JESUÏTES educació

DAWM07UF4

SERVEIS WEB

UF4.6.2 REST AMB JAVA. JAX-RS

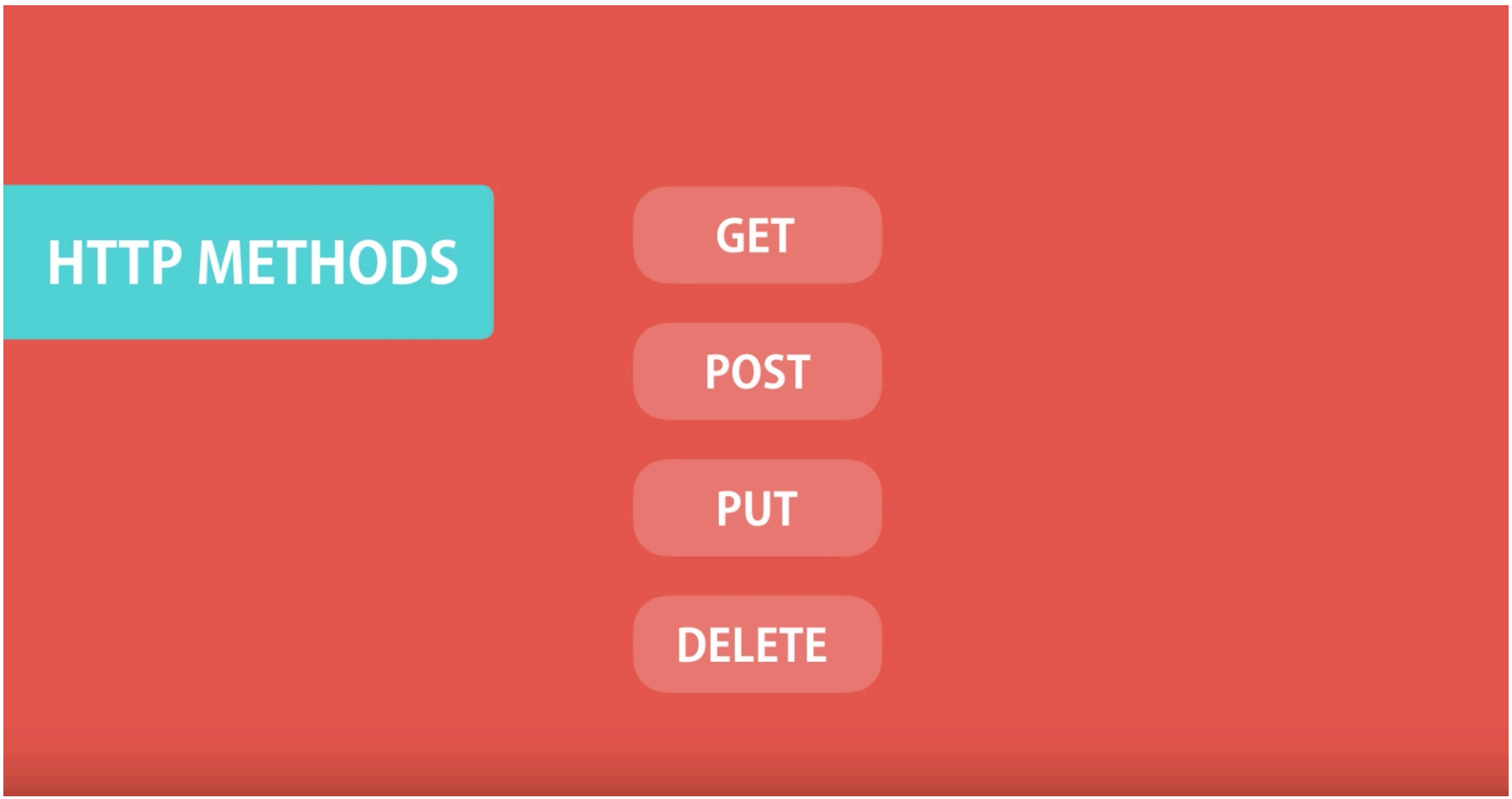
CFGS Desenvolupament d'Aplicacions Web
M07. Desenvolupament web en entorn servidor
Fundació Jesuïtes Educació - Escola del Clot
Sergi Grau sergi.grau@fje.edu

OBJECTIUS

- Configurar IntellijCrear per a crar Serveis Web
- Utilitzar el framework Jersey, JAX-RS
- Crear un servei REST amb l'API JAX-RS

- Els serveis web RESTful es basen en mètodes HTTP i el concepte de REST.

