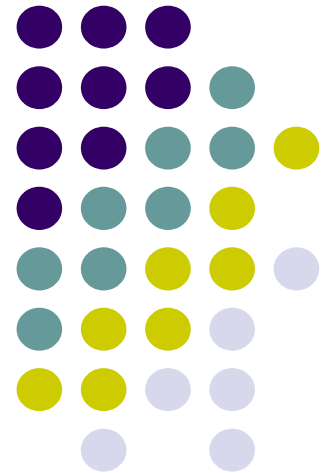


5.- AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEEⁿ ESKEMAK

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

2023-2024

Egileak: Maider Huarte Arrayago, Gorka Prieto Agujeta



2023-2024 ZTA 5 AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEEEn - ESKEMAK.odp



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



2023-2024 ZTA 5 AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEEEn - ESKEMAK.odp lana, Maider Huartek eta Gorka Prietok egina, 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 5 AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEEEn - 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.

AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEEñ EDUKIA

- 1.- SARRERA
 - 2.- REST MOTAKO WEB ZERBITZUAK
 - 3.- APLIKAZIOAK KONFIGURATZEA
- ERREFERENTZIAK

AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEE_n EDUKIA

1.- SARRERA

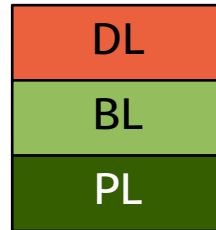
2.- REST MOTAKO WEB ZERBITZUAK

3.- APLIKAZIOAK KONFIGURATZEA

ERREFERENTZIAK

1.- SARRERA

- Zer da Web Zerbitzu bat? HTTP bidez atzitutako aplikazio informatikoa, AURKEZPENIK GABEKOA
 - JavaEEren definizioa
 - Aurkezpeneko web aplikazioekin ezberdintasunak
 - SOAren erlazioa SOA arkitektura jarraituta inplementatzen dira.
- Zer da SOA (Service Oriented Architecture)?
 - SWa eraikitze metodoa Moduluen konbinazioa
 - Zerbitzuak Moduluak: Software unitate sinpleak
 - Berrerabilpena
 - Elkarlana
 - Teknologiekiko independentzia Edozein lengoia egin daitezke
- Zer da SOA zerbitzu bat? Software unitate funtzional desakoplatua



1.- SARRERA

- Web Zerbitzu motak
 - Web Zerbitzu “astunak”
 - SOAP Erabiltzeko, SML mezu bereziak HTTP bidez garraiatuak
 - WSDL
 - Web Zerbitzu “arinak” Erabiltzeko, HTTPko mezuak zuzenean
 - REST funtzionaltasunak
 - Egoera mantentzea behar ez dutenak
 - Hardware mugak dituzten bezeroentzakoak
- REST motako Web zerbitzu ezagunak: GoogleMaps, Twitter,...

AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEE_n EDUKIA

1.- SARRERA

2.- REST MOTAKO WEB ZERBITZUAK

2.1.- Sarrera

2.2.- REST baliabideak JavaEErekin kodifikatzea

2.3.- JAX-RS bezeroa

3.- APLIKAZIOAK KONFIGURATZEA

ERREFERENTZIAK

2.- REST MOTAKO WEB ZERBITZUAK

2.1.- Sarrera

- 2 tier-eko sw arkitektura, egoerarik mantentzen ez duen protokoloarekin
- Baliabideak: URL-ekin atzigarriak
- REST aplikazioetarako printzipioak
 - Zerbitzuak webguneko baliabide mota bat dira
 - Baliabideekin lan egiteko HTTP eragiketak

JAVAEEn: Klase bateko metodoak

HTTP-request mezu motak

AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEE_n EDUKIA

1.- SARRERA

2.- REST MOTAKO WEB ZERBITZUAK

2.1.- Sarrera

2.2.- REST baliabideak JavaEErekin kodifikatzea

2.3.- JAX-RS bezeroa

3.- APLIKAZIOAK KONFIGURATZEA

ERREFERENTZIAK

2.- REST MOTAKO WEB ZERBITZUAK

2.2.- REST baliabideak JavaEerekin kodifikatzea

- REST zerbitzuak: POJO klasea+JAX-RS oharrak

- JAX-RS oharrak (klase/metodoak)

Nola deitu?
Zer bidali?
Zer jaso?

- @Path
- @GET, @POST, @PUT, @DELETE
- @PathParam, @QueryParam, @FormParam
- @Consumes, @Produces

```
Metodoa URL HTTP_bertsioa  
Goiburu1:balio1  
Goiburu2:balio2
```

```
...  
lerro_hutsa  
[Mezuaren_gorputza]
```

HTTP-request

- REST baliabide izateko Java metodoak

DERRIGORREZKO EZAUGARRIAK

- Atzipena `public`
- Parametroak `metodoIzena(...)`
 - Parametrorik gabe
 - Java mota primitiboa(k) edo **String**
 - JB objektua: BAKARRA
- Itzulera balioa `return(...)`
 - **void**
 - **javax.ws.rs.core.Response**
 - Java mota primitiboa edo **String**
 - JB objektua

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

```
...  
lerro_hutsa  
[Mezuaren_gorputza: testua/XML/JSON]
```

HTTP-response

2.- REST MOTAKO WEB ZERBITZUAK

2.2.- REST baliabideak JavaEerekin kodifikatzea

- Oinarrizko adibidea
 - URL: `http://domeinua/proiektua/rest/Ad1`

```
package bl;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

@Path("/Ad1")
public class Ad1 {

    @GET
    public Response ongietorriMezua() {

        String result = "KAIXO MUNDUA!";

        return Response.status(200).entity(result).build();

    }
}
```



2.- REST MOTAKO WEB ZERBITZUAK

2.2.- REST baliabideak JavaEerekin kodifikatzea

- REST baliabideak programatzeko JAX-RS oharrak

1: Baliabidea deitzeko URLa zehazteko oharrak Zein URL-rekin deitu REST zerbitzua

- `@Path("URL")`: Java klase eta metodoetarako

2: Java metodoaren parametroei buruzko oharrak metodoIzena(...)

