TUTORIAL DE INTRODUÇÃO AO CEWOLF

CRIANDO GRÁFICOS COM JSP, CEWOLF/JFREECHART E MYSQL

1. Introdução

O Cewolf (http://cewolf.sourceforge.net/) é uma biblioteca que facilita a criação de gráficos dinâmicos em uma aplicação web, distribuído sob a licença LGPL. O Cewolf é todo baseado no JFreeChart (http://www.jfree.org/jfreechart/index.html), pois ele aproveita os mesmos mecanismos de desenho dos gráficos do JFreeChart.

É possível gerar vários tipos de gráficos desde os mais simples, como de linha, barras, setoriais (pizza) até gráficos combinados, em 3D, etc.

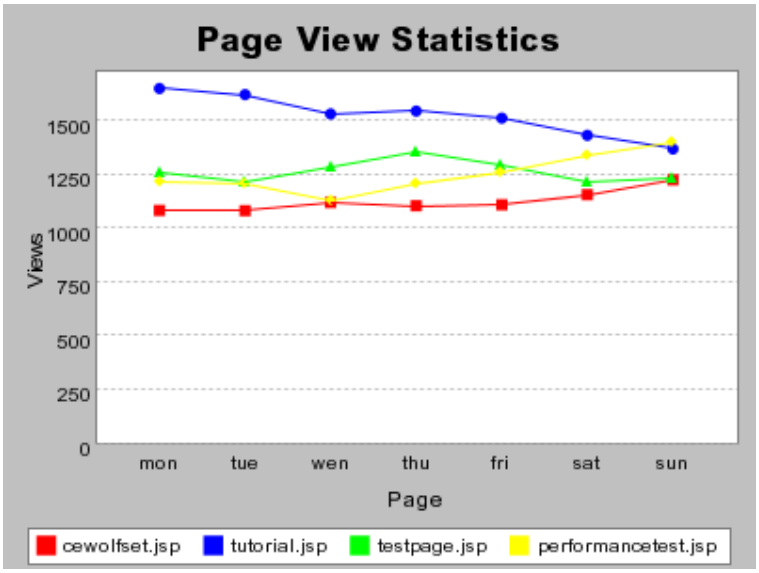


Figura 1. Gráfico de linhas.

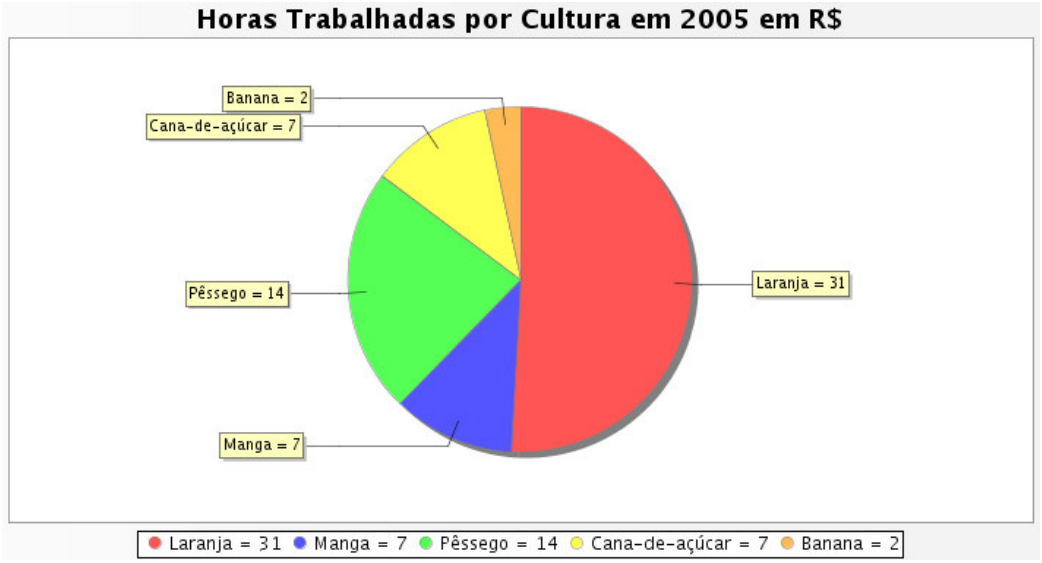


Figura 2. Gráfico setorial.