- Un servei web RESTful normalment defineix el URI base per als serveis, els tipus MIME suportats (XML, text, JSON, definit per l'usuari, ...) i el conjunt d'operacions (POST, GET, PUT, DELETE) que estan suportats .


HTTP METHODS
GET
POST
PUT
DELETE

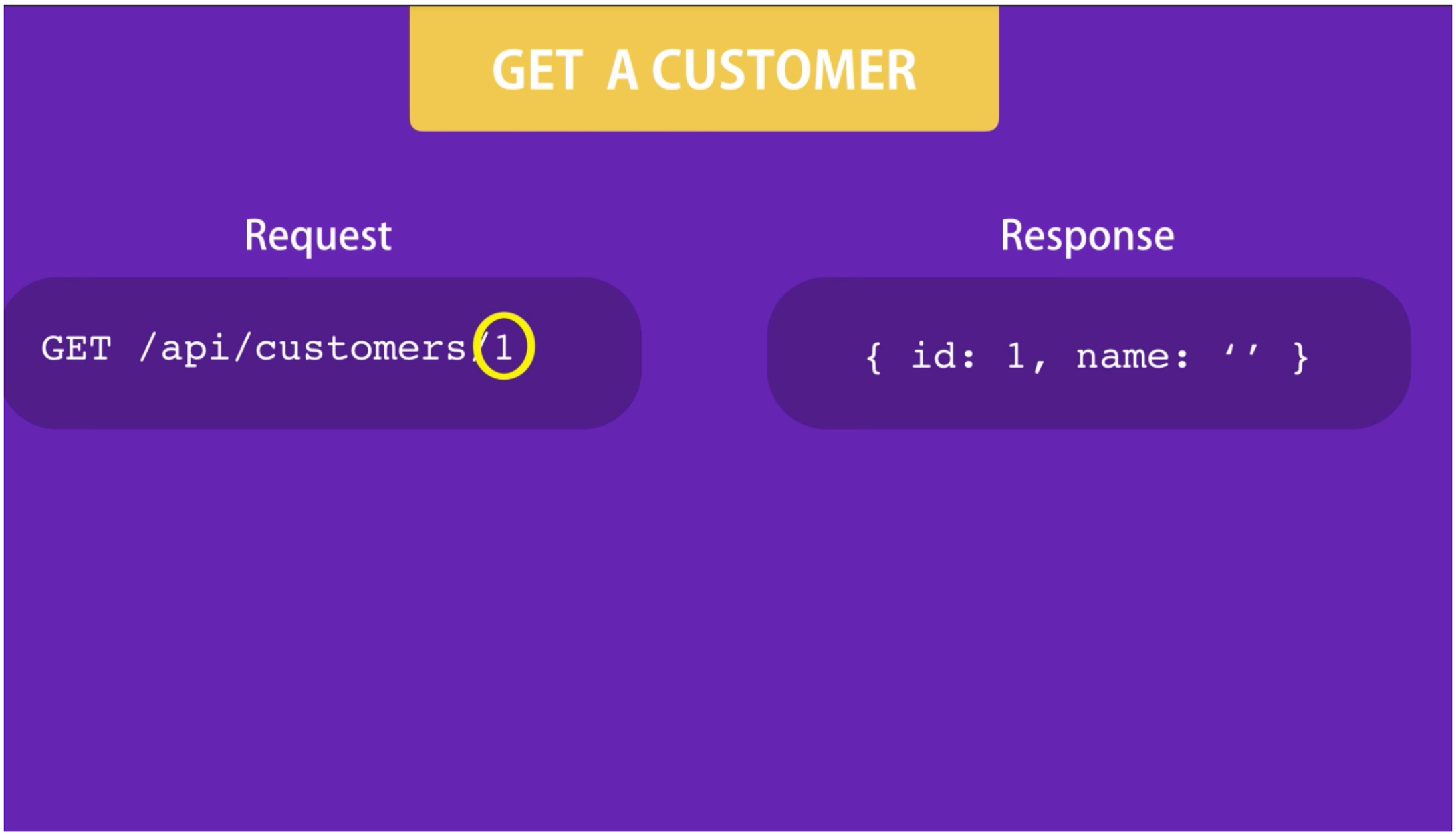
GET CUSTOMERS

Request

```
GET /api/customers
```

Response

```
[
    { id: 1, name: '' },
    { id: 2, name: '' },
    …
]
```


GET A CUSTOMER
Request
GET /api/customers/1
Response
{ id: 1, name: '' }

UPDATE A CUSTOMER

Request

```
PUT /api/customers/1

{ name: '' }
```

Response

DELETE A CUSTOMER

Request

```
DELETE /api/customers/1
```

Response

```
GET /api/customers
GET /api/customers/1
PUT /api/customers/1
DELETE /api/customers/1
POST /api/customers
```

- JAX-RS: Java API for RESTful Web Services és una API del llenguatge de programació Java que proporciona suport en la creació de serveis web d'acord amb l'estil arquitectònic Representational State Transfer (REST). JAX-RS utilitza anotacions ,introduïdes en Java SE 5 , per simplificar el desenvolupament i desplegament dels clients i punts finals dels serveis web.

- A partir de la versió 1.1 en endavant, JAX-RS és una part oficial de Java EE 6. Una característica notable de ser part oficial de Java EE és que no es requereix configuració per començar a utilitzar JAX-RS. Per als entorns que no són Java EE 6 es requereix una (petita) entrada al descriptor de desplegament web.xml.

ESPECIFICACIÓ

- JAX-RS proporciona algunes anotacions per ajudar a mapejar una classe recurs (POJO) com un recurs web. Entre aquestes anotacions s'inclouen:

- @Path especifica la ruta d'accés relativa per a una classe recurs o mètode.

- @GET, @ PUT, @ POST, @ DELETE i @ HEAD especifiquen el tipus de petició HTTP d'un recurs.

