WebServices

Web service é uma solução utilizada na **integração de sistemas** e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *Web services* são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, um formato intermediário como *XML, Json, CSV, etc*.

Para as empresas, os *Web services* podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana.

Essencialmente, o Web Service faz com que os recursos da aplicação do software estejam disponíveis sobre a rede de uma forma normalizada. Outras tecnologias fazem a mesma coisa, como por exemplo, os browsers da Internet acessam às páginas Web disponíveis usando por norma as tecnologias da Internet, HTTP e HTML. No entanto, estas tecnologias não são bem sucedidas na comunicação e integração de aplicações. Existe uma grande motivação sobre a tecnologia Web Service pois possibilita que diferentes aplicações comuniquem - se entre si e utilizem recursos diferentes.

Utilizando a tecnologia Web Service, uma aplicação pode invocar outra para efetuar tarefas simples ou complexas mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Por outras palavras, os Web Services fazem com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos pelo Web Service.

Os Web Services são identificados por um URI (Uniform Resource Identifier), descritos e definidos usando XML (Extensible Markup Language). Um dos motivos que tornam os Web Services atractivos é o facto deste modelo ser baseado em tecnologias standards, em particular XML e HTTP (Hypertext Transfer Protocol). Os Web Services são utilizados para disponibilizar serviços interactivos na Web, podendo ser acessados por outras aplicações usando, por exemplo, o protocolo SOAP (Simple Object Access Protocol).

O **objetivo dos Web Services** é a comunicação de aplicações através da Internet. Esta comunicação é realizada com intuito de facilitar a EAI (Enterprise Application Integration) que significa a integração das aplicações de uma empresa, ou seja, interoperabilidade entre a informação que circula numa organização nas diferentes aplicações como, por exemplo, o comércio electrónico com os seus clientes e seus fornecedores. Esta interação constitui o sistema de informação de uma empresa. E para além da interoperabilidade entre as aplicações, a EAI permite definir um workflow entre as aplicações e pode constituir uma alternativa aos ERP (Enterprise Resource Planning). Com umworkflow é possível otimizar e controlar processos e tarefas de uma determinada organização.

XML

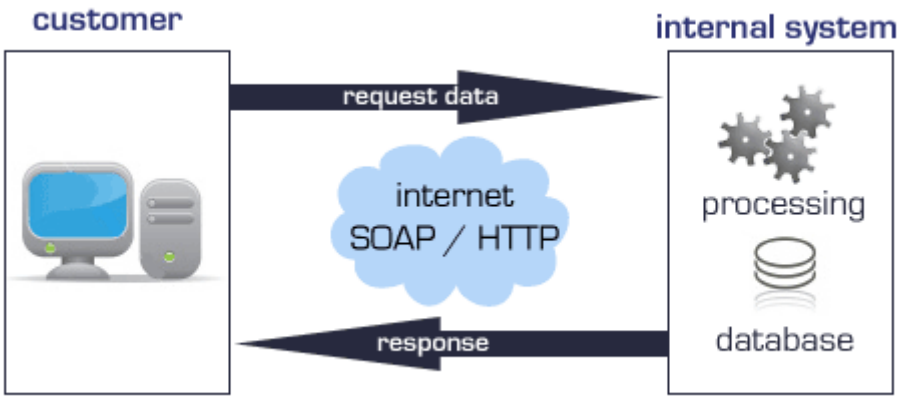
Extensible Markup Language (XML) é a base em que os Web Services são construídos. O XML fornece a descrição, o armazenamento, o formato da transmissão para trocar os dados através dos Web Services e também para criar tecnologias Web Services para a troca dos dados. A sintaxe de XML usada nas tecnologias dos Web Services especifica como os dados são representados genericamente, define como e com que qualidades de serviço os dados são transmitidos, pormenoriza como os serviços são publicados e descobertos.
Os Web Services decodificam as várias partes de XML para interagir com as várias aplicações.

WSDL

É a sigla de *Web Services Description Language*, padrão baseado em XML para descrever o serviço como no COM, onde ele traz os métodos do *Web Service*. Funciona como uma espécie de "*TypeLibrary*" do *Web Service*, além de ser usado para a validação das chamadas dos métodos.