Mais exemplos podem ser vistos no link: http://cewolf.sourceforge.net/new/demo.html

2. Ambiente de Desenvolvimento

Será utilizado o Eclipse 3.2M3 (www.eclipse.org) , com os plug-ins do projeto Amateras. Os links para downloads, a instalação e configuração é apresentada nesse link: http://www.guj.com.br/posts/list/31160.java

A versão do Cewolf utilizada é a 0.12.0 e a do Tomcat é a 5.5.9.

O MySQL(http://dev.mysql.com/) utilizado é o 5.0.16. O driver do MySQL está junto com o material disponibilizado para download (http://www.furutani.eti.br/tutoriais/Graficos.zip).

3. Codificando

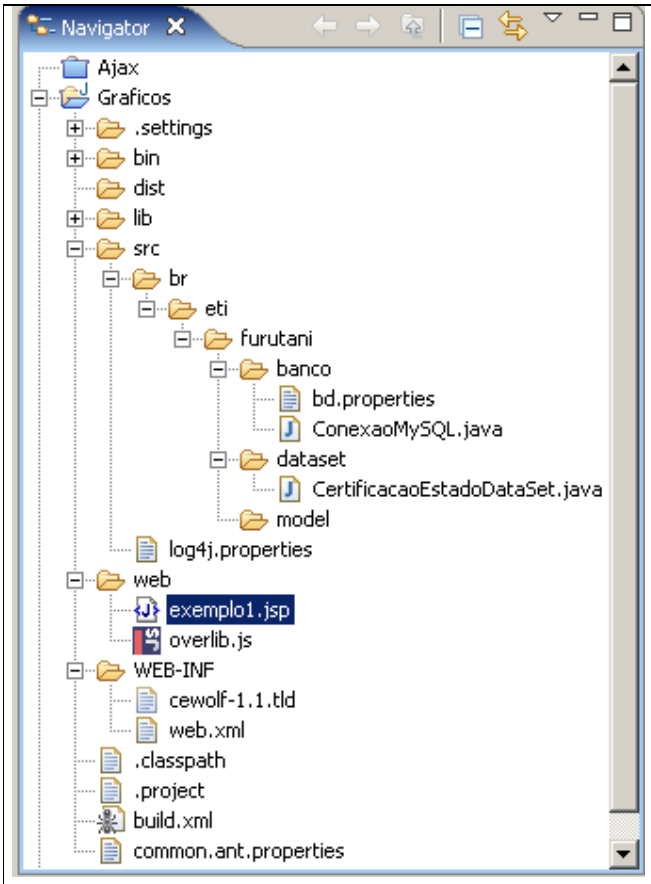
Inicialmente é preciso dizer um pouco da organização dos arquivos fontes feita dentro de eclipse. O arquivo disponibilizado para download(http://www.furutani.eti.br/tutoriais/Graficos.zip) contém um projeto para ser importado no eclipse, selecione menu file, Import/Export e utilize o wizard que é bem intuitivo.

Após a importação abra o arquivo common.ant.properties, a primeira linha está assim:

```
Deploy.Dir=D:/tomcat-5.5.9/webapps
```

Altere para o diretório webapps do seu tomcat.

Os diretórios estão separados assim:



Bin – aqui ficam as classes compiladas.

Dist – aqui fica pacote war.

Lib – aqui ficam todas as bibliotecas utilizadas.

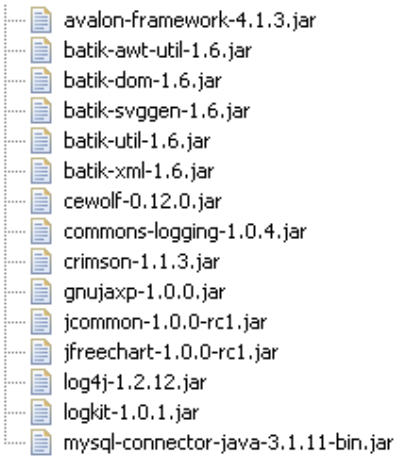
Src – aqui ficam os fontes .java

Web – ficam páginas jsp, css, javascript, etc.

Web-inf – onde fica o web.xml e um monte de outras coisas.

O arquivo build.xml é um script do ANT (http://ant.apache.org/) para automatizar o deployment da aplicação. Toda vez que vc fizer uma alteração no projeto, clique com o botão direito sobre o build.xml e selecione Run As, Ant build para empacotar toda a aplicação, criar o war e jogar no webapps do tomcat.