- @Produces especifica els tipus de mitjans MIME de resposta.

- @Consumes especifica els tipus de mitjans de petició acceptats.

ESPECIFICACIÓ

- A més, proporciona anotacions addicionals per als paràmetres de mètode per extreure informació de la sol · licitud. Totes les anotacions @ * Param prenen una clau d'alguna manera que s'utilitza per buscar el valor requerit.

- @PathParam enllaça el paràmetre a un segment de ruta.

- @QueryParam enllaça el paràmetre al valor d'un paràmetre de consulta HTTP.

- @MatrixParam enllaça el paràmetre al valor d'un paràmetre de matriu de HTTP.

- @HeaderParam enllaça el paràmetre a un valor de capçalera HTTP.

- @CookieParam enllaça el paràmetre a un valor de galeta.
- @FormParam enllaça el paràmetre a un valor de formulari.
- @DefaultValue especifica un valor per defecte per als enllaços anteriors quan la clau no és trobada.
- @Context torna tot el context de l'objecte. (Per exemple: @ Context HttpServletRequest request)

JERSEY, SERVEIS WEB RESTFUL EN JAVA

- EL Desenvolupament de serveis web que suporten RESTful sense problemes l'exposició de les seves dades en una varietat de tipus de mitjans de representació i abstreure els detalls de baix nivell de la comunicació client-servidor no és una tasca fàcil, sense un bon conjunt d'eines.

- Per tal de simplificar el desenvolupament de serveis web RESTful i els seus clients en Java, disposem de l'API JAX-RS. El bastiment Jersey RESTful Web Services és de codi obert,amb QoS, dissenyat per al desenvolupament de serveis web RESTful en Java i proporciona suport per per a JAX-RS (JSR 311 i JSR 339)

Jersey

RESTful Web Services in Java.

- Jersey és la implementació de referència d'aquesta especificació .

