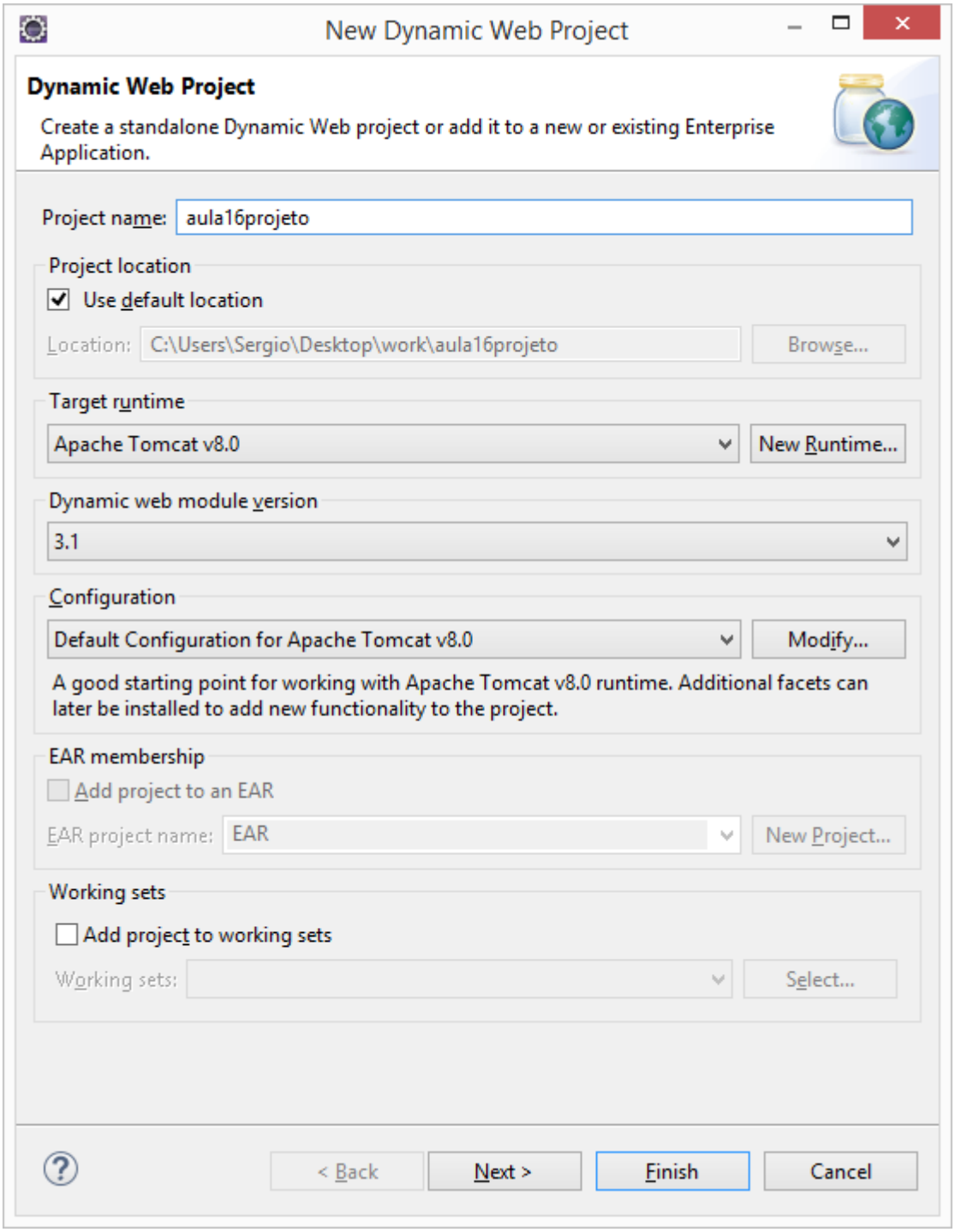
Criando o projeto:

File > New > Dynamic Web Project



Criando e mapeando a entidade Funcionario:

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

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "funcionario")
public class Funcionario {