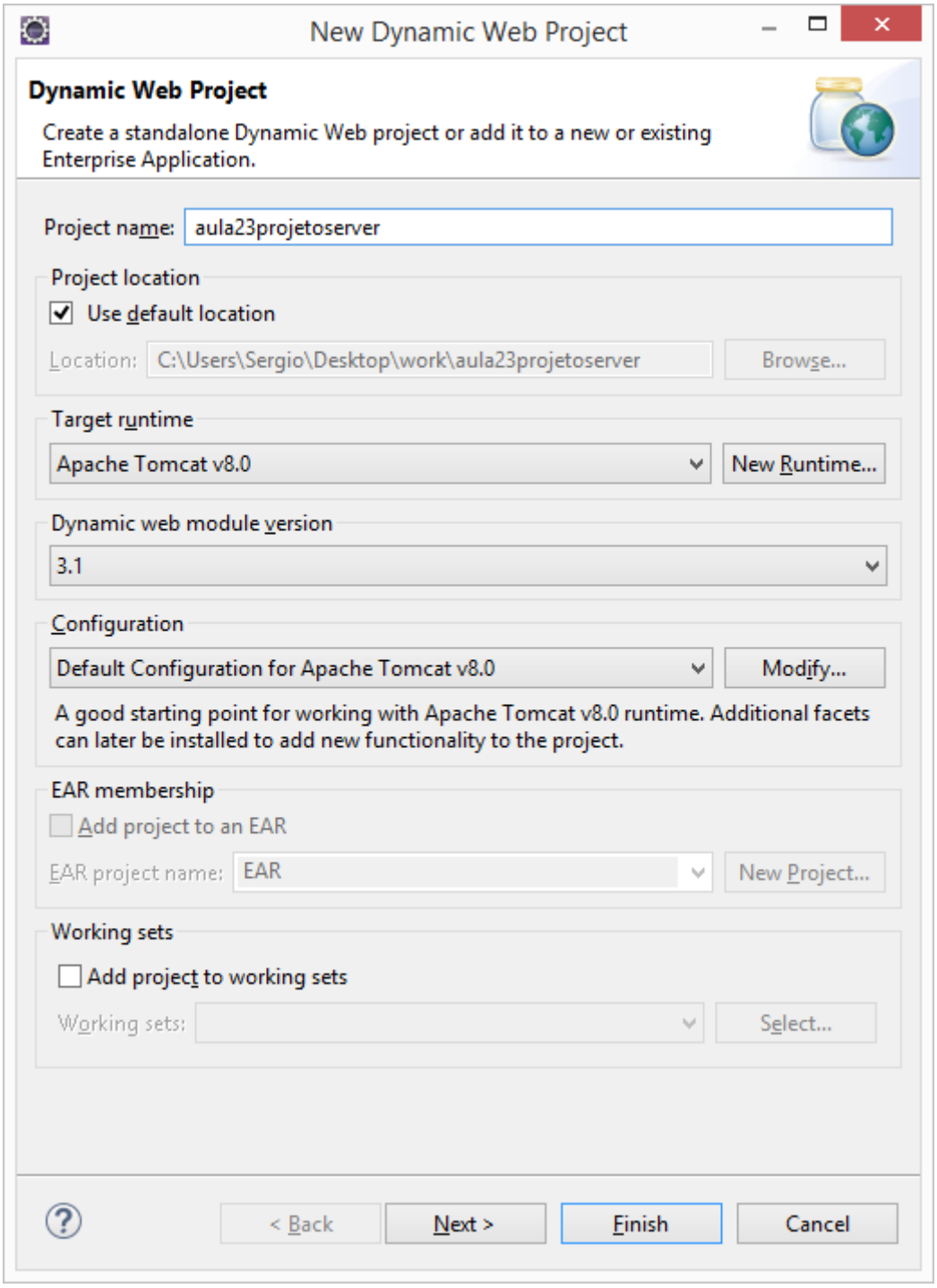
O WSDL é uma especificação desenvolvida pelo W3C.

O WSDL é extensível para permitir a descrição dos serviços e suas mensagens, independentemente dos formatos de mensagem e dos protocolos de rede que sejam usados. No entanto, é comum usar-se o MIME (Multipurpose Internet Mail Extensions) e o HTtp://SOAP. O WSDL descreve os serviços disponibilizados à rede através de uma semântica XML, este providencia a documentação necessária para se chamar um sistema distribuído e o procedimento necessário para que esta comunicação se estabeleça. Enquanto que o SOAP especifica a comunicação entre um cliente e um servidor, o WSDL descreve os serviços oferecidos.



