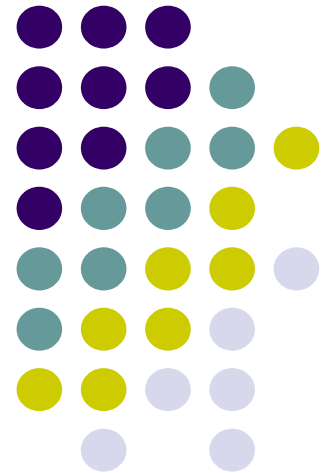


# 3.- WEB ORRIALDE DINAMIKOAK ESKEMAK

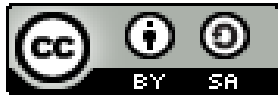
27380 ZERBITZU TELEMATIKO AURRERATUAK  
Telekomunikazio Teknologiako Ingeniaritza Gradua, 3. maila

2023-2024

Egileak: Maider Huarte Arrayago, Gorka Prieto Agujeta



# 2023-2024 ZTA 3 WEB ORRIALDE DINAMIKOAK - ESKEMAK.odp



Copyright © 2013-2024 Maider Huarte Arrayago, Gorka Prieto Agujeta



2023-2024 ZTA 3 WEB ORRIALDE DINAMIKOAK - ESKEMAK.odp lana, Maider Huartek eta Gorka Prietok eginia, Creative Commons-en Attribution-Share Alike 3.0 Unported License baimenaren menpe dago. Baimen horren keria bat ikusteko, <http://creativecommons.org/licenses/by-sa/3.0/> webgunea bisitatu edo gutun bat bidali ondoko helbidera: Creative Commons, 171 2nd Street, Suite 300, San Francisco, California, 94105, USA.

2023-2024 ZTA 3 WEB ORRIALDE DINAMIKOAK - ESKEMAK.odp by Maider Huarte and Gorka Prieto is licensed under a Creative Commons Attribution-Share Alike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/> or, send a letter to Creative Commons, 171 2nd Street, Suite 300, San Francisco, California, 94105, USA.

# WEB ORRIALDE DINAMIKOAK

## EDUKIA

- 1.- SARRERA
- 2.- CLIENT-SIDE SCRIPTING
- 3.- SERVER-SIDE SCRIPTING
- 4.- DISEINURAKO PATROIAK  
ERREFERENTZIAK

# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

2.- CLIENT-SIDE SCRIPTING

3.- SERVER-SIDE SCRIPTING

4.- DISEINURAKO PATROIAK  
ERREFERENTZIAK

# 1.- SARRERA

- Web orrialdeak

- Definizioa Informazioa formatu elektronikaon duen dokumentua.  
WWW zerbitzura egokituta web nabigatzaiile batez bistaratzeko

- Estatikoak vs dinamikoak

- Estatikoak: HTML

- Dinamikoak

- Client-side: DHTML
    - Server-side: CGI, Servlets,...

-Edukia aldakorra, ezaugarri batzuen arabera -Elkarrekin batera

# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

2.- CLIENT-SIDE SCRIPTING

2.1.- Sarrera

2.2.- DHTML

3.- SERVER-SIDE SCRIPTING

4.- DISEINURAKO PATROIAK

ERREFERENTZIAK

# 2.- CLIENT-SIDE SCRIPTING

## 2.1.- Sarrera

- Definizioa Web nabigatzaileak exekutatutako dinamismoa, deskargatutako orrialdean zehaztutako gertaeren arabera.
- DHTML: Osagaiak
  - HTML
  - CSS
  - Javascript
  - DOM

# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

**2.- CLIENT-SIDE SCRIPTING**

2.1.- Sarrera

**2.2.- DHTML**

3.- SERVER-SIDE SCRIPTING

4.- DISEINURAKO PATROIAK

ERREFERENTZIAK



# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- HTML: bezeroarekin dinamismoa eragiteko elementuak
  - HTML gertaerak
    - Erabiltzailearekiko interaktibitatea ahalbidetzen du Zerbait gertatzean: script funtzio bat exekutatzen da.
    - Zenbait HTML elementuetarako atributuak: *gertaeraAtributua="scriptFuntzioreaDeia"*
      - Leihoko gertaerak (<body>): **onload, onunload,...**
      - Formularioko (eta barneko elementuetako) gertaerak: **onsubmit, onblur/onfocus, onchange, onselect,...**
      - Teklatuko gertaerak: **onkeydown, onkeyup, onkeypress**
      - Saguko gertaerak: **onclick,...**
  - DHTML estandarrerako HTML elementuak
    - CSS
      - **<style type="text/css">...</style>**
      - **<link>**
    - Script kodeak
      - **<script>...</script>** Adibidez, gertaera baten ondoren zer egin.
      - **<noscript>...</noscript>**

# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- CSS Estatikoaren berdin funtzionaten du
  - DHTML estandarraren osagarrietako bat da
  - Ez du dinamismorik sartzen orrialdeetan
  - Orrialdeen itxura errazago kudeatzeko

# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

NOT RELATED TO JAVA IN ANY WAY SHAPE OR FORM!!

- JavaScript

- Definizioa Scripting language: Interpretatua da  
Printzipioz gauza txikietarako diseinatuta

- JS kodeen kokapena

- HTML barruan sartzea

Posiblea: <head> EDO <body> barruan  
Gomendioa: <head> barruan

- `<script>...</script>`

- `<noscript>...</noscript>`

- .js fitxategia: `<script src="fitxategia.js"></script>`

- Exekuzioa

- JS sententziak Renderizazioan

- Funtzioak Barruko sententziak, funtzioa deitzen denean bakarrik exekutatu dira

# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- JavaScript

- Sintaxia

- Iruzkina: *//Iruzkina, /\* Iruzkina \*/*

- Sententziak: ;

- Funtzioak

```
function izena(argumentuak) {  
    Sententziak  
}
```

goiburuan mota EZ: TYPELESS!

What a FUCKING SHITSHOW

return zeozer;

- Aldagaiak eta datu-motak

- Deklarazioa: **var** *izena*;

- Deklarazioa+hasieratzea: **var** *izena=balioa*;

- Esparruak: lokala vs globala

KONTUZ! Existitzen den aldagai bat erabiltzea EZ da errore sintaktikoa,  
IZEN HORREKIN ALDAGAI GLOBALA egingo du!

- Eragileak: esleipena (=), aritmetikoak (+, -, ...), alderatzea (>, >=, ...), logikoak (&&, ...)

- Baldintzazko sententziak: **if**, **switch**

- Sententzia errepikakorrak: **for**, **while**

do while ez da existitzen. womp womp

# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- JavaScript

- Sintaxia

- **Objektuak**

Informazio multzoak, eremu ezberdinetan antolatuak

- Eremuak

- Propietateak: balioak

- Metodoak: funtzioak

- Jadanik definitutako objektuak

- **String, Number, Array, Math**

- **window:** alert(), confirm(), prompt(), open(), close()