As libs que devem ser acrescentadas no classpath para que a aplicação funcione são:



Para a aplicação será usada uma base de dados simples, como pode ser visto na figura abaixo.


```
CREATE TABLE Pessoa (
  idPessoa INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nome VARCHAR(100) NOT NULL,
  Cidade VARCHAR(50) NOT NULL,
  Estado VARCHAR(2) NOT NULL,
  PRIMARY KEY(idPessoa)
);

CREATE TABLE Certificacao (
  idCertificacao INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nome VARCHAR(10) NOT NULL,
  Descricao VARCHAR(100) NOT NULL,
  DataLancamento DATETIME NOT NULL,
  PRIMARY KEY(idCertificacao)
);

CREATE TABLE Pessoa_has_Certificacao (
  FK_idPessoa INTEGER UNSIGNED NOT NULL,
  FK_idCertificacao INTEGER UNSIGNED NOT NULL,
  DataExame DATETIME NOT NULL,
  Nota INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(FK_idPessoa, FK_idCertificacao),
  FOREIGN KEY(FK_idPessoa)
    REFERENCES Pessoa(idPessoa)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(FK_idCertificacao)
    REFERENCES Certificacao(idCertificacao)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);
```

Depois dessa apresentação, vamos prosseguir com o tutorial.

Inicialmente temos que configurar e mapear o servlet responsável por renderizar os gráficos. No arquivo web.xml foram acrescentadas as seguinte linhas:

```
<servlet>
    <servlet-name>CewolfServlet</servlet-name>
    <servlet-class>de.laures.cewolf.CewolfRenderer</servlet-class>
    <init-param>
        <param-name>storage</param-name>
        <param-value>de.laures.cewolf.storage.TransientSessionStorage</param-value>
    </init-param>
    <init-param>
        <param-name>overliburl</param-name>
```

```
        <param-value>overlib.js</param-value>
    </init-param>
<!—Torna o debug ativo -->
    <init-param>
        <param-name>debug</param-name>
        <param-value>true</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>CewolfServlet</servlet-name>
    <url-pattern>/cewolf/*</url-pattern>
  </servlet-mapping>
```

Agora iremos criar um DatasetProducer para um gráfico do tipo setorial (pizza) que terá o nome `CertificacaoEstadoDataSet`. Esse gráfico irá apresentar a quantidade de certificações por Estado.

```
package br.eti.furutani.dataset;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import java.util.Map;

import org.apache.log4j.Logger;
import org.jfree.data.general.DefaultPieDataset;

import br.eti.furutani.banco.ConexaoMySQL;
import de.laures.cewolf.DatasetProduceException;
import de.laures.cewolf.DatasetProducer;

public class CertificacaoEstadoDataSet implements DatasetProducer {

        private static final long serialVersionUID = 1L;
        // vamos utilizar o log4j
        Logger log = Logger.getLogger(CertificacaoEstadoDataSet.class);

        // Retorna uma Dataset com os dados que irão alimentar o gráfico
        public Object produceDataset(Map arg0) throws DatasetProduceException {

        // Criando um Dataset para o gráfico
        DefaultPieDataset ds = new DefaultPieDataset();

        PreparedStatement ps = null;

        String query = "SELECT p.estado, COUNT(p.estado) as qtde FROM `pessoa_has_certificacao` phc "
```

```
        + "INNER JOIN `certificacao` c ON c.idCertificacao = phc.FK_idCertificacao "
        + "INNER JOIN `pessoa` p ON p.idPessoa = phc.FK_idPessoa "
        + "GROUP BY p.estado";
                try {
                        // Recupera uma conexão com o banco
                        Connection conexao = ConexaoMySQL.getConn();
                        ps = conexao.prepareStatement(query);

                        ResultSet rs = ps.executeQuery();

                        while (rs.next()) {
                        // Inserindo no DataSet o estado e a quantidade de pessoas certificadas
                                ds.setValue(rs.getString(1), new Double(rs.getInt(2)));
                        }
                } catch (SQLException e) {
                        log.info(e);
                }
                // retorna o DataSet
                return ds;
        }

        // O dados deste dataset é invalidado imediatamente.
        // O tempo de retorno influencia no cachê de dados do cewolf
        // Retornando sempre true, os dados do dataset são sempre atualizados a cada request.
        public boolean hasExpired(Map arg0, Date arg1) {
                return true;
                // Se quiséssemos que os dataset expirasse em 5 segundos colocaríamos
                // return (System.currentTimeMillis() - since.getTime())  > 5000;
        }

        // Retorna um Id único para este dataset
        public String getProducerId() {
                return "CertificacaoPorEstado";
        }
}
```

Agora vamos desenvolver o JSP. As tags do Cewolf são bem fáceis de usar.