Criando o projedor de serviços:

File > New > Dynamic WebProject



Criando a entidade modelo:

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

public class Produto {

	private Integer idProduto;
	private String nome;
	private Double preco;

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

	public Produto(Integer idProduto, String nome, Double preco) {
		super();
		this.idProduto = idProduto;
		this.nome = nome;
		this.preco = preco;
	}

	public Integer getIdProduto() {
		return idProduto;
	}

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

	public String getNome() {
		return nome;
	}

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

	public Double getPreco() {
		return preco;
	}

	public void setPreco(Double preco) {
		this.preco = preco;
	}

	@Override
	public String toString() {
		return "Produto [idProduto=" + idProduto + ", nome="
			+ nome + ", preco=" + preco + "]";
	}

}
```

Camada de persistência de dados com JDBC:

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

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class DAO {

	private static final String DRIVER = "com.mysql.jdbc.Driver";
	private static final String URL = "jdbc:mysql://localhost:3306/aula23";
	private static final String USER = "root";
	private static final String PASSWORD = "brqbrq";

	protected Connection con;
	protected PreparedStatement stmt;
	protected CallableStatement call;
	protected ResultSet rs;

	protected void openConnection() throws Exception{
		Class.forName(DRIVER);
		con = DriverManager.getConnection(URL, USER, PASSWORD);
	}

	protected void closeConnection() throws Exception{
		if(con != null){
			con.close();
		}
	}
}
```

Criando a classe de persistência para Produto:

/persistence/DAOProduto.java

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

import br.com.brq.entities.Produto;

public class DAOProduto extends DAO{

	public void insert(Produto p) throws Exception{

		openConnection(); //abrir conexão..

		stmt = con.prepareStatement("insert into produto(nome, preco)
					values(?, ?)");
```

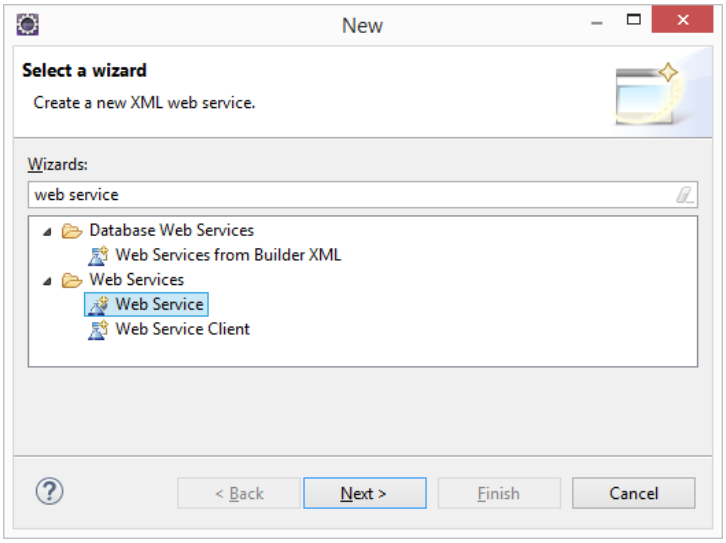
```
                stmt.setString(1, p.getNome());
                stmt.setDouble(2, p.getPreco());
                stmt.execute();
                stmt.close();

                closeConnection(); //fechar conexão..
        }
}
```

Criando a classe de serviço:

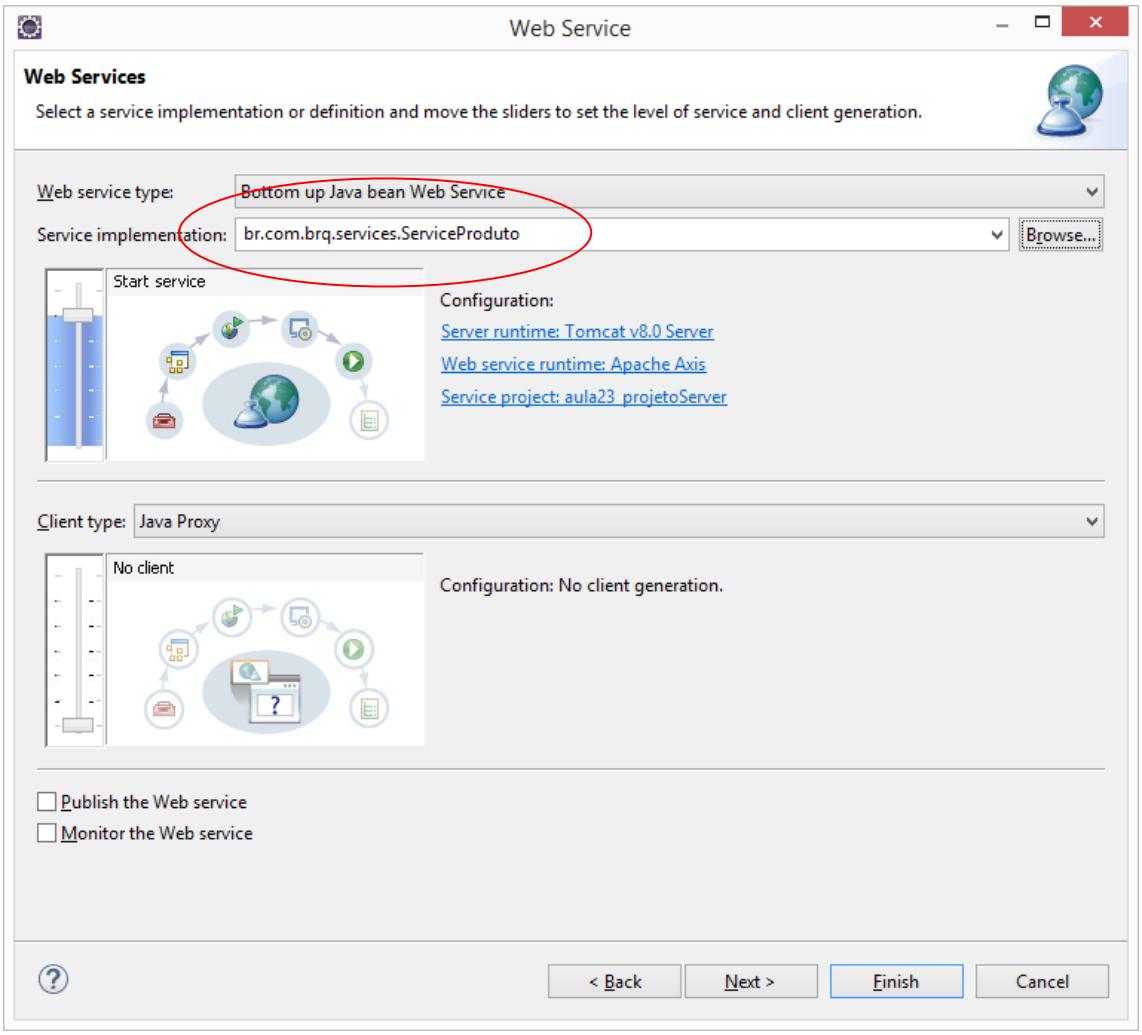
Serviço para cadastro de produto...