- Parametroen balioak **HTTP-request** mezuan non iritsiko diren
- Java mota primitiboa(k) edo **String**: parametroetarako oharrak

- Parametroei banan-banan jartzeko, **guztiak berdin**

- `@QueryParam("datuName")` URL?datuName1=balio1&...%datuNameN=balioN

3000 karaktereko maximoa URLan!!

- `@FormParam("datuName")` HTTP-request gorputzean: datuName1=balio1&...%datuNameN=balioN

- JB objektua: java metodorako oharra BAKARRA

- `@Consumes(MIMEmota)` HTTP-request gorputzean: objektua MIMEmota formatuan

Sarrerako gauzetarako

- *MIMEmota*: “application/xml”, “application/json”

- **Error HTTP 415 Unsupported Media Type** "Jasotako JAXB objektua ez dago formatu egokian"

2.- REST MOTAKO WEB ZERBITZUAK

2.2.- REST baliabideak JavaEerekin kodifikatzea

- REST baliabideak programatzea

3: Baliabidea deitzeko HTTP-request motari buruzko oharra

- Metodora oharra

- **@GET**

- Java metodoaren parametroa(k) @QueryParam

EDO BATA EDO BESTEA

- **@POST**

- Java metodoaren parametroa(k) @FormParam @Consumes

NAHASTEK EZ DUTE
FUNZIONATUKO

4: Baliabideak HTTP-response mezuan itzuliko duenari buruzko oharra

- Java metodora oharra
- Java mota primitiboa edo **String**: oharrik gabe edo **@Produces("text/plain")**
- JB objektua: **@Produces(MIMEmota)** Irteerako gauzetarako
 - *MIMEmota*: "application/xml", "application/json"
 - **Error HTTP 406 Not Acceptable** "Zerbitzuak emaitza ez du eskatutako formatuan sortzen"
- **void**: oharrik gabe

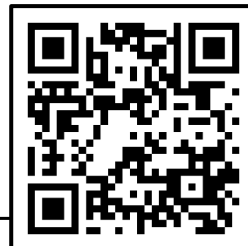
2.- REST MOTAKO WEB ZERBITZUAK

2.2.- REST baliabideak JavaEerekin kodifikatzea

- Formularioen adibidea

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Formularioak dituen orrialdea, frogak egiteko</title>
</head>
<body>
  <h1>Formularioak dituen orrialdea, frogak egiteko</h1>
  <form action="rest/Ad3/GETformularioa" method="get">
    <h2>GET FORMULARIOA</h2>
    <input type="text" name="er1" />
    <select name="er">
      <option value="+">+</option>
      <option value="-">-</option>
      <option value="*">*</option>
      <option value="/">/</option>
    </select>
    <input type="text" name="er2" />
    <br>
    <input type="submit" value="KALKULATU" />
  </form>
  <form action="rest/Ad3/POSTformularioa" method="post">
    <h2>POST FORMULARIOA</h2>
    <input type="text" name="er1" />
    <select name="er">
      <option value="+">+</option>
      <option value="-">-</option>
      <option value="*">*</option>
      <option value="/">/</option>
    </select>
    <input type="text" name="er2" />
    <br>
    <input type="submit" value="KALKULATU" />
  </form>
</body>
</html>
```

formularioak.html



2.- REST MOTAKO WEB ZERBITZUAK

2.2.- REST baliabideak JavaEerekin kodifikatzea

- Formularioen adibidea

```
package bl;

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import javax.ws.rs.FormParam;
import javax.ws.rs.core.Response;

@Path("/Ad3")
public class Ad3 {

    @Path("/GETformularioa")
    @GET
    public Response GETmetodoa(@QueryParam("er1") float er1,@QueryParam("er") String er,@QueryParam("er2") float er2) {

        String mezua="Formularioan adierazitako eragiketa hau izan da: "+er1+er+er2;

        return Response.status(200).entity(mezua).build();
    }

    @Path("/POSTformularioa")
    @POST
    public Response POSTmetodoa(@FormParam("er1") float er1,@FormParam("er") String er,@FormParam("er2") float er2) {

        String mezua="Formularioan adierazitako eragiketa hau izan da: "+er1+er+er2;

        return Response.status(200).entity(mezua).build();
    }
}
```

bl.Ad3.java

2.- REST MOTAKO WEB ZERBITZUAK

2.2.- REST baliabideak JavaEerekin kodifikatzea

- MIME adibidea

```
package bl;

import java.io.File;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.ResponseBuilder;

@Path("/Ad4")
public class Ad4 {

    private static final String FILE = System.getProperty("user.home")+"/PATH/frogetarakoFitxategia.pdf";

    @GET
    @Produces("application/pdf")
    public Response getFile() {
        File file = new File(FILE);

