatbázisszerver
- tetszés szerinti böngésző