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	@Column(name = "idfuncionario")
	private Integer idFuncionario;

	@Column(name = "nome", length = 50, nullable = false)
	private String nome;

	@Column(name = "email", length = 50, nullable = false, unique = true)
	private String email;

	@Column(name = "salario", nullable = false)
	private Double salario;

	@Column(name = "funcao", length = 100, nullable = false)
	private String funcao;

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

	public Funcionario(Integer idFuncionario, String nome, String email,
			Double salario, String funcao) {
		this.idFuncionario = idFuncionario;
		this.nome = nome;
		this.email = email;
		this.salario = salario;
		this.funcao = funcao;
	}

	public Integer getIdFuncionario() {
		return idFuncionario;
	}
```

```
	public void setIdFuncionario(Integer idFuncionario) {
		this.idFuncionario = idFuncionario;
	}

	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;
	}

	public Double getSalario() {
		return salario;
	}

	public void setSalario(Double salario) {
		this.salario = salario;
	}

	public String getFuncao() {
		return funcao;
	}

	public void setFuncao(String funcao) {
		this.funcao = funcao;
	}

	@Override
	public String toString() {
		return "Funcionario [idFuncionario=" + idFuncionario + ", nome="
				+ nome + ", email=" + email + ", salario="
				+ salario + ", funcao=" + funcao + "]";
	}

}
```

Configurando o Hibernate para acesso ao MySql

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-
configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">
             org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">
             com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">
             jdbc:mysql://localhost:3306/aula16</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">brqbrq</property>

    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.format_sql">true</property>

    <mapping class="br.com.brq.entities.Funcionario"/>

 </session-factory>
</hibernate-configuration>
```

Gerando as tabelas no banco de dados:

```
package br.com.brq.util;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.cfg.Configuration;
import org.hibernate.tool.hbm2ddl.SchemaExport;

public class GenerateTables {

        public static void main(String[] args) {

                try{
                        Configuration cfg = new AnnotationConfiguration();
                        cfg.configure("br/com/brq/config/mysql_hibernate.cfg.xml");

                        SchemaExport s = new SchemaExport(cfg);
                        s.create(true, true);
                }
                catch(Exception e){
                        System.out.println(e.getMessage());
                }
        }
}
```

Apache Struts 2

O objetivo do Struts é separar o *model* (modelo - lógica de aplicativo que interage com um banco de dados) do *view* (visualização - páginas HTML apresentadas para o cliente) e do *controller* (controlador - instância que transmite informações entre a exibição e o modelo).

O Struts é um framework, baseado em open-source pelo projeto Jakarta, auxiliando a criação de aplicações para a Web. O Struts foi criado em Java, e seu núcleo é formado por uma camada flexível, proveniente das tecnologias Java Servlets, JavaBeans e XML. Contamos ainda com o desenvolvimento de aplicações do moedlo MVC (Model-View-Controller).

Configurando o Struts no projeto web:
\WEB-INF\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">

  <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>

   <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>
```

Criando as páginas do projeto:

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Projeto</title>

	<link rel="stylesheet" type="text/css"
		href="css/bootstrap.min.css"/>
	<link rel="stylesheet" type="text/css"
		href="css/bootstrap-theme.min.css"/>

</head>
<body class="container">

	<h2>Projeto Controle de Funcionarios</h2>
	Struts2 (MVC) com Hibernate e JPA
	<hr/>

	<ul>
		<li> <a href="cadastro.jsp">
			Cadastrar Funcionarios</a> </li>
		<li> <a href="consultarfuncionarios.action">
			Consultar Funcionarios</a> </li>
	</ul>

</body>
</html>
```

cadastro.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

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

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Projeto</title>

	<link rel="stylesheet" type="text/css"
		href="css/bootstrap.min.css"/>

	<link rel="stylesheet" type="text/css"
		href="css/bootstrap-theme.min.css"/>
```

```
        <style type="text/css">
                .label { color: #000; }
                .errorMessage { color: #FF0000; font-size: 10pt; }
        </style>

