



Estácio

Missão Prática | Nível 4 | Mundo 3

POLO UNAMAR - CABO FRIO - RJ
DESENVOLVIMENTO FULL STACK

Vamos Integrar Sistemas

9001

2024.3 flex

Gerson Ferreira Cotrim Júnior

1º Procedimento | Camadas de Persistência e Controle

Objetivos:

- 1-Implementar persistência com base em JPA.
- 2-Implementar regras de negócio na plataforma JEE, através de EJBs.
- 3-Implementar sistema cadastral Web com base em Servlets e JSPs.
- 4-Utilizar a biblioteca Bootstrap para melhoria do design.
- 5-No final do exercício, o aluno terá criado todos os elementos necessários para exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para lidar com contextos reais de aplicação.

Classe Produto.java

```
package com.estacio.persistence.entities;

import jakarta.persistence.*;

import java.math.BigDecimal;
import java.util.Objects;

@Entity
public class Produto {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "idProduto")
    private int idProduto;
    @Basic
    @Column(name = "nome")
    private String nome;
    @Basic
    @Column(name = "quantidade")
    private Integer quantidade;
    @Basic
    @Column(name = "precoVenda")
    private BigDecimal precoVenda;

    public int getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(int idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }
}
```

```

    }

    public void setName(String nome) {
        this.nome = nome;
    }

    public Integer getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(Integer quantidade) {
        this.quantidade = quantidade;
    }

    public BigDecimal getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(BigDecimal precoVenda) {
        this.precoVenda = precoVenda;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Produto produto = (Produto) o;
        return idProduto == produto.idProduto && Objects.equals(nome,
produto.nome) && Objects.equals(quantidade, produto.quantidade) &&
Objects.equals(precoVenda, produto.precoVenda);
    }

    @Override
    public int hashCode() {
        return Objects.hash(idProduto, nome, quantidade, precoVenda);
    }
}

```

Classe ProdutoDao.java

```

package com.estacio.persistence.dao;

import com.estacio.persistence.entities.Produto;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import jakarta.persistence.TypedQuery;

import java.math.BigDecimal;
import java.util.List;
@Stateless
public class ProdutoDao {

    @PersistenceContext
    private EntityManager em;

    public Produto criarProduto(String nome, BigDecimal preco) {

```

```

        Produto produto = new Produto();
        produto.setNome(nome);
        produto.setPrecoVenda(preco);
        em.persist(produto);
        return produto;
    }

    public Produto buscarPorId(Long id) {
        return em.find(Produto.class, id);
    }

    public List<Produto> listarTodos() {
        //return em.createQuery("SELECT p FROM Produto p",
        Produto.class).getResultList();

        TypedQuery<Produto> query= em.createQuery("SELECT p FROM Produto p",
        Produto.class);
        return query.getResultList();
    }
}

```

Classe ProdutoListServlet.java

```

package com.estacio.cadastroee;
import com.estacio.persistence.dao.ProdutoDao;
import com.estacio.persistence.entities.Produto;
import jakarta.ejb.EJB;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.util.List;

@WebServlet("/ServletProduto")
public class ProdutoListServlet extends HttpServlet {

    @EJB
    private ProdutoDao produtoDao;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        try {
            List<Produto> produtos = produtoDao.listarTodos();

            resp.setContentType("text/html;charset=UTF-8");
            resp.getWriter().write("<h1>Servlet ServletProduto at /
CadastroEE-WAR</h1>");
            for (Produto produto : produtos) {
                resp.getWriter().write("<p>ID: " + produto.getIdProduto() +
                ", Nome: " + produto.getNome() + ", Preço: " + produto.getPrecoVenda() +
                "</p>");
            }
        } catch (Exception e) {

```

```

        resp.getWriter().write("Erro ao listar produtos: " +
e.getMessage());
    }
}
}

```

pom.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>com.cadastroee</groupId>
    <artifactId>CadastroEE</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>CadastroEE</name>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.target>11</maven.compiler.target>
        <maven.compiler.source>11</maven.compiler.source>
        <junit.version>5.9.2</junit.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.eclipse.persistence</groupId>
            <artifactId>eclipselink</artifactId>
            <version>3.0.2</version>
        </dependency>
        <dependency>
            <groupId>jakarta.persistence</groupId>
            <artifactId>jakarta.persistence-api</artifactId>
            <version>3.0.0</version>
        </dependency>
        <dependency>
            <groupId>jakarta.validation</groupId>
            <artifactId>jakarta.validation-api</artifactId>
            <version>3.0.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jakarta.ejb</groupId>
            <artifactId>jakarta.ejb-api</artifactId>
            <version>4.0.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jakarta.servlet</groupId>
            <artifactId>jakarta.servlet-api</artifactId>
            <version>5.0.0</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>

```

```

        <groupId>org.glassfish.web</groupId>
        <artifactId>jakarta.servlet.jsp.jstl</artifactId>
        <version>3.0.1</version>
    </dependency>

    <dependency>
        <groupId>com.microsoft.sqlserver</groupId>
        <artifactId>mssql-jdbc</artifactId>
        <version>9.4.0.jre11</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jaxb</groupId>
        <artifactId>jaxb-runtime</artifactId>
        <version>3.0.2</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>
    <finalName>CadastroEE-WAR</finalName>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <release>11</release>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>

```

Resultados da execução dos códigos

1- Download, configuração do pool de conexão com SQL Server
Criação de recurso e start do GlassFish Server 6.2.1:

baixar servidor

The screenshot shows a web browser with two tabs. The first tab is titled 'localhost:8080/CadastroEE-WAR' and the second is 'Eclipse GlassFish Downloads'. The address bar shows 'glassfish.org/download.html'. The page content includes a link to the Jakarta EE Platform Specification Project and a list of download links for Eclipse GlassFish 6.2.5. A red box highlights the text 'All Eclipse GlassFish 6.x Downloads' and the link 'Eclipse GlassFish 6.x Downloads'. A red arrow points from this box to the second tab, which is titled 'Eclipse GlassFish 6.x Downloads' and shows the address 'glassfish.org/download_gf6.html'. The page content for the second tab includes a list of download links for Eclipse GlassFish 6.2.2 and a section for Eclipse GlassFish 6.2.1. A red box highlights the link 'Eclipse GlassFish 6.2.1, Jakarta EE Platform, 9.1'.

For more details on Jakarta EE 9.1, please see the [Jakarta EE Platform Specification Project](#).

- [Eclipse GlassFish 6.2.5, Jakarta EE Platform, 9.1](#)
- [Eclipse GlassFish 6.2.5, Jakarta EE Web Profile, 9.1](#)

All Eclipse GlassFish 6.x Downloads

Download all GlassFish 6.x releases at the [Eclipse GlassFish 6.x Downloads](#) page.

localhost:8080/CadastroEE-WAR x EF Eclipse GlassFish 6.x Downloads x +

glassfish.org/download_gf6.html

For more details on Jakarta EE 9.1, please see the [Jakarta EE Platform Specification Project](#).

- [Eclipse GlassFish 6.2.2, Jakarta EE Platform, 9.1](#)
- [Eclipse GlassFish 6.2.2, Jakarta EE Web Profile, 9.1](#)

Eclipse GlassFish 6.2.1

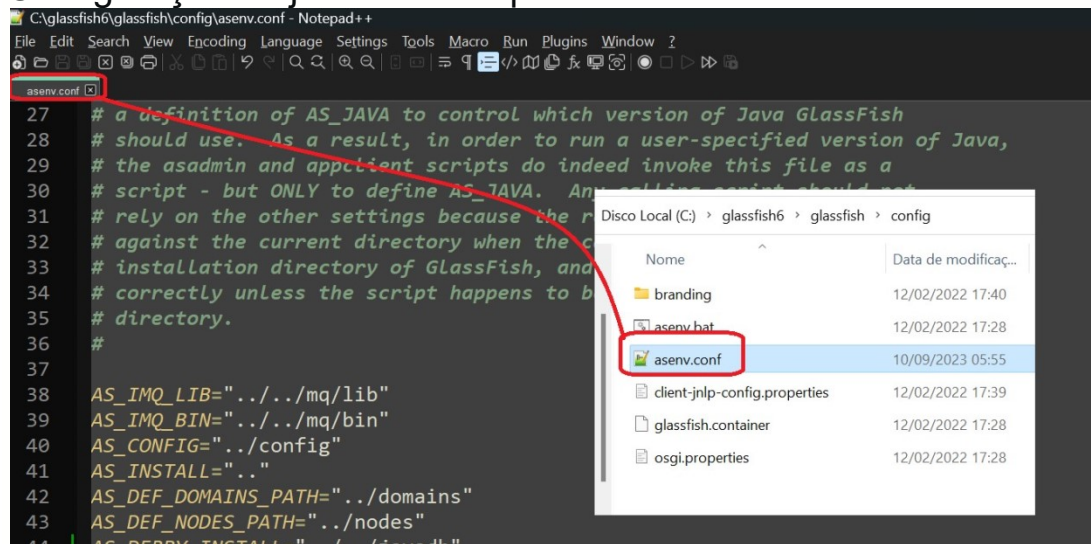
GlassFish 6.2.1 brings integration of Eclipse Exousia, component updates, bug fixes.

For more details on Jakarta EE 9.1, please see the [Jakarta EE Platform Specification Project](#).

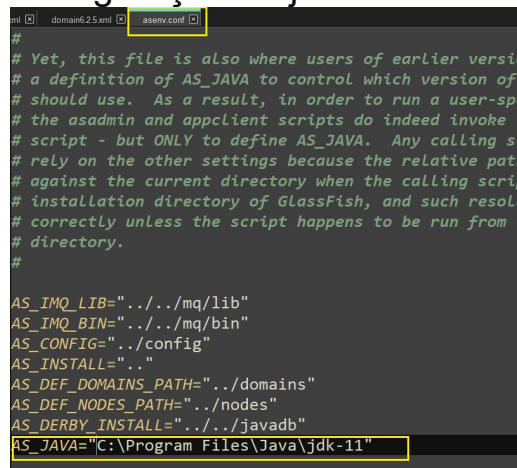
The major change is support for JDK 17.

- [Eclipse GlassFish 6.2.1, Jakarta EE Platform, 9.1](#)
- [Eclipse GlassFish 6.2.1, Jakarta EE Web Profile, 9.1](#)

Configuração do jdk – abrir arquivo



Configuração do jdk 11



Configurando os jars de conexão no domínio a ser acionado



Iniciar servidor

```
GF 6.2.1 start    GF 6.2.1 deploy    GF 6.2.1 config
PS D:\estudo\servidores\glassfish6.2.1\glassfish\bin> ./asadmin start-domain
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain Location: D:\estudo\servidores\glassfish6.2.1\glassfish\domains\domain1
Log File: D:\estudo\servidores\glassfish6.2.1\glassfish\domains\domain1\logs\server.log
Admin Port: 4848
Command start-domain executed successfully.
PS D:\estudo\servidores\glassfish6.2.1\glassfish\bin> |
```

Criar conexão

```
GF 6.2.5 Start    GF 6.2.5 config
PS D:\estudo\servidores\glassfish6.2.5\bin> ./asadmin create-jdbc-connection-pool --datasourceclassname com.microsoft.s
qlserver.jdbc.SQLServerDataSource --restype javax.sql.DataSource --property user=loja:password=loja:databaseName=loja:se
rverName=localhost:portNumber=1434 SqlSeverPool
JDBC connection pool SqlSeverPool created successfully.
Command create-jdbc-connection-pool executed successfully.
PS D:\estudo\servidores\glassfish6.2.5\bin> |
```

Criar recurso

```
GF 6.2.5 Start    GF 6.2.5 config
PS D:\estudo\servidores\glassfish6.2.5\bin> ./asadmin create-jdbc-resource --connectionpoolid SqlSeverPool jdbc/Loja
JDBC resource jdbc/Loja created successfully.
Command create-jdbc-resource executed successfully.
PS D:\estudo\servidores\glassfish6.2.5\bin> |
```

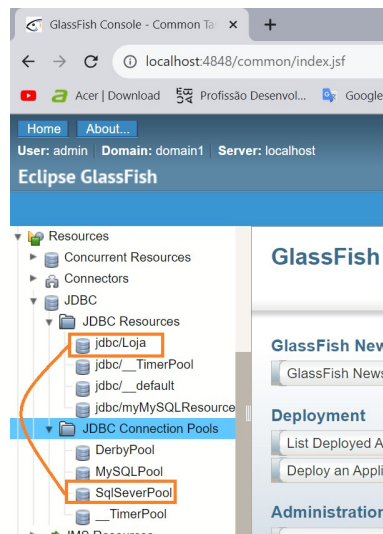
Configurações adicionais para mssql

```
PS D:\estudo\servidores\glassfish6.2.5\bin> ./asadmin set domain.resources.jdbc-connection-pool.SqlSeverPool.property.tr
ustServerCertificate=true
domain.resources.jdbc-connection-pool.SqlSeverPool.property trustServerCertificate=true
Command set executed successfully.
PS D:\estudo\servidores\glassfish6.2.5\bin> ./asadmin set domain.resources.jdbc-connection-pool.SqlSeverPool.property.en
crypt=true
domain.resources.jdbc-connection-pool.SqlSeverPool.property encrypt=true
Command set executed successfully.
PS D:\estudo\servidores\glassfish6.2.5\bin> |
```

Acionando o domínio

```
GF 6.2.5 Start    GF 6.2.5 config
PS D:\estudo\servidores\glassfish6.2.5\bin> ./asadmin start-domain domain1
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain Location: D:\estudo\servidores\glassfish6.2.5\glassfish\domains\domain1
Log File: D:\estudo\servidores\glassfish6.2.5\glassfish\domains\domain1\logs\server.log
Admin Port: 4848
Command start-domain executed successfully.
PS D:\estudo\servidores\glassfish6.2.5\bin> |
```

Abrindo adm



Conferindo config

Edit JDBC Resource

Edit an existing JDBC data source.

[Load Defaults](#)

JNDI Name:

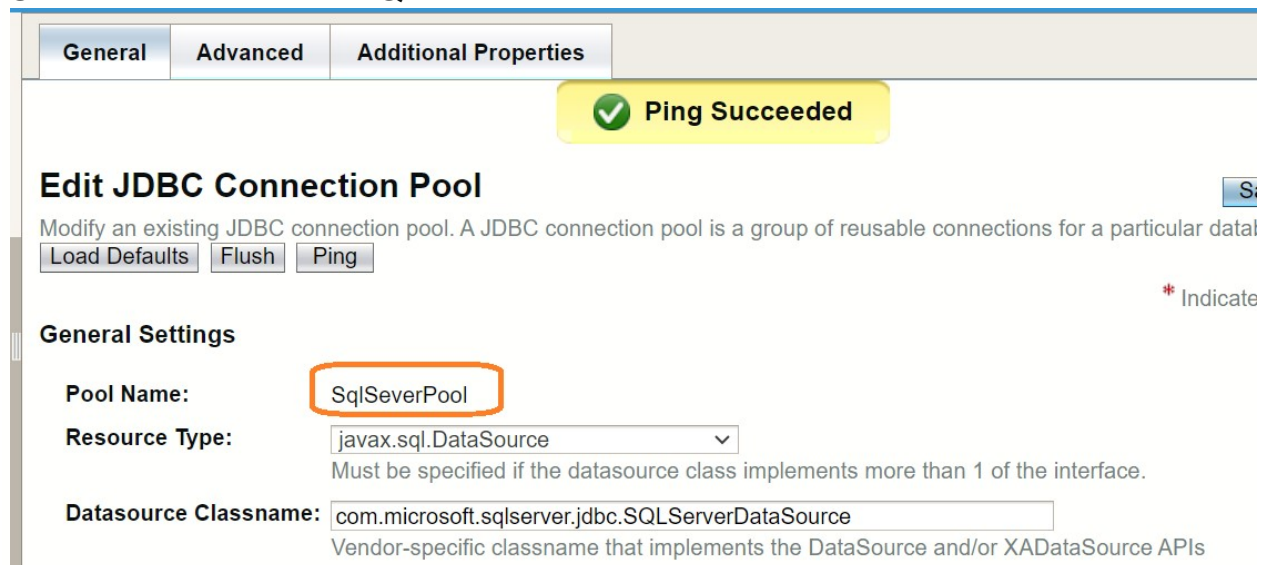
jdbc/Loja

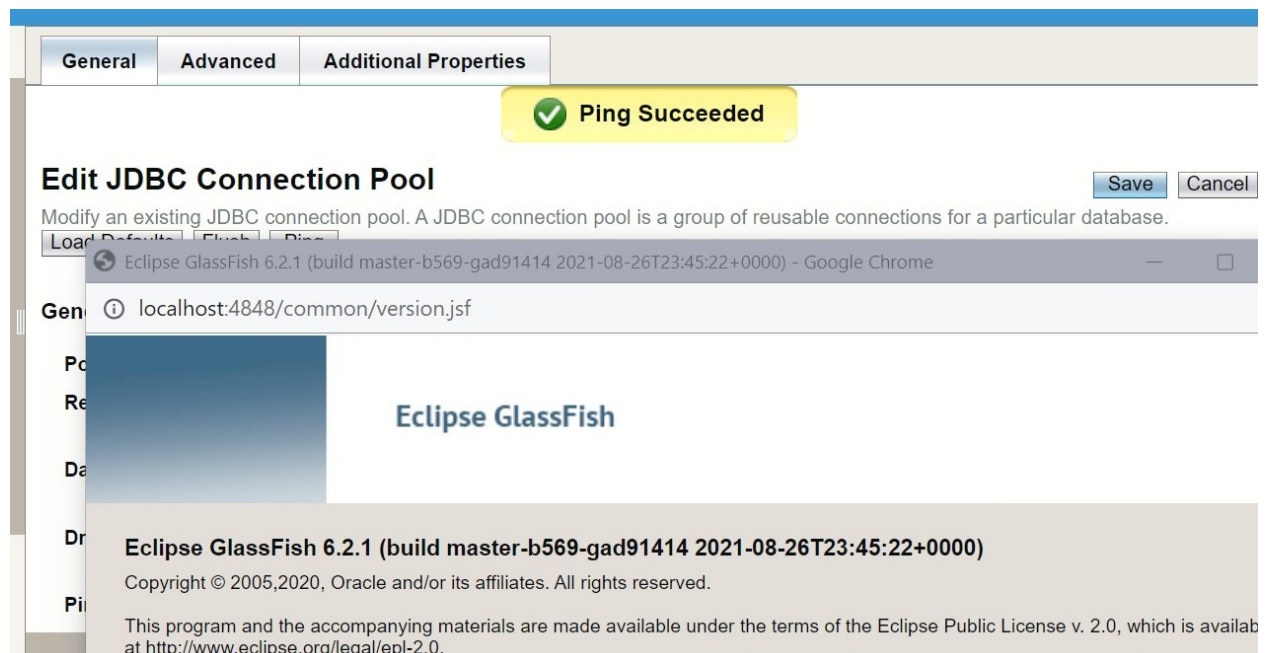
Pool Name:

SqlSeverPool ▾

Use the JDBC Connection Pools page to create new pools

Conferindo conexão SQL Server





Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

Pool Name: SqlSeverPool

Additional Properties (7)		
<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="button" value="Add Property"/> <input type="button" value="Delete Properties"/>		
Select	Name	Value
<input type="checkbox"/>	password	loja
<input type="checkbox"/>	databaseName	loja
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	user	loja
<input type="checkbox"/>	portNumber	1434
<input type="checkbox"/>	trustServerCertificate	true
<input type="checkbox"/>	encrypt	true

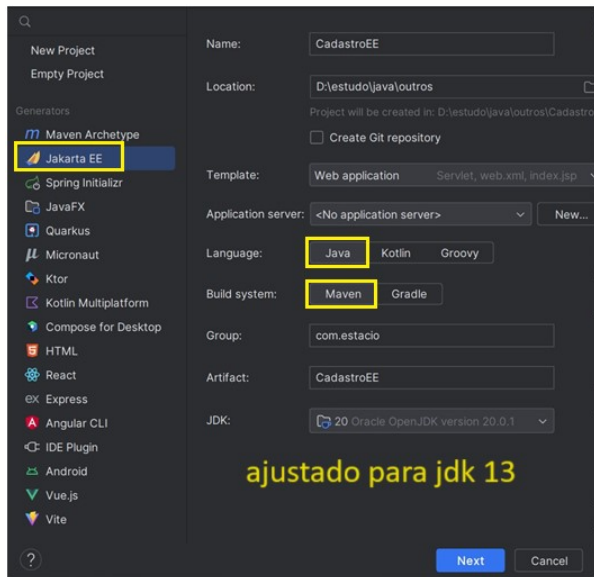
Ping conexão SQL Server via cmd

```

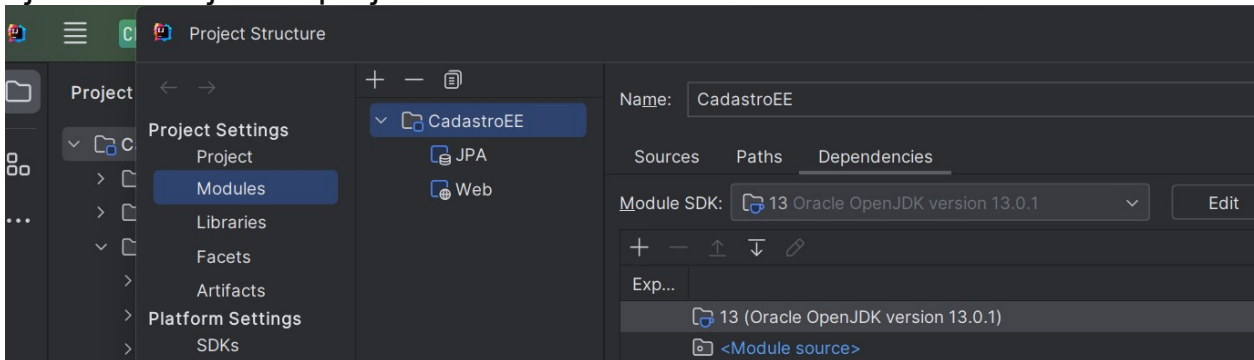
GF 6.2.5 Start x GF 6.2.5 config x + v
PS D:\estudo\servidores\glassfish6.2.5\bin> ./asadmin ping-connection-pool SqlSeverPool
Command ping-connection-pool executed successfully.
PS D:\estudo\servidores\glassfish6.2.5\bin> |

```

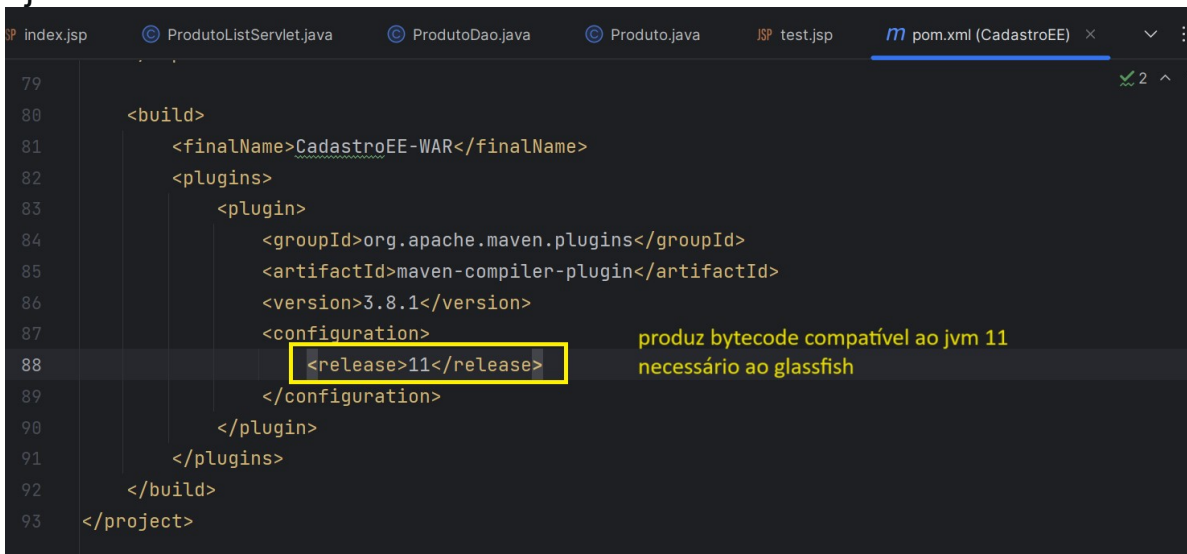
2- Criar o aplicativo corporativo



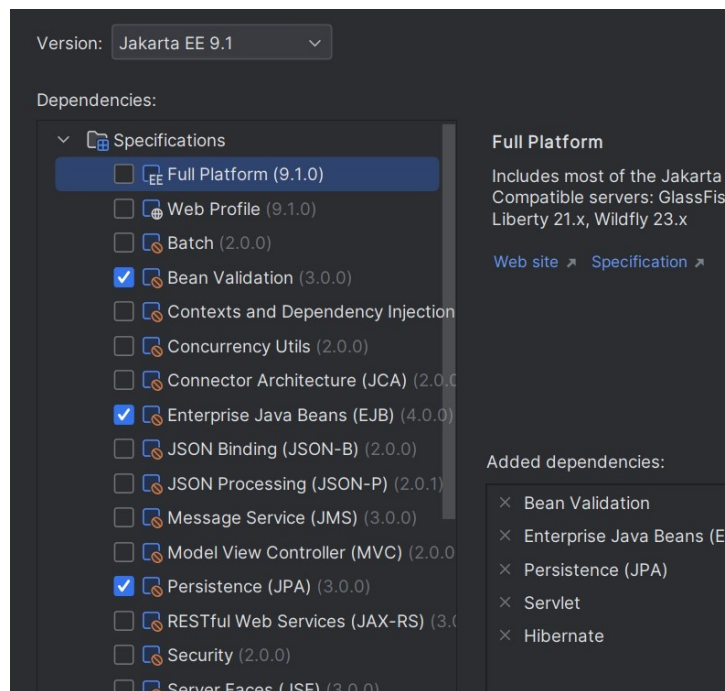
Ajustando o jdk do projeto



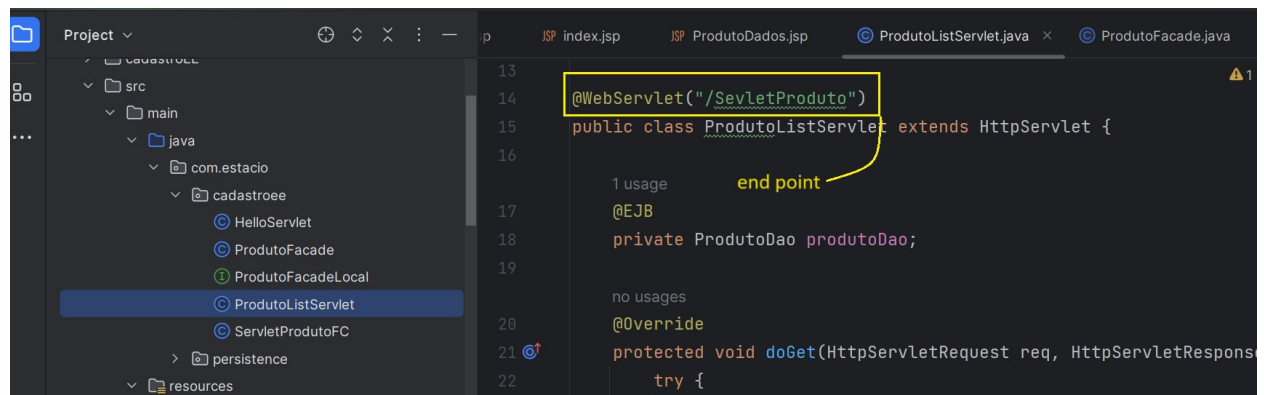
Ajustando o buider



Selecionado dependencias

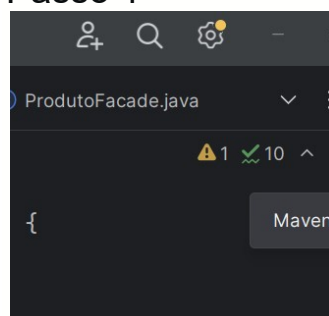


Servlet solicitado

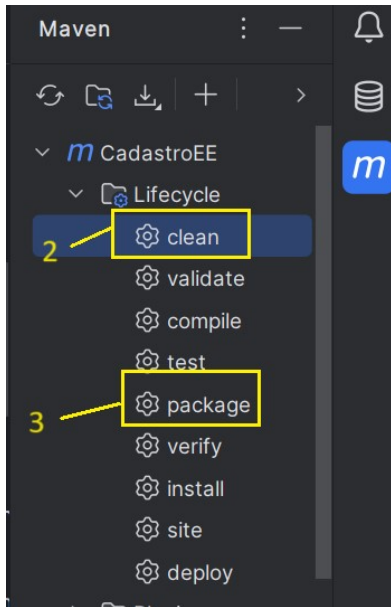


3- A execução deve ser efetuar com o uso de Run ou Deploy no projeto principal (CadastroEE)

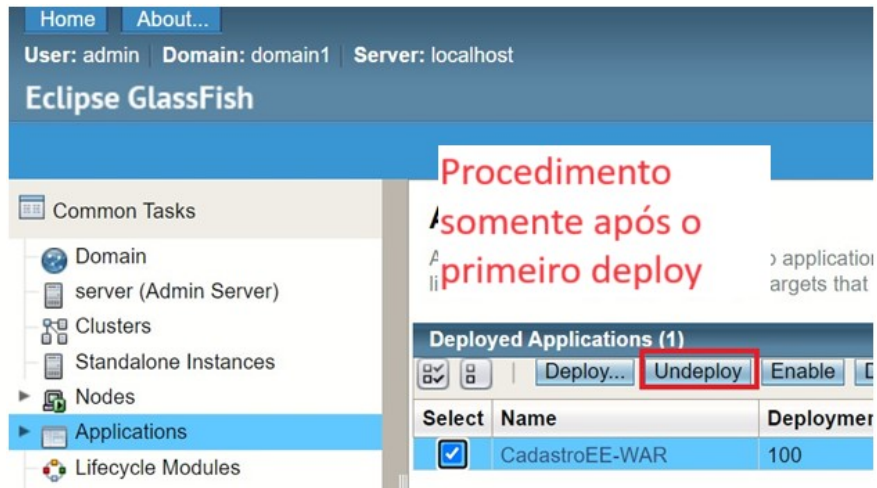
Passo 1



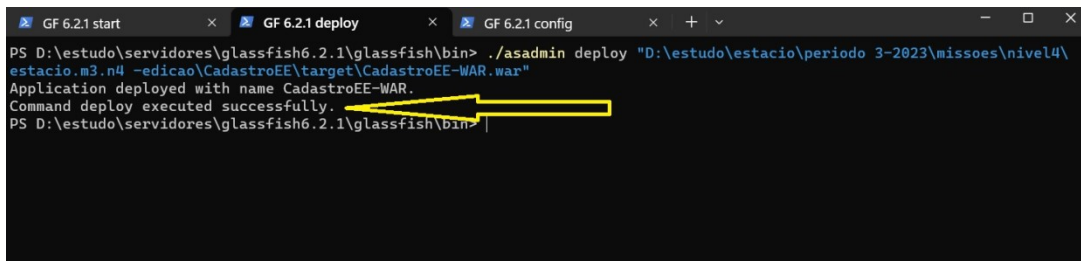
Passo 2



Passo 3

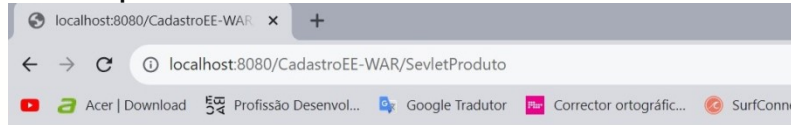


Passo 4



4- Acessar o endereço a seguir, para testar o Servlet `http://localhost:8080/CadastroEE-war/ServletProduto` já tendo alimentado a base

url : `http://localhost:8080/CadastroEE-WAR/SevletProduto`



Servlet ServletProduto at / CadastroEE-WAR

ID: 1, Nome: laranjas, Preço: 30.10

ID: 2, Nome: maçãs, Preço: 38.15

idProduto	nome	quantidade	precoVenda
1	laranjas	10	30,10
2	maçãs	20	38,15

Análise e Conclusão:

1-Como é organizado um projeto corporativo no IntelliJ ?

Estrutura do Projeto como sendo pastas:

`src/main/java`: código fonte em Java.

`src/main/resources`: recursos, como arquivos de configuração.

`src/test/java`: código de teste.

`src/test/resources`: recursos usados durante os testes.

Pacotes: Dentro da estrutura `src/main/java`, você pode organizar seu código em pacotes para separar funcionalidades, por exemplo:

`com.empresa.modelo`: Classes que representam os modelos de dados.

`com.empresa.servico`: Classes de serviços com lógicas de negócios.

`com.empresa.controlador`: Controladores, por exemplo, para uma aplicação web.

Configuração de Build e Dependências:

Utilização de ferramentas como Maven ou Gradle para gerenciar as dependências e o processo de build.

.

Controle de Versão:

Utilização do .gitignore para excluir arquivos temporários e diretórios de saída.

2-Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?

Os EJBs lidam com a lógica de negócios e invocam a camada de persistência (JPA) para interagir com o banco de dados. A combinação dessas duas tecnologias permite que os desenvolvedores construam aplicações robustas, escaláveis e transacionais para a plataforma web no ambiente Java

3-Como o IntelliJ viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?

Na combinação de recursos de edição, validação, integração com bancos de dados e outras ferramentas de suporte faz da IDE uma escolha poderosa para desenvolvedores que trabalham com JPA e EJB. Estas características visam acelerar o processo de desenvolvimento, reduzir erros e melhorar a qualidade do código.

Validação:

A IDE verifica e valida automaticamente seu código JPA e EJB.

Suporte a JPQL:

Ao escrever consultas JPQL (Java Persistence Query Language).

Integração com bancos de dados:

Capacidade de conectar-se diretamente a uma fonte de dados. Com isso, se pode gerar entidades JPA a partir de tabelas existentes.

Geradores de Código:

Getters/setters, construtores e outros membros baseados em campos de classe etc.

Refatoração Avançada:

As ferramentas de refatoração da IntelliJ IDEA são inteligentes o suficiente para reconhecer padrões associados ao JPA e EJB.

Visualização Gráfica de Entidades:

A IntelliJ IDEA oferece uma visualização gráfica das entidades JPA e seus relacionamentos.

etc ...

4-O que são Servlets, e como o IntelliJ oferece suporte à construção desse tipo de componentes em um projeto Web?

Servlets são componentes Java que são executados no servidor e são usados para desenvolver aplicações web

Criando-se diretamente através da interface da IDE uma classe do tipo Servlet

5-Como é feita a comunicação entre os Servlets e os Session Beans do pool de EJBs?

Uma das maneiras mais comuns de acessar EJBs em Servlets no ambiente Java EE é através da injeção de dependência usando a anotação @EJB