- Bàsicament conté un servidor i un client REST . El client de nucli proporciona una biblioteca per comunicar-se amb el servidor .

- Al costat del servidor Jersey utilitza un servlet que escaneja classes predefinides per identificar els recursos REST . L'arxiu de configuració web.xml que és proporcionat per la distribució Jersey registra aquest servlet de l'aplicació Web .

- La URL base d'aquest servlet és :

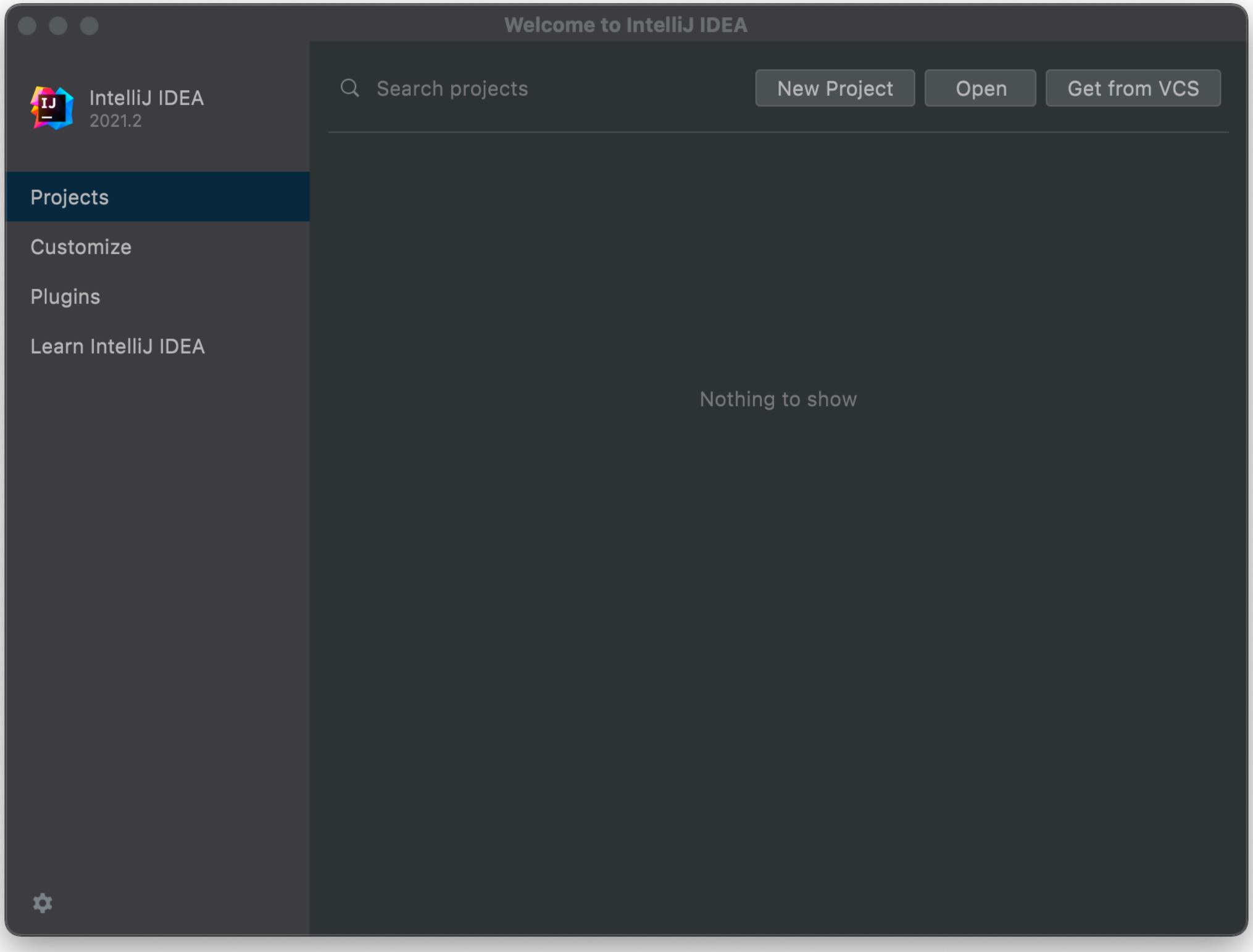
- http://domini:port/nom_app/url/ruta_classe_rest

- Aquest servlet analitza la sol · licitud HTTP entrant i selecciona la classe i el mètode correcte per a respondre a aquesta sol · licitud . Aquesta selecció es basa en les anotacions de la classe i els mètodes .