- Eremuak erabiltzeko: . eragilea

- **Objektuak sortzea:** objektuaren definizioa, new eragilea

- **Hauek izan ezik: String, Number**

new erabili sortzeko

- Erroreak: throw, try, catch



ez da klaserik egin behar:  
definitzea nahikoa da

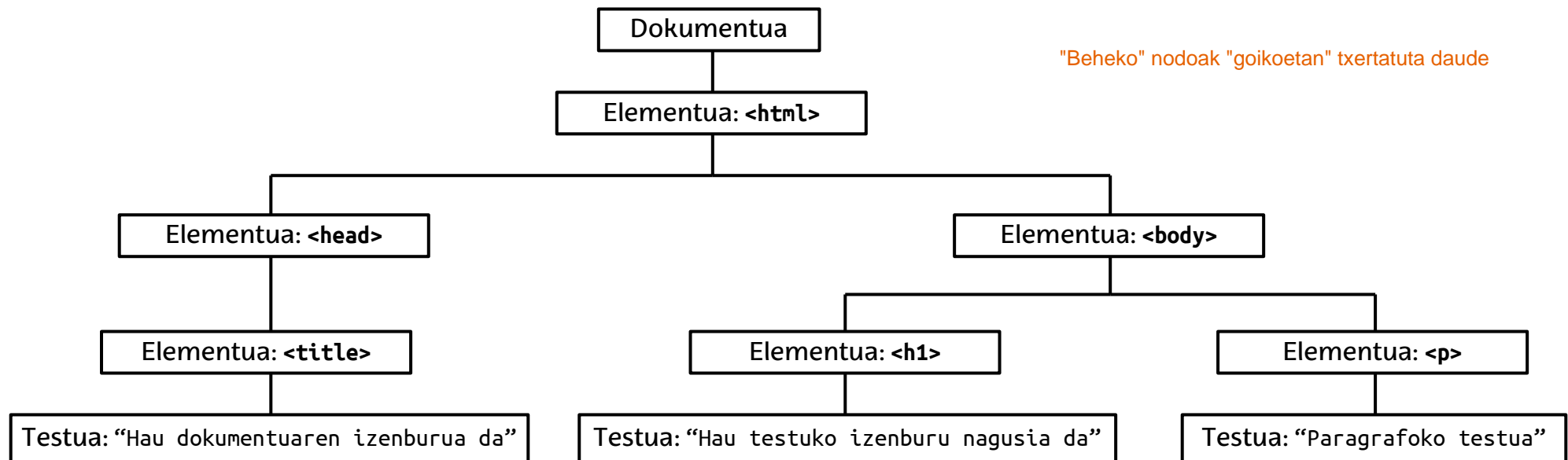
# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- DOM

Document Object Mode

- Definizioa HTML dokumentuetako elementuak eta edukiak memorian nola antolatu zehazten duen estandarra
- Elementu eta edukien egitura hierarkikoa: nodoen zuhaitza



- Client-side scripting moduko lengoaiei, nodoak erabiltzea ahalbidetzeko


# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- DOM
  - JSen jadanik definitutako DOM objektuak
    - **node: edozein nodorentzat**
      - Propietateak: `nodeName`, `childNodes`
      - Metodoak: `appendChild()`, `replaceChild()`, `removeChild()`
    - **document: erro-nodoa, ez dauka amarik.** node objektuko guztia +
      - Propietateak: `body`, `forms`, `images`, `title`, ...
      - Metodoak: `getElementById()`, `getElementsByTagName()`, `createElement()`, `renameNode()`, `open()`, `write()`, `close()`, ...
    - **element: edozein HTML elementurako.** node objektuko guztia +
      - Propietateak: `id`, `innerHTML`, `tagName`, ...
      - Metodoak: `getAttribute()`, `setAttribute()`, ...

# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- DOM  DOM-entzako hitz klabe asko daude!
- JSen jadanik definitutako DOM objektuak
  - **body**: **node** eta **element** objektuetako guztia +
    - Propietateak: **background**, **bgColor**, **onload**,...
    - Metodoak: -
  - **form**: **node** eta **element** objektuetako guztia +
    - Propietateak: **name**, **action**, **method**, **target**, **length**,...
    - Metodoak: **reset()**, **submit()**
  - **checkbox**: **node** eta **element** objektuetako guztia +
    - Propietateak: **checked**, **disabled**, **name**, **value**,...
    - Metodoak: -
  - **image**, **link**, **table**: **node** eta **element** objektuetako guztia +
  - **style**: CSS estiloekin lan egiteko

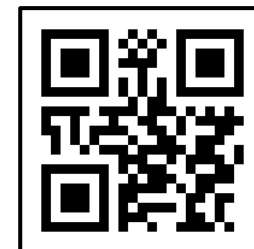


# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- JAVASCRIPT+DOM ADIBIDEAK

```
<!DOCTYPE html>
<html>
<head>
<script>
function agurraBistaratu() {
    var e1=prompt("Nor zara?");
    document.getElementById("demo").innerHTML="Kaixo "+e1+"! hau da une honetako data eta ordua: "+Date();
}
</script>
</head>
<body>
    <h1>ZTA 2023-2024: Nire lehenengo JavaScript kodea duen orrialdea</h1>
    <p id="demo" >Erantzun galdera eta hemen agurtuko zaitut...</p>
    <button type="button" onclick="agurraBistaratu()">Ikusi agurra</button>
</body>
</html>
```



Zer getatuko litzateke <h1 id="demo"> jarri izan bagenu?

Header-aren testua editatuko litzateke

Eta >buttonid="demo" type="button" onclick...>

Botoiaren barruko testua editatuko litzateke

```
<!DOCTYPE html>
<html>
<head>
<script>
function kontuzWrite() {
    document.open();
    document.write("SUNTSIKETAAAA!!!!");
    document.close();
}
</script>
</head>
<body>
    <h1>ZTA 2023-2024: Nire bigarren JavaScript</h1>
    <p id="demo">Hau paragrafo bat da ere.</p>
    <button type="button" onclick="kontuzWrite()">Saiatu document.write</button>
</body>
</html>
```

# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- JAVASCRIPT+DOM ADIBIDEAK



```
<!DOCTYPE html>
<html>
<head>
<script>
function aldatuNodoak() {
    var ps=document.getElementsByTagName("p");
    ps[0].innerHTML+=" ";
    var subp0=document.createElement("strong");
    subp0.innerHTML="Hau paragrafoaren amaierako testu garrantzitsua da";
    ps[0].appendChild(subp0);

    var ns=document.body.childNodes;
    document.body.removeChild(ns[5]);
    document.body.removeChild(ns[4]);
}

function aldatuEstiloa() {
    var p1=document.getElementById("p1");
    p1.style.backgroundColor="red";
}
</script>

</head>
<body>
    <h1>ZTA 2023-2024: HAU DOKUMENTUKO LEHENENGO IZENBURUA DA</h1>
    <p id="p1">Hau dokumentuko lehenengo paragrafoa da</p>
    <h2>HAU DOKUMENTUKO LEHENENGO IZENBURUKO LEHENENGO AZPI-IZENBURUA DA</h2>
    <p>Hau dokumentuko bigarren paragrafoa da</p>
    <p>Hau dokumentuko hirugarren paragrafoa da</p>

    <button type="button" onclick="aldatuNodoak()">Aldatu paragrafoak</button>
    <button type="button" onclick="aldatuEstiloa()">Aldatu atzealdea</button>

</body>
</html>
```

# 2.- CLIENT-SIDE SCRIPTING

## 2.2.- DHTML

- JAVASCRIPT+DOM ADIBIDEAK

```
<!DOCTYPE html>
<html>
<head>
<script>
function egiaztatuFormularioa() {
    var e=true;
    var o=document.getElementById("i1").value;
    var m=document.getElementById("i2").value;

    if(o==null||o=="||o<0||o>23||m==null||m=="||m<0||m>59) {
        alert("Ordua ez da baliozkoa");
        document.getElementById("i1").value="";
        document.getElementById("i2").value="";
        e=false;
    }
    else
        alert("Ordua egokia da");
    return e;
}
</script>
</head>
<body>
    <h1>ZTA 2023-2024: Nire laugarren JavaScript, formularioak egiaztatze</h1>
    <form action="DatuakProzesatukoDituenURL" onsubmit="return egiaztatuFormularioa()" method="post">
        Sar ezazu ordua <br>
        <input id="i1" type="text" name="ordua" maxlength="2" pattern="\d{2}" placeholder="oo">
        :
        <input id="i2" type="text" name="min"    maxlength="2" pattern="\d{2}" placeholder="mm">
        <br>
        <input type="submit" value="BIDALI">
    </form>
</body>
</html>
```

# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

2.- CLIENT-SIDE SCRIPTING

**3.- SERVER-SIDE SCRIPTING**

**3.1.- Sarrera**

3.2.- Servlet

4.- DISEINURAKO PATROIAK

ERREFERENTZIAK

# 3.- SERVER-SIDE SCRIPTING

## 3.1.- Sarrera

EZ DA NAHIKOA Apache

-HTTP-request Jaso + HTTP response bidali

-HTML kodea sortu: EZ fitxategi batetik irakurri

- Exekuzio ingurunea

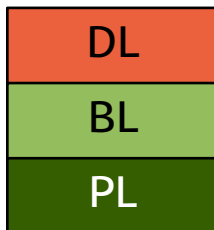
- Web zerbitzaria+CGI: Apache, CGI moduluarekin
- Web zerbitzaria+Aplikazio Zerbitzarirako connector+Aplikazio Zerbitzaria: Apache zerbitzaria Tomcat-ekin
- Aplikazioen Zerbitzaria bakarrik
  - Tomcat
  - Glassfish
  - Jboss/Wildfly
  - Jetty
- Multitier arkitekturak: DB

- Bezeroak

- Nabigatzaileak
- Neurrira egindakoa APP-ak

- Implementazioak: 3 mailetako arkitekturak

- Data Access Layer
- Business Layer
- Presentation Layer



Biltegitratutako informazioa: fitxategiak, DBak...

Kalkuluak

Erabiltzailearekiko interfazeak

# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

2.- CLIENT-SIDE SCRIPTING

**3.- SERVER-SIDE SCRIPTING**

3.1.- Sarrera

**3.2.- Servlet**

4.- DISEINURAKO PATROIAK

ERREFERENTZIAK

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- Aurretik berdinerako bazegoen CGI ordezkatzeko Java proposamena
  - Eraginkortasuna Memoria gutxiago erabiltzen dute CGI programekin konparatuz
  - Komenigarritasuna Java oso lengoaia zabaldua da
  - Ahalmena Eragiketa orokorretarako liburutegiak: Datu Baseak, sesioak...
  - Eramangarritasuna Sistema eragilearekiko menpekotasunik ez JVM-ri esker
- Servleten Java hierarkia
  - **javax.servlet.Servlet** interface

**java.lang.Object**

↳ **javax.servlet.GenericServlet (abstract): javax.servlet.Servlet, ...** class

↳ **javax.servlet.http.HttpServlet (abstract)** class

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- HTTPrako servlet baten oinarritzko egitura

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

HTTPrako servlet baten oinarritzko egitura

PL

```
@WebServlet("/ServletIzena")
```

Servlet deitzeko URL-rako izena

```
public class ServletIzena extends HttpServlet {
    private static final long serialVersionUID = 1L;
    //Beste atributuak
```

```
public ServletIzena() {
}
```

```
...//Beste eraikitzaileak
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //GET moduko HTTP-request eskaerari erantzuteko exekutatu beharreko kodea
}
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //POST moduko HTTP-request eskaerari erantzuteko exekutatu beharreko kodea
}
```

```
//Beste metodoak, berriak edo heredatuak (doHead,...)
}
```

- Gaur egun, edozein programaziorako IDE berrik txantiloiak eskaintzen ditu



# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- HTTPrako servlet baten oinarrizko egitura
  - Zerbitzu-metodoen parametroak: `doXXX(...)`
    - **request:** HTTP-request mezuko informazioa
      - `.getMethod()`
      - `.getHeader("HTTPgoiburua")`
      - Erabiltzailearen datuak (adibidez, formularioan sartuak)
        - `.getParameter("datuName")` String Radio botoietan, testu kutxetan... baliogarria
        - `.getParameterValues("multzoName")` String[] Checkbox-etan eta select-moduko zerrendatan baliogarria
      - ...

```
Metodoa path_edo_URL HTTP_bertsioa
Goiburu1:balio1
Goiburu2:balio2
...
Lerro_hutsa
Mezuaren_gorputza HTTP-request
```

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- HTTPrako servlet baten oinarritzko egitura
  - Zerbitzu-metodoen parametroak: `doXXX(...)`

- **response: HTTP-response mezurako informazioa**

- HTTP-response mezuan gorputzik egongo bada DERRIGORREZKOA

- **HTML kodea**

ASCII testu fitxategi bat balitz moduan egin daiteke

- **PrintWriter objektuan idatzi**

- `response.getWriter()`

- **Fitxategi bitarra**

MULTIMEDIA fitxategietarako, adibidez!

- `ServletOutputStream` objektuan idatzi

- `response.getOutputStream()`

- AUKERAZKOA

- HTTP-response mezuaren 1. lerroko eremuak
    - HTTP-response mezuaren goiburuak:  
`.setContentType("mime"), .addCookie(...),...`

```
HTTP_bertsioa Kodea Azalpena
Goiburu1:balio1
Goiburu2:balio2

...
lerro_hutsa
Mezuaren_gorputza:_adib_HTML_kodea

HTTP-response
```

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- HTTPrako servlet baten oinarrizko egitura
  - Zerbitzu-metodo baten eskema orokorra

Guztiak beti? Ez!! Baina guztiak behar baditugu ORDEN HONETAN.

- 1: HTTP-request mezutik informazioa atera
- 2: Kanpoko baliabideak atzitu
- 3: Datu eta baliabide guztiekin lan egin
- 4: HTTP-response mezurako idatzi: ordenean
  - HTTP-response mezuaren goiburuak
  - HTTP-response mezuaren gorputza



- Dagokion stream objektua ireki: HTML kodea edo bitarra
- *Stream* objektuarekin idatzi
- Erabilitako *stream* objektua itxi

HTTP\_bertsioa Kodea Azalpena

Goiburu1:balio1

Goiburu2:balio2

...

GoiburuN:balioN

- - - (Lerro hutsa)

Gorputza

...



# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: HTTP-request mezuan jasotako informazioaz arduratu gabe, HTML kodea sortzen

```
package pl;

//import lerroak

@WebServlet("/KaixoWWWServlet")
public class KaixoWWWServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Kaixo WWW</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello WWW</h1>");
        out.println("</body>");
        out.println("</html>");

        return;
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request,response);
        return;
    }
}
```

4. Pausua da guztia! 1, 2 eta 3 pausuak ez dira behar demo honetan.

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Servlet sinplea, klase laguntzaile (POJO) batekin hobetua

Modularitatearekin hobetutako Servlet sinplea: pl.KaixoHTMLServlet.java

```
package pl;

//import lerroak

@WebServlet("/KaixoHTMLServlet")
public class KaixoHTMLServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println(TxantiloiaPOJO.HASTERA);
        out.println(TxantiloiaPOJO.headTitle("KaixoWWW_MOD"));
        out.println("<body>");
        out.println("<h1>KAIXO MUNDU MODULARRA!!!!</h1>");
        out.println(TxantiloiaPOJO.AMAIERA);
        out.close();

        return;
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request,response);

        return;
    }
}
```

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Servlet sinplea, klase laguntzaile (POJO) batekin hobetua

```
package pl;  
  
public class TxantiloiaPOJO {  
  
    public static final String HASIERA = "<!DOCTYPE html>\n<html>";  
  
    public static String headTitle(String tit) {  
        String s="<head>";  
        s+="\n<title>" + tit + "</title>";  
        s+="\n</head>";  
  
        return s;  
    }  
  
    public static final String AMAIERA = "</body>\n</html>";  
  
}
```