        ResponseBuilder response = Response.ok((Object) file);
        response.header("Content-Disposition", "attachment; filename="+FILE);
        return response.build();
    }
}
```

bl.Ad4.java



2.- REST MOTAKO WEB ZERBITZUAK

2.2.- REST baliabideak JavaEerekin kodifikatzea

- POJO klase bat REST zerbitzu bezala eskaintzea Zein ohar?
 - **@Path**: klasean eta baliabide izatea nahi den metodoetan
 - Baliabide metodoen araberakoak
 - Parametroak
 - Java mota primitiboa(k) edo **String** @QueryParam(...) parametro(et)an @GET java metodoan
@FormParam(...) parametro(et)an @POST java metodoan
 - JB objektua @Consumes(...) eta @POST java metodoan etodoan
 - Beste kasu batzuk: **List** bat, JB batzuk, + datu sinple batzuk,...
 - JB wrapper klase berria sortu Barruan List, JB objektuak + datu sinpleak
 - Oharrak @Consumes(...) eta @POST java metodoan
 - Itzulera balioa: KONTUZ!
 - Java mota primitiboa / **String** @Produces("text/plain") oharra java
 - JB objektua @Produces(...) oharra java metodoan
 - Beste kasu batzuk (**List** bat, JB batzuk, + datu sinple batzuk,...): JB wrapper sortu
 - **void** → JB ?

AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEE_n EDUKIA

1.- SARRERA

2.- REST MOTAKO WEB ZERBITZUAK

2.1.- Sarrera

2.2.- REST baliabideak JavaEErekin kodifikatzea

2.3.- JAX-RS bezeroa

3.- APLIKAZIOAK KONFIGURATZEA

ERREFERENTZIAK

2.- REST MOTAKO WEB ZERBITZUAK

2.3.-JAX-RS bezeroa

- Pausoak REST zerbitzuak beste JAVAEE klaseetatik erabiltzeko
 - Bezero objektua sortu: `Client c= ClientBuilder.newClient();`
 - WebTarget objektua sortu
`WebTarget wt=c.target("URL");` Zerbitzuaren URL: PATH? QUERY-STRING
 - HTTP-request sortu
`Invocation.Builder req=wt.request(MediaType);` HTTP-response gorputzean JASOKO denaren MIME formatua
 - HTTP-request bidali
`req.get(...), req.post(...), req.delete(...), req.put(...),...`
 - GET: () edo `(XXX.class)` HTTP-response gorputzean JASOKO dena zein klaseko objektuan sartu
 - POST: () edo `(XXX.class)` edo `(Entity.entity(bean,mime), XXX.class)` HTTP-request gorputzean BIDALTZEN dena
- Guztia pauso bakarrean
`ClientBuilder.newClient().target("URL").request(MediaType).get(...)`
`ClientBuilder.newClient().target("URL").request(MediaType).post(...)`

2.- REST MOTAKO WEB ZERBITZUAK

2.3.-JAX-RS bezeroa

- `get(...): @Produces(MediaType.APPLICATION_JSON)` zerbitzu baterako
 - JAXB objektua (`@Xml` oharrekin apaindutako JavaBean klasea) lortzeko

```
...
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.client.Entity;
...
    HTTP-response gorputzean jasotakoa
    JavaBean bean = ClientBuilder.newClient()
        .target("http://localhost:8080/P4-2/rest/zerbitzuak/produktuakLortuDBURL")
        .request(MediaType.APPLICATION_JSON)    HTTP-response gorputzean jasoko denaren MIME formatua
        .get(JavaBean.class);

//JavaBean: get mezuaren HTTP-response gorputzean jasotzen diren datuak biltzen dituen JAXB objektuaren klasea
```

- HTTP-Response mezu osoarekin lan egin ahal izateko

```
...
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.client.Entity;
...
    HTTP-response mezu osoa
    Response r = ClientBuilder.newClient()
        .target("http://localhost:8080/P4-2/rest/zerbitzuak/produktuakLortuDBURL")
        .request(MediaType.APPLICATION_JSON)    HTTP-response gorputzean jasoko denaren MIME formatua
        .get();

    int statusCode=r.getStatus(); //HTTP-response mezuaren STATUS kodea lortzeko
    JavaBean bean=r.readEntity(JavaBean.class); //HTTP-response mezuaren gorputzeko JAXB objektua lortzeko
```

2.- REST MOTAKO WEB ZERBITZUAK

2.3.-JAX-RS bezeroa

- **post(...): @Consumes+@Produces(MediaType.APPLICATION_JSON)** zerbitzu baterako
 - JAXB objektua (@Xml oharrekin apaindutako JavaBean klasea) lortzeko

```
...
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.MediaType;
...
    JavaBean1 bean1 = new JavaBean1(...);
...
    JavaBean2 bean2 = ClientBuilder.newClient()
        .target("http://localhost:8080/P4-2/rest/zerbitzuak/produktuaSartuDBURL")
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(bean1, MediaType.APPLICATION_JSON), JavaBean2.class);
//bean1: post mezuan bidali nahi diren datuak biltzen dituen JAXB objektua, JavaBean1 klasekoa
//JavaBean2: post mezua HTTP-response erantzunean jaso nahi diren datuak biltzen dituen JAXB objektuaren klasea
```

HTTPS-reponse gorputzean jasotakoa

HTTP-response gorputzean bidalitakoa

- HTTP-Response mezu osoarekin lan egin ahal izateko

