**JAX-RS DERS NOTLARI**

Jax-RS yani restful,dışarıya web servisleri açmaya yarayan bir teknolojidir.Soap’tan sonra soapın zorlukluklarını ortadan kaldırmak için ortaya çıkmıştır.Dışarıya web servisleri açmak yada api yazmak demek yazdığımız kodların diğer teknolojilerle iletişim kurması demek gibi birşeydir.Örneğin biz java kodları yazıyoruz ve bu kodları frontçu alıp kendi uygulamasında kullanacak işte jax-rs sayesinde yazdığımız kodları json,xml gibi formatlara dönüştürüp veriyoruz.Onlarda oradan alıp kendi işlerinde kullanıyorlar.

**Nasıl Rest projesi açarız?**

Rest projeleri 2 ye ayrılır.RESTEasy ve Jersey.Jersey’in kullanımı daha çok yaygındır.Ben ikisinide aşağıya yazacağım.

**Jersey için:**

Ctrl+N > dynamic web project i diyoruz fakat finish demiyoruz.Son sayfaya gelip oraya WebContent yazıp altındaki web.xml create butonuna kesinlikle tıklamamız gerek.Daha sonrasında bu jax-rs’i projemizde kullanabilmemiz için projemizin anayasasına yani web.xml’ine aşağıdaki kodları yazıyoruz.Bu web.xml e yazdığımız kodlar jersey kullanımı için geçerlidir\*.

Web.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>

<display-name>01.restful.jersey</display-name>

//sırası ile ilk açılacak sayfalar

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>jersey-servlet</servlet-name>

<servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>

<init-param>

<param-name>com.sun.jersey.config.property.packages</param-name>

<param-value>com.project</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>jersey-servlet</servlet-name>

<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

</web-app>

---Daha sonra pom.xml e de şu alttaki dependencyleri eklemiz gerekiyor.

<dependencies>

<!-- https://mvnrepository.com/artifact/junit/junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.jersey/jersey-server -->

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-server</artifactId>

<version>1.19.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.jersey/jersey-servlet -->

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-servlet</artifactId>

<version>1.19.1</version>

</dependency>

<!-- Java 11 ve sonrasında java EE modüllerine desteğini kaldırdı JaxB

EE modülü olduğu için bunu javaya maven ile manuel eklememiz gerek alttaki

3 kod,jaxb yani ee modülünü kullanmamız için gereken dependecyler -->

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-core</artifactId>

<version>2.3.0.1</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-impl</artifactId>

<version>2.3.1</version>

</dependency>

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

</dependencies>

**MAİN** /app/selam’dan önce /rest yazıyoruz.

@Path("/app")

**public** **class** Main {

// -----------------------------------

// Standart Rest API

@GET

@Path("/selam")

@Produces(value = MediaType.***TEXT\_PLAIN***)

**public** String selam1() {

**return** "Selam";}

// -----------------------------------

/\*Alternatif Yol,200 yanıtı olumlu sonuç döndüğünde bu metodu çağıracak, 404 yazarak sorun olmasaydı bile kullanıcıya 404 not found döndürebiliriz. \*/

@GET

@Path("/selamAlternatif")

@Produces(value = MediaType.***TEXT\_PLAIN***)

**public** Response selamAlternatif() {

**return** Response.*status*(200).entity("Selam Alternatif Yol").build();

}

// -----------------------------------

// Parametreli yol

@GET

@Path("/selam/{adi}")

@Produces(value = MediaType.***TEXT\_PLAIN***)

**public** Response selamParametreli(@PathParam(value = "adi") String adi) {

**return** Response.*status*(200).entity("Selam " + adi).build();

}

// -----------------------------------

// 2 Parametreli yol

@GET

@Path("/selam/{adi}/{soyadi}")

@Produces(value = MediaType.***TEXT\_PLAIN***)