HTML koderako POJO klasea: pl.TxantiloiaPOJO.java

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Web Aplikazio sinplea, Servlet eta klase laguntzaileekin (POJO)

```
<!DOCTYPE html>
<html>
<head>
<title>Aplikazio aritmetikoaren hasierako orrialdea</title>
<link rel="stylesheet" type="text/css" href="aplikAritEstiloak.css">
</head>
<body>
<h1>Sar itzazu beharrezko datuak</h1>
<form method="post" action="FormularioaJasoEtaEmitzaSortzekoServlet">
  <p>
    <input type="number" name="er1" maxlength="2" size="12" placeholder="1. eragigai">
    <select name="er">
      <option value="+">+</option>
      <option value="-">-</option>
      <option value="*">*</option>
      <option value="/">/</option>
    </select>
    <input type="number" name="er2" maxlength="2" size="11" placeholder="2. eragigai">
  </p>
  <p>
    <input type="submit" value="Bidali">
  </p>
</form>
</body>
</html>
```

Erabiltzailearen datuak jasotzeko html orrialde estatikoa: formularioa.html

```
h1 {text-align:center;}
p {text-align:center;}
```

aplikAritEstiloak.css

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Web Aplikazio sinplea, Servlet eta klase laguntzaileekin (POJO)

```
package pl;

//import lerroak

@WebServlet("/FormularioaJasoEtaEmitzaSortzekoServlet")
public class FormularioaJasoEtaEmitzaSortzekoServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println(TxantiloiaPOJO.HASIERA);
        out.println(TxantiloiaPOJO.headTitleStyle("Atzipen errorea","aplikAritEstiloak.css"));
        out.println("<body>");
        out.println("<h1>ERROREA: servlet hau POST metodoarekin bakarrik atzitu daiteke</h1>");
        out.println(TxantiloiaPOJO.AMAIERA);
        out.close();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String sEr1=request.getParameter("er1");
        String sEr=request.getParameter("er");
        String sEr2=request.getParameter("er2");

        KalkuluZerbitzuaPOJO kL=new KalkuluZerbitzuaPOJO();
        float em=kL.kalkulatu(Float.valueOf(sEr1),sEr.charAt(0),Float.valueOf(sEr2));

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println(TxantiloiaPOJO.HASIERA);
        out.println(TxantiloiaPOJO.headTitleStyle("Emitza","aplikAritEstiloak.css"));
        out.println("<body>");
        out.println("<h1>ESKATUTAKO ERAGIKETAREN EMAITZA</h1>");
        out.println("<p>"+sEr1+sEr+sEr2+"="+em+"</p>");
        out.println(TxantiloiaPOJO.AMAIERA);
        out.close();
    }
}
```

Formularioan sartutako datuak jaso, prozesatzera bidali eta bistaratzeko HTML kodea sortzen duen Servleta:  
pl.FormularioaJasoEtaEmitzaSortzekoServlet.java



# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Web Aplikazio sinplea, Servlet eta klase laguntzaileekin (POJO)

```
package pl;
```

HTML koderako POJO klasea: pl.TxantiloiaPOJO.java

```
public class TxantiloiaPOJO {  
    public static final String HASIERA = "<!DOCTYPE html>\n<html>";  
  
    public static String headTitleStyle(String tit,String styleSheetFileName) {  
        String s="<head>";  
        s+="\n<title>"+tit+"</title>";  
        s+="\n<link rel=\"stylesheet\" type=\"text/css\" href=\""+styleSheetFileName+">";  
        s+="\n</head>";  
  
        return s;  
    }  
  
    public static final String AMAIERA = "</body>\n</html>";  
}
```

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Web Aplikazio sinplea, Servlet eta klase laguntzaileekin (POJO)

Aplikazioko logikarako POJO moduko klasea: bl.KalkuluZerbitzuaPOJO.java

```
package bl;

public class KalkuluZerbitzuaPOJO {
    public float kalkulatu(float er1, char er, float er2) {
        float em;

        switch(er) {
            case '+':
                em=er1+er2;
                break;

            case '-':
                em=er1-er2;
                break;

            case '*':
                em=er1*er2;
                break;

            case '/':
                em=er1/er2;
                break;

            default:
                em=0.0f;
        }

        return em;
    }
}
```

BL klase honek ez daki Serverlet edo app lokal baten funtzionatzen ari den.

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- Servletekin sesioak kudeatzen
  - HTTPk ez du egoera mantentzen
  - Zerbitzariak memorian mantendu dezake sesioko informazioa duen objektua
  - HTTP-request mezuetan sesioko identifikatzailea bidali behar du bezeroak
    - Eskaera lerroko URLan
    - Cookie goiburuan
    - Gorputzean: formularioko eremu ezkutuan `<input type="hidden" ...>`
  - Sesioaren informazioa servletetan erabiltzeko
    - `HttpSession` objektua: `request.getSession()`
    - Sesioaren identifikatzailea: `.getId()`
    - Informazioa sartu/atera: `.setAttribute("izena", objektua)`, `.getAttribute("izena")`
    - Sesioa amaitzea: `.invalidate()`
  - Nabigatzea: nola? `<body>` barruko botoi/loturen bidez
    - `response.encodeURL("URLarrunta")` `<form action = "urlIDRekin">`  
`<a href="urlIDRekin">`

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Sesioen kudeaketa servletekin

pl.Servlet1.java: sesioa sortzen du

```
package pl;

//import lerroak

@WebServlet("/Servlet1")
public class Servlet1 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        HttpSession s=request.getSession();

        s.setAttribute("kontagailua", "0");      Kontagailua datua sortu eta 0 balioa eman, sesioaren barruan.

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println(TxantiloiaPOJO.HASIERA);
        out.println(TxantiloiaPOJO.headTitleStyle("Servlet1","aplikAritEstiloak.css"));
        out.println("<body>");
        out.println("<h1>SESIO BERRI BAT SORTU DA</h1>");
        out.println("<p>SESIOAREN ID: "+s.getId()+"</p>");

        String urlIDrekin=response.encodeURL("Servlet2");

        out.println("<form action=\""+urlIDrekin+"\" method=\"get\">");
        out.println("<p><input type=\"submit\" value=\"Hasi exekuzioa\"></p>");
        out.println("</form>");
        out.println("<a href=\""+urlIDrekin+"\" target=\"_blank\">Hurrengo</a>");

        out.println(TxantiloiaPOJO.AMAIERA);
        out.close();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Sesionen kudeaketa servletekin

pl.Servlet2.java: sesioa erabiltzen du

```
package pl;

//import lerroak

@WebServlet("/Servlet2")
public class Servlet2 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        HttpSession s=request.getSession();

        if(!s.isNew()){ //Sesioa bada existitzen dela konprobatzen du.

            String[] inc=request.getParameterValues("gehikuntzak");

            if(inc!=null) {
                int c=Integer.parseInt((String)s.getAttribute("kontagailua"));

                for(int i=0;i<inc.length;i++)
                    c+=Integer.parseInt(inc[i]);

                s.setAttribute("kontagailua",""+c);
            }

            //HEMENGOKO KODE GUZTIA HURRENGO ORRIALDEAN DAGO
        }
    }
}
```

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Sesioen kudeaketa servletekin

//HEMENGOKODEGUZTIAAURREKOORRIALDEANDAGO

pl.Servlet2.java: sesioa erabiltzen du

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();

out.println(TxantiloiaPOJO.HASIERA);
out.println(TxantiloiaPOJO.headTitleStyle("Servlet2","aplikAritEstiloak.css"));
out.println("<body>");
out.println("<h1>Sesio berdinean gaude</h1>");
out.println("<p>SESIONAREN ID: "+s.getId()+"</p>");
out.println("<h1>Kontagailuaren balioa orain: "+s.getAttribute("kontagailua")+"</h1>");

String urlIDrekin=response.encodeURL("Servlet2");
out.println("<form action='"+urlIDrekin+"' method='get'>");
out.println("<p>Aukeratu zenbat gehitu nahi duzun kontagailura:</p>");

out.println("<p>+1<input type='checkbox' name='gehikuntzak' value='1'></p>");
out.println("<p>+2<input type='checkbox' name='gehikuntzak' value='2'></p>");
out.println("<p>+3<input type='checkbox' name='gehikuntzak' value='3'></p>");

out.println("<p><input type='submit' value='Gehitu kontagailuan'></p>");
out.println("</form>");

urlIDrekin=response.encodeURL("Servlet3");
out.println("<a href='"+urlIDrekin+"' target='_blank'>Hurrengo</a>");

out.println(TxantiloiaPOJO.AMAIERA);
out.close();

}
else
    System.out.println("SESION BERRIA: Servlet HAU EZIN DA SESION BERRIAN EXEKUTATU");
}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    doGet(request,response);
}
}
```

