

# Üdvözli Önöket A Programozás Technológiai Eszközei c. tantárgy!!

a.k.a: Szoftverfejlesztési projekt munka a gyakorlatban III. (PGY3)

Bakay Árpád dr.

NETvisor kft

(30) 385 1711

[arpad.bakay@netvisor.hu](mailto:arpad.bakay@netvisor.hu)

# Tartalom – idén

WEB UI programozási technológiák

Web-es alkalmazás Tudor/Szeráj/Sing-Sing

Szoftvertechnológiai eszközök a gyakorlatban

Subversion

Atlassian Jira

Szerver és kliens oldalt tesztelő eszközök

Szerver és kliens oldali hibakeresés

Build / config mgmt eszközök: Ant, Ivy, Maven

Java EE és Web UI technológiai barangolások

Mobil alkalmazások (?)

# Tantárgyi weblap

- <http://telco.ikkk.inf.elte.hu/progtechgyak3.htm>
- Ld. A Syllabuszt is!

# Követelmények

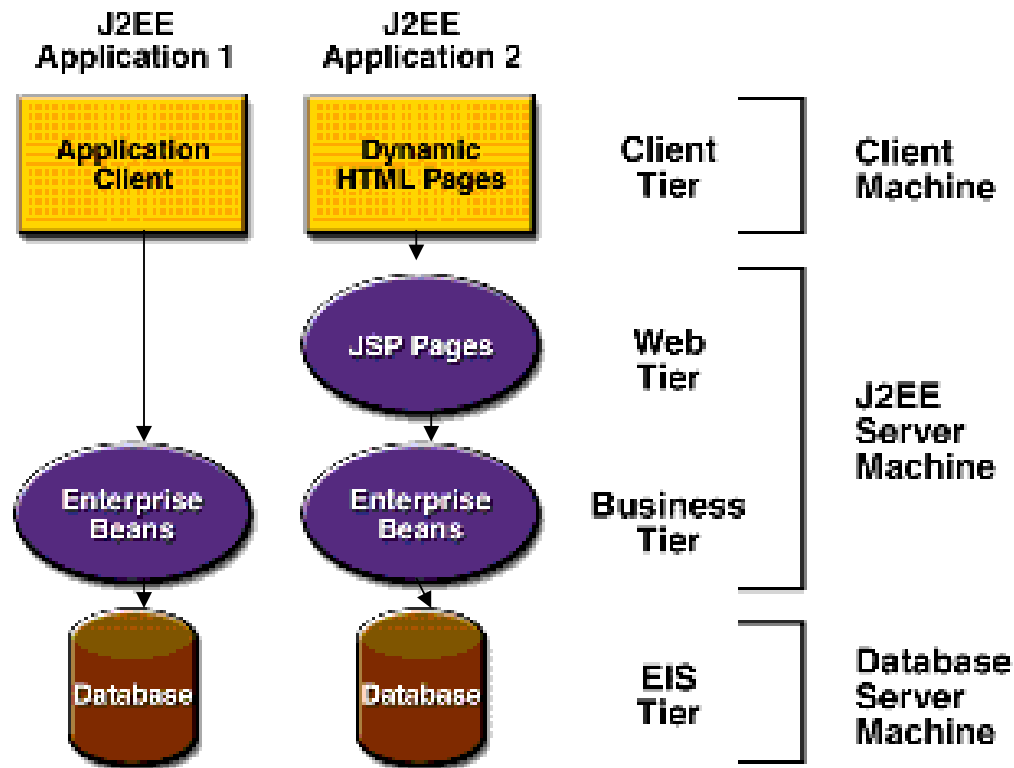
- Kis ZH-k, 4-6 db
- Egyszerű Java EE web alkalmazás (okt. 16)
- Technológiai áttekintés, esszé (nov 10.)
- Komplet web alkalmazás (dec. 10)
  - A PGY2-ben elkészített üzleti logika (Java, JSP/JSF) interfésze
  - vagy
  - A Programozás Technológia 2 tantárgy alkalmazása, teammunkában

# Tartalom - ma

- Általános tudnivalók
  - Órarend, követelmények, vizsga, konzultációk
- Mai anyag:
  - Java EE komponens-technológiai áttekintés
  - Alap-ui technológiák:
    - HTTP, HTML, dinamikus tartalom
    - Servletek és JSP-k