**public** Response selam2Parametreli(@PathParam(value = "adi") String adi, @PathParam(value = "soyadi") String soyadi) {

**return** Response.*status*(200).entity("Selam " + adi +" " + soyadi).build();

}

// -----------------------------------

// Burada gonderilen cevap text değil html'dir

@GET

@Path("/selamhtml")

@Produces(MediaType.***TEXT\_HTML***)

**public** Response selamHtmlGonder() {

**return** Response.*status*(200).entity("<font size=25> Selam HTML </font>").build();

}

// -----------------------------------

// Burada gonderilen cevap text değil html'dir

@GET

@Path("/selamxml")

@Produces(value = MediaType.***TEXT\_XML***)

**public** Response selamXmlGonder() {

String mesaj ="<selam> Selam XML </selam>";

**return** Response.*status*(200).entity(mesaj).build();

}

}

**RESTEasy için:**

Ctrl+N > dynamic web project i diyoruz fakat finish demiyoruz.Son sayfaya gelip oraya WebContent yazıp altındaki web.xml create butonuna kesinlikle tıklamamız gerek.RESTEasy’i projemizde kullanabilmemiz için Jersey’de olduğu gibi web.xml’e bir kod yazmamız gerekmiyor.Fakat restEasyServices paketi altında RestEasyServices Classı açıp oraya şu alttaki kodu yazmamız gerek.Bu aşağıdaki kod standarttır.Google’den falan bulabirsin.

@ApplicationPath("/rest") //sunucuyla programı çalıştırırken en başta urlye /rest yazıyoruz.

**public** **class** RestEasyServices **extends** Application {

**private** Set<Object> singletons = **new** HashSet<Object>();

**public** RestEasyServices() {

singletons.add(**new** Main()); //RestEasy kullancağımız classı burada belirtiyoruz.

}

@Override

**public** Set<Object> getSingletons() {

**return** singletons;

}

}

---Daha sonra pom.xml e de şu alttaki dependencyleri eklemiz gerekiyor.

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-jaxrs -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-jaxrs</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-client -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-client</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-servlet-initializer -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-servlet-initializer</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

</dependencies>

**MAİN**

Main metodumuz Jersey’deki ile aynı./app/selam’dan önce /rest yazıyoruz.

@Path("/app")

**public** **class** Main {

// -----------------------------------

// Standart Rest API

@GET

@Path("/selam")

@Produces(value = MediaType.***TEXT\_PLAIN***)

**public** String selam1() {

**return** "Selam";

}

// -----------------------------------

/\*Alternatif Yol,200 yanıtı olumlu sonuç döndüğünde bu metodu çağıracak, 404

yazarak sorun olmasaydı bile kullanıcıya 404 not found döndürebiliriz. \*/

@GET

@Path("/selamAlternatif")

@Produces(value = MediaType.***TEXT\_PLAIN***)

**public** Response selamAlternatif() {

**return** Response.*status*(200).entity("Selam Alternatif Yol").build();

}

// -----------------------------------

// Parametreli yol

@GET

@Path("/selam/{adi}")

@Produces(value = MediaType.***TEXT\_PLAIN***)

**public** Response selamParametreli(@PathParam(value = "adi") String adi) {

**return** Response.*status*(200).entity("Selam " + adi).build();

}

// -----------------------------------

// 2 Parametreli yol

@GET

@Path("/selam/{adi}/{soyadi}")

@Produces(value = MediaType.***TEXT\_PLAIN***)

**public** Response selam2Parametreli(@PathParam(value = "adi") String adi, @PathParam(value = "soyadi") String soyadi) {

**return** Response.*status*(200).entity("Selam " + adi +" " + soyadi).build();

}

// -----------------------------------

// Burada gonderilen cevap text değil html'dir

@GET

@Path("/selamhtml")

@Produces(MediaType.***TEXT\_HTML***)