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Sesionen kudeaketa servletekin

pl.Servlet3.java: sesioa amaitzen du

```
package pl;

//import lerroak

@WebServlet("/Servlet3")
public class Servlet3 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
        HttpSession s=request.getSession();

        if(!s.isNew()) {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            out.println(TxantiloiaPOJO.HASIERA);
            out.println(TxantiloiaPOJO.headTitleStyle("Servlet3","aplikAritEstiloak.css"));
            out.println("<body>");
            out.println("<h1>Sesioaren amaiera</h1>");
            out.println("<p>SESIONAREN ID: "+s.getId()+"</p>");
            out.println("<h1>Kontagailuaren azken balioa: "+s.getAttribute("kontagailua")+"</h1>");

            s.invalidate();

            out.println(TxantiloiaPOJO.AMAIERA);
            out.close();
        }
        else
            System.out.println("SESION BERRIA: Servlet HAU EZIN DA SESION BERRIAN EXEKUTATU");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```

//AURREKO ADIBIDEKO KODE BERDINA

HTML koderako POJO klasea: pl.TxantiloiaPOJO.java

//AURREKO ADIBIDEKO KODE BERDINA

HTML koderako CSS fitxategia: aplikAritEstiloak.css

# 3.- SERVER-SIDE SCRIPTING

## 3.2.- Servlet

- SERVLET ADIBIDEAK: Sesionen kudeaketa servletekin



# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

2.- CLIENT-SIDE SCRIPTING

3.- SERVER-SIDE SCRIPTING

**4.- DISEINURAKO PATROIAK**

**4.1.- Sarrera**

4.2.- JAVA inplementazioetako klase-motak

4.3.- MVC patroia

4.4.- JavaEEn erabilitako patroien adibideak

**ERREFERENTZIAK**

# 4.- DISEINURAKO PATROIAK

## 4.1.- Sarrera

- Definizioa Programazio-arazo baten irtenbide berrerabilgarria eta eraginkorra  
OOP: Klase eta objektu batzuen egitura
- Garrantzia
- Patroi motak
  - Sorkuntza: factory, builder, singleton,...
  - Egitura: composite, adapter,...
  - Portaera: iterator, observer,...
  - Concurrence: lock, monitor, scheduler,...
  - Arkitektura: module, layer, tier, MVC,...
- Kontzeptu orokorrak
  - Lengoaia bakoitzak bere implementazio zehatza

# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

2.- CLIENT-SIDE SCRIPTING

3.- SERVER-SIDE SCRIPTING