```
package br.com.brq.services;

import javax.jws.WebMethod;
import javax.jws.WebService;

import br.com.brq.entities.Produto;
import br.com.brq.persistence.DAOProduto;

@WebService //serviço para integração de sistemas web..
public class ServiceProduto {

        //operação para cadastro de produto..
        //este método será executado por outros sistemas (clientes do serviço..)
        @WebMethod
        public String cadastrarProduto(String nome, Double preco){
                try{

                        Produto p = new Produto(); //entidade..
                        p.setNome(nome); //recebendo nome..
                        p.setPreco(preco); //recebendo preco..

                        DAOProduto d = new DAOProduto(); //persistencia..
                        d.insert(p); //gravando..

                        return "Produto " + p.getNome() + ", cadastrado com sucesso.";
                }
                catch(Exception e){
                        //retornar mensagem de erro..
                        return "Ocorreu um erro: " + e.getMessage();
                }
        }

}
```

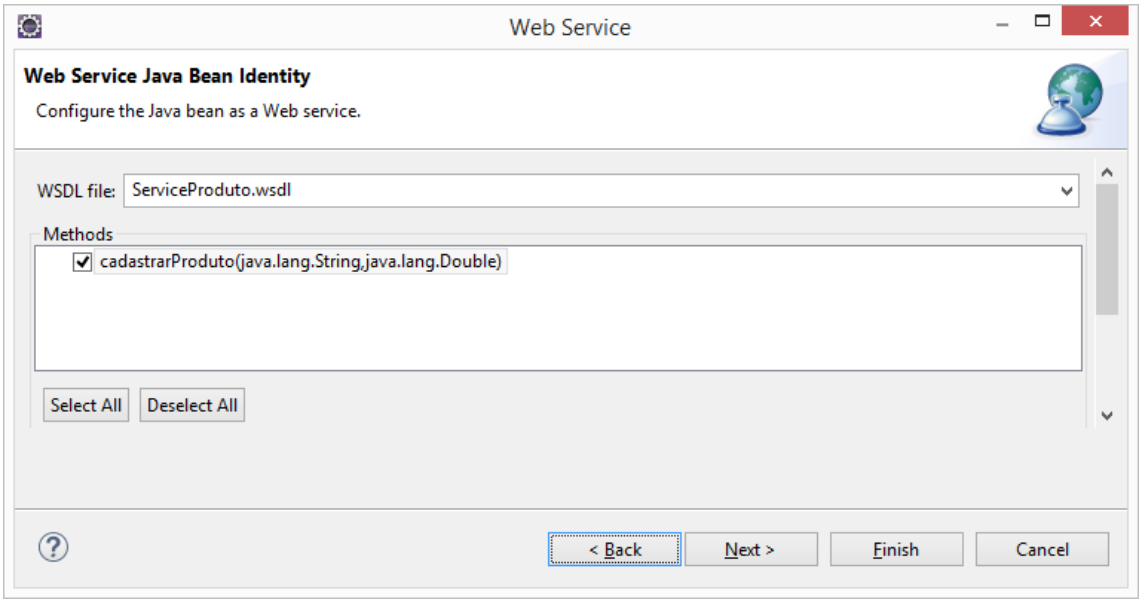
Gerando o WebService baseado na classe de serviço:



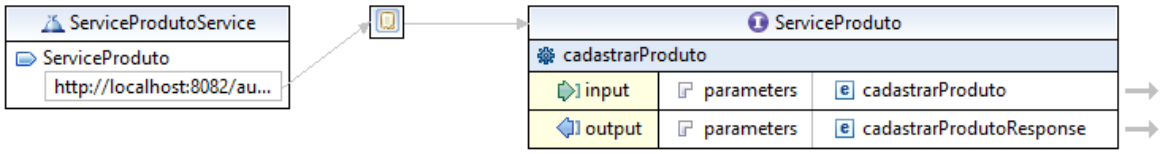
Selecione a classe de serviço:



Método / Operação do serviço:



Arquivo WSDL gerado para conectar o serviço ao seu cliente:
ServiceProduto.wsdl