</head>
<body class="container">

        <h2>Cadastro de Funcionarios</h2>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>

        <div>
                Para cadastrar um novo funcionario, informe os dados abaixo:
        </div>
        <br/>

        <!-- formulário padrão struts -->
        <s:form name="formulario" method="post" action="cadastrarfuncionario">

                <s:textfield label="Nome do Funcionario"
                             name="funcionario.nome"/>
                <s:textfield label="Email"
                             name="funcionario.email"/>
                <s:textfield label="Salário"
                             name="funcionario.salario"/>
                <s:textfield label="Função"
                             name="funcionario.funcao"/>

                <s:submit value="Cadastrar Funcionario"
                             cssClass="btn btn-success btn-sm"/>

        </s:form>

        <h4>${mensagem}</h4>

</body>
</html>
```

consulta.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

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

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Projeto</title>

        <link rel="stylesheet" type="text/css"
                href="css/bootstrap.min.css"/>
        <link rel="stylesheet" type="text/css"
                href="css/bootstrap-theme.min.css"/>
```

```
</head>
<body class="container">

	<h2>Consulta de Funcionarios</h2>
	<a href="index.jsp">Voltar</a> para a página inicial.
	<hr/>

	<div>
		Relação de Funcionários cadastrados:
	</div>

	<h4>${mensagem}</h4>

	<table class="table table-hover">
		<thead>
			<tr>
				<th>Código</th>
				<th>Nome do Funcionario</th>
				<th>Email</th>
				<th>Salário</th>
				<th>Função</th>
				<th>Operações</th>
			</tr>
		</thead>
		<tbody>

			<!-- varrer a listagem de funcionarios.. -->
			<s:iterator value="listagemFuncionarios">
				<tr>
					<td> <s:property value="idFuncionario"/> </td>
					<td> <s:property value="nome"/> </td>
					<td> <s:property value="email"/> </td>
					<td> <s:property value="salario"/> </td>
					<td> <s:property value="funcao"/> </td>
					<td>

						<a href="excluirfuncionario.action
								?id=${idFuncionario}"
						   class="btn btn-danger btn-sm">
						    Excluir
						</a>

					</td>
				</tr>
			</s:iterator>

		</tbody>
		<tfoot>
			<tr>
				<td colspan="6">
					Quantidade de Funcionarios:
					<s:property value="listagemFuncionarios.size()"/>
				</td>
			</tr>
		</tfoot>
	</table>

</body>
</html>
```

Controller:

ControleFuncionario.java

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

import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

import br.com.brq.entities.Funcionario;
import br.com.brq.persistence.DAOFuncionario;

//Classe de controle do struts 2..
//Toda Classe de controle do Struts, deverá herdar ActonSupport
//Toda Classe de controle do Struts deverá implementar a interface
//ModelDriven através do qual irá definir a sua classe de modelo (dados)
public class ControleFuncionario extends ActionSupport implements ModelDriven<Funcionario>{

	//atributo..
	private Funcionario funcionario; //entidade..
	private List<Funcionario> listagemFuncionarios; //consulta.

	//construtor..
	public ControleFuncionario() {
		//espaço de memória para o atributo funcionario..
		funcionario = new Funcionario(); //instanciando..
	}

	//método para receber o submit do formulário de cadastro...
	public String cadastrar(){

		//criar o request e o response...
		HttpServletRequest request = (HttpServletRequest) ActionContext
			.getContext().get(ServletActionContext.HTTP_REQUEST);

		try{
			DAOFuncionario d = new DAOFuncionario(); //persistencia..
			d.saveOrUpdate(funcionario); //gravando..

			request.setAttribute("mensagem", "Funcionario " + funcionario.getNome()
					+ ", cadastrado com sucesso.");

			funcionario = new Funcionario(); //novo espaço de memória..
		}
		catch(Exception e){
			//mensagem de erro..
			request.setAttribute("mensagem", "Erro: " + e.getMessage());
		}
```