4.- DISEINURAKO PATROIAK

4.1.- Sarrera

4.2.- **JAVA inplementazioetako klase-motak**

4.3.- MVC patroia

4.4.- JavaEEn erabilitako patroien adibideak

ERREFERENTZIAK

# 4.- DISEINURAKO PATROIAK

## 4.2.- JAVA inplementazioetako klase-motak

- POJO (Plain Old Java Object)

- Java klase arruntak
- Derrigorrez bete beharreko baldintzarik gabe

- JavaBean BEANS!!

- Serializagarria den POJOa
- Derrigorrez bete beharreko baldintzak

- Parametrorik gabeko eraikitzaile publikoa

Baina parametrodun eraikitzaileak ere egin daitezke, hau da, eraikitzaile hau EZ DA ZERTAN BAKARRA IZAN BEHAR.

- Atributu guztiak pribatuak

- Beste baldintzak
  - Atributuen atzipenerako getter eta setter publikoak
    - getter: **get***AtributuIzena()*
    - setter: **set***AtributuIzena(atributurakoParametroa)*
- Serializable interfazea inplementatzea

# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

2.- CLIENT-SIDE SCRIPTING

3.- SERVER-SIDE SCRIPTING

4.- DISEINURAKO PATROIAK

4.1.- Sarrera

4.2.- JAVA inplementazioetako klase-motak

**4.3.- MVC patroia**

4.4.- JavaEEn erabilitako patroien adibideak

ERREFERENTZIAK

# 4.- DISEINURAKO PATROIAK

## 4.3.- MVC patroia

- Giza-erabiltzailentzako aplikazioak sortzeko oso erabilia

- Atalak

- **Model: M** Datuen eredua

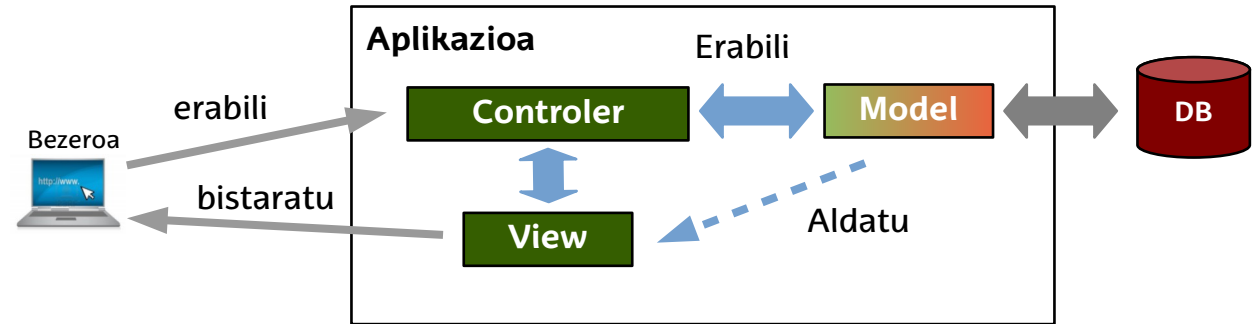
- Negozioaren logika eta biltegitratutako datuak
    - Beste bi atalekin erlazionatzen da

- **View: V** Erabiltzailearekiko Interfazea

- Erabiltzaileari M adierazten dio
    - Erabiltzaileari datuak sartu eta C erabiltzeko gaitasuna ematen dio

- **Controller: C** Aplikazioaren gidaritza

- Erabiltzailearen eskaerak jasotzen ditu V bidez
    - M erabiltzen du
    - V kontrolatzen du



# WEB ORRIALDE DINAMIKOAK

## EDUKIA

1.- SARRERA

2.- CLIENT-SIDE SCRIPTING

3.- SERVER-SIDE SCRIPTING

**4.- DISEINURAKO PATROIAK**

4.1.- Sarrera

4.2.- JAVA inplementazioetako klase-motak

4.3.- MVC patroia

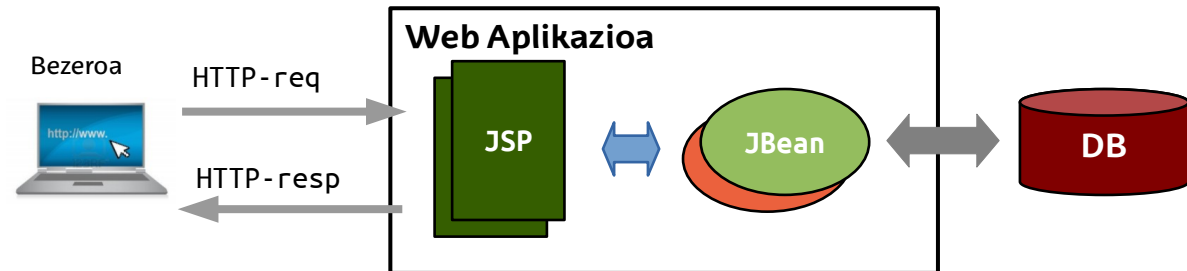
**4.4.- JavaEEn erabilitako patroien adibideak**

ERREFERENTZIAK

# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

- Model 1 *Abandonatuta dago orain gehien bat*
  - Web Aplikazio sinpleetarako
  - JavaBeans