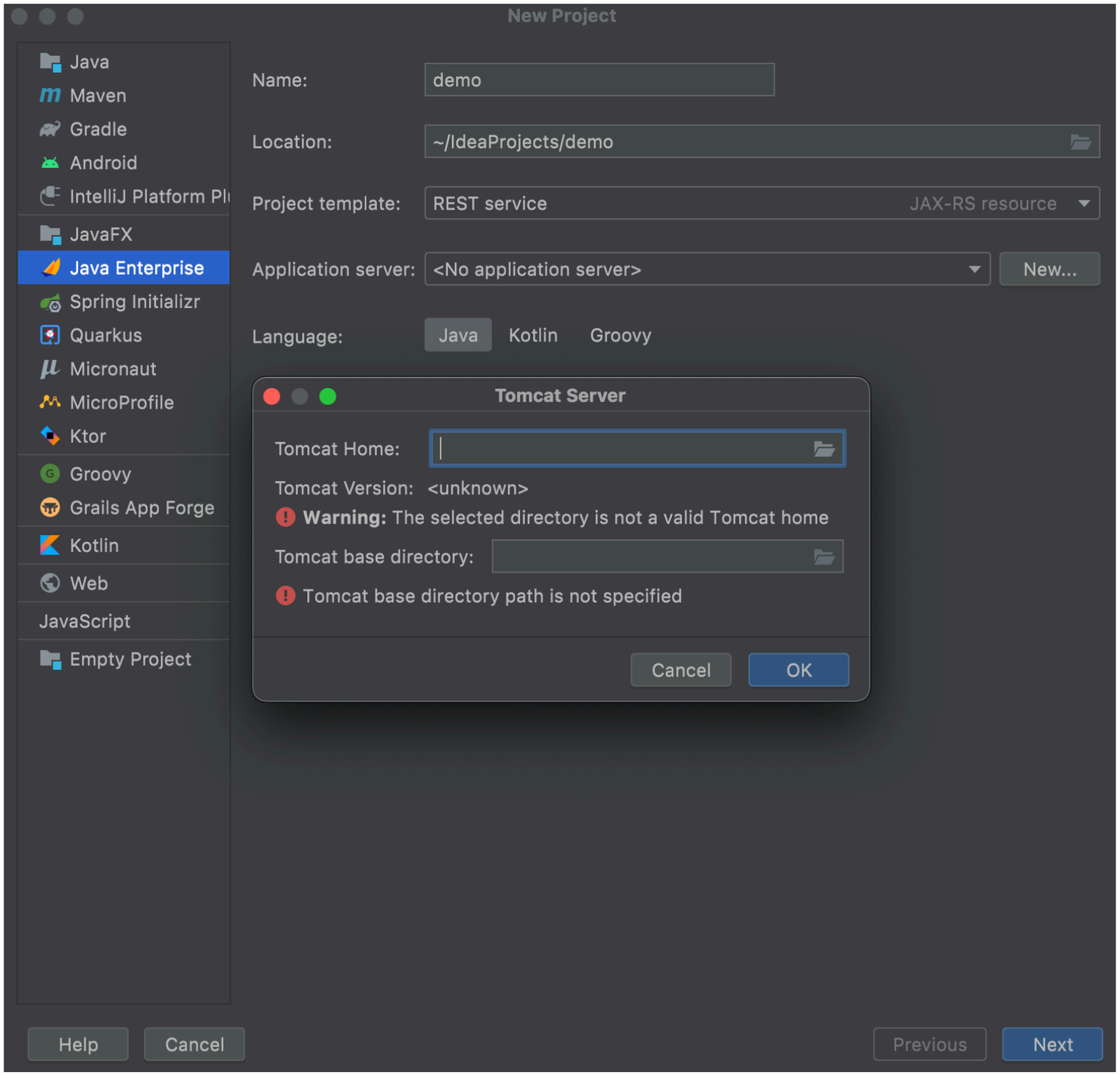
- Una aplicació web REST consisteix , per tant , fora de les classes de dades (recursos) i serveis. Aquests dos tipus es mantenen generalment en diferents paquets com el servlet Jersey es defineix a través del web.xml per escanejar alguns paquets per a les classes de dades .