```
...
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.MediaType;
...
    Response resp = ClientBuilder.newClient()
        .target("http://localhost:8080/P4-2/rest/zerbitzuak/produktuaSartuDBURL")
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(bean, MediaType.APPLICATION_JSON));
//bean: post mezuan bidali nahi diren datuak biltzen dituen JAXB objektua
```

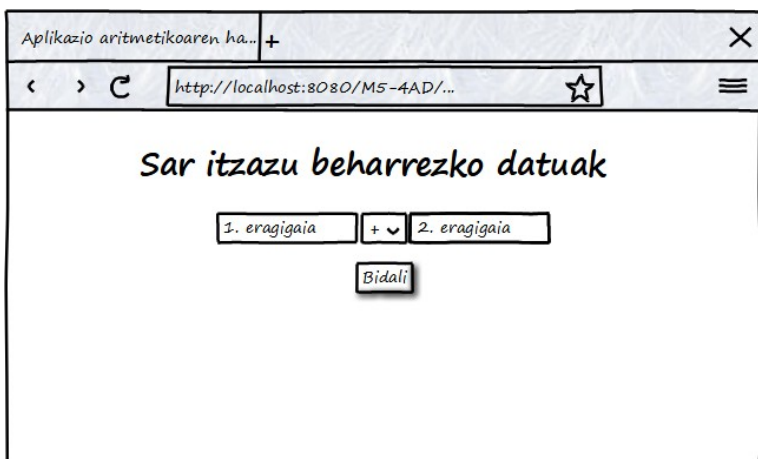
HTTP-response gorputzean bidalitakoa

2.- REST MOTAKO WEB ZERBITZUAK

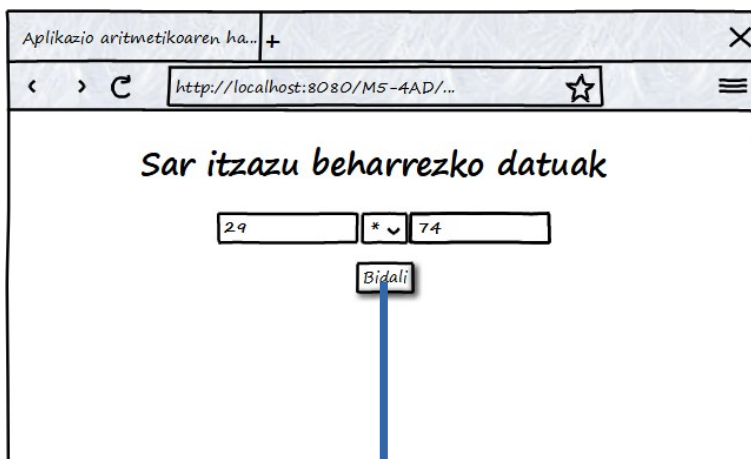
2.3.-JAX-RS bezeroa

- ADIBIDEA: JAX-RS zerbitzuak eta bezeroak

1. pantaila: Hasierako egoera

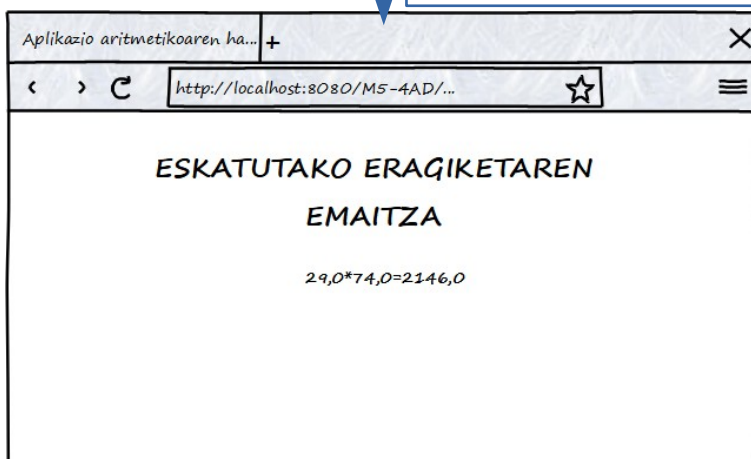


1. pantaila: Erabiltzaileak formularioa beteta



HTML estatikoa

2. pantaila: Emaita



Kalkulua REST zerbitzu batekin egin
2. pantailara

Zerbitzariko
Web aplikazioan

Web aplikazioak dinamikoki
sortutako HTML-a

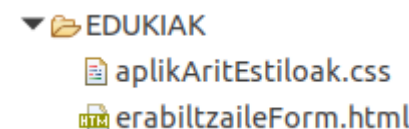
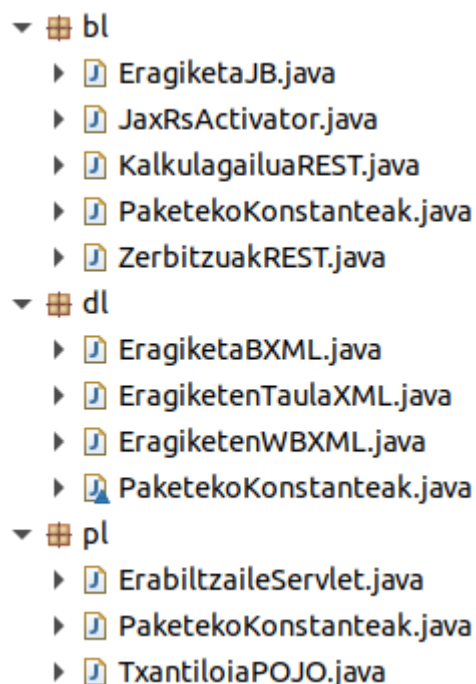
2.- REST MOTAKO WEB ZERBITZUAK

2.3.-JAX-RS bezeroa

- ADIBIDEA: JAX-RS zerbitzuak eta bezeroak

- JAVA osagaiak: **java**

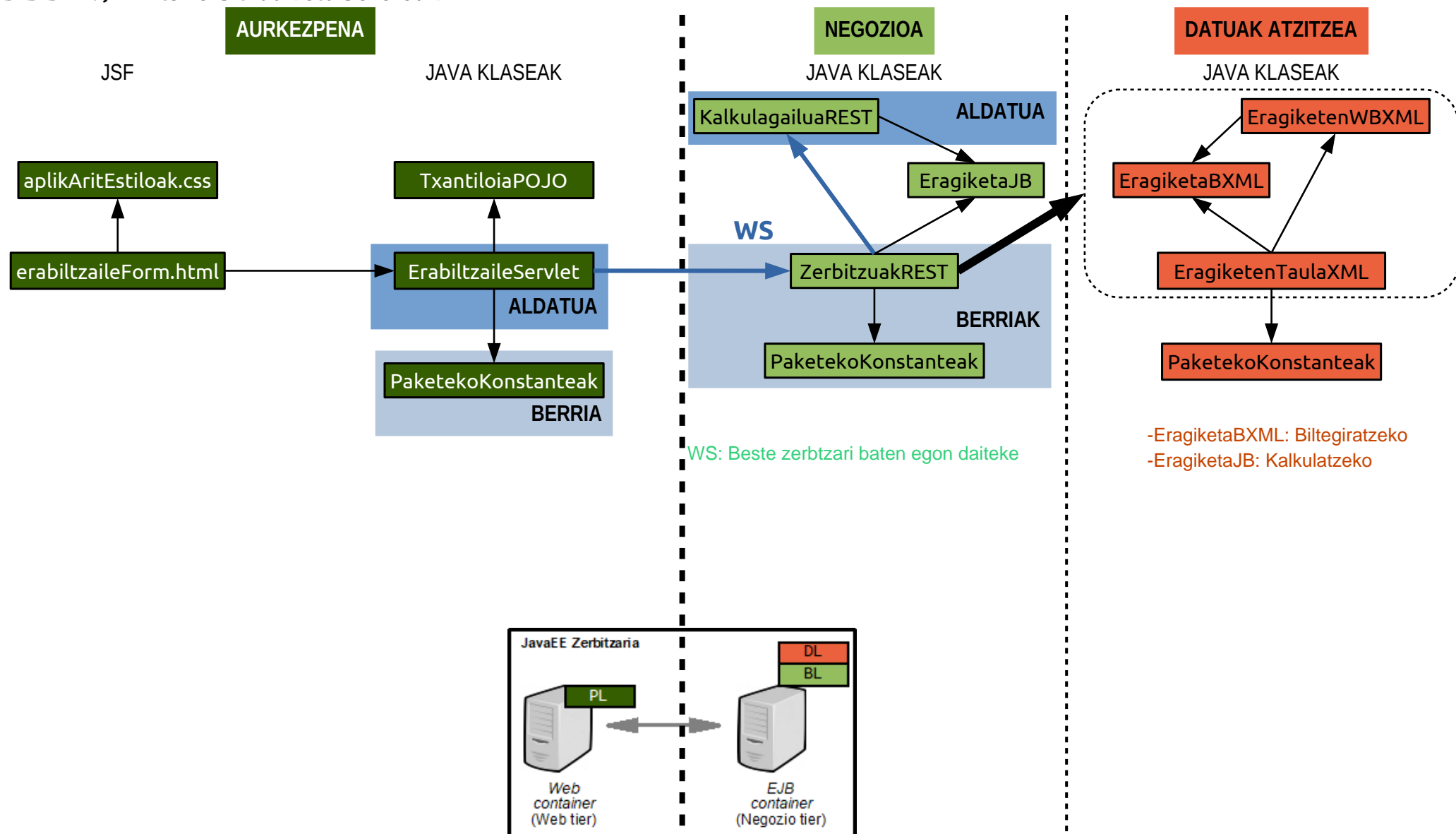
- HTML osagaiak: **webapp**



2.- REST MOTAKO WEB ZERBITZUAK

2.3.-JAX-RS bezeroa

- ADIBIDEA: JAX-RS zerbitzuak eta bezeroak



2.- REST MOTAKO WEB ZERBITZUAK

2.3.-JAX-RS bezeroa

- ADIBIDEA: JAX-RS zerbitzuak eta bezeroak