    - HTTP-response mezuaren gorputza sortzeko informazioa
  - Aplikazio konplexuetan arazoak
    - JSPak: HTML eta scriptlet kode zatien nahasketa
    - Nabigatze deszentralizatua





# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

- Model 2

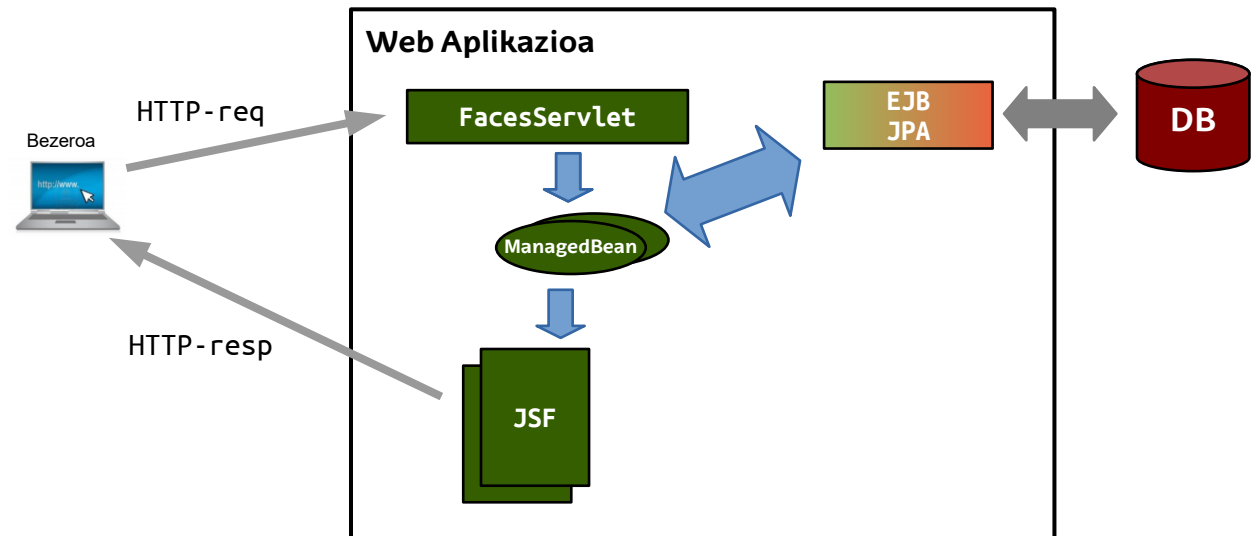
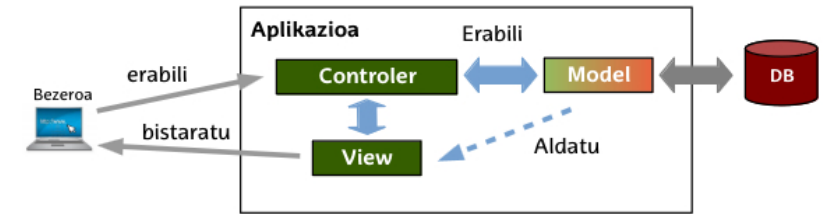
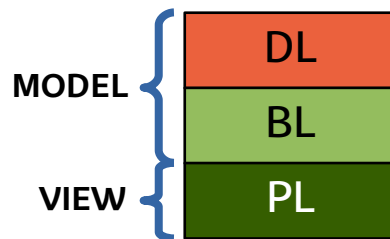
- Web Aplikazio konplexuak: kasu orokorra
- MVC patroia egokitzapena

- Model: JavaBeans, EJBs, WebServices, POJOs
- View: JSFs (hasiera batean JSPs), **ManagedBeans**, JavaBeans
- Controller: Servlet (**FacesServlet**)

JavaBeans bereziak  
Controller Zatia: Guk ez dugu programatu behar:  
Liburutegi bat da "FacesServlet"

- Implementazioa

- 3 mailetako arkitektura

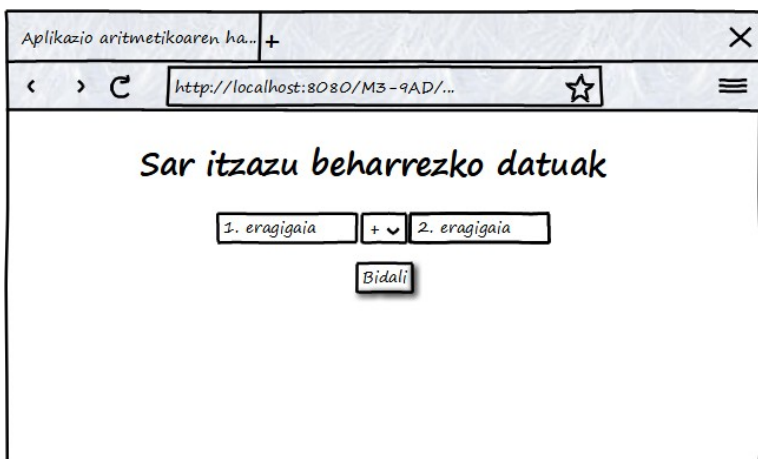


# 4.- DISEINURAKO PATROIAK

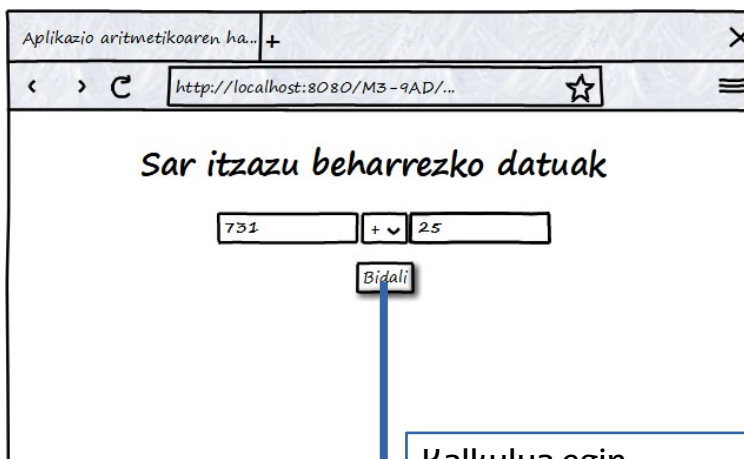
## 4.4.- JavaEEn erabilitako patroien adibideak

- SERVLET ADIBIDEAK: MVC servletekin

1. pantaila: Hasierako egoera



1. pantaila: Erabiltzaileak formularioa beteta



Atal hau estatikoki programatu dezakegu HTML arruntarekin.

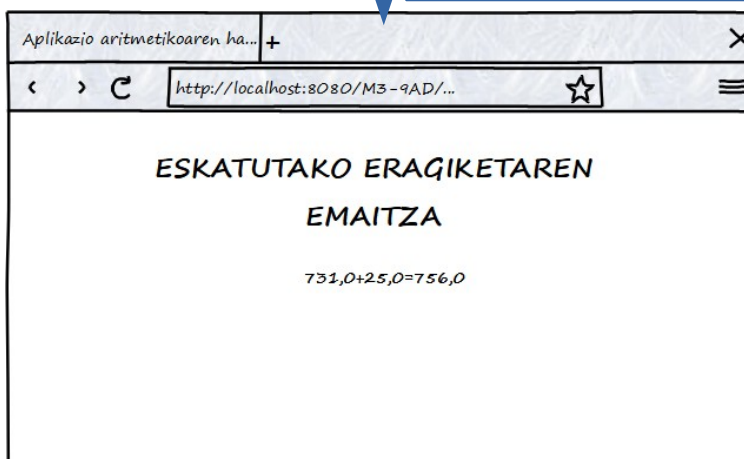
Zerbitzariko  
Web Aplikazioan

Kalkulua egin

Eragiketa CSV fitxategi batean sartu

2. pantailara

2. pantaila: Emaita



Zerbitzariaren web aplikazioan, emandako sarrerak izanda dinamikoki sortutako HTML-a

Servlet batekin, JavaBeans batekin...

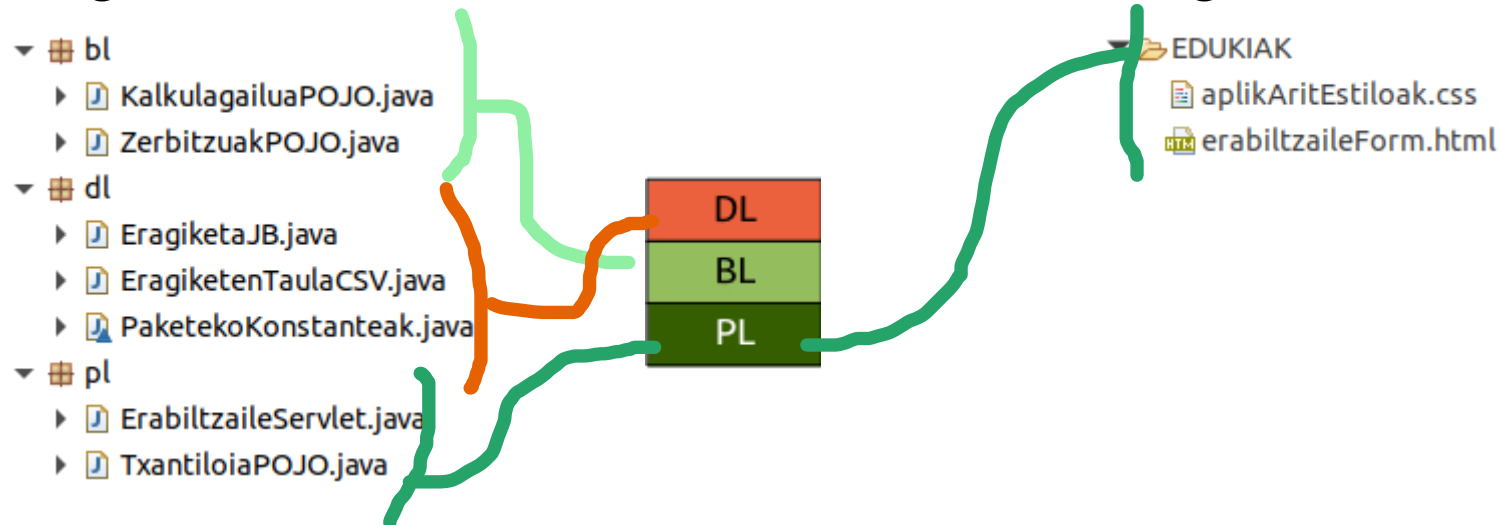
# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

- SERVLET ADIBIDEAK: MVC servletekin

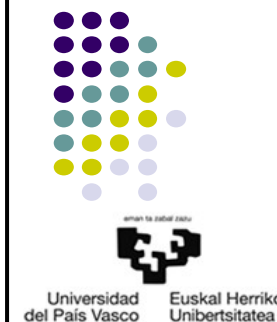
– JAVA osagaiak: **java**

– HTML osagaiak: **webapp**

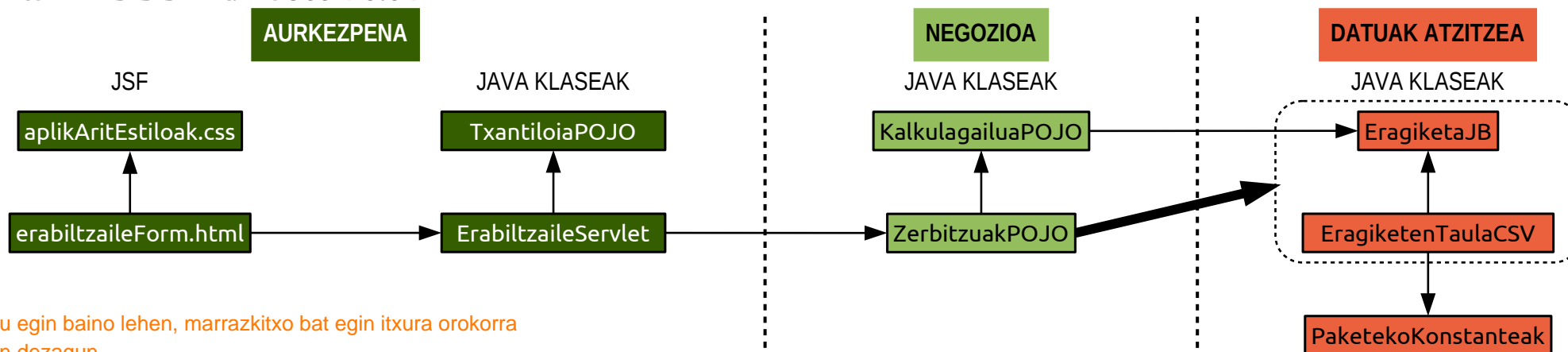


# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak



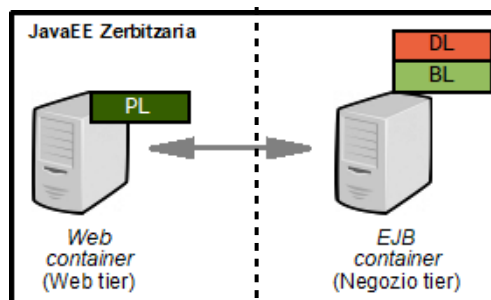
- SERVLET ADIBIDEAK: MVC servletekin



Hau egin baino lehen, marrazkitxo bat egin itxura orokorra izan dezagun.  
Baita ere DL parteko fitxategian/datu basean zer gorde behar dugun.

Programatzeko azken ataletako bat izan behar da, beste guztiaren dependentea delako!

Beti DL partetik programatzen hasiko gara, dependentziarik ez dutelako beste parteeetara.



# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

DL

- SERVLET ADIBIDEAK: MVC servletekin JB = JavaBean