- JAX - RS dóna suport a la creació d' XML i JSON a través de l'arquitectura Java per vinculació XML (JAXB) .

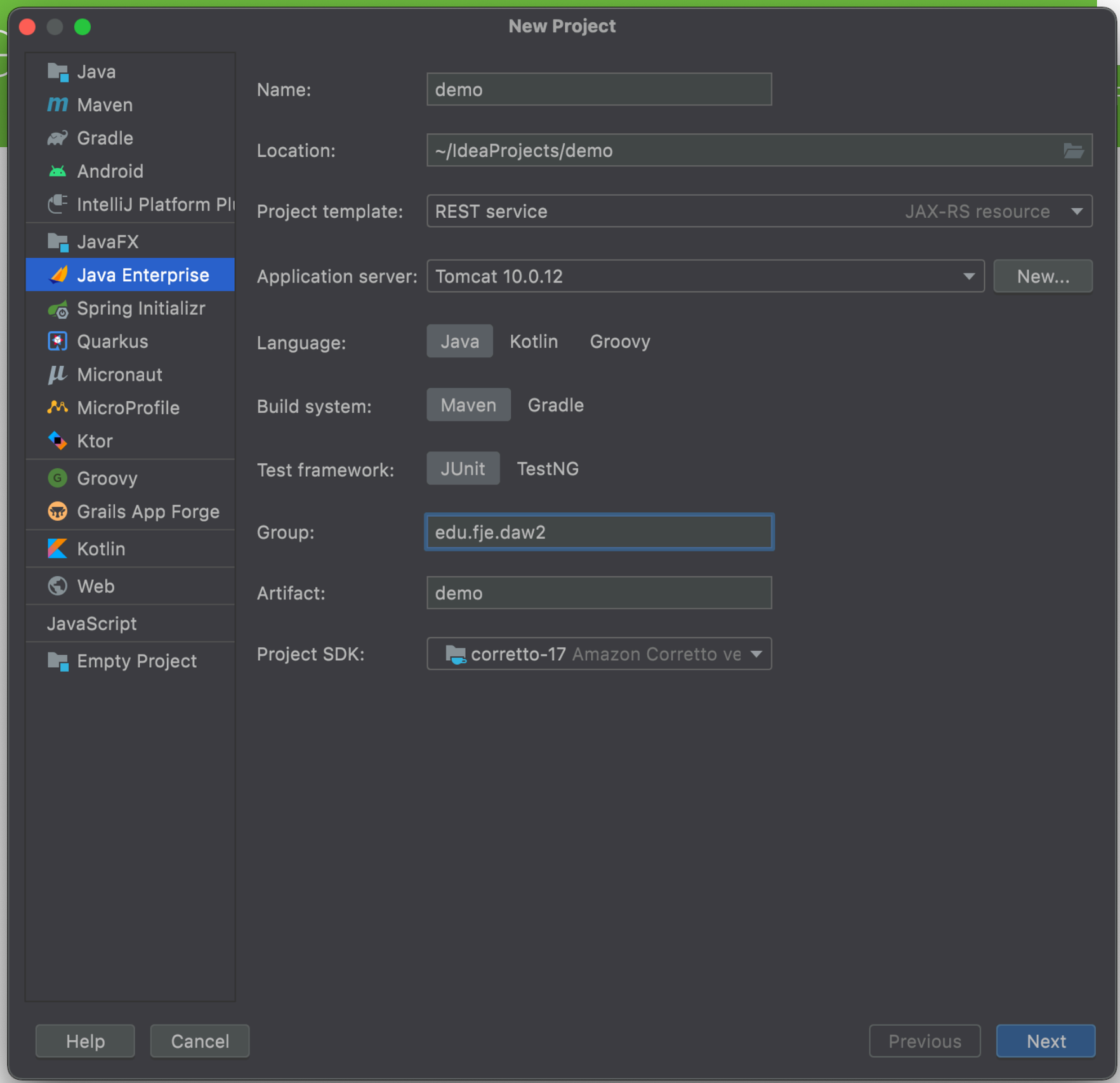
CONFIGURAR INTELLIJ IDEA



CONFIGURAR INTELLIJ IDEA



JESUÏTES educació
F4 sergi.grau@fje.edu



JESUÏTES educació
F4 sergi.grau@fje.edu

New Project

Version: Jakarta EE 9

Dependencies:

- Specifications
  - [ ] Full Platform (9.0.0)
  - [ ] Web Profile (9.0.0)
  - [ ] Batch (2.0.0)
  - [ ] Bean Validation (3.0.0)
  - [x] Contexts and Dependency Injection (CDI) (3.0.0)
  - [ ] Concurrency Utils (2.0.0)
  - [ ] Connector Architecture (JCA) (2.0.0)
  - [ ] Enterprise Java Beans (EJB) (4.0.0)
  - [ ] JSON Binding (JSON-B) (2.0.0)
  - [ ] JSON Processing (JSON-P) (2.0.0)
  - [ ] Message Service (JMS) (3.0.0)
  - [ ] Model View Controller (MVC) (2.0.0)
  - [ ] Persistence (JPA) (3.0.0)
  - [x] RESTful Web Services (JAX-RS) (3.0.0)
  - [ ] Security (2.0.0)
  - [ ] Server Faces (JSF) (3.0.0)
  - [x] Servlet (5.0.0)
  - [ ] Transaction (JTA) (2.0.0)
  - [ ] WebSocket (2.0.0)
  - [ ] XML Web Services (JAX-WS) (3.0.1)
- Implementations
  - [x] Eclipse Jersey Server (3.0.2)
  - [ ] Eclipse Jersey Client (3.0.2)
  - [ ] EclipseLink (3.0.1)

RESTful Web Services (JAX-RS)

Provides support in creating web services according to the Representational State Transfer (REST) architectural pattern (JSR 370).