**public** Response selamHtmlGonder() {

**return** Response.*status*(200).entity("<font size=25> Selam HTML </font>").build();

}

// -----------------------------------

// Burada gonderilen cevap text değil html'dir

@GET

@Path("/selamxml")

@Produces(value = MediaType.***TEXT\_XML***)

**public** Response selamXmlGonder() {

String mesaj ="<selam> Selam XML </selam>";

**return** Response.*status*(200).entity(mesaj).build();

} }

**JAXB (Jakarta XML Binding)**

-RestEasy de Kullanımı

Xml formatında çıktı vermek için kullanıyoruz.Bunu RESTEasy’li projede kullanabilmek için RESTEasy dependencylerine ek bunu ekliyoruz.

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-jaxb-provider -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-jaxb-provider</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

-Jerseyde Kullanımı

Bunu Jersey’li projede kullanabilmek için herhangi bir dependency eklememize gerek yok.Yukarıdaki Jersey dependencylerini ekleyince xml formatında çıktı alabiliriz.

**Jackson (JSON)**

-RestEasy de Kullanımı

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-jaxrs -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-jaxrs</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-client -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-client</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-servlet-initializer -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-servlet-initializer</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-core</artifactId>

<version>2.3.0.1</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-impl</artifactId>

<version>2.3.1</version>

</dependency>

//Projede Jackson Kullanabilmek için bu dependency

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-jackson-provider</artifactId>

<version>3.0.4.Final</version>

</dependency>

</dependencies>

-Jerseyde Kullanımı

<dependencies>

<!-- https://mvnrepository.com/artifact/junit/junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.jersey/jersey-server -->

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-server</artifactId>

<version>1.19.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.jersey/jersey-servlet -->

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-servlet</artifactId>

<version>1.19.1</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-core</artifactId>

<version>2.3.0.1</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-impl</artifactId>

<version>2.3.1</version>

</dependency>

</dependencies>

**RestFul Servisleri Açma(Elementleri Belirli olan)**

Elementleri belirli olan demek herhangi bir pojodan elementleri ve root elementini otomatik alan restful servisleri demek.XML olarak çıktı vereceksek aşağıdaki gibi root element,xml element gibi anotasyonlarla donatmamız lazım fakat JSON için düz anotasyonsuz pojo yeterlidir.Tabi yukarıdan hangisini kullanacaksak ona uygun dependencyleri eklemeliyiz ilk önce.Ayrıca RESTEasy yada Jersey hangisi ile yapacaksak ona göre restEasyServices yada jersey web.xml ayarlarınıda yapmamız lazım

**User POJOSU**

@XmlRootElement(name = "users")

@XmlAccessorType(XmlAccessType.***FIELD***)

**public** **class** User {

@XmlElement(name = "ıd")

**private** **int** userId;

@XmlElement(name ="uname")

**private** String username;

@XmlElement(name = "sname")

**private** String surname;

**public** User() {

}

PARAMETRELİ CONSTRUCTOR

SET – GET

TO STRİNG VAR AŞAĞIDA

**MAİN**

@Path(value = "/app")

**public** **class** Main {

@GET

@Path(value = "/userinfo/{id}")

@Produces(MediaType.***APPLICATION\_XML***)

//@Produces("application/xml")

**public** Response getUserInfo(@PathParam(value = "id") Integer id) {

User user = **new** User(id,"Samet" ,"Çavur");

**return** Response.*status*(200).entity(user).build();

}

@GET

@Path("/userinfo/{id}")

@Produces(MediaType.***APPLICATION\_JSON***)

**public** Response getUserInfo(@PathParam (value = "id") Integer id) {

User user = **new** User(id, "Samet", "Cavur");

**return** Response.*status*(200).entity(user).build();

}

}

**RestFul Servisleri Açma(Elementleri Belirsiz olan)**

Bunda xml dosyasının yada json dosyasının elementlerini api’yi yazarken veriyoruz.Ortada bir pojo falan yok.Yine yukarıdaki gibi gerekli dependencyleri pom.xml’e ekliyoruz,RESTEasy ise ona göre restEasyServices classı yada Jersey’se ona göre web.xml ayarlıyoruz.

