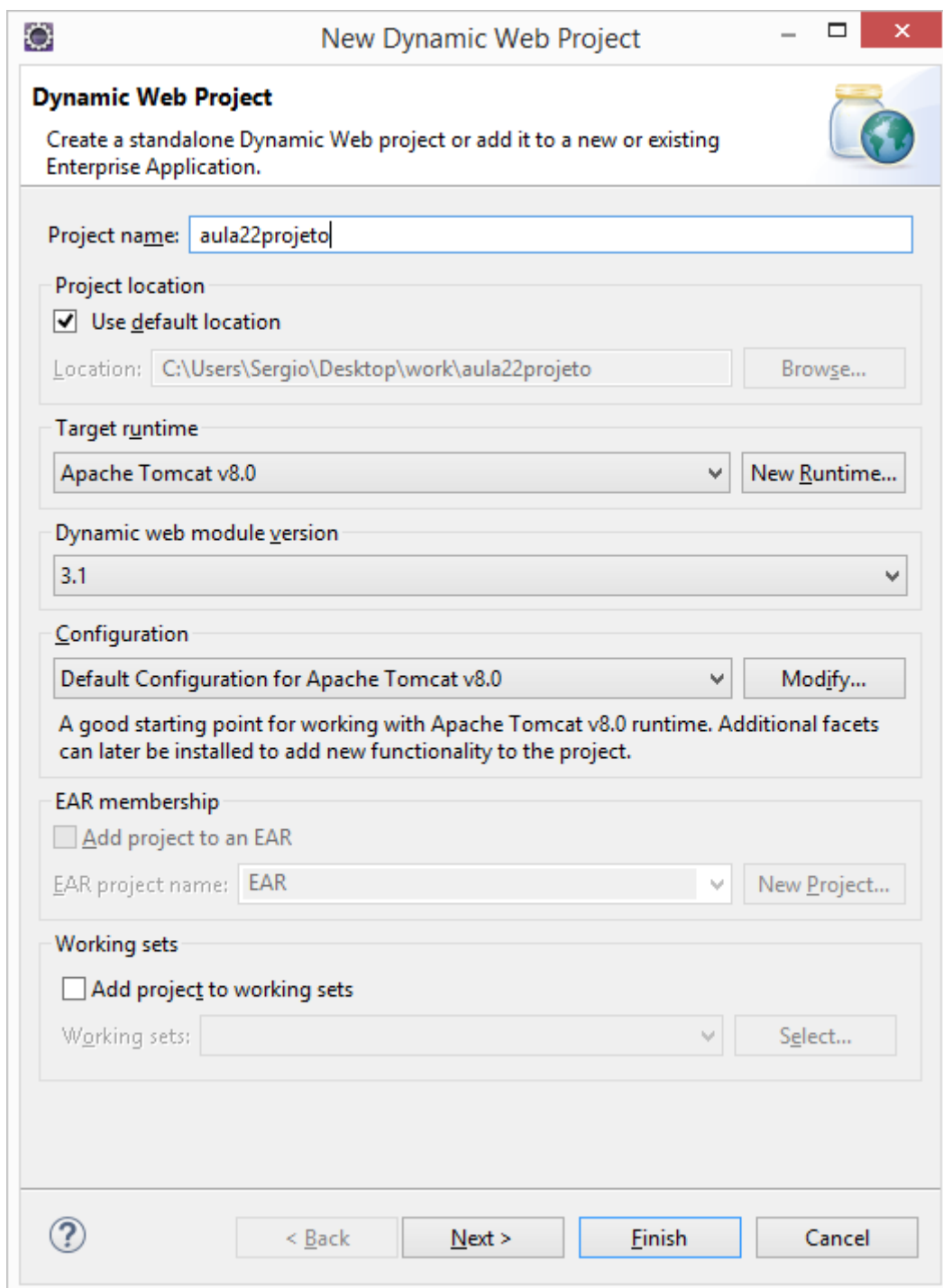


Criando o projeto:

## File > New > Dynamic Web Project



The screenshot shows the 'New Dynamic Web Project' dialog box in the Eclipse IDE. The dialog is titled 'New Dynamic Web Project' and has a close button (X) in the top right corner. It contains several sections for configuring the project:

- Dynamic Web Project**: A section with a description: 'Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.' and a small icon of a globe and a jar.
- Project name:** A text field containing 'aula22projeto'.
- Project location**: A section with a checkbox 'Use default location' which is checked. Below it is a text field 'Location:' containing 'C:\Users\Sergio\Desktop\work\aula22projeto' and a 'Browse...' button.
- Target runtime**: A dropdown menu showing 'Apache Tomcat v8.0' and a 'New Runtime...' button.
- Dynamic web module version**: A dropdown menu showing '3.1'.
- Configuration**: A dropdown menu showing 'Default Configuration for Apache Tomcat v8.0' and a 'Modify...' button. Below this is a descriptive text: 'A good starting point for working with Apache Tomcat v8.0 runtime. Additional facets can later be installed to add new functionality to the project.'
- EAR membership**: A section with a checkbox 'Add project to an EAR' which is unchecked. Below it is a text field 'EAR project name:' containing 'EAR' and a 'New Project...' button.
- Working sets**: A section with a checkbox 'Add project to working sets' which is unchecked. Below it is a text field 'Working sets:' and a 'Select...' button.

At the bottom of the dialog, there is a row of buttons: a help button (question mark), '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

## Configurando o projeto:

### - web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <display-name>struts2_spring</display-name>
  <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>

  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>

  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

</web-app>
```

### - applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans>

</beans>
```

## - beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans>

    <!-- mapeamento da conexão com o banco de dados -->
    <bean id="conexao" class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">

        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/aula22" />
        <property name="username" value="root" />
        <property name="password" value="brqbrq" />

    </bean>

    <!-- mapeamento para o JDBC do Spring (JDBC TEMPLATE) -->
    <bean id="jdbcConfiguracao"
        class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="conexao"></property>
    </bean>

    <!-- INJETAR DEPENDENCIA na classe DAOCliente -->
    <!-- Enviar a configuração para o atributo jdbcTemplate -->
    <bean id="daoCliente" class="br.com.brq.persistence.DAOCliente">
        <property name="jdbcTemplate" ref="jdbcConfiguracao"/>
    </bean>

</beans>
```

## - struts.xml

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

    <constant name="struts.devMode" value="true" />

    <!-- mapear todas as ações de cliente no sistema -->
    <package name="cliente" extends="struts-default">

        <!-- ação de cadastro de cliente -->
        <action name="cadastrar" method="cadastrar"
            class="br.com.brq.control.ControleCliente">
            <!-- redirecionamento -->
            <result name="success">/cadastro.jsp</result>

        </action>
    </package>
</struts>
```

## Criando a entidade:

```
package br.com.brq.entities;

public class Cliente {

    private Integer idCliente;
    private String nome;
    private String email;

    public Cliente() {
        // TODO Auto-generated constructor stub
    }

    public Cliente(Integer idCliente, String nome, String email) {
        super();
        this.idCliente = idCliente;
        this.nome = nome;
        this.email = email;
    }

    public Integer getIdCliente() {
        return idCliente;
    }

    public void setIdCliente(Integer idCliente) {
        this.idCliente = idCliente;
    }

    public String getNome() {
        return nome;
    }

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

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "Cliente [idCliente=" + idCliente + ", nome="
            + nome + ", email=" + email + "];"
    }
}
```