```
//AURREKO ADIBIDEKO KODE BERDINAK
```

dl.EragiketaBXML.java
dl.EragiketenWBXML.java
dl.EragiketenTaulaXML.java

```
//AURREKO ADIBIDEKO KODE BALIOKIDEA
```

dl.PaketekoKonstanteak.java

```
//AURREKO ADIBIDEKO KODE BERDINA
```

bl.EragiketaJB.java

package bl; bl.KalkulagailuaREST.java

```
import javax.ws.rs.POST;  
import javax.ws.rs.Path;  
import javax.ws.rs.Consumes;  
import javax.ws.rs.Produces;
```

```
@Path("/kalkulagailua")  
public class KalkulagailuaREST {
```

```
    @Path("/kalkulatuURL")  
    @POST  
    @Consumes("application/json")  
    @Produces("application/json")  
    public EragiketaJB kalkulatu(EragiketaJB eragiketaJB) {  
        float er1=eragiketaJB.getEragigai1();  
        char er=eragiketaJB.getEragilea().charAt(0);  
        float er2=eragiketaJB.getEragigai2();
```

```
        ...
```

```
        eragiketaJB.setEmaitza(em);
```

```
        return eragiketaJB; HTTP-responsen mezua bidaltzeko
```

```
    }  
}
```

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

4. Gaiko adibideetako KalkulagailuaPOJO metodoa, WS-ra bihurtuta

- @Path
- Parametroa JB: @Post @Consumes
- Itzulera void -> JB @Produces

2.- REST MOTAKO WEB ZERBITZUAK

2.3.-JAX-RS bezeroa

- ADIBIDEA: JAX-RS zerbitzuak eta bezeroak