@Path(value = "/app")

**public** **class** Main {

XML İÇİN BÖYLE

@GET

@Path(value = "/xml")

@Produces(MediaType.***APPLICATION\_XML***)

**public** String getUserInfo() {

String name = "Samet";

String surname = "Cavur";

**return** "<user>\r\n"

+ " <name>"+name+"</name>\r\n"

+ " <sname>"+surname+"</sname>\r\n"

+ "</user>";

}

JACKSON İÇİN BÖYLE

@GET

@Produces(MediaType.***APPLICATION\_JSON***)

@Path("/json")

**public** Response getJson() {

JSONObject jsonObject = **new** JSONObject();

jsonObject.put("name", "Smet");

jsonObject.put("surname", "Cavur");

**return** Response.*status*(200).entity(jsonObject.toString()).build();

}

}

**Client**

1.Bu clienti çalıştırmak için ilk önce bir sunucunun çalışması lazım. İlk önce 06.restful.jackson(json).jersey projesini çalıştırdım,daha sonra alttaki WebResource ye bu çalıştırdığım sunucunun url'sini yazdım.

2.Client olaylarını try-catch içinde yazmak daha sağlıklıdır.Zorunlu olmadığım halde try-catch içine aldım.

**Apache Client**

Apachenin Clientini eklemek için aşağıdaki dependecy’i ekliyoruz.

<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.5.13</version>

</dependency>

**public** **class** ClientWithApache {

**public** **static** **void** main(String[] args) {

*getInfo*();

*getInfoParam*("Samco","Urfa");

}

**public** **static** **void** getInfo() {

**try** {

HttpClient httpClient = HttpClientBuilder.*create*().build();

HttpGet httpGet = **new** HttpGet("http://localhost:8080/04.2.restful.jackson/rest/app/json");

httpGet.addHeader("accept", "application/json");

HttpResponse httpResponse = httpClient.execute(httpGet);

**if** (httpResponse.getStatusLine().getStatusCode() != 200) {

System.***out***.println("Hata Meydana Geldi,Status Code : " + httpResponse.getStatusLine().getStatusCode());

} **else** {

BufferedReader bufferedReader = **new** BufferedReader(

**new** InputStreamReader(httpResponse.getEntity().getContent()));

String result;

**while** ((result = bufferedReader.readLine()) != **null**) {

System.***out***.println(result);

}

}

} **catch** (Exception e) {

System.***out***.println(e);

}

}

URL KISMINDA PARAMETRE VARSA BU CLİENT KULLANIRIZ

**public** **static** **void** getInfoParam(String name,String surname) {

**try** {

HttpClient httpClient = HttpClientBuilder.*create*().build();

HttpGet httpGet = **new** HttpGet("http://localhost:8080/04.2.restful.jackson/rest/app/"+name+"/"+surname+""); //Aldığımız URL Adresi bu

httpGet.addHeader("accept", "application/json"); //Aldığımız Url’deki dosyanın JSON olduğunu burada belirtiyoruz

HttpResponse httpResponse = httpClient.execute(httpGet);

**if** (httpResponse.getStatusLine().getStatusCode() != 200) {

System.***out***.println("Hata Meydana Geldi,Status Code : " + httpResponse.getStatusLine().getStatusCode());

} **else** {

BufferedReader bufferedReader = **new** BufferedReader(

**new** InputStreamReader(httpResponse.getEntity().getContent()));

String result;

**while** ((result = bufferedReader.readLine()) != **null**) {

System.***out***.println(result);

}

}

} **catch** (Exception e) {

System.***out***.println(e);