Em http://cewolf.sourceforge.net/new/taglib.html tem uma listagem completa das tags e os atributos suportados pelo cewolf.

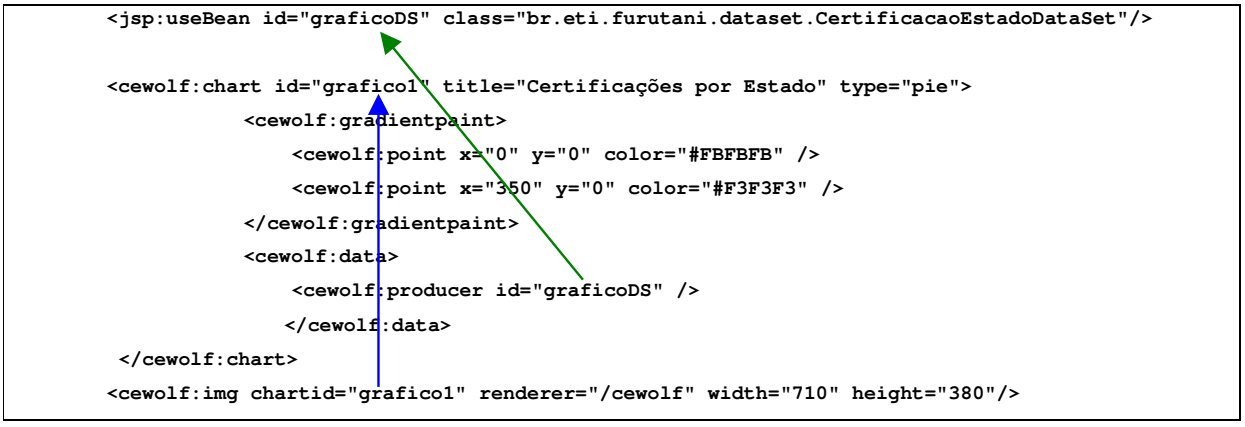
```
        <%@ taglib uri="/WEB-INF/cewolf-1.1.tld" prefix="cewolf"%>
        <html><head>
        <meta http-equiv="Content-Type" content="text/html; charset=Cp1252"/>
        <title>Exemplo 1</title>
        </head><body>
        <!– Criando um bean -->
        <jsp:useBean id="graficoDS" class="br.eti.furutani.dataset.CertificacaoEstadoDataSet"/>

        <center><h3>GRÁFICOS DE PIZZA</h3></center>
```

```
                    <!-- GRÁFICO SETORIAL/PIZZA -->
                    <cewolf:chart id="grafico1" title="Certificações por Estado" type="pie">
                        <cewolf:gradientpaint>
                            <cewolf:point x="0" y="0" color="#FBFBFB" />
                            <cewolf:point x="350" y="0" color="#F3F3F3" />
                        </cewolf:gradientpaint>
                        <cewolf:data>
                            <cewolf:producer id="graficoDS" />
                           </cewolf:data>
                    </cewolf:chart>
                    <cewolf:img chartid="grafico1" renderer="/cewolf" width="710" height="380"/>

                    <br/>

                    <!-- GRÁFICO SETORIAL/PIZZA EM 3D -->
                    <cewolf:chart id="grafico2" title="Certificações por Estado" type="pie3d">
                        <cewolf:gradientpaint>
                            <cewolf:point x="0" y="0" color="#FBFBFB" />
                            <cewolf:point x="350" y="0" color="#F3F3F3" />
                        </cewolf:gradientpaint>
                        <cewolf:data>
                            <cewolf:producer id="graficoDS" />
                        </cewolf:data>
                    </cewolf:chart>
                    <cewolf:img chartid="grafico2" renderer="/cewolf" width="710" height="380"/>
        </body>
        </html>
```

