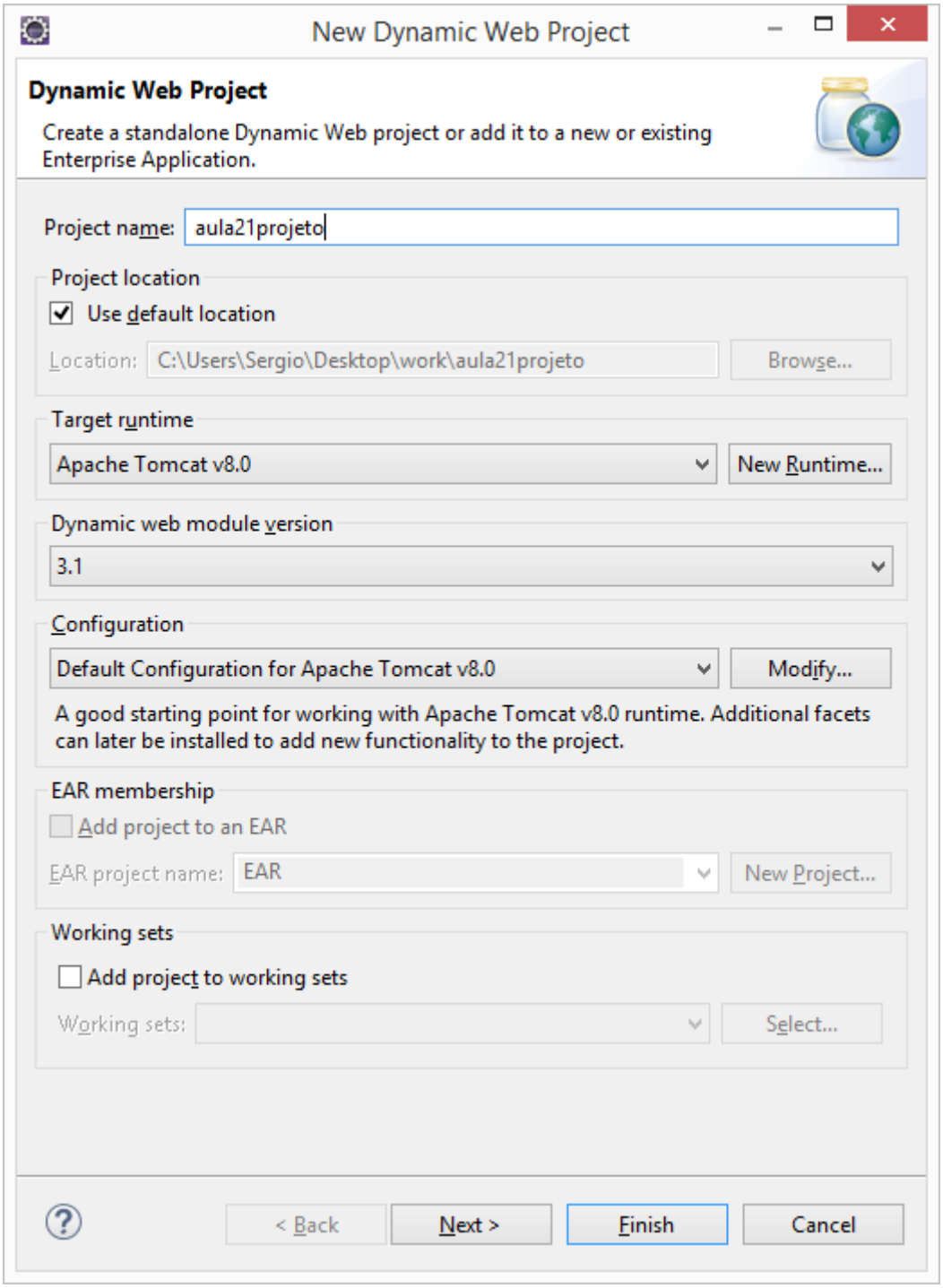
Novo projeto:

- File / New / Dynamic web Project



Spring Framework

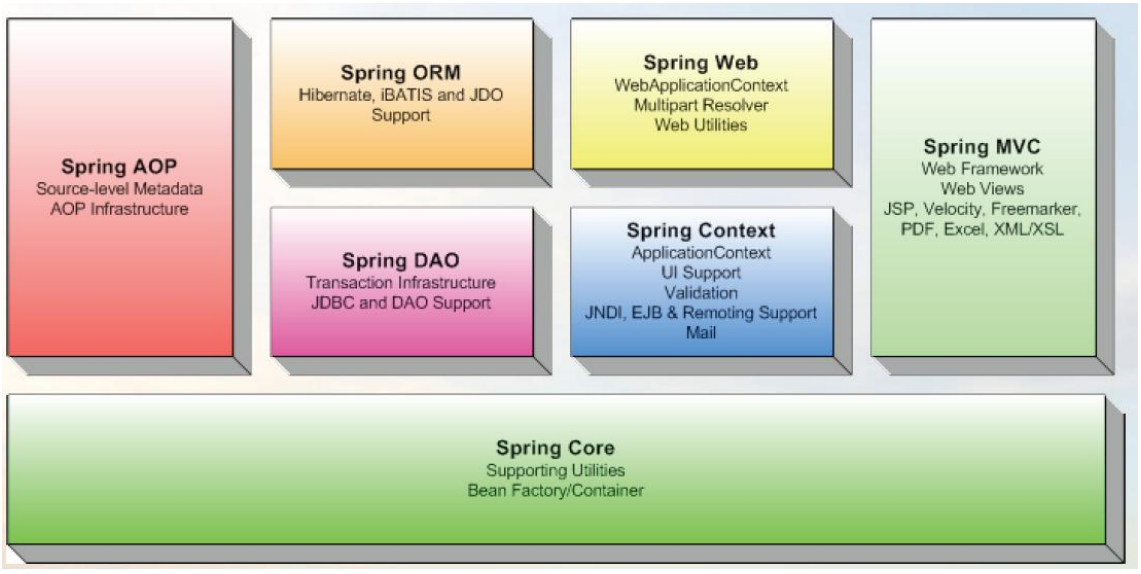
https://spring.io/

O Spring é um *framework open source* para a plataforma Java criado por *Rod Johnson* e descrito em seu livro "Expert One-on-One: JEE Design e Development". Trata-se de um *framework* não intrusivo, baseado nos padrões de projeto inversão de controle (IoC) e injeção de dependência.

No Spring o *container* se encarrega de "instanciar" classes de uma aplicação Java e definir as dependências entre elas através de um arquivo de configuração em formato XML, inferências do framework, o que é chamado de auto-wiring ou ainda anotações nas classes, métodos e propriedades. Dessa forma o Spring permite o baixo acoplamento entre classes de uma aplicação orientada a objetos.

O Spring possui uma arquitetura baseada em interfaces e POJOs (Plain Old Java Objects), oferecendo aos POJOs características como mecanismos de segurança e controle de transações. Também facilita testes unitários e surge como uma alternativa à complexidade existente no uso de EJBs. Com Spring, pode-se ter um alto desempenho da aplicacao.

Esse *framework* oferece diversos módulos que podem ser utilizados de acordo com as necessidades do projeto, como módulos voltados para desenvolvimento Web, persistência, acesso remoto e programação orientada a aspectos.



Injeção de Dependencia:

Injeção de dependência (Dependency Injection, em inglês) é um padrão de desenvolvimento de programas de computadores utilizado quando é necessário manter baixo o nível de acoplamento entre diferentes módulos de um sistema. Nesta solução as dependências entre os módulos não são

definidas programaticamente, mas sim pela configuração de uma infraestrutura de software (container) que é responsável por "injetar" em cada componente suas dependências declaradas. A Injeção de dependência se relaciona com o padrão Inversão de controle, mas não pode ser considerado um sinônimo deste.

Criando a entidade Pessoa:

```
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.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "pessoa")
@NamedQueries(
		{
			@NamedQuery(name  = Pessoa.FIND_ALL,
					query = "from Pessoa")
		}
)
public class Pessoa {

	public static final String FIND_ALL = "pessoa.findall";

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

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

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

	public Pessoa() {
	}
```

```
	public Pessoa(Integer idPessoa, String nome, String email) {
		this.idPessoa = idPessoa;
		this.nome = nome;
		this.email = email;
	}

	public Integer getIdPessoa() {
		return idPessoa;
	}

	public void setIdPessoa(Integer idPessoa) {
		this.idPessoa = idPessoa;
	}

	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 "Pessoa [idPessoa=" + idPessoa + ", nome=" + nome
			+ ", email=" + email + "]";
	}
}
```

Utilizando o HibernateTemplate

Implementação do hibernate como componente do SpringFramework

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

import java.util.List;

import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateTemplate;
```

```
import br.com.brq.entities.Pessoa;

public class DAOPessoa {

	//Hibernate embedded no spring...
	private HibernateTemplate hibernate; //null.

	//INJEÇÂO DE DEPENDENCIA!!
	//método para receber a configuração de SessionFactory do applicationContext.xml
	//esta configuração será enviada por injeção de dependencia, através do Spring..
	public void setSessionFactory(SessionFactory sessionFactory){
		//utilizar este sessionFactory recebido para inicializar o HibernateTemplate..
		hibernate = new HibernateTemplate(sessionFactory);
	}


	//método para cadastrar pessoa..
	public void insert(Pessoa p) throws Exception{
		hibernate.persist(p); //gravando..
	}

	//método para atualizar pessoa..
	public void update(Pessoa p) throws Exception{
		hibernate.merge(p); //atualizando..
	}

	//método para excluir pessoa..
	public void delete(Pessoa p) throws Exception{
		hibernate.delete(p); //excluindo..
	}

	//método para buscar 1 pessoa pelo id..
	public Pessoa findById(Integer idPessoa) throws Exception{
		return (Pessoa) hibernate.get(Pessoa.class, idPessoa);
	}

	//método para listar todos os registros..
	@SuppressWarnings("unchecked")
	public List<Pessoa> findAll() throws Exception{
		return hibernate.findByNamedQuery(Pessoa.FIND_ALL);
	}
}
```

Configurando o SpringFramework:

- \WEB-INF\web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" version="3.1">

  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>

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

  <listener>
    <listener-class>
        org.springframework.web.context.request.RequestContextListener
    </listener-class>
  </listener>

</web-app>
```