}

}

}

**Jersey Client**

Web.xml’e jersey kodlarını yine yazıyoruz.Ayrıca aşağıdak dependecy’i de jersey dependencylerine ek olarak yazıyoruz

<!-- https://mvnrepository.com/artifact/com.sun.jersey/jersey-client -->

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-client</artifactId>

<version>1.19.1</version>

</dependency>

**public** **class** ClientWithJersey {

**public** **static** **void** main(String[] args) {

//aşağıya yazdığımız clientleri burada çalıştırmalıyız

*getInfo*();

*getInfoParam*("ssset", "cavur");

}

**public** **static** **void** getInfo() {

**try** {

Client client = Client.*create*();

WebResource webResource = client.resource("http://localhost:8080/04.2.restful.jackson/rest/app/json");

ClientResponse clientResponse = webResource.accept(MediaType.***APPLICATION\_JSON***).get(ClientResponse.**class**);

**if** (clientResponse.getStatus() != 200) {

System.***out***.println("Hata Meydana Geldi,Status Code : " + clientResponse.getStatus());

} **else** {

System.***out***.println(clientResponse.getEntity(String.**class**));

}

} **catch** (Exception e) {

System.***out***.println(e);

}

}

//Parametreli versiyonuda böyledir.

**public** **static** **void** getInfoParam(String name,String surname) {

**try** {

Client client = Client.*create*();

WebResource webResource = client.resource("http://localhost:8080/04.2.restful.jackson/rest/app/"+name+"/"+surname+"");

ClientResponse clientResponse = webResource.accept(MediaType.***APPLICATION\_JSON***).get(ClientResponse.**class**);

**if** (clientResponse.getStatus() != 200) {

System.***out***.println("Hata Meydana Geldi,Status Code : " + clientResponse.getStatus());

} **else** {

System.***out***.println(clientResponse.getEntity(String.**class**));

}

} **catch** (Exception e) {

System.***out***.println(e);

}

}

}

**JDBC-RestFul Examples**

**DBHelper Classı**

**public** **class** DBHelper {

**public** Connection getConnection() **throws** SQLException, ClassNotFoundException {

String url = "jdbc:mysql://localhost:3306/jpa";

String ıd="root";

String pass="1234";

Class.*forName*("com.mysql.jdbc.Driver");

**return** DriverManager.*getConnection*(url, ıd, pass);

}

**public** **void** showErrorMessage(SQLException e) {

System.***out***.println("SQL Hatası:" + e.getLocalizedMessage());

System.***out***.println("Hata Kodu:" + e.getErrorCode());

}

**User Pojosu**

@XmlRootElement(name = "user")

@XmlAccessorType(XmlAccessType.***NONE***)

**public** **class** User {

@XmlElement

**private** **int** userId;

@XmlElement

**private** String userName;

@XmlElement

**private** String userSurname;

@XmlElement

**private** String birthOfDate;

**public** User() {

}

**public** User(**int** userId, String userName, String userSurname,String birthOfDate) {

**this**.userId = userId;

**this**.userName = userName;

**this**.userSurname = userSurname;

**this**.birthOfDate = birthOfDate;

}

**public** **int** getUserId() {

**return** userId;

}

**public** **void** setUserId(**int** userId) {

**this**.userId = userId;

}

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** String getUserSurname() {

**return** userSurname;

}

**public** **void** setUserSurname(String userSurname) {

**this**.userSurname = userSurname;

}

**public** String getbirthOfDate() {

**return** birthOfDate;

}

**public** **void** setbirthOfDate(String birthOfDate) {

**this**.birthOfDate = birthOfDate;

}

@Override

**public** String toString() {

**return** "User [userId=" + userId + ", userName=" + userName + ", userSurname=" + userSurname + ", birthOfDate="

+ birthOfDate + "]";