## Criando a classe de persistência para Cliente:

```
package br.com.brq.persistence;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

import br.com.brq.entities.Cliente;

public class DAOCliente {

    //ferramenta para desenvolvimento JDBC com Spring...
    private JdbcTemplate jdbcTemplate;

    //INJEÇÃO DE DEPENDENCIA..
    //Método set para que o atributo jdbcTemplate receba a configuração
    //definida no arquivo applicationContext.xml
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    //método para gravar um cliente na base de dados..
    public void insert(Cliente c) throws Exception{

        String query = "insert into cliente(nome, email) values(?, ?)";
        jdbcTemplate.update(query,
            new Object[]{ c.getNome(), c.getEmail() } );
    }

    //método para atualizar um cliente na base de dados..
    public void update(Cliente c) throws Exception{

        String query = "update cliente set nome = ?, email = ? where idcliente = ?";

        jdbcTemplate.update(query,
            new Object[]{ c.getNome(), c.getEmail(), c.getIdCliente() } );
    }

    //método para excluir 1 cliente..
    public void delete(Integer idCliente) throws Exception{

        String query = "delete from cliente where idcliente = ?";
        jdbcTemplate.update(query, new Object[]{ idCliente } );
    }
}
```

```
}

//método para listar todos os clientes..
public List<Cliente> findAll() throws Exception{

    String query = "select * from cliente order by nome";

    @SuppressWarnings("unchecked")
    List<Cliente> lista = jdbcTemplate.query(query, new ClienteMapper());
    return lista;
}

//método para buscar 1 cliente pelo id..
public Cliente findById(Integer idCliente) throws Exception{

    String query = "select * from cliente where idcliente = ?";

    @SuppressWarnings("unchecked")
    List<Cliente> lista = jdbcTemplate.query(query,
        new Object[]{ idCliente }, new ClienteMapper());

    if(lista != null && !lista.isEmpty()){
        return lista.get(0); //retornar o primeiro elemento da lista..
    }
    else{
        return null; //vazio..
    }
}

}

//classe auxiliar para ler os campos da tabela cliente e montar um objeto..
class ClienteMapper implements RowMapper{

    @Override
    public Object mapRow(ResultSet rs, int rowNum) throws SQLException {

        Cliente c = new Cliente(); //entidade..

        c.setIdCliente(rs.getInt("idcliente"));
        c.setNome(rs.getString("nome"));
        c.setEmail(rs.getString("email"));

        return c;
    }

}
```

### Script da base de dados:

```
drop database if exists aula22;
create database aula22;
use aula22;

create table cliente(
    idcliente          integer          auto_increment,
    nome               varchar(50)      not null,
    email              varchar(50)      not null,
    primary key(idcliente));

desc cliente;
```

---

### Página de cadastro de clientes:

cadastro.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!-- taglib do struts -->
<%@ taglib uri="/struts-tags" prefix="s" %>

<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Projeto</title>
</head>

<body>

    <h3>Cadastro de Clientes</h3>
    Projeto Struts 2 com Spring e JDBC
    <hr/>

    <s:form action="cadastrar">

        <s:textfield label="Nome do Cliente" name="cliente.nome"/>
        <s:textfield label="Endereço de Email" name="cliente.email"/>

        <s:submit value="Cadastrar Cliente"/>

    </s:form>

    <h4> ${mensagem} </h4>

</body>
</html>
```

### Classe de controle:

ControleCliente.java

```
package br.com.brq.control;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts2.ServletActionContext;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.opensymphony.xwork2.ActionSupport;

import br.com.brq.entities.Cliente;
import br.com.brq.persistence.DAOCliente;

public class ControleCliente extends ActionSupport {

    // atributo para armazenar os dados do cliente do formulário..
    private Cliente cliente;

    //atributo para a classe de persistencia..
    private DAOCliente daoCliente;

    // construtor..
    public ControleCliente() {
        cliente = new Cliente(); // instanciando..

        ApplicationContext ctx = new ClassPathXmlApplicationContext("beans.xml");

        daoCliente = (DAOCliente) ctx.getBean("daocliente"); //nome no xml..
    }

    //método para a ação de cadastro..
    public String cadastrar(){

        HttpServletRequest request = ServletActionContext.getRequest();

        try{

            daoCliente.insert(cliente); //gravando

            request.setAttribute("mensagem", "Dados gravados.");
            cliente = new Cliente(); //instanciando..
        }
        catch(Exception e){
            e.printStackTrace();
            request.setAttribute("mensagem", e.getMessage());
        }

        return SUCCESS;
    }
}
```

```
public Cliente getCliente() {  
    return cliente;  
}  
  
public void setCliente(Cliente cliente) {  
    this.cliente = cliente;  
}  
}
```

- **Estrutura do projeto:**