- \WEB-INF\faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
      http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
    version="2.2">

<application>

   <el-resolver>
      org.springframework.web.jsf.el.SpringBeanFacesELResolver
   </el-resolver>

</application>

</faces-config>
```

-\WEB-INF\applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:sec="http://www.springframework.org/schema/security"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
        http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
        http://www.springframework.org/schema/security
       http://www.springframework.org/schema/security/spring-security-
       3.1.xsd">

       <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/aula21" />
              <property name="username" value="root" />
              <property name="password" value="brqbrq" />
       </bean>

       <bean id="hibernate"
       class="org.springframework.orm.hibernate3.annotation
              .AnnotationSessionFactoryBean">
              <property name="dataSource" ref="conexao" />
              <property name="annotatedClasses">
                     <list>
                            <value>br.com.brq.entities.Pessoa</value>
                     </list>
              </property>

              <property name="hibernateProperties">
                     <props>
                            <prop key="hibernate.dialect">
                                   org.hibernate.dialect.MySQLDialect
                            </prop>
                            <prop key="hibernate.show_sql">true</prop>
                     </props>
              </property>
       </bean>

       <!-- Mapear a injeção de dependencia para a classe DAOPessoa -->
       <bean id="daopessoa" class="br.com.brq.persistence.DAOPessoa">
              <!-- Enviando a configuração da sessionFactory.. -->
              <property name="sessionFactory" ref="hibernate"/>
       </bean>
</beans>
```

Criando a página de cadastro/consulta:

manterpessoas.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui" >

      <h:head>
            <title>Projeto</title>
      </h:head>

      <h:body>

      <p:panel header="Cadastro de Pessoas">

      <h:outputText value="Para cadastrar uma pessoa,
            informe os dados abaixo:"/>

      <h:form id="formcadastro">

            <h:panelGrid columns="2">

            <p:outputLabel value="Nome da Pessoa:" for="nome"/>
            <p:inputText id="nome" required="true"
                  requiredMessage="Por favor, informe o nome da pessoa."
                  value="#{mbpessoa.pessoa.nome}" />

            <p:outputLabel value="Endereço de Email:" for="email"/>
            <p:inputText id="email" required="true"
                  requiredMessage="Por favor, informe o email da pessoa."
                  value="#{mbpessoa.pessoa.email}" />

            <p:commandButton value="Cadastrar Pessoa" ajax="true"
            update=":formcadastro" action="#{mbpessoa.cadastrar}"/>

            </h:panelGrid>

      <div>
            <p:messages/>
            <p:growl/>
      </div>

      </h:form>

      </p:panel>

      <p:panel header="Consulta de Pessoas">

            <h:outputText value="Listagem de pessoas cadastradas:"/>
```

```
        <p:separator/>

        <h:form id="formconsulta">

        <p:dataTable paginator="true" rows="10"
               emptyMessage="Não há registros.">

               <p:column headerText="Código">
               </p:column>

               <p:column headerText="Nome da Pessoa">
               </p:column>

               <p:column headerText="Endereço de Email">
               </p:column>

               <p:column headerText="Operações">
               <p:commandButton value="Excluir" ajax="true"
                     update=":formconsulta"/>
               </p:column>

        </p:dataTable>

        </h:form>

        </p:panel>

        </h:body>

</html>
```

Criando o ManagedBean:

ManagedBeanPessoa.java

```
package br.com.brq.managedbeans;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.ViewScoped;
import javax.faces.context.FacesContext;

import br.com.brq.entities.Pessoa;
import br.com.brq.persistence.DAOPessoa;

@ManagedBean(name = "mbpessoa")
@ViewScoped
public class ManagedBeanPessoa {

        // atributo para resgatar os campos do formulário..
        private Pessoa pessoa;
```

```
//atributo para receber a configuração mapeada no spring..
@ManagedProperty(value = "#{daopessoa}") //nome mapeado no spring..
private DAOPessoa daoPessoa; //Injeção de dependencia..

// construtor..
public ManagedBeanPessoa() {
	pessoa = new Pessoa(); // instanciando..
}

//método para cadastrar pessoa..
public void cadastrar(){
	String mensagem = null;

	try{
		daoPessoa.insert(pessoa); //gravando..

		mensagem = "Pessoa " + pessoa.getNome() + ", cadastrado com sucesso.";
		pessoa = new Pessoa(); //instanciando..
	}
	catch(Exception e){
		mensagem = e.getMessage();
	}

	FacesContext.getCurrentInstance().addMessage("formcadastro",
							new FacesMessage(mensagem));
}

public Pessoa getPessoa() {
	return pessoa;
}

public void setPessoa(Pessoa pessoa) {
	this.pessoa = pessoa;
}

public DAOPessoa getDaoPessoa() {
	return daoPessoa;
}

public void setDaoPessoa(DAOPessoa daoPessoa) {
	this.daoPessoa = daoPessoa;
}
}
```