```
package bl;
//import lerroak

@Path("/zerbitzuak")
public class ZerbitzuakREST {
    EragiketenTaulaXML etXML;

    public ZerbitzuakREST() {
        etXML=new EragiketenTaulaXML();
    }

    @Path("/eragiketaEginBiltegitratuURL")
    @POST
    @Consumes("application/json")
    @Produces("application/json")
    public EragiketaBXML eragiketaEginBiltegitratu(EragiketaBXML eragiketaBXML) {
        EragiketaJB eragiketaJB=new EragiketaJB(eragiketaBXML.getEragigai1(), eragiketaBXML.getEragilea(),
            eragiketaBXML.getEragigai2(), 0.0f);
        eragiketaJB=ClientBuilder.newClient()
            .target(PaketekoKonstanteak.kalkulatuURL)
            .request("application/json")
            .post(Entity.entity(eragiketaJB, MediaType.APPLICATION_JSON),EragiketaJB.class);
        eragiketaBXML.setEmaizta(eragiketaJB.getEmaizta());
        etXML.eragiketaSartuDB(eragiketaBXML);

        return eragiketaBXML;
    }
}
```

bl.ZerbitzuakREST.java

4. Gaiko adibideetako KalkulagailuaPOJO metodoa, WS-ra bihurtuta

- @Path
- Parametroa JB: @Post @CONsumes
- Itzulera void -> JB @Produces

```
package bl;
class PaketekoKonstanteak {
    public static final String kalkulatuURL="http://localhost:8080/ProiektuarenIzena/rest/kalkulagailua/kalkulatuURL";
}
```

dl.PaketekoKonstanteak.java

2.- REST MOTAKO WEB ZERBITZUAK

2.3.-JAX-RS bezeroa

- ADIBIDEA: JAX-RS zerbitzuak eta bezeroak

//AURREKO ADIBIDEKO KODE BERDINA

pl.TxantiloiaPOJO.java

```
package pl;

//import lerroak

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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //AURREKO ADIBIDEKO KODE BERDINA
    }

    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");

        Date dataOrdua=new Date();
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

        EragiketaXML eragiketaBXML=new EragiketaBXML(Float.valueOf(sEr1), sEr, Float.valueOf(sEr2), 0.0f, sdf.format(dataOrdua));
        eragiketaBXML=ClientBuilder.newClient()
            .target(PaketekoKonstanteak.eragiketaEginBiltegitratuURL)
            .request("application/json")
            .post(Entity.entity(eragiketaBXML, MediaType.APPLICATION_JSON),EragiketaBXML.class);

        //4. GAIKO ADIBIDEKO KODE BERDINA
    }
}
```

pl.ErabiltzaileServlet.java

```
package pl;

class PaketekoKonstanteak {
    public static final String eragiketaEginBiltegitratuURL="http://localhost:8080/ProiektuarenIzena/rest/zerbitzuak/eragiketaEginBiltegitratuURL";
}
```

pl.PaketekoKonstanteak.java

//AURREKO ADIBIDEKO KODE BERDINA

erabiltzaileForm.html

//AURREKO ADIBIDEKO KODE BERDINA

aplikAritEstiloak.css

AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEEñ

EDUKIA

- 1.- SARRERA
- 2.- REST MOTAKO WEB ZERBITZUAK
- 3.- APLIKAZIOAK KONFIGURATZEA
 - 3.1.- JAX-RS liburutegia
 - 3.2.- Eclipse

ERREFERENTZIAK

3.- APLIKAZIOAK KONFIGURATZEA

3.1.- JAX-RS liburutegia

- JAX-RS inplementazioak
 - Jersey (Oracle), RESTeasy (JBoss), etab.
 - Java EE 6tik aurrera, integratua
- Java EE EZ diren inguruneetarako Apache Tomcat
 - Liburutegia deskargatu
 - Javako WS klaseak pakete hierarkia baten barruan egon behar dira
 - Aplikazioak web.xml fitxategia eduki behar du

```
...
<servlet>
  <servlet-name>Jersey REST Service</servlet-name>
  <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>jersey.config.server.provider.packages</param-name>
    <param-value>GaratutakoWSklaseenPaketearenIzena</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Jersey REST Service</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
...
```

web.xml

3.- APLIKAZIOAK KONFIGURATZEA

3.1.- JAX-RS liburutegia

- Java EE BADIREN inguruneetarako Wildfly
 - Ez dira liburutegiak sartu behar ezta web.xml ukitu behar
 - Konfiguraziorako klase berezia: JaxRsActivator
 - Application klasearen alaba
 - **@ApplicationPath**

REST zerbitzuak klaseak dauden paketean