Added dependencies:

- Contexts and Dependency Injection (CDI)
- RESTful Web Services (JAX-RS)
- Servlet
- Eclipse Jersey Server
- Weld SE

Help | Cancel | Previous | Finish

JESUÏTES educació
F4 sergi.grau@fje.edu

New Project

Version: Jakarta EE 9

Dependencies:

- Specifications
  - [ ] Full Platform (9.0.0)
  - [ ] Web Profile (9.0.0)
  - [ ] Batch (2.0.0)
  - [ ] Bean Validation (3.0.0)
  - [x] Contexts and Dependency Injection (CDI) (3.0.0)
  - [ ] Concurrency Utils (2.0.0)
  - [ ] Connector Architecture (JCA) (2.0.0)
  - [ ] Enterprise Java Beans (EJB) (4.0.0)
  - [ ] JSON Binding (JSON-B) (2.0.0)
  - [ ] JSON Processing (JSON-P) (2.0.0)
  - [ ] Message Service (JMS) (3.0.0)
  - [ ] Model View Controller (MVC) (2.0.0)
  - [ ] Persistence (JPA) (3.0.0)
  - [x] RESTful Web Services (JAX-RS) (3.0.0)
  - [ ] Security (2.0.0)
  - [ ] Server Faces (JSF) (3.0.0)
  - [x] Servlet (5.0.0)
  - [ ] Transaction (JTA) (2.0.0)
  - [ ] WebSocket (2.0.0)
  - [ ] XML Web Services (JAX-WS) (3.0.1)
- Implementations
  - [x] Eclipse Jersey Server (3.0.2)
  - [ ] Eclipse Jersey Client (3.0.2)
  - [ ] EclipseLink (3.0.1)

RESTful Web Services (JAX-RS)

Provides support in creating web services according to the Representational State Transfer (REST) architectural pattern (JSR 370).