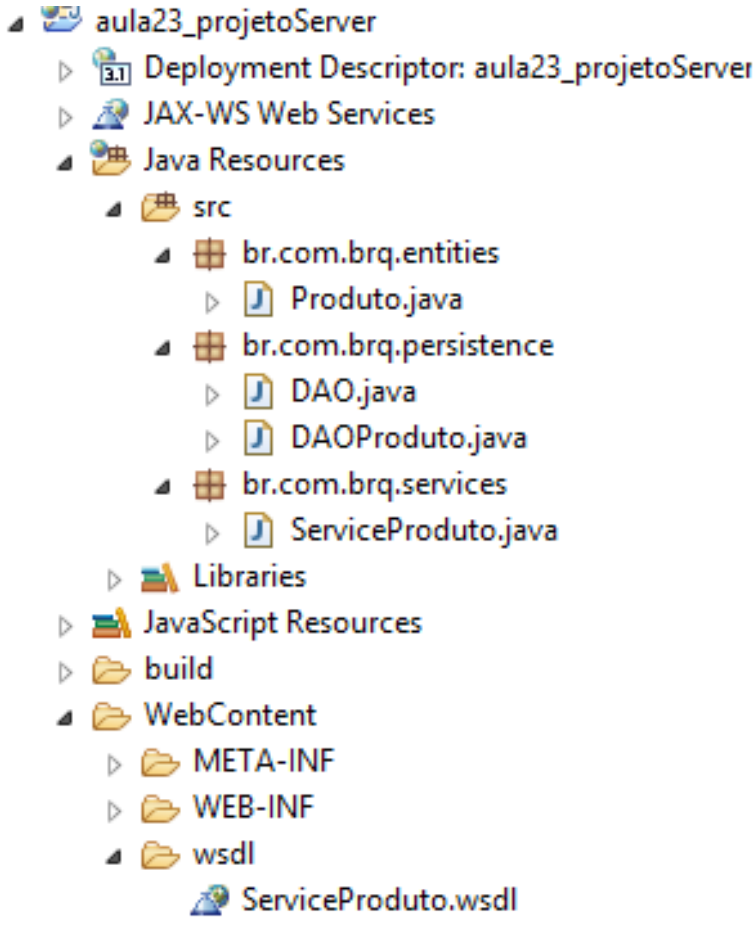



```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://services.brq.com.br"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://services.brq.com.br"
xmlns:intf="http://services.brq.com.br"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema elementFormDefault="qualified"
      targetNamespace="http://services.brq.com.br"
      xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="cadastrarProduto">
    <complexType>
     <sequence>
      <element name="nome" type="xsd:string"/>
      <element name="preco" type="xsd:double"/>
     </sequence>
    </complexType>
   </element>
```

```
    <element name="cadastrarProdutoResponse">
     <complexType>
      <sequence>
       <element name="cadastrarProdutoReturn" type="xsd:string"/>
      </sequence>
     </complexType>
    </element>
   </schema>
  </wsdl:types>

    <wsdl:message name="cadastrarProdutoResponse">

       <wsdl:part element="impl:cadastrarProdutoResponse" name="parameters">

       </wsdl:part>

    </wsdl:message>

    <wsdl:message name="cadastrarProdutoRequest">

       <wsdl:part element="impl:cadastrarProduto" name="parameters">

       </wsdl:part>

    </wsdl:message>

    <wsdl:portType name="ServiceProduto">

       <wsdl:operation name="cadastrarProduto">

          <wsdl:input message="impl:cadastrarProdutoRequest"
              name="cadastrarProdutoRequest">

