

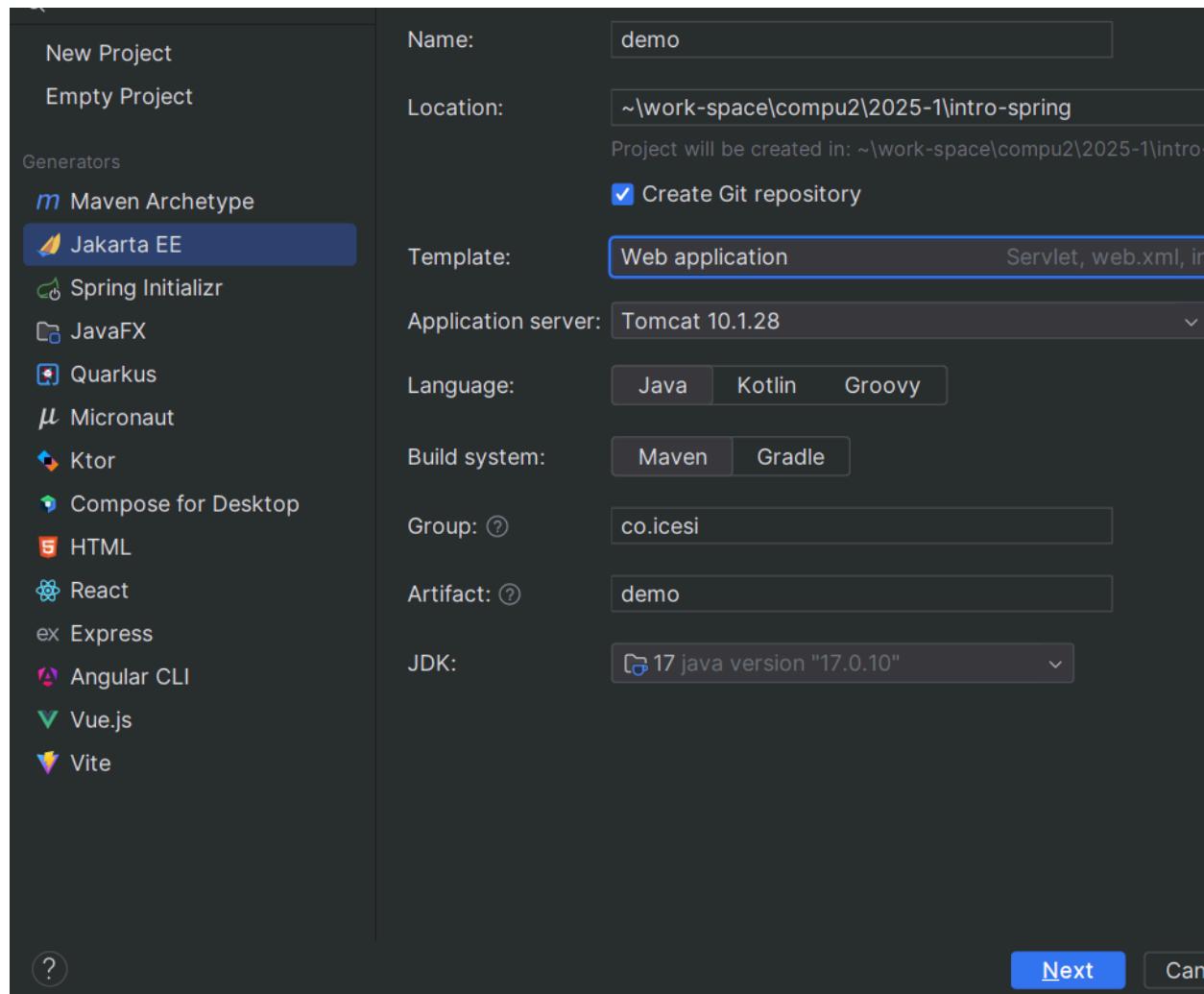
Demo: Developing Dynamic Web Project

1. Instalación y verificación de herramientas:

- Java 17: `java -version`
- IDE: IntelliJ o vscode.
- Maven: `mvn -v`
- Tomcat 10: `catalina.sh -v`

2. Crear proyecto:

- Si está usando IntelliJ cree uno con Jakarta EE usando tomcat 10



- En vscode: `ctrl + p` y escriba: > Java: Create java Project, luego seleccione maven. Luego seleccione **maven-archetype-webapp**, las demás opciones dependen de ustedes.

Select an archetype ...

No Archetype...
Create a basic Maven project directly.

More...
Find more archetypes available in remote catalog.

	Popular Archetypes
 maven-archetype-quickstart org.apache.maven.archetypes	Popular Archetypes
An archetype which contains a sample Maven project.	
 archetype-quickstart-jdk8 com.github.ngeor	
A Maven archetype for a simple Java app, updated for Java 8	
 maven-reactjs-blank-archetype am.ik.archetype	
Blank Project for React.js	
 maven-archetype-webapp org.apache.maven.archetypes	
An archetype which contains a sample Maven Webapp project.	
 spring-boot-blank-archetype am.ik.archetype	
Blank Project for Spring Boot	
 vaadin-archetype-application com.vaadin	
This archetype creates a Vaadin Flow application for basic	servlet container and can be used...
 mvc-1.0-blank-archetype am.ik.archetype	
MVC 1.0 Blank Project	

Cuando se cree el proyecto agregar esta dependencia al pom.xml, adicionalmente agregue la carpeta para los archivos fuente y resources.

```
<dependency>
    <groupId>jakarta.servlet</groupId>
    <artifactId>jakarta.servlet-api</artifactId>
    <version>5.0.0</version>
    <scope>provided</scope>
</dependency>
```

Añadir la clase:

```
import java.io.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebServlet(name = "HelloServlet", value = "/hello-servlet")
public class HelloServlet extends HttpServlet {
    private String message;

    public void init() {
        message = "Hello World!";
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        if (message == null) {
            message = "Hello World!";
        }
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(message);
    }
}
```

```
}

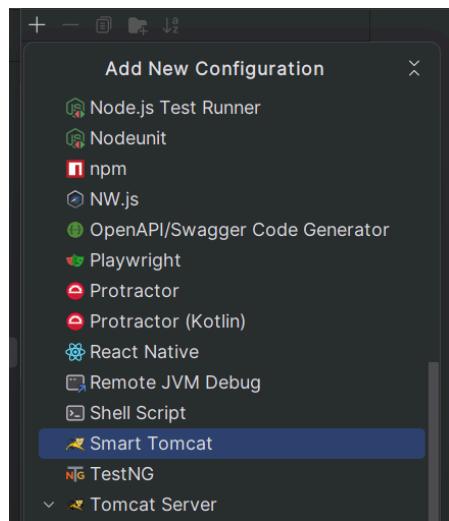
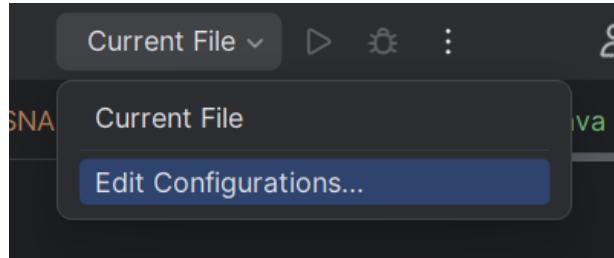
    public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException {
        response.setContentType("text/html");

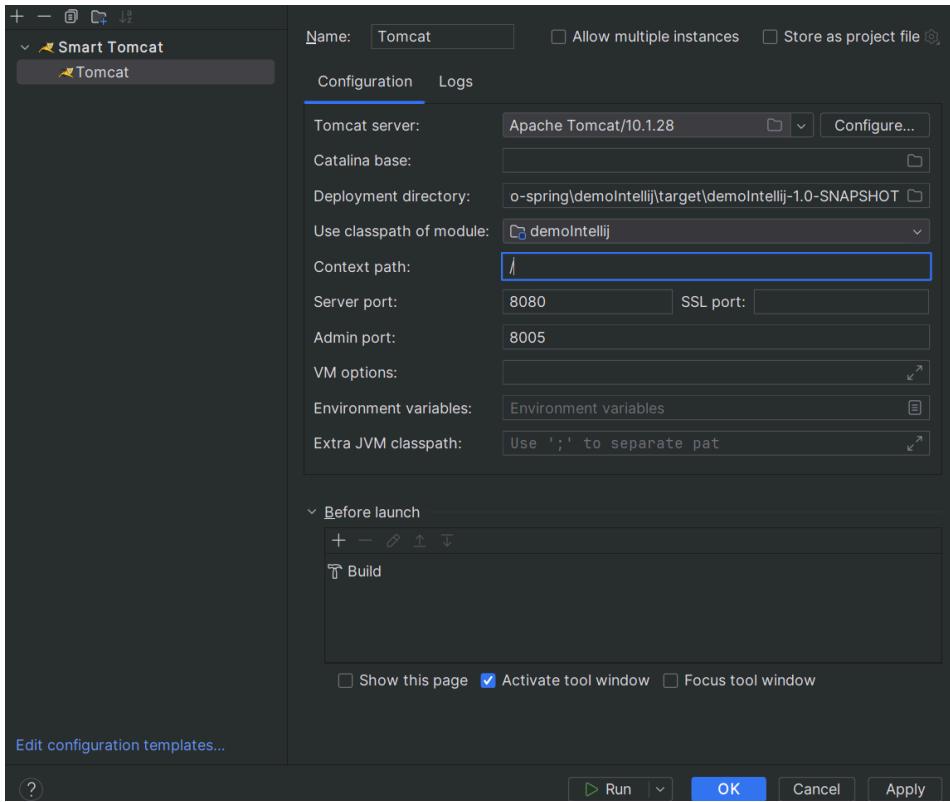
        // Hello
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>" + message + "</h1>");
        out.println("</body></html>");
    }

    public void destroy() {
}
}
```

Compile el proyecto para que la carpeta target sea generada: mvn clean compile

3. Seleccionar el servidor Tomcat para el deployment

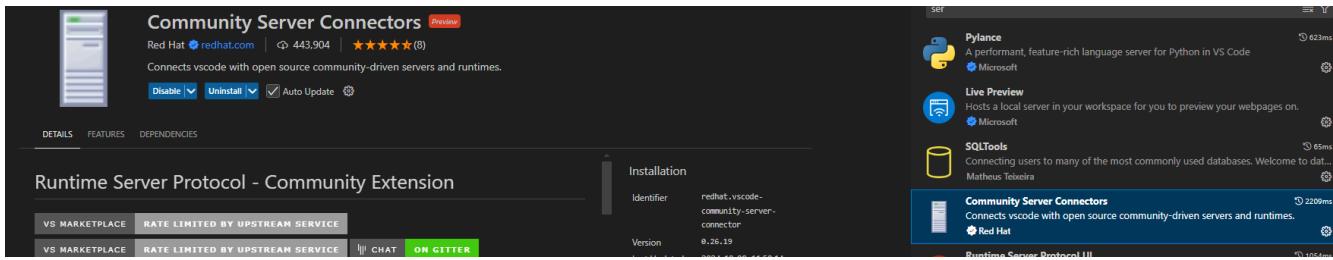


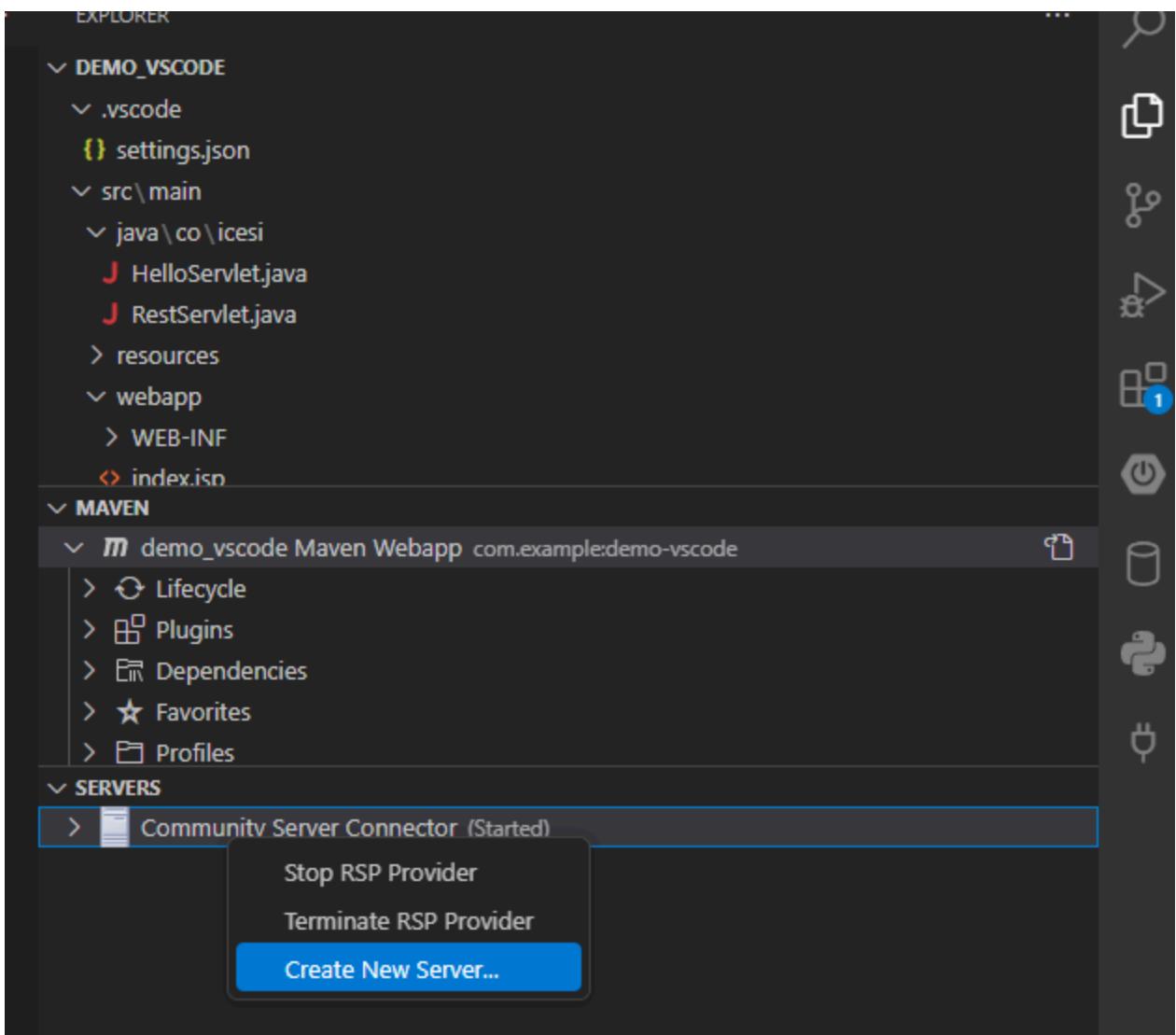


Correr el proyecto con esta configuración.

- Despliegue con vscode

Instale la siguiente extensión.

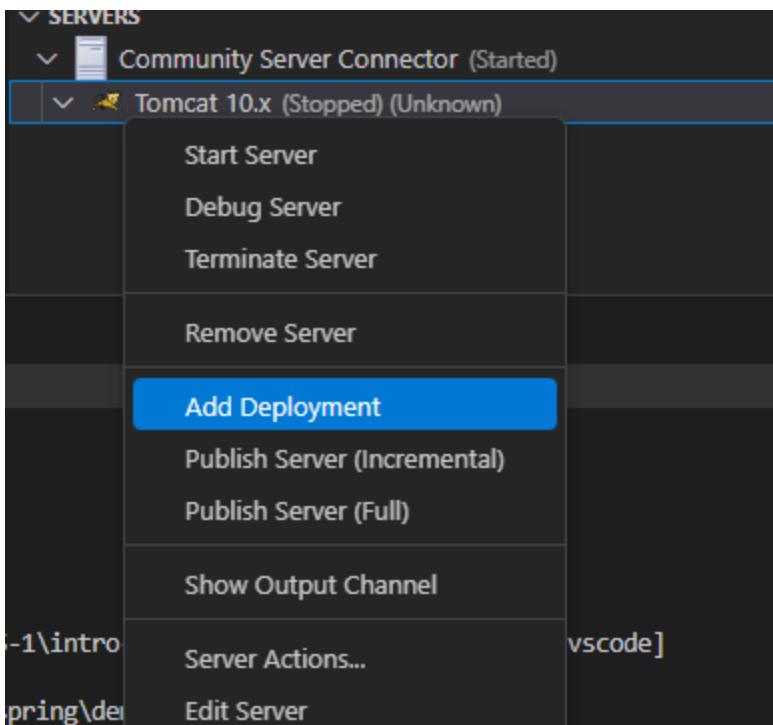




Use el tomcat que debió descargar antes.

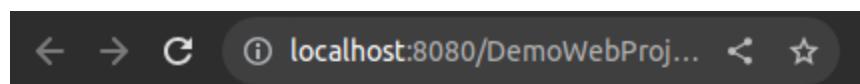
Ejecute: mvn clean package.

Esto genera la compilación del proyecto en la carpeta target

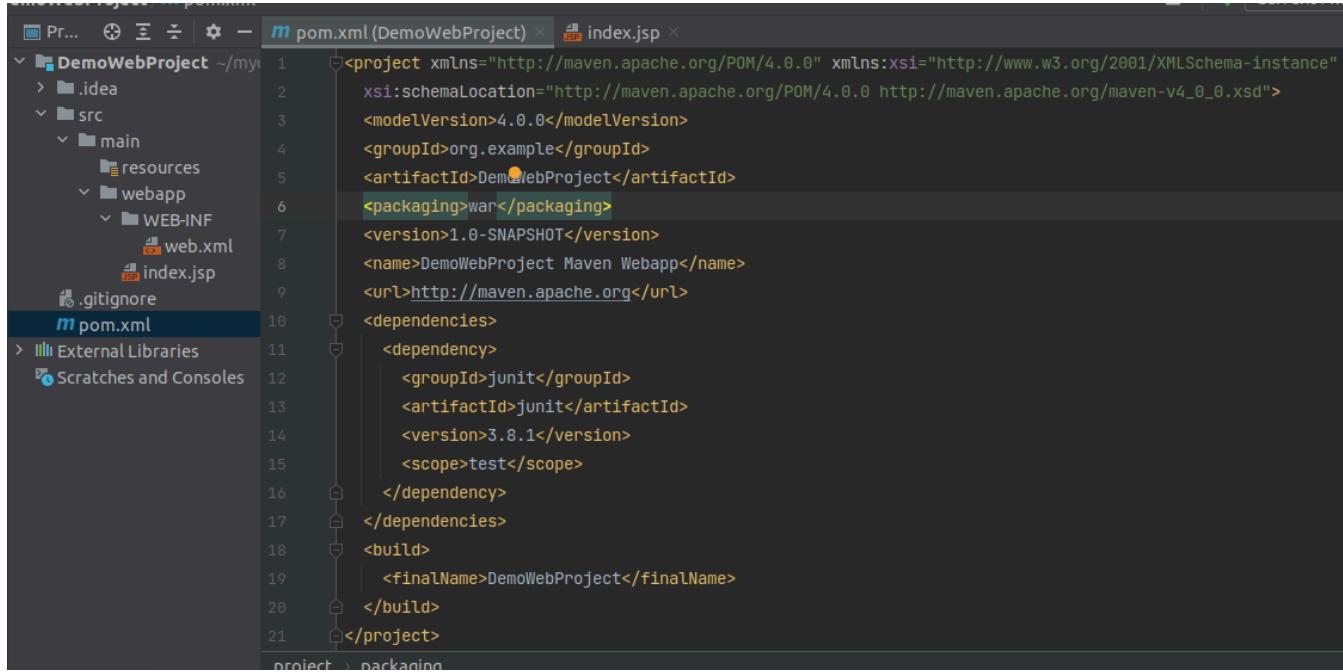


Seleccione la carpeta de binarios generada en la carpeta target.

1. En el panel "SERVERS" de la barra lateral → clic en "+"
 2. Selecciona "Apache Tomcat" y elige la versión (descárgala desde ahí mismo o apunta a una instalación existente)
 3. Genera el **.war**:
 5. Clic derecho en el servidor → "Add Deployment" → selecciona el **.war** de target
 6. Clic derecho → "Start"
 7. Abre **http://localhost:8080** en el navegador
-
4. Podemos intentar ejecutar el proyecto No olvide darle clic derecho al deploy y darle start server



5. Demos un vistazo al archivo pom.xml



```
<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 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>DemoWebProject</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>DemoWebProject Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <finalName>DemoWebProject</finalName>
  </build>
</project>
```

Demo: Developing Spring Web Project (IoCContainer)

1. Buscamos los artefactos para adicon de dependencias en la siguiente url:

<https://mvnrepository.com/artifact/org.springframework>

2. Seleccionamos el Spring Context

 3. Spring Context
org.springframework » [spring-context](#)

13,307 usages
Apache

Spring Context provides access to configured objects like a registry (a context). It inherits its features from Spring Beans and adds support for internationalization, event propagation, resource loading, and the transparent creation of contexts.

Last Release on Jul 13, 2023

Y copiamos el fragmento xml para adicionar la dependencia.

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>6.0.11</version>
</dependency>
```

3.

:-\$ mvn clean install

4. Adicionar la clase HolaMundo como bean

```
2 usages
public class HolaMundo {

    2 usages
    private String message;
    no usages
    public String getMessage() { return message; }

    no usages
    public void setMessage(String message) { this.message = message; }
}
```

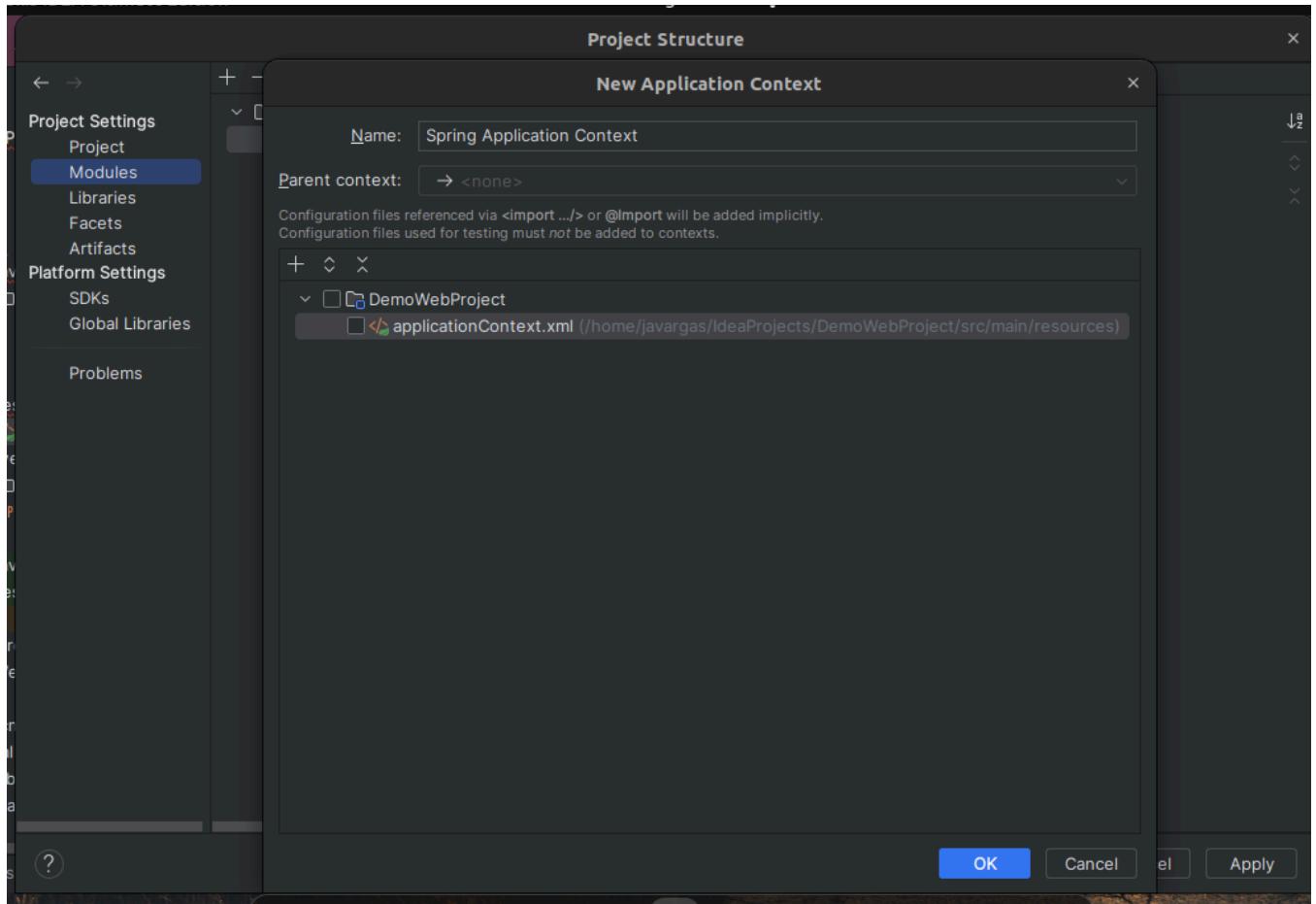
5. En el applicationContext.xml

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows a project named "DemoWebProject" with the following structure:
 - .idea
 - src
 - main
 - java
 - Aplicacion
 - HolaMundo
 - resources
 - applicationContext.xml
 - webapp
 - WEB-INF
 - web.xml
 - JSP index.jsp
 - target
- Editors:** The right side has several tabs open:
 - pom.xml (DemoWebProject)
 - HolaMundo.java
 - applicationContext.xml (selected)
 - Aplicacion.java

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           https://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="HolaMundo" class="HolaMundo">
        <property name="message" value="Hola Mundo spring!!" />
    </bean>
</beans>
```

File -> Project Structure -> Modules -> Add Spring → Add Spring Application Context



◀ Previous

```
1 package com.javargass.demowebproject;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 public class Application {
7
8     public static void main(String[] args) {
9         ApplicationContext context = ClassPathXmlApplicationContext("applicationContext.xml")
10
11
12
13 }
```

```

1
2
3 import org.springframework.context.ApplicationContext;
4
5
6 public class Application {
7
8     public static void main(String[] args) {
9         ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
10        HelloWorld obj = (HelloWorld) context.getBean("helloWorld");
11        System.out.println(obj);
12    }
13
14 }
15

```

package com.example.demowebproject;

```

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Aplicacion {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        HolaMundo hola = (HolaMundo) context.getBean("HolaMundo");

    }
}

```

En el index.jsp

```

<%
    ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
    HolaMundo hola = (HolaMundo) context.getBean("HolaMundo");
%>

```