Added dependencies:

- Contexts and Dependency Injection (CDI)
- RESTful Web Services (JAX-RS)
- Servlet
- Eclipse Jersey Server
- Weld SE

Help | Cancel | Previous | Finish

JESUÏTES
educació
i.grau@fje.edu

Run/Debug Configurations

Tomcat Server
Tomcat 10.0.12

Name: Tomcat 10.0.12 — Store as project file

Open browser

After launch — Firefox — ... — with JavaScript debugger

URL: http://localhost:8080/demo_war_exploded/

VM options:

On 'Update' action: Update classes and resources — Show dialog

On frame deactivation: Do nothing

JRE: Default (corretto-17 - project SDK)

Tomcat Server Settings

HTTP port: 8080 — Deploy applications configured in Tomcat instance

HTTPs port: — Preserve sessions across restarts and redeploys

JMX port: 1099

AJP port:

Before launch

Build

Build 'demo:war exploded' artifact

Edit configuration templates...

Cancel Apply OK

JESUÏTES
educació
i.grau@fje.edu

Run/Debug Configurations

Tomcat Server

Tomcat 10.0.12

Name: Tomcat 10.0.12

Store as project file

Server | Deployment | Logs | Code Coverage | Startup/Connection

Deploy at the server startup

demo:war exploded

Application context: /demo_war_exploded

Before launch

Edit configuration templates...

Cancel | Apply | OK

POM.XML


```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>edu.fje.daw2</groupId>
    <artifactId>demo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>demo</name>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.compiler.source>1.8</maven.compiler.source>
        <junit.version>5.7.1</junit.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>jakarta.servlet</groupId>
            <artifactId>jakarta.servlet-api</artifactId>
            <version>5.0.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jersey.containers</groupId>
            <artifactId>jersey-container-servlet</artifactId>
            <version>3.0.2</version>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jersey.media</groupId>
            <artifactId>jersey-media-json-jackson</artifactId>
            <version>3.0.2</version>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jersey.inject</groupId>
            <artifactId>jersey-cdi2-se</artifactId>
            <version>3.0.2</version>
        </dependency>
        <dependency>
            <groupId>org.jboss.weld.se</groupId>
            <artifactId>weld-se-core</artifactId>
            <version>4.0.1.SP1</version>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.3.1</version>
            </plugin>
        </plugins>
```

CREAR EL RECURS, POJO

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

@Path("/hola-mon")
public class HolaMon {

    @GET
    @Produces("text/plain")
    public String getClichedMessage() {
        return "Hello, World!";
    }
}
```

CREAR EL RECURS, APLICACIO

```
import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;
import java.util.HashSet;
import java.util.Set;

@ApplicationPath("/")
public class Aplicacio extends Application {
    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> h = new HashSet<>();
        h.add( HelloWorld.class );
        return h;
    }
}
```

CLIENT PER AL RECURS SIMPLE AMB JAX-RS


```
Client client = ClientBuilder.newClient();
WebTarget target =
client.target("http://localhost:8080").path("/Jersey/
webresources/servei");
String bean =
target.request(MediaType.TEXT_HTML).get(String.class);
System.out.println(bean);
target =
client.target("http://localhost:8080").path("/Jersey/
webresources/servei/hola");
bean = target.request(MediaType.TEXT_PLAIN).get(String.class);
System.out.println(bean);
```

JESUÏTES educació
DAWM06UF4 sergi.grau@fje.edu

```
//s'utilitza com
http://localhost:8080/Jersey/webresources/servei/consulta?
curs=a&nom=sergi&nom=joan
@GET
@Path("/consulta")
public Response getPersones(
@QueryParam("curs") String curs,
@QueryParam("nom") List<String> noms) {
return Response
.status(200)
.entity("curs : " + curs + ", noms" +
noms.toString()).build();
}
```

POST I PUT


```
@PUT
@Produces(MediaType.TEXT_PLAIN)
public void putXml(String content) {
System.out.println("PUT");
}

@POST
@Consumes(MediaType.TEXT_PLAIN)
public Response postPersona( String nom) {
String output = "POST:Jersey te les seguents dades : " + nom;
return Response.status(200).entity(output).build();
}
```