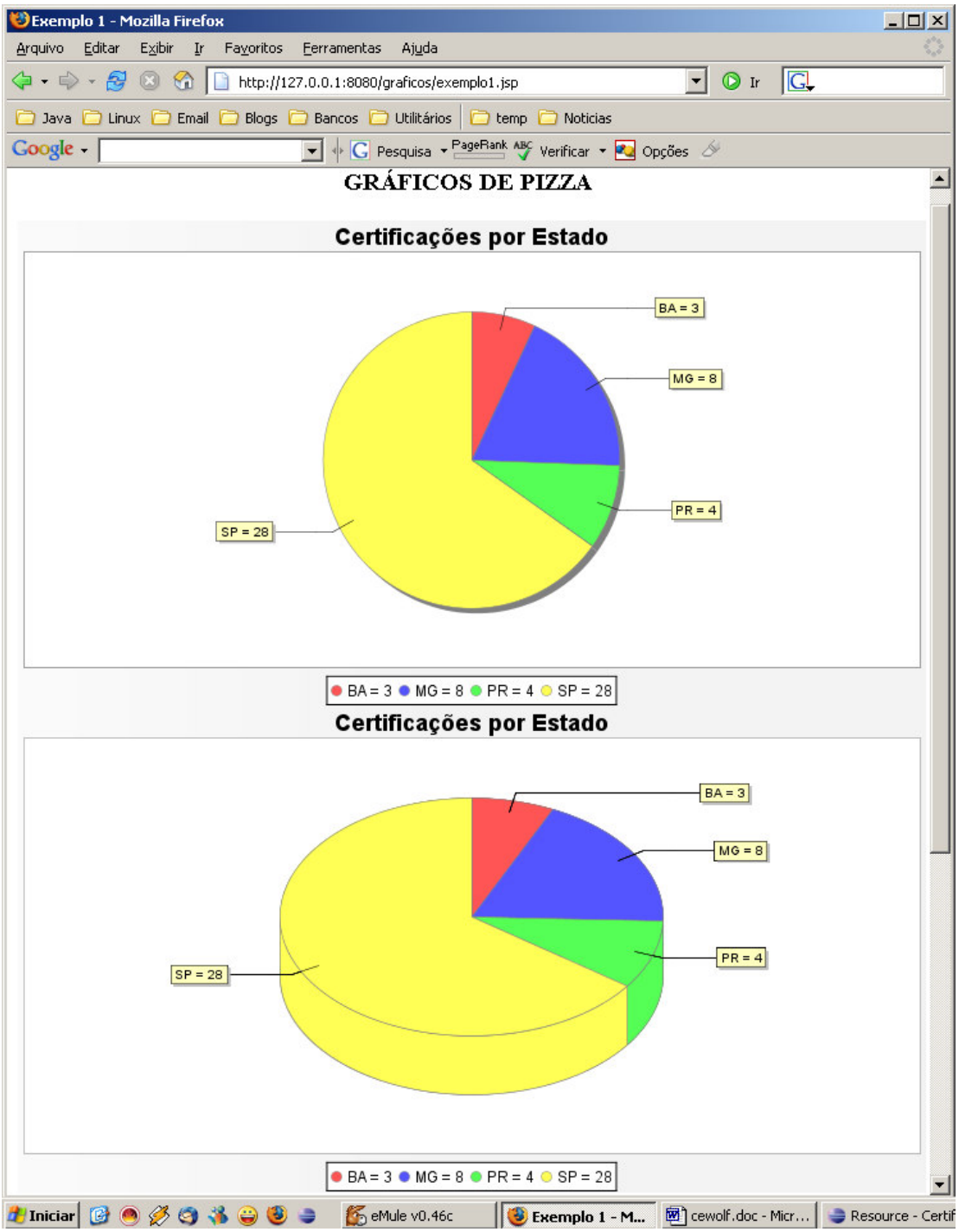
Duas relações são importantes dentro desse monte de tags, a primeira é a do id do jsp:useBean e o id do cewolf:producer.

A segunda é o id do cewolf:chart e o id do cewolf:img.

Observe as flechas abaixo.



O resultado é o seguinte:



Agora iremos criar um novo gráfico, para exibir a quantidade de tipos de certificações por estado. O DatasetProducer vai se chamar PessoasCertificadasDataSet.

```
package br.eti.furutani.dataset;
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import java.util.Map;

import org.apache.log4j.Logger;
import org.jfree.data.category.DefaultCategoryDataset;

import br.eti.furutani.banco.ConexaoMySQL;
import de.laures.cewolf.DatasetProduceException;
import de.laures.cewolf.DatasetProducer;

public class PessoasCertificadasDataSet implements DatasetProducer {

        private static final long serialVersionUID = 1L;
        Logger log = Logger.getLogger(PessoasCertificadasDataSet.class);

        // Retorna uma Dataset com os dados que irão alimentar o gráfico
        public Object produceDataset(Map arg0) throws DatasetProduceException {

        // Criando um Dataset para o gráfico de linhas
        DefaultCategoryDataset ds = new DefaultCategoryDataset();

        PreparedStatement ps = null;

        String query = "SELECT COUNT(c.idCertificacao), p.estado,c.Nome FROM
`pessoa_has_certificacao` phc "
        + "INNER JOIN `certificacao` c ON c.idCertificacao = phc.FK_idCertificacao "
        + "INNER JOIN `pessoa` p ON p.idPessoa = phc.FK_idPessoa "
        + "GROUP BY p.estado,c.idCertificacao";
        try {

        Connection conexao = ConexaoMySQL.getConn();
        ps = conexao.prepareStatement(query);

        ResultSet rs = ps.executeQuery();

        while (rs.next()) {
        // Inserindo no DataSet a quantidade de pessoas certificadas, o nome da certf. e o
estado.
                ds.addValue(rs.getInt(1), rs.getString(2), rs.getString(3));
        }
        } catch (SQLException e) {
                log.info(e);
        }

        return ds;
        }
```

```
		// O dados deste dataset é invalidado imediatamente. O tempo de retorno influencia no
cache do cewolf
		public boolean hasExpired(Map arg0, Date arg1) {
			return true;
			// Se quiséssemos que os dataset expirasse em 5 segundos colocaríamos
			// return (System.currentTimeMillis() - since.getTime())  > 5000;
		}

		// Retorna um Id único para o dataset
		public String getProducerId() {
			return "PessoasCertificadasPorEstado";
		}

	}
```

A maior diferença entre esse DatasetProducer para o outro é que trocamos o DefaultPieDataset para o DefaultCategoryDataset.

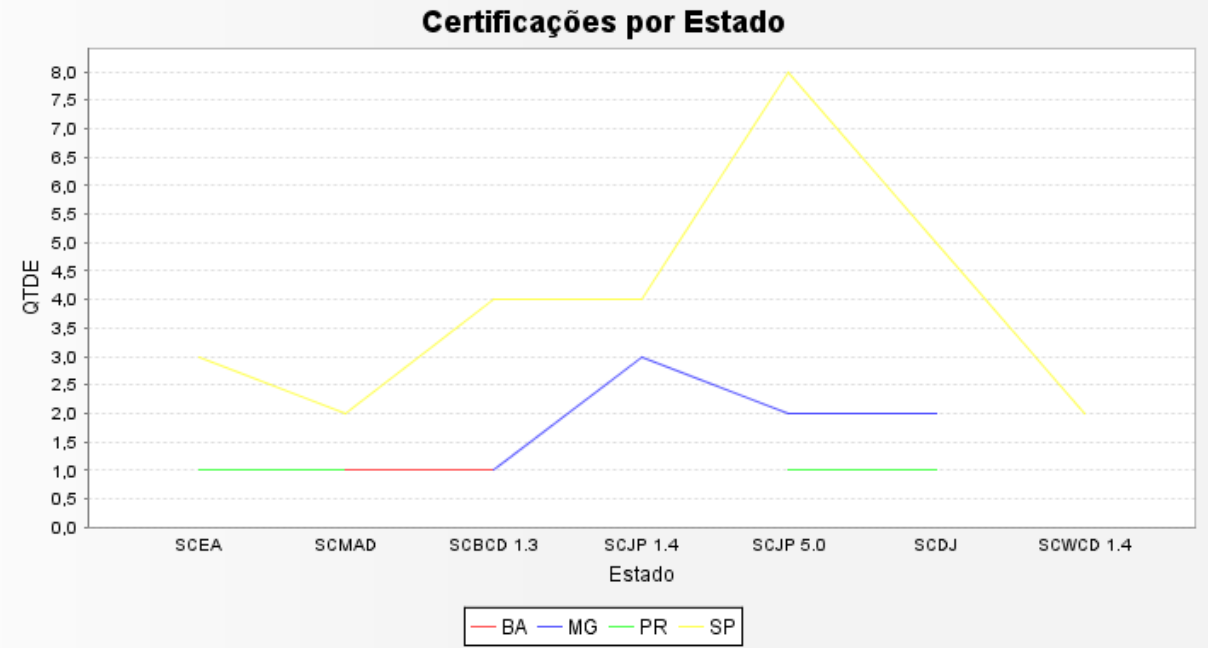
Vejam agora o JSP. A única diferença para o JSP anterior é o atributo type da tag cewolf:chart em azul.