          </wsdl:input>

          <wsdl:output message="impl:cadastrarProdutoResponse"
              name="cadastrarProdutoResponse">

          </wsdl:output>

       </wsdl:operation>

    </wsdl:portType>

    <wsdl:binding name="ServiceProdutoSoapBinding" type="impl:ServiceProduto">

       <wsdlsoap:binding style="document"
             transport="http://schemas.xmlsoap.org/soap/http"/>

       <wsdl:operation name="cadastrarProduto">

          <wsdlsoap:operation soapAction=""/>

          <wsdl:input name="cadastrarProdutoRequest">
```

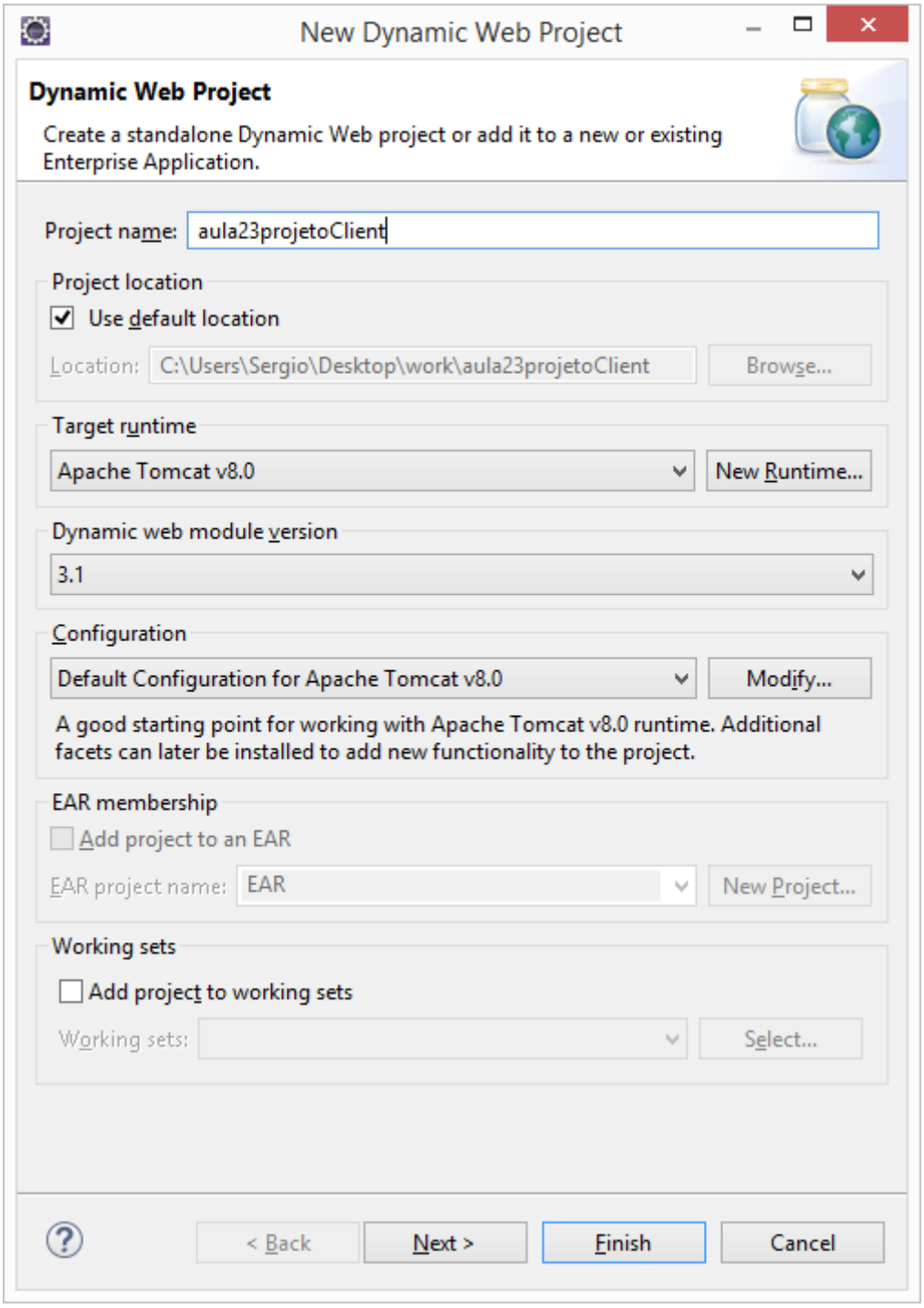
```
            <wsdlsoap:body use="literal"/>
         </wsdl:input>