```
package dl;

public class EragiketaJB {
    private float eragigai1;
    private String eragilea;
    private float eragigai2;
    private float emaitza;

    public EragiketaJB() {
    }

    public EragiketaJB(float eragigai1, String eragilea, float eragigai2, float emaitza) {
        this.eragigai1 = eragigai1;
        this.eragilea = eragilea;
        this.eragigai2 = eragigai2;
        this.emaitza = emaitza;
    }

    public float getEragigai1() {
        return eragigai1;
    }

    public void setEragigai1(float eragigai1) {
        this.eragigai1 = eragigai1;
    }

    //Beste atributuen getter eta setter metodoak

    public String laburpena() {
        return "" + eragigai1 + eragilea + eragigai2 + "=" + emaitza;
    }

    public String toCSV() {
        return eragigai1 + ";" + eragilea + ";" + eragigai2 + ";" + emaitza;
    }
}
```

dl.EragiketaJB.java

# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEn erabilitako patroien adibideak

DL

- SERVLET ADIBIDEAK: MVC servletekin

dl.EragiketenTaulaCSV.java

```
package dl;

//import lerroak

public class EragiketenTaulaCSV {

    private List<EragiketaJB> guztiakIrakurri() {
        File fitxategiaCSV = new File(PaketekoKonstanteak.csvFitxategiIzena);
        List<EragiketaJB> eragiketakJB=new ArrayList<EragiketaJB>();

        if(fitxategiaCSV.exists()) {
            BufferedReader br;
            try {
                br = new BufferedReader(new FileReader(fitxategiaCSV));

                String lerroa;
                String[] datuak;
                EragiketaJB eJB;
                while( (lerroa=br.readLine()) != null ) {
                    datuak = lerroa.split(";");
                    eJB=new EragiketaJB(Float.parseFloat(datuak[0]), datuak[1], Float.parseFloat(datuak[2]), Float.parseFloat(datuak[3]));
                    eragiketakJB.add(eJB);
                }
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        return eragiketakJB;
    }

    //HEMENGOKO KODE GUZTIA HURRENGO ORRIALDEAN DAGO
}
```

# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

DL

- SERVLET ADIBIDEAK: MVC servletekin

//HEMENGOKODEGUZTIAAURREKOORRIALDEANDAGO

dl.EragiketenTaulaCSV.java

```
private void guztiakIdatzi(List<EragiketaJB> eragiketakJB) {
    try {
        PrintWriter wr = new PrintWriter(new FileWriter(PaketekoKonstanteak.csvFitxategiIzena));

        EragiketaJB eJB;
        for(int i=0;i<eragiketakJB.size();i++) {
            eJB=eragiketakJB.get(i);
            wr.println(eJB.toCSV()); Fitxategira printeatu
        }

        wr.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void eragiketaSartuDB(EragiketaJB eragiketaJB) {
    List<EragiketaJB> eragiketakJB=guztiakIrakurri();

    eragiketakJB.add(eragiketaJB);

    guztiakIdatzi(eragiketakJB);

    return;
}
```

package dl;

dl.PaketekoKonstanteak.java

```
class PaketekoKonstanteak {
    static final String csvFitxategiIzena=System.getProperty("user.home")+"/ProiektuarenKarpetaHometik/Eragiketak.csv"; Gure karpetaren arabera!
}
```

# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

BL

- SERVLET ADIBIDEAK: MVC servletekin

bl.KalkulagailuaPOJO.java

```
package bl;

import dl.EragiketaJB;

public class KalkulagailuaPOJO {
    public void kalkulatu(EragiketaJB eragiketaJB) {
        float er1=eragiketaJB.getEragigai1();
        char er=eragiketaJB.getEragilea().charAt(0);
        float er2=eragiketaJB.getEragigai2();

        float em;

        switch(er) {
            case '+':
                em=er1+er2;
                break;

            case '-':
                em=er1-er2;
                break;

            case '*':
                em=er1*er2;
                break;

            case '/':
                em=er1/er2;
                break;

            default:
                em=0.0f;
        }

        eragiketaJB.setEmaizta(em);
    }
}
```

JavaBean-aren atributuan jartzen dugu emaitza, ez zuzenean irteerataraz!



# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

BL

- SERVLET ADIBIDEAK: MVC servletekin

```
package bl;

import dl.EragiketenTaulaCSV;
import dl.EragiketaJB;

public class ZerbitzuakPOJO {
    EragiketenTaulaCSV etCSV;
    KalkulagailuaPOJO kPOJO;

    public ZerbitzuakPOJO() {
        etCSV=new EragiketenTaulaCSV();
        kPOJO=new KalkulagailuaPOJO();
    }

    public void eragiketaEginBiltegitatu(EragiketaJB eragiketaJB) {
        kPOJO.kalkulatu(eragiketaJB);
        etCSV.eragiketaSartuDB(eragiketaJB);
    }
}
```

bl.ZerbitzuakPOJO.java

# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

- SERVLET ADIBIDEAK: MVC servletekin

```
package pl;
//import lerroak

@WebServlet("/EDUKIAK/ErabiltzaileServlet") Servlet-a /EDUKIAK/ karpetan balego bezala jarriko dugu, gauzak sinplifikatzeko.
public class ErabiltzaileServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println(TxantiloiaPOJO.HASIERA);
        out.println(TxantiloiaPOJO.headTitleStyle("Atzipen errorea","aplikAritEstiloak.css"));
        out.println("<body>");
        out.println("<h1>ERROREA: servlet hau POST metodoarekin bakarrik atzitu daiteke</h1>");
        out.println(TxantiloiaPOJO.AMAIERA);
        out.close();
    }

    //HEMENGOKO KODE GUZTIA HURRENGO ORRIALDEAN DAGO
}
```

pl.ErabiltzaileServlet.java

# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEn erabilitako patroien adibideak

- SERVLET ADIBIDEAK: MVC servletekin

//HEMENGOKO KODE GUZTIA AURREKO ORRIALDEAN DAGO

pl.ErabiltaileServlet.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    String sEr1=request.getParameter("er1");  
    String sEr=request.getParameter("er");  
    String sEr2=request.getParameter("er2");  
  
    EragiketaJB eragiketaJB=new EragiketaJB(Float.valueOf(sEr1), sEr, Float.valueOf(sEr2), 0.0f);  
    ZerbitzuakPOJO zPOJO=new ZerbitzuakPOJO();  
    zPOJO.eragiketaEginBiltegitatu(eragiketaJB);  
  
    String mezua=eragiketaJB.laburdura();  
  
    response.setContentType("text/html");  
  
    PrintWriter out = response.getWriter();  
    out.println(TxantiloiaPOJO.HASIERA);  
    out.println(TxantiloiaPOJO.headTitleStyle("Emaiza","aplikAritEstiloak.css"));  
    out.println("<body>");  
    out.println("<h1>ESKATUTAKO ERAGIKETAREN EMAITZA</h1>");  
    out.println("<p>"+mezua+"</p>");  
    out.println(TxantiloiaPOJO.AMAIERA);  
    out.close();  
}
```

1: Request-eko informazioa lortu

2: Beste baliabideak lortu

3: datuak prozesatu

4: HTTP-response mezua prestatu

//AURREKO ADIBIDEKO KODE BERDINA

pl.TxantiloiaPOJO.java

# 4.- DISEINURAKO PATROIAK

## 4.4.- JavaEEn erabilitako patroien adibideak

- SERVLET ADIBIDEAK: MVC servletekin

EDUKIAK/erabiltzaileForm.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Aplikazio aritmetikoaren hasierako orrialdea</title>
  <link rel="stylesheet" type="text/css" href="./aplikAritEstiloak.css">
</head>
<body>
  <h1>Sar itzazu beharrezko datuak</h1>
  <form method="post" action="ErabiltzaileServlet">
    <p>
      <input type="number" name="er1" maxlength="2" size="12" placeholder="1. eragigai">
      <select name="er">
        <option value="+">+</option>
        <option value="-">-</option>
        <option value="*">*</option>
        <option value="/">/</option>
      </select>
      <input type="number" name="er2" maxlength="2" size="12" placeholder="2. eragigai">
    </p>
    <p>
      <input type="submit" value="Bidali">
    </p>
  </form>
</body>
</html>
```

//AURREKO ADIBIDEKO KODE BERTINA

EDUKIAK/aplikAritEstiloak.css

# WEB ORRIALDE DINAMIKOAK

## ERREFERENTZIAK

- [1] HTML: <http://www.w3schools.com/html/default.asp>
- [2] CSS: <http://www.w3schools.com/css/default.asp>
- [3] JavaScript: <http://www.w3schools.com/js/default.asp>
- [4] DOM: <http://www.w3schools.com/jsref/default.asp>
- [5] Java Servlet Technology:  
<http://docs.oracle.com/javaee/7/tutorial/servlets.htm#BNAFD>
- [6] “Design Patterns. Elements of Reusable Object-Oriented Software”. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (GoF- Gang of Four). Addison Wesley