}

}

}

**User Repository İnterfacesi ve Onu İmplement eden Class**

**public** **interface** IUserRepository {

**public** List<User> findAllUsers() **throws** ClassNotFoundException;

**public** User findUserById(**int** userId) **throws** ClassNotFoundException;

**public** **void** insertUser(User user) **throws** ClassNotFoundException;

**public** User updateUser(**int** userId,User user) **throws** ClassNotFoundException;

**public** **void** deleteUser(**int** userId) **throws** ClassNotFoundException;

}

**public** **class** UserRepository **implements** IUserRepository {

DBHelper dbHelper = **new** DBHelper();

Connection connection = **null**;

PreparedStatement statement = **null**;

ResultSet resultset;

**public** List<User> findAllUsers() **throws** ClassNotFoundException {

List<User> users = **new** ArrayList<User>();

**try** {

connection = dbHelper.getConnection();

statement = connection.prepareStatement("SELECT \* FROM User;");

resultset = statement.executeQuery();

**while** (resultset.next()) {

User user = **new** User(resultset.getInt(1), resultset.getString(3),resultset.getString(4), resultset.getString(2));

users.add(user);

}

} **catch** (SQLException e) {

dbHelper.showErrorMessage(e);

}

**return** users;

}

@Override

**public** User findUserById(**int** userId) **throws** ClassNotFoundException {

User user = **new** User();

**try** {

connection = dbHelper.getConnection();

statement = connection.prepareStatement("SELECT \* FROM User WHERE userId=?");

statement.setInt(1, userId);

resultset = statement.executeQuery();

**while** (resultset.next()) {

user = **new** User(resultset.getInt(1), resultset.getString(3), resultset.getString(4),resultset.getString(2));

}

} **catch** (SQLException e) {

dbHelper.showErrorMessage(e);

}

**return** user;

}

@Override

**public** **void** insertUser(User user) **throws** ClassNotFoundException {

String insertUserQuery = "INSERT INTO user (birthOfDate, userName, userSurname) VALUES (?, ?, ?)";

**try** {

connection = dbHelper.getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(insertUserQuery);

preparedStatement.setString(1, user.getbirthOfDate());

preparedStatement.setString(2, user.getUserName());

preparedStatement.setString(3, user.getUserSurname());

preparedStatement.execute();

} **catch** (SQLException e) {

dbHelper.showErrorMessage(e);

}

}

@Override

**public** User updateUser(**int** userId,User user) **throws** ClassNotFoundException {

String updateUserQuery = "UPDATE user SET birthOfDate=?, userName =?, userSurname =? WHERE userId =?";

**try** {

connection = dbHelper.getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(updateUserQuery);

preparedStatement.setString(1, user.getbirthOfDate());

preparedStatement.setString(2, user.getUserName());

preparedStatement.setString(3, user.getUserSurname());

preparedStatement.setInt(4, user.getUserId());

preparedStatement.executeUpdate();

} **catch** (SQLException e) {

dbHelper.showErrorMessage(e);

}

**return** user;

}

@Override

**public** **void** deleteUser(**int** userId) **throws** ClassNotFoundException{

String deleteUserQuery = "DELETE FROM User WHERE userId=?";

**try** {

connection = dbHelper.getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(deleteUserQuery);

preparedStatement.setInt(1,userId);

preparedStatement.execute();

}

**catch** (SQLException e) {

dbHelper.showErrorMessage(e);

}

} }

**RESTEasy Service Classı**

@ApplicationPath("/rest")

**public** **class** RestEasyServices **extends** Application {

**private** Set<Object> singletons = **new** HashSet<Object>();

**public** RestEasyServices() {

singletons.add(**new** UserRestfulServices());

}

@Override

**public** Set<Object> getSingletons() {

**return** singletons;

}

}

**UserRestfulServices CLASSI**

@Path(value = "/app")

**public** **class** UserRestfulServices {

//@Consumes(MediaType.APPLICATION\_XML) > Programa bir xml yada json yollacaksak bu