         <wsdl:output name="cadastrarProdutoResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>

   </wsdl:binding>

   <wsdl:service name="ServiceProdutoService">
      <wsdl:port binding="impl:ServiceProdutoSoapBinding"
             name="ServiceProduto">

         <wsdlsoap:address
                   location="http://localhost:8082/aula23_projetoServer/
                   services/ServiceProduto"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Estrutura do projeto:



Criando o Cliente do Serviço:

- File > New > DynamicWebProject

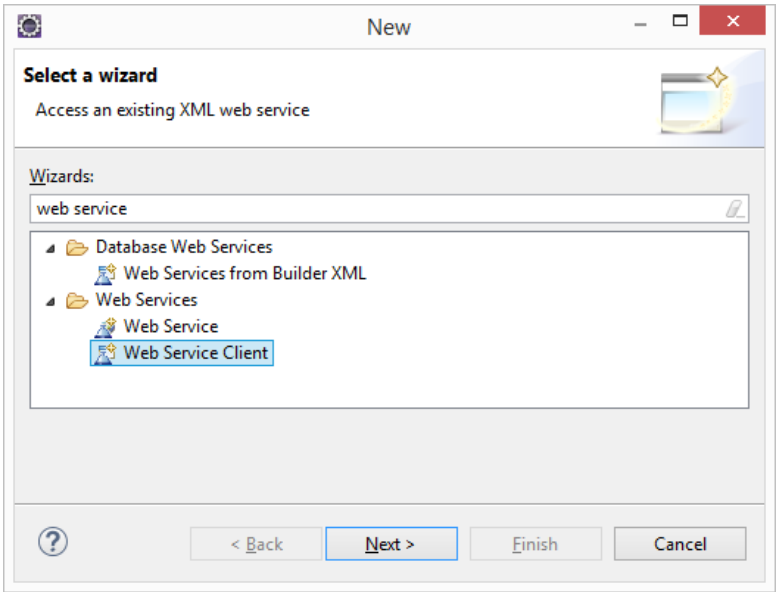


Criando a página para cadastro de produtos:

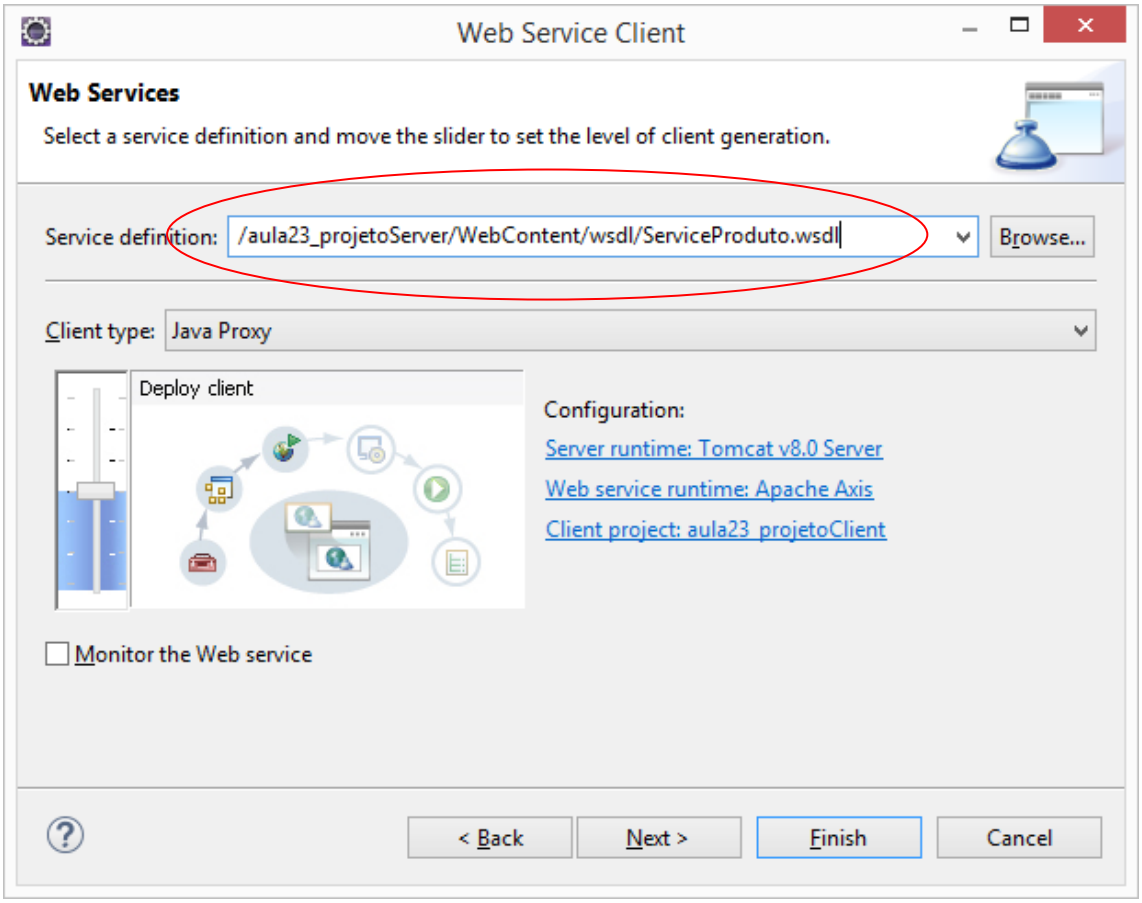
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>