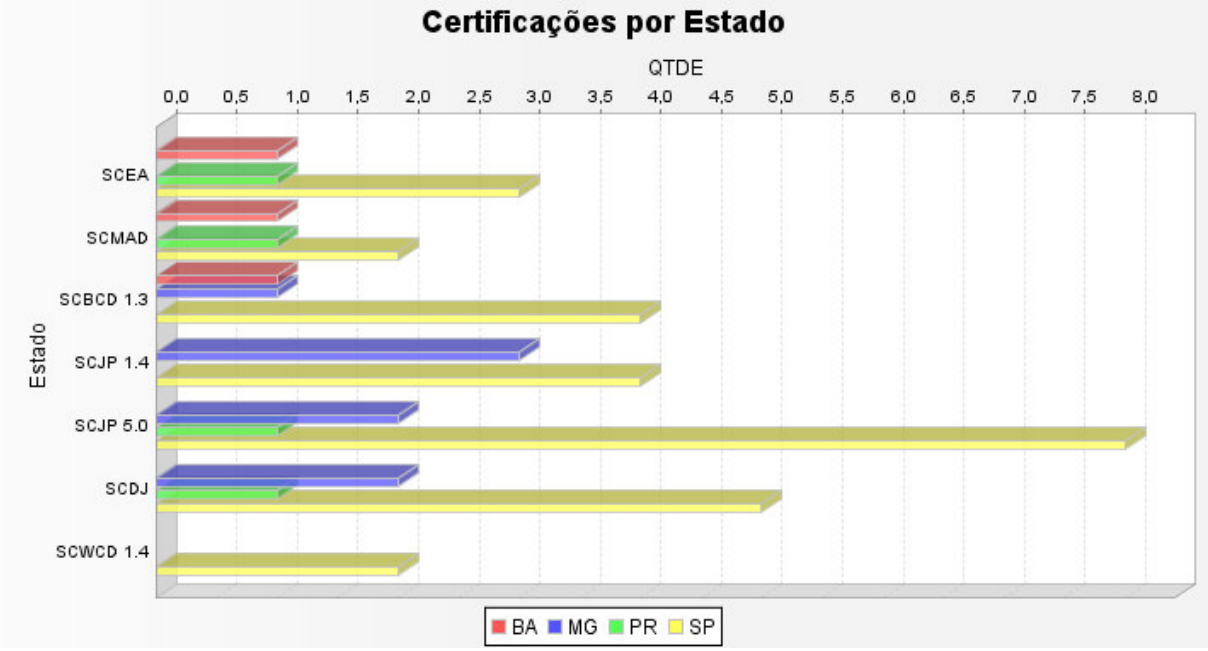
```
<%@ page contentType="text/html; charset=iso-8859-1" errorPage="" %>
<%@ taglib uri="/WEB-INF/cewolf-1.1.tld" prefix="cewolf"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=Cp1252"/>
<title>Exemplo 2</title>
</head>
<body>
<jsp:useBean id="graficoDS" class="br.eti.furutani.dataset.PessoasCertificadasDataSet"/>

<center><h3>GRÁFICOS DE LINHAS E COLUNAS</h3></center>

	<!-- GRÁFICO LINHAS E COLUNA -->
		<cewolf:chart id="grafico1" title="Certificações por Estado" type="line"
				xaxislabel="Estado" yaxislabel="QTDE">
		    <cewolf:gradientpaint>
		        <cewolf:point x="0" y="0" color="#FBFBFB" />
		        <cewolf:point x="350" y="0" color="#F3F3F3" />
		    </cewolf:gradientpaint>
		    <cewolf:data>
		        <cewolf:producer id="graficoDS" />
		       </cewolf:data>
		</cewolf:chart>
		<cewolf:img chartid="grafico1" renderer="/cewolf" width="710" height="380"/>


		<br/><br/>

		<!-- GRÁFICO COLUNA HORIZONTAL 3D -->
```