//@Produces(MediaType.APPLICATION\_XML) > Programdan bir xml yada json dosyası alacaksak bu

// Her ikisinide kullanabiliriz.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@GET

@Path(value = "/userinfo")

@Produces(MediaType.***APPLICATION\_XML***)

**public** List<User> findAllUsers() **throws** ClassNotFoundException{

UserRepository repository = **new** UserRepository();

List<User> users = repository.findAllUsers();

**return** users;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@GET

@Path(value = "/userinfo/{userId}")

@Produces(MediaType.***APPLICATION\_XML***)

**public** User findUserById(@PathParam ("userId") **int** userId) **throws** ClassNotFoundException{

UserRepository repository = **new** UserRepository();

User user = repository.findUserById(userId);

**return** user;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//1.Postmana gir url satırına bu alttaki url yi gir ve POST olarak seç daha sonra altta 'Body' diye bir kısım var oraya istediğim bilgileri 'xml' formatında gir ve send'e bas

2.Metodun üstüne POST yazdığımız için gönderilen xml formatındaki Useri alıp bu metodda kullanacak

@POST

@Path("/userinfo/insertuser")

@Consumes(MediaType.***APPLICATION\_XML***)

**public** Response insertUser(User user) **throws** ClassNotFoundException {

UserRepository repository = **new** UserRepository();

repository.insertUser(user);

**return** Response.*status*(200).build();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//1.Parametrede ki userId ile güncellemek istediğimiz useri bulduk ve

//yine parametredeki User user ile güncellenmiş veriyi döndürdük

//2.Hem güncellenmiş veriyi gönderdik hemde bunu xml formatında yazdırdığımız için

//aşağıda hem consumes hem produces yazdık,sadece gönderecek olsaydık consumes yazmamız yeterliydi.

@PUT

@Path("/userinfo/updateuser/{userId}")

@Consumes(MediaType.***APPLICATION\_XML***)

@Produces(MediaType.***APPLICATION\_XML***)

**public** User updateUser(@PathParam ("userId") **int** userId,User user) **throws** ClassNotFoundException {

UserRepository repository = **new** UserRepository();

repository.updateUser(userId, user);

**return** user;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Herhangi bir xml gönderip almayacağız sadece bir veri sileceğimiz için produces yada consumes kullanmadık.

@DELETE

@Path("/userinfo/deleteuser/{userId}")

**public** Response deleteUser(@PathParam ("userId") **int** userId){

UserRepository repository = **new** UserRepository();

**try** {

repository.deleteUser(userId);

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

**return** Response.*status*(404).build();

}

**return** Response.*status*(200).build();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

**POM.XML**

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-jaxrs -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-jaxrs</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-servlet-initializer -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-servlet-initializer</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-client -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-client</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-jaxb-provider -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-jaxb-provider</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.jboss.resteasy/resteasy-jackson-provider -->

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-jackson-provider</artifactId>

<version>4.0.0.Beta1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/net.sf.scannotation/scannotation -->

<dependency>

<groupId>net.sf.scannotation</groupId>

<artifactId>scannotation</artifactId>

<version>1.0.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple

Json dosyalarını alıp parçalamak için,(UserServices Classı) -->

<dependency>

<groupId>com.googlecode.json-simple</groupId>

<artifactId>json-simple</artifactId>

<version>1.1.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java --

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.24</version>

</dependency>

</dependencies>

**YENİ EĞİTİM**

**Json dosyasını Java’da okuma**

\*Öncelikle şu alttaki dependency’i pom’a eklemeliyiz.Alttaki sarı işaretli anotasyon şu demek.Json dosyasının içindeki son property’e bak from yazıyor ama student pojosunun içinde from diye bir değişken yok.Eğer o sarı işaretli anotasyonu koymasaydık o jsonı javada okuyamazdık ama o anotasyon sayesinde okuyabiliyoruz.

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.12.3</version>

</dependency>