<head>

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

</head>

<body>

	<h3>Cadastro de Produtos</h3>
	<hr/>

	<p>
		Informe os dados abaixo:
	</p>

	<form name="formcadastro" method="post"
		action="ControleProduto?action=cadastrar">

		Nome do Produto: <br/>
		<input type="text" name="nome" placeholder="Digite aqui"
			required="required"
			title="Por favor, informe o nome do produto"/>
		<br/><br/>

		Preço: <br/>
		<input type="text" name="preco" placeholder="Digite aqui"
			required="required"
			title="Por favor, informe o preço do produto."/>
		<br/><br/>

		<input type="submit" value="Cadastrar Produto"/>

		<div>
			<h4>${mensagem}</h4>
		</div>

	</form>

</body>
</html>
```

Consumindo o serviço web:

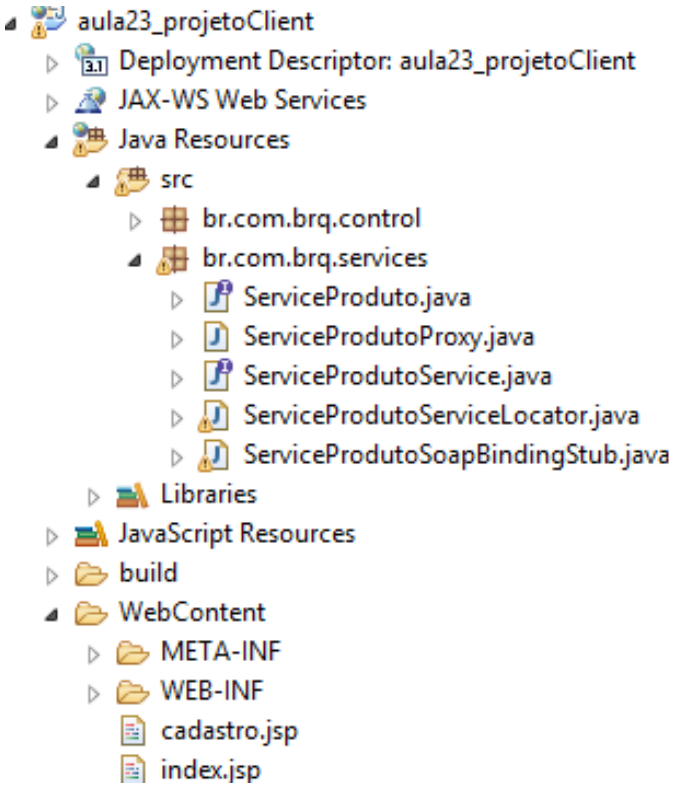
Acessando o serviço através do seu endereço WSDL



Cole o endereço do WSDL do serviço:
/aula23_projetoServer/WebContent/wsdl/ServiceProduto.wsdl



Classes e interfaces de Proxy geradas para executar chamados ao serviço:



Classe Servlet para executar a ação de cadastro:

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

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import br.com.brq.services.ServiceProdutoProxy;

@WebServlet("/ControleProduto")
public class ControleProduto extends HttpServlet {
	private static final long serialVersionUID = 1L;

	public ControleProduto() {
		super();
	}
```

```
	protected void execute(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {

		String action = request.getParameter("action");

		if("cadastrar".equalsIgnoreCase(action)){
			try{

				String nome = request.getParameter("nome");
				Double preco = Double.parseDouble
						(request.getParameter("preco"));

				ServiceProdutoProxy service = new ServiceProdutoProxy();
				String mensagem = service.cadastrarProduto(nome, preco);

				request.setAttribute("mensagem", mensagem);
			}
			catch(Exception e){
				e.printStackTrace();
				request.setAttribute("mensagem", e.getMessage());
			}
			finally{
				request.getRequestDispatcher("cadastro.jsp")
					.forward(request, response);
			}
		}
	}

	protected void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		execute(request, response);
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		execute(request, response);
	}

}
```