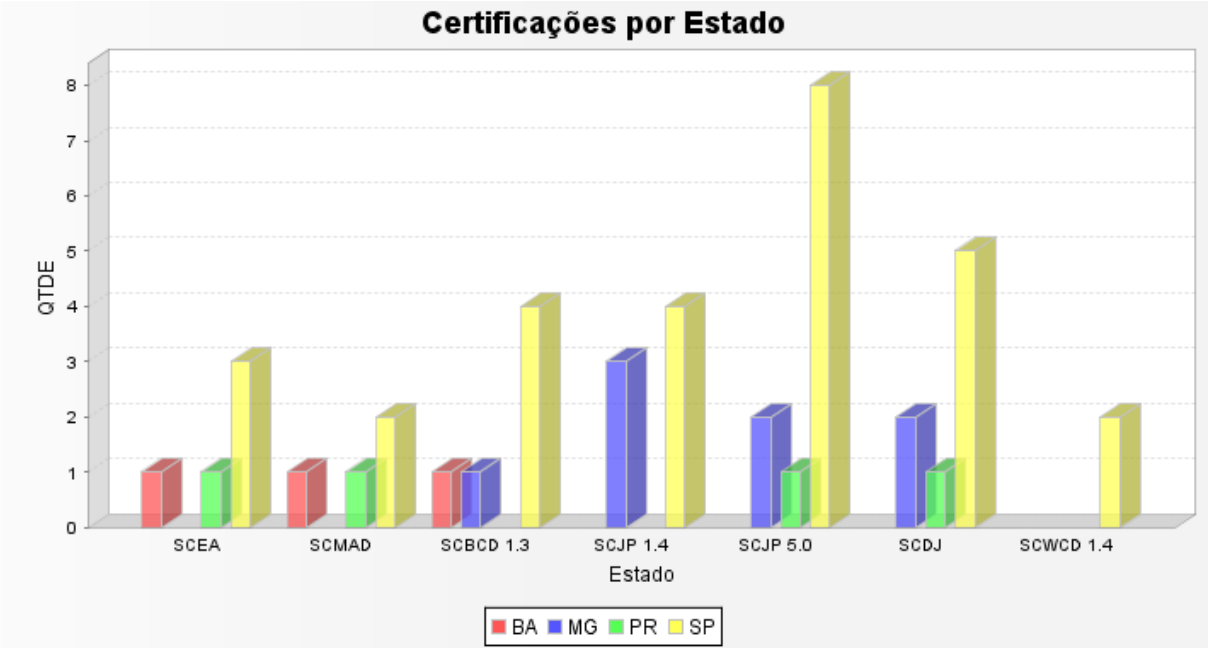
```
                <cewolf:chart id="grafico2" title="Certificações por Estado" type="horizontalbar3d"
                              xaxislabel="Estado" yaxislabel="QTDE">
                    <cewolf:gradientpaint>
                        <cewolf:point x="0" y="0" color="#FBFBFB" />
                        <cewolf:point x="350" y="0" color="#F3F3F3" />
                    </cewolf:gradientpaint>
                    <cewolf:data>
                        <cewolf:producer id="graficoDS" />
                       </cewolf:data>
                </cewolf:chart>
                <cewolf:img chartid="grafico2" renderer="/cewolf" width="710" height="380"/>

                <br/><br/>

                <!-- GRÁFICO COLUNA 3D -->
                <cewolf:chart id="grafico3" title="Certificações por Estado" type="verticalbar3d"
                              xaxislabel="Estado" yaxislabel="QTDE">
                    <cewolf:gradientpaint>
                        <cewolf:point x="0" y="0" color="#FBFBFB" />
                        <cewolf:point x="350" y="0" color="#F3F3F3" />
                    </cewolf:gradientpaint>
                    <cewolf:data>
                        <cewolf:producer id="graficoDS" />
                       </cewolf:data>
                </cewolf:chart>
                <cewolf:img chartid="grafico3" renderer="/cewolf" width="710" height="380"/>

                <br/><br/>

                <!-- GRÁFICO DE AREA -->
                <cewolf:chart id="grafico4" title="Certificações por Estado" type="area"
                              xaxislabel="Estado" yaxislabel="QTDE">
                    <cewolf:gradientpaint>
                        <cewolf:point x="0" y="0" color="#FBFBFB" />
                        <cewolf:point x="350" y="0" color="#F3F3F3" />
                    </cewolf:gradientpaint>
                    <cewolf:data>
                        <cewolf:producer id="graficoDS" />
                       </cewolf:data>
                </cewolf:chart>
                <cewolf:img chartid="grafico4" renderer="/cewolf" width="710" height="380"/>

                <br/><br/>

                <!-- GRÁFICO COLUNA -->
                <cewolf:chart id="grafico5" title="Certificações por Estado"
type="stackedverticalbar3d"
                              xaxislabel="Estado" yaxislabel="QTDE">
                    <cewolf:gradientpaint>
                        <cewolf:point x="0" y="0" color="#FBFBFB" />
                        <cewolf:point x="350" y="0" color="#F3F3F3" />
                    </cewolf:gradientpaint>
```