# 1. Ismétlés, áttekintés

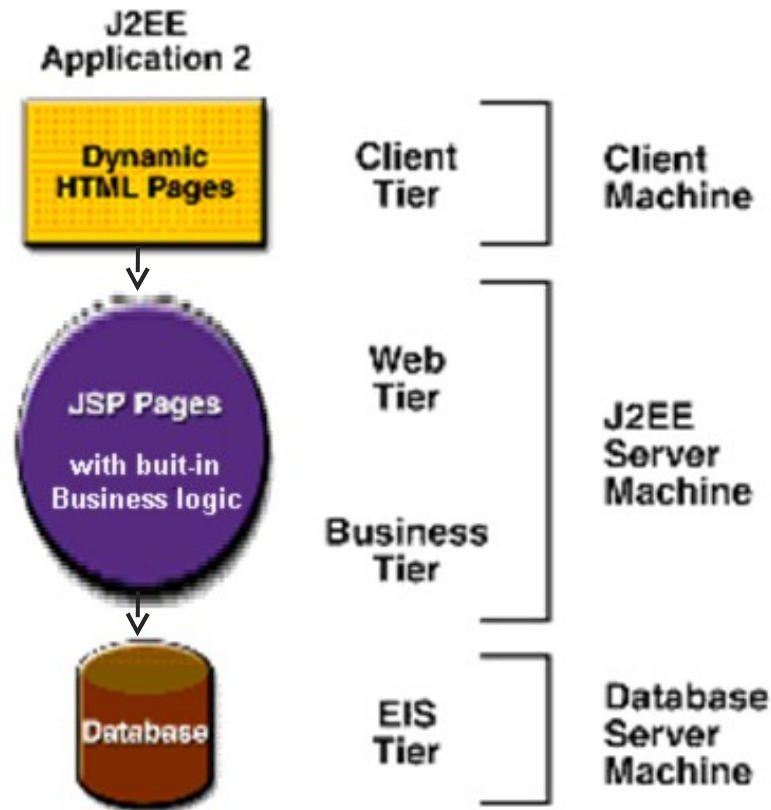
# Hagyományos és Web alapú többrétegű architektúra



- Alternatívák
  - Normál GUI
    - v. java Applet
  - Web GUI (browser)
- Számunkra a lényeg a middle tier!
- Web Tier és Enterprise Beans

JSP pages: a Web Tier egyik jellemző technológiája

# Egyszerűsített többrétegű Web alkalmazás architektúra



özvetlenül a Web containerbe integrált üzleti logika

Egyszerűbb, sokszor praktikusabb elterjedtebb

# Kliens oldali alternatívák

- „fat client”
  - Üzleti logikát tartalmaz, nem illik a 3-rétegű szemléletbe
- Önálló, de „thin” kliens alkalmazás
  - A telepítési problémák továbbra is fennállnak
  - Java EE szerver esetén leginkább ez is Java (az RMI miatt)
- Java applet „thin” kliens
  - Weblapba ágyazva, automatikusan letöltődik a browserbe
  - Viszont: funkcionális és teljesítmény-problémák
- Egyéb weblapokba beágyazható aktív objektumok:
  - Adobe Flash, Silverlight, SVG
- Pure .html kliens
  - A GUI prezentációs logika is (részben) a szerver oldalon
    - „web tier”: JSP-k, servletek, stb.
  - Kliens oldalon JavaScript kóddal dinamikussá tehető
  - Pillanatnyilag ez a legnépszerűbb
    - Összességében ezzel van a legkevesebb probléma!!!!
    - Akár egy „okostelefonon” is elfut!!
- HTML 5: a jövő szabványa
  - Grafikus (canvas) és Audio/Video media-lejátszási képességek
  - További újítások: drag&drop, local data storage stb.
  - Kiválthatóvá teszi a különféle beágyazott aktív objektumokat
  - Részben már implementált browserekben)