```
package bl;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("rest")
public class JaxRsActivator extends Application {
}
```

Hitz klabea da, edozein hitz izan daiteke, baina rest edo antzeko bat erabiltzea komenigarria da.

JaxRsActivator.java

AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEEñ

EDUKIA

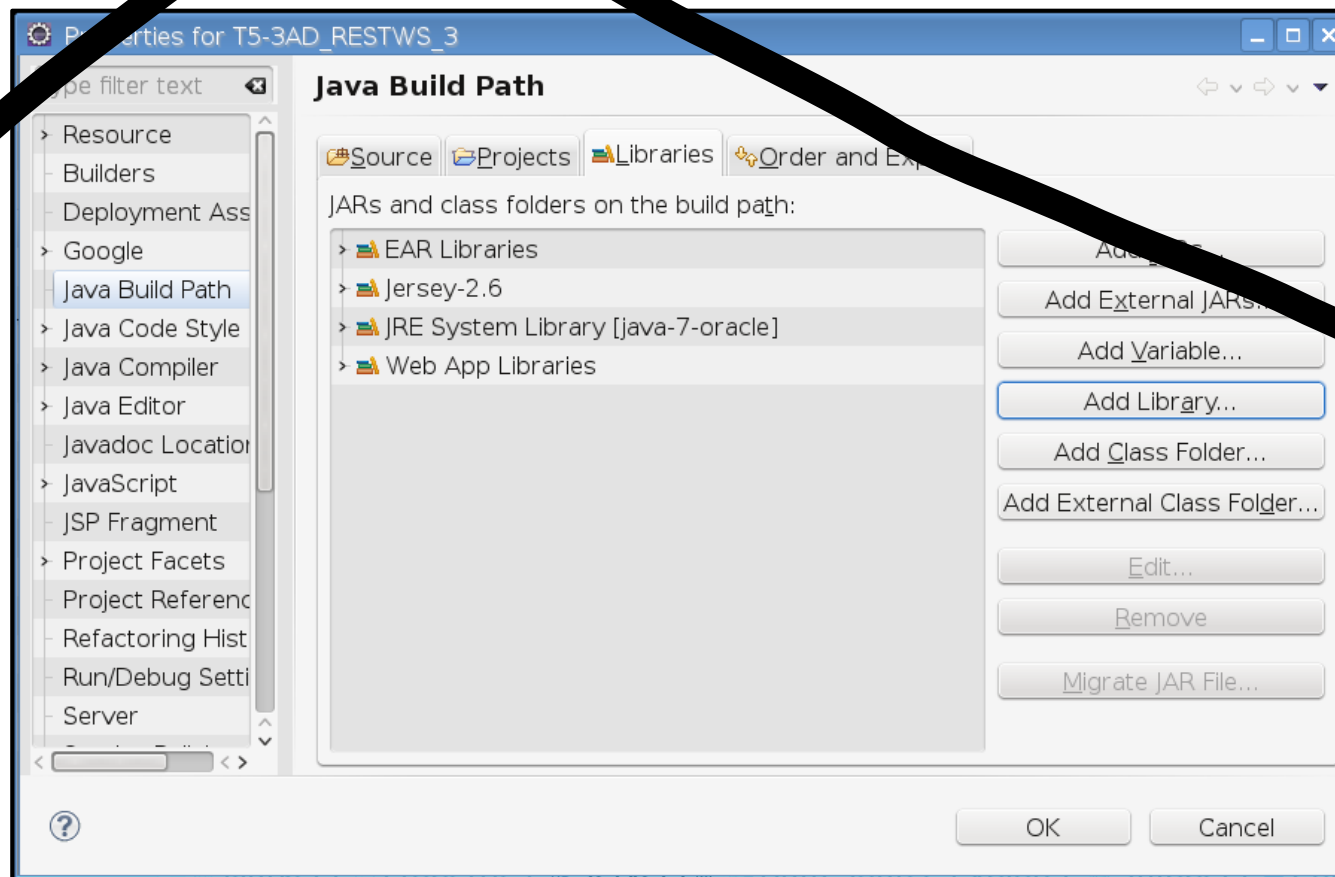
- 1.- SARRERA
- 2.- REST MOTAKO WEB ZERBITZUAK
- 3.- APLIKAZIOAK KONFIGURATZEA**
 - 3.1.- JAX-RS liburutegia
 - 3.2.- Eclipse**

ERREFERENTZIAK

3.- APLIKAZIOAK KONFIGURATzea

3.2.- Eclipse

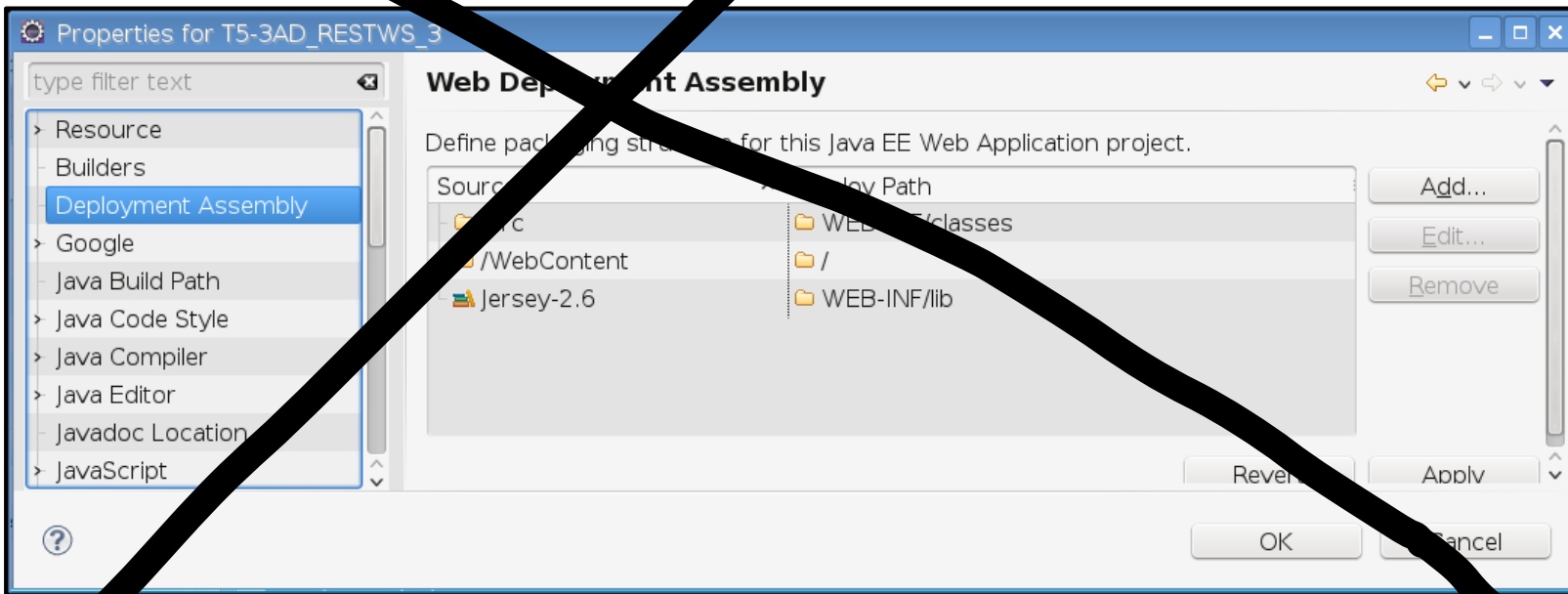