```
		return SUCCESS;
	}

	//método para executar a consulta de funcionarios..
	public String consultar(){

		HttpServletRequest request = (HttpServletRequest) ActionContext.getContext()
						.get(ServletActionContext.HTTP_REQUEST);

		try{
			DAOFuncionario d = new DAOFuncionario(); //persistencia..
			listagemFuncionarios = d.findAll(); //executando a consulta..
		}
		catch(Exception e){
			request.setAttribute("mensagem", "Erro: " + e.getMessage());
		}

		return SUCCESS;
	}

	public String excluir(){

		HttpServletRequest request = (HttpServletRequest) ActionContext.getContext()
				.get(ServletActionContext.HTTP_REQUEST);

		try{
			//resgatando o id enviado pela URL..
			Integer idFuncionario = Integer.parseInt(request.getParameter("id"));

			//buscar o funcionario pelo id..
			DAOFuncionario d = new DAOFuncionario(); //persistencia..
			Funcionario f = d.findById(idFuncionario); //obtendo..

			d.delete(f); //excluindo..

			request.setAttribute("mensagem", "Funcionario " + f.getNome()
						+ ", excluido com sucesso.");

			listagemFuncionarios = d.findAll();
		}
		catch(Exception e){
			request.setAttribute("mensagem", "Erro: " + e.getMessage());
		}

		return SUCCESS;
	}



	public Funcionario getFuncionario() {
		return funcionario;
	}
```

```
	public void setFuncionario(Funcionario funcionario) {
		this.funcionario = funcionario;
	}

	public List<Funcionario> getListagemFuncionarios() {
		return listagemFuncionarios;
	}

	@Override
	public Funcionario getModel() {
		//retornar um objeto instanciado da classe de modelo..
		return funcionario; //retornando o atributo..
	}
}
```

XML para validação dos dados do Funcionario:

ControleFuncionario-validation.xml

```
<!DOCTYPE validators PUBLIC
        "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
        "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">

<validators>

	<!-- Arquivo para mapear as validações do ControleFuncionario -->
	<!-- iremos validar o atributo 'funcionario' -->

	<!-- Validar o nome do funcionario -->
	<field name="funcionario.nome">

		<!-- campo obrigatorio -->
		<field-validator type="requiredstring">
			<message>Por favor, informe o nome do
				Funcionario.</message>
		</field-validator>

		<!-- expressao regular -->
		<field-validator type="regex">
			<param name="expression">^[A-Za-zÀ-Üà-ü\s]{3,50}$</param>
			<message>Erro. Informe um nome valido para o
				Funcionario.</message>
		</field-validator>

	</field>


	<!-- Validar o email do funcionario -->
	<field name="funcionario.email">

		<!-- campo obrigatorio -->
		<field-validator type="requiredstring">
```

```
                <message>Por favor, informe o email do
                        Funcionario.</message>
            </field-validator>

            <!-- tipo email -->
            <field-validator type="email">
                <message>Erro. Informe um email valido para o
                        Funcionario.</message>
            </field-validator>

        </field>


        <!-- Validar o salario do funcionario -->
        <field name="funcionario.salario">

            <!-- campo obrigatorio -->
            <field-validator type="required">
                <message>Por favor, informe o salario do
                        Funcionario.</message>
            </field-validator>

            <!-- intervalo -->
            <field-validator type="double">
                <param name="minInclusive">890.0</param>
                <param name="maxInclusive">10000.0</param>
                <message>Por favor, informe um salario entre
                        ${minInclusive} e ${maxInclusive}</message>
            </field-validator>

        </field>


        <!-- Validar a funcao do funcionario -->
        <field name="funcionario.funcao">

            <!-- campo obrigatorio -->
            <field-validator type="requiredstring">
                <message>Por favor, informe a função do
                        Funcionario.</message>
            </field-validator>

            <!-- expressao regular -->
            <field-validator type="regex">
                <param name="expression">^[A-Za-zÀ-Üà-ü\s]{3,100}$</param>
                <message>Erro. Informe uma função valida para o
                        Funcionario.</message>
            </field-validator>

        </field>

</validators>
```