***Ezek érdekelnek minket!***

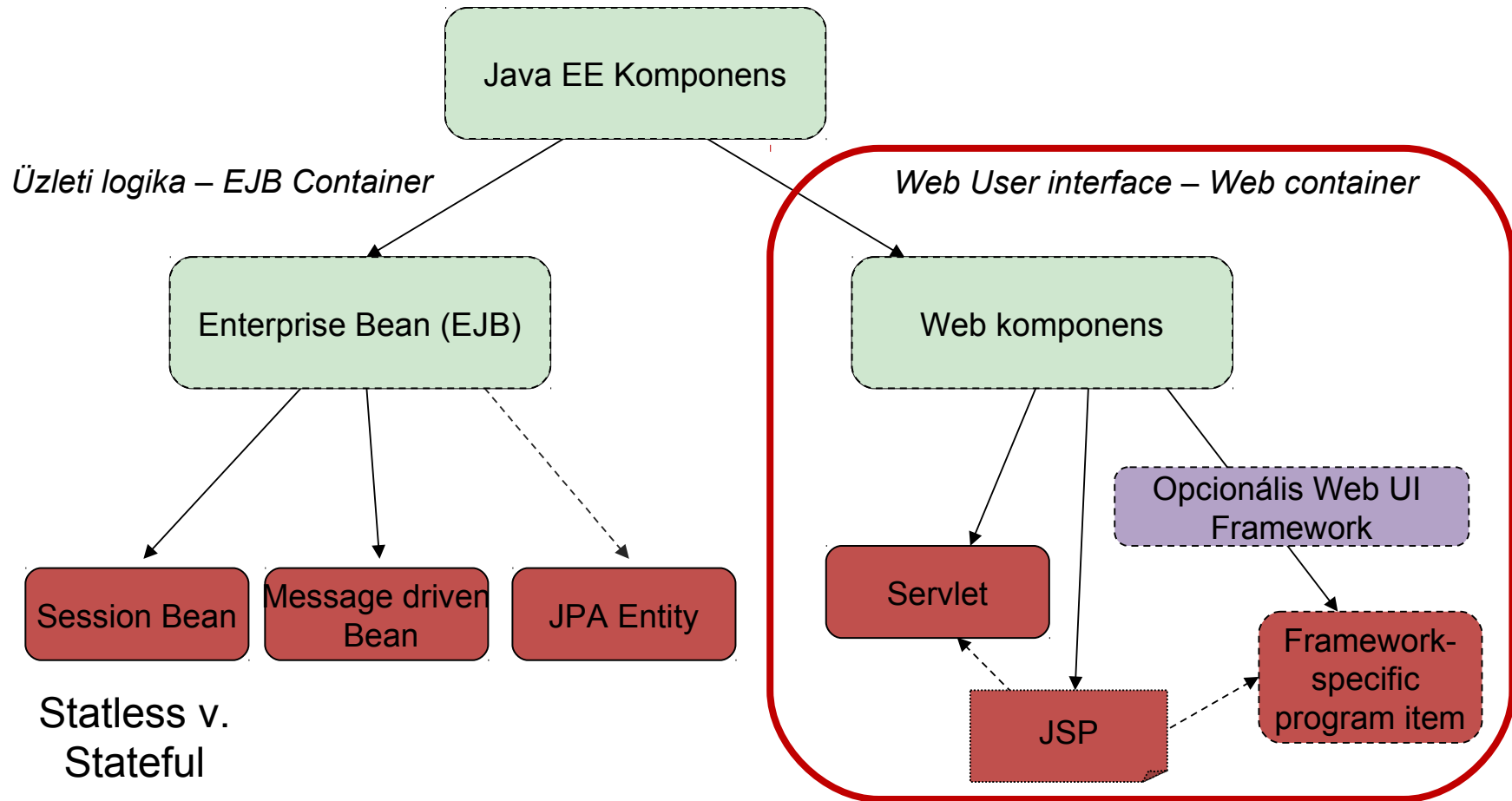
# A Java Enterprise Edition

- A Java Standard Edition-ra épül
  - Java SE: OO környezet -- Java EE: komponens környezet
- Futtató környezet komponensek számára
  - Az alkalmazás logikája szabványos architektúrájú komponensekben
    - Bevált, kiforrott tervminták alkalmazása
  - Premium szolgáltatások a környezettől
  - Szinte teljesen szabványos,
    - azaz egy alkalmazás bármely Java EE szerverben futtatható
- Logikusan következik belőle egy alkalmazás architektúra
  - Leginkább általános üzleti alkalmazásokhoz
- A Java EE rendszerint maga is „pure Java” azaz a Java SE-n fut

# A komponens

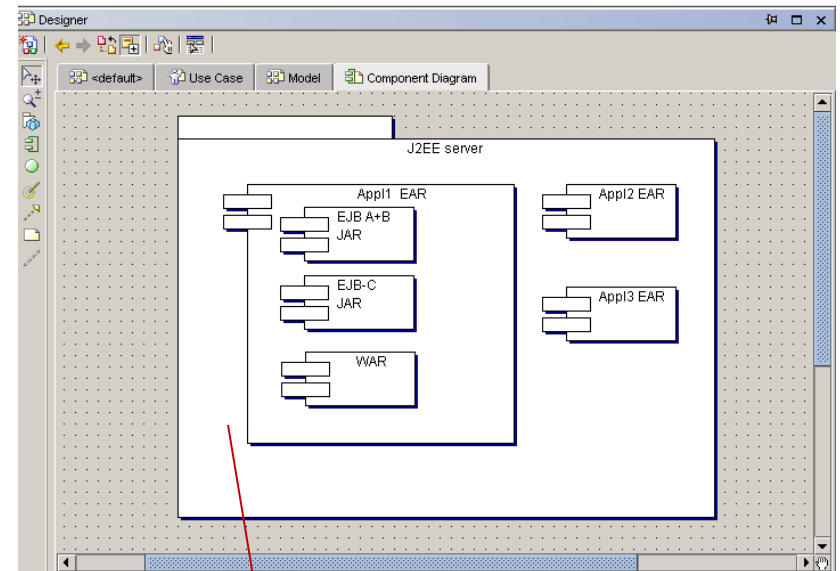
- *„egy darab szoftver, amelyet változtatás nélkül használnak más, a komponens íróitól függetlenül elkészített alkalmazások,“ -- Martin Fowler, [www.martinfowler.com](http://www.martinfowler.com)*
- Önálló logikai funkciót valósít meg
  - Koherencia, kohézió
  - Ujrahasználható
- Meghatározott architekturális környezetben (container-ben) futtatjuk
  - Védetten és ellenőrzöten fut (pl. ha elszállna)
  - Támaszkodik a környezet szolgáltatásaira
  - Java EE EJB Container és Web container
- Egyéb jellemzők:
  - Rendszerint egyetlen fő file / osztály definiálja
  - Automatikus telepítés, frissítés.

# Java EE Komponensek „családfája”



# Java EE komponensek csomagolása és telepítése

- **Alkalmazás telepítése egyetlen file-ban!!**
  - egymásba skatulyázott jar (zip) fileok
- xml formátumú komponens descriptorok
  - + szerver-specifikus kiegészítő descriptorok
- Pl:
  - **App1.ear**
    - AandB.jar
      - ABean.class, A.class, AHome.class
      - BBean.class, B.class, BHome.class
      - **ejb-jar.xml + jboss.xml**
    - C.jar
      - CBean.class, C.class, CHome.class
      - **ejb-jar.xml + jboss.xml**
    - Gui.war
      - index.jsp
      - main.jsp
      - **web.xml + jboss-web.xml**
    - **application.xml**
  - **App2.ear**
    - ... <http://www.apache.org/foundation/how-it-works.html>
  - **App3.ear**
    - ...



**Web komponens helye az alkalmazás archívumban**

# Legfontosabb rövidítések

- Java SE - Standard Edition
  - Az alap-java környezet
- Java EE - Enterprise Edition
- EJB - Enterprise Bean
  - Egyfajta komponens,  
(nem JavaBean!!)
- XML - Extensible Markup Language
  - Általános jelölőnyelv
- HTML - Hypertext Markup Language
  - Weblapok és hasonló dokumentumok jelölőnyelve
- HTTP - Hypertext Transfer Protocol
  - Hálózati átviteli protokoll, (nemcsak HTML-re!)
- JSP - Java Server Pages
  - Egyféle Web presentation  
komponens/technológia
- AJAX - Asynchronous Javascript and XML
  - A dinamikus web-alkalmazások alaptermészetűje
- CSS - Cascading Stylesheets
  - HTML tartalom javasolt formázási technológiája