- Java EE EZ diren inguruneetarako
 - Liburutegia erreferentzia gisa, beharrezkoa
 - Build Path: User Library (berria)



3.- APLIKAZIOAK KONFIGURATZEA

3.2. Eclipse

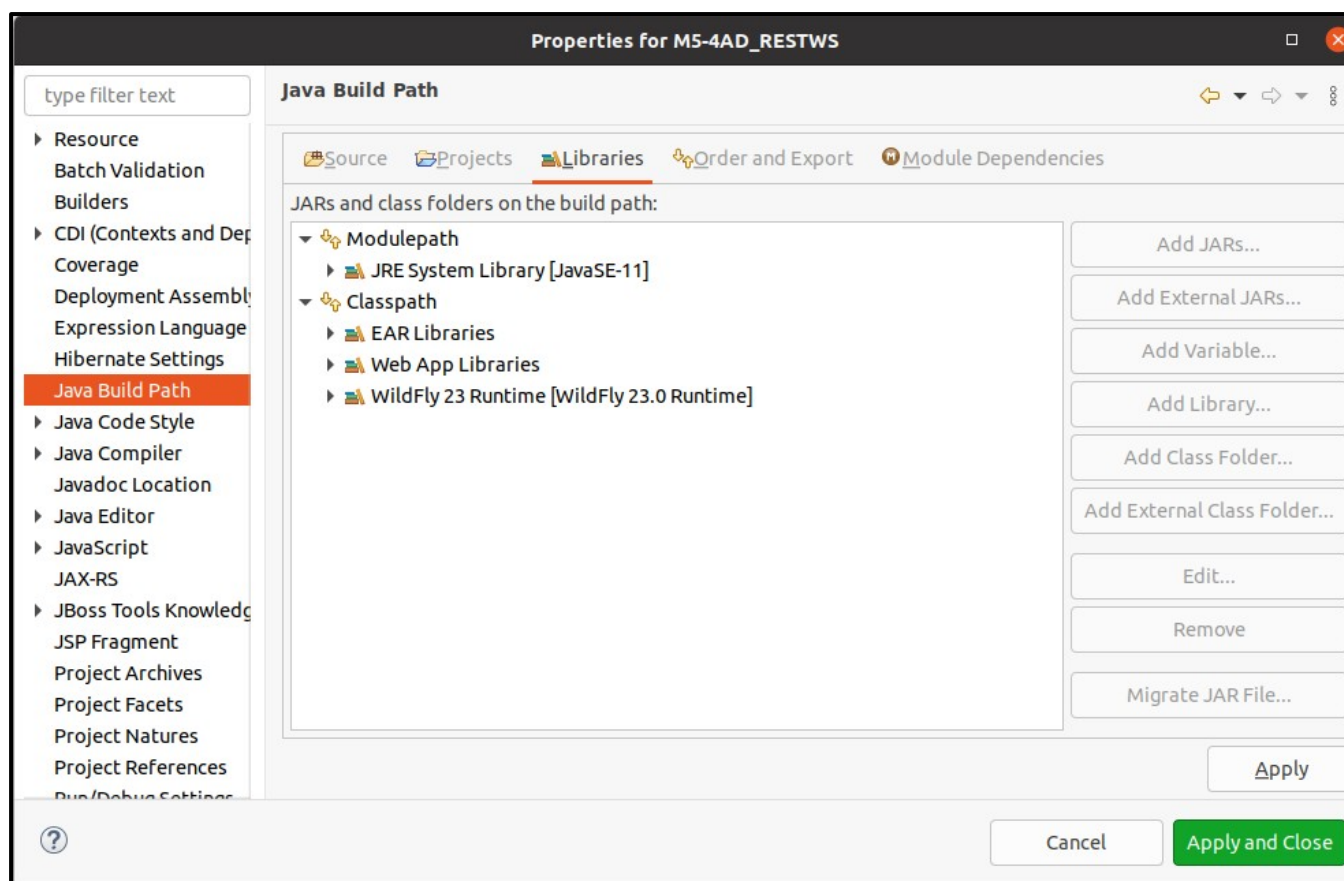
- Java EE EZ giren inguruneetarako
 - Web Deployment Assembly



3.- APLIKAZIOAK KONFIGURATZEA

3.2.- Eclipse

- Java EE BADIREN inguruneetarako
 - Liburutegia, Java EE ingurunearen Runtime-ak emango du
 - Orokorrean, Java EE moduko runtime batera lotutako proiektua sortzea nahikoa da



AURKEZPENIK GABEKO WEB APLIKAZIOAK JAVAEE_n ERREFERENTZIAK

[1] JAVAEE TUTORIAL: <https://docs.oracle.com/javaee/7/tutorial/>