Már ismerjük

Ezzel  
részletesen  
foglalkozunk  
majd

# 2. A Web alkalmazások működése röviden

- Átviteli protokoll: HTTP (Hypertext Transfer Protocol)
  - \_ Az rfc2616 IETF szabvány definiálja
  - \_ Alapelv: mindig a kliens kérdez, a szerver erre válaszol
  - \_ Http 1.0: 1 TCP kapcsolat - 1 kérés-válasz,
    - Http 1.1-ben már lehet több is - connection reuse
- A címzés URL alapján történik
  - \_ „Uniform Resource Locator”, rfc1738
  - 1. Protokoll (általában http), szerver, azon belül a kért lap azonosítása  
<http://www.apache.org/foundation/how-it-works.html>
- A kéréshez extra paraméterek, adatok csatolhatók az URL végén (http GET művelet), vagy a http kérésű parancshoz csatolva (http POST)  
<http://www.youtube.com/watch?v=dYP-wBaqQAI>
- A válasz rendszerint valami adatot tartalmaz.
  - \_ Ennek típusát a http válasz header jelöli

# Http üzenetek

- Parancs/válasz + headerek + tartalom
- Mind a kérés, mind a válasz tartalmaz

## \_ Egysoros kérést, ill. választ

K: GET <http://www.cisco.com> HTTP/1.1

V: 200 OK

## \_ Header mezőket

- Ezek név-érték párok, jó részük szabványos
- K: browser típus, nyelv preferencia, usernév és password, cookiek, content type, stb.
- V: content type, dátum, hossz, cookie-k, szerver azonosító, stb.

**Content-type: image/jpeg**

**Content-length: 31313**

## \_ Tartalmat

- A content type-nak megfelelően
- (GET kérésnél általában nincs tartalom)

# Hogyan alakul ki egy web-oldal képe?

- A User által megadott URL általában egy HTML oldalt eredményez
  1. Http kérésként a browser megküldi az URL-t
  2. Szerver megkeresi vagy generálja
  3. HTTP protokoll válaszként átküldi
  4. Browser megjeleníti a W3C HTML szabványai szerint → rendering
- HTML-ből hivatkozhatók beágyazott képek, videók stb.
  - \_ Saját URL-jük van, és ezek is további HTTP-kérésekkel töltődnek le.
- A HTML-be beágyazhatók aktív működésű önálló objektumok is: pl. Flash, Java Applet, ActiveX elemek
  - \_ Ezek kódja is rendszerint egy szerverről töltődik le.
  - \_ Az oldal egy téglalapnyi részét önállóan kezeli
  - \_ A browserbe telepített bővítő modulok futtatják őket
- További kapcsolódó technológiák:
  - \_ Pl. megjelenítés, stílusokkal,
  - \_ Különbéle grafikus és média tartalom lejátszása,
  - \_ Security (titkosítás, autentikáció, tanúsítványok),
  - \_ Proxy-k és cache-ek.

# A HTML és az XML

- Közös ős: Standard Generalized Markup Language - 1986
  - (népszerűtlen) szabvány dokumentumok szemantikai leírására
- 1. utód: HTML (1991-):
  - laza, engedékeny szintaxis,
    - Pl. `<p align=center>AA<p align=left>BB</p>`
    - V.ö: `<p align="center">AA</p><p align="left">BB</p>`
  - ezért viszonylag nehezen parse-olható és verifikálható
  - megjelenítéséhez széles körben elfogadott szabályok léteznek (rendering, W3C.org által definiálva)
- 2. utód: XML:
  - népszerű egyszerűsítés („subset”)
  - könnyen parse-olható
  - Mindenféle adatra
  - Megjelenítés általában nem kötött
- XHTML:
  - a fenti 2 újraraházasítása

# A HTML technológia fejlődése

- Statikus dokumentumok
- Interaktív form-ok és dinamikusan generált tartalom
  - A szerver az előző lapokról származó input adatok alapján din. változó oldalakat generál
- JavaScript
  - Kliens oldali programnyelv dinamikus viselkedés implementálásához
  - Semmi köze a Java-hoz: run-time eval, laza változó, függvény és objektum típuskezelés stb.
- DOM: hozzáférés a teljes dokumentum struktúrához
  - Dinamikus HTML oldalak készíthetők
- AJAX
  - Asynchronous Javascript Through XML = a Javascript egy külön csatornán kommunikál a szerverrel, és új lap-kérések sorozata helyett csak változtatja a megjelenített oldalt.
- A browser bár nem „böngésző”, hanem User Interface futtató platform/konzol.
  - RIA: Rich Internet Application

# HTML Tartalom és Megjelenés

- Ideális esetben a HTML markup a szöveg tartalmát gazdagítja
- A megjelen(ít)és ettől elméletileg független kell legyen
  - ... és részben a browser belügye
    - Pl. speciális megjelenítés fogyatékosoknak, PDA-kra, stb.
  - „Semantic HTML” „Separation of concerns”
  - Ennek egyik fontos eszköze a Style Sheet-ek alkalmazása
    - ld. table-less html
- De: a látványos. interaktív web GUI-k fejlesztésénél a prezentáció részleteit is egyre szorosabban kézben kell tartani.

# Interaktív web alkalmazások - alapszabványok

1. A browser letölt egy `<form>`-ot ill. input elemeket tartalmazó (akár statikus) HTML lapot
  - `<form method="get" action="getUsers.jsp" > ... </form>`
  - `<input type=...>` („text”, „button”, „radio”, „checkbox”, ill. „submit” v. „reset”)
  - `<select> ... <option1> <option2>`
  - `<textarea>`
  - `<button>` - specifikus elavult tag-ek bizonyos input komponenseknek
2. Egy `<input type="submit">` gomb hatására betölt egy új URL-t.
  - a kérésben a input elemek állapota is benne van („request parameter”-ekként)
  - GET v. POST: alternatív HTTP műveletek, kb. azonos hatással
    - A különbség a request paraméterek átadásában van.
3. A szerver egy, a paraméterek értéke alapján dinamikusan generált HTML dokumentumot küld vissza
  - Ez további form-okat és interaktív elemeket tartalmazhat.

# Interaktív HTML alkalmazások állapota

- A HTTP alapvetően állapotmentes
  - A szerver csak az aktuális kérést nézi, nem ismeri a kapcsolat előéletét
  - A kliens is minden oldalt az előzményektől függetlenül jelenít meg
- De: kerülő megoldásokkal a kommunikáció állapotossá tehető
  - A. A szerver által visszaadott HTML-ekben szereplő linkek URL-jeibe kódolt adatokkal (elavult)
  - B. Cookie-k: browserben tárolt adatok, amiket a adott szerver felé irányuló kérésekhez csatol (egy HTTP feature)
- Éltszerű példák az állapotra
  - \_ Usernév, bevásárlókocsi, tranzakció státusza stb.,
  - \_ v. egyetlen „session key”, ami egy kulcs, a szerveren tárolt, tetszőleges méretű és tartalmú session-specifikus adatokhoz.

# Hagyományos dinamikus web szerver technológiák

- CGI – Common Gateway Interface
  - \_ A Web szerver a dinamikus tartalomhoz egy külső programot hív meg (pl. shell v. perl script)
  - \_ Ez a paramétereket parancssorban v. környezeti változóiban kapja meg
  - \_ A program kimenetét közvetlenül megkapja a browser
- PHP – HyperText Preprocessor
  - \_ A HTML-be ágyazható speciális kódokat a szerverbe épített engine feldolgozza, és helyükre új tartalmat generál.

```
<?php  
echo 'Hello' + GET[„Username] + '!' ;  
?>
```

# Ellenőrző kérdések

1. Hogyan csomagoljuk a Web komponenst az Alkalmazás archívumba?
2. Mik az alapvető web komponensek és mi a fő különbség köztük?
3. Mit tartalmazhat egy web-es URL?
4. Milyen HTTP metódusokkal kérhetünk le adatokat egy szervertől?
5. Mik a HTML technológia fejlődésének lépcsői?
6. Vázolja röviden egy interaktív web-alkalmazás alapműködését!
7. Az állapotmentes HTTP-vel hogyan lehet adatokat egy többlépéses interakción át megőrizni? Adjon egy példát?
8. Ismertessen nem Java alapú dinamikus szerver-technológiákat! Mi a két alapvető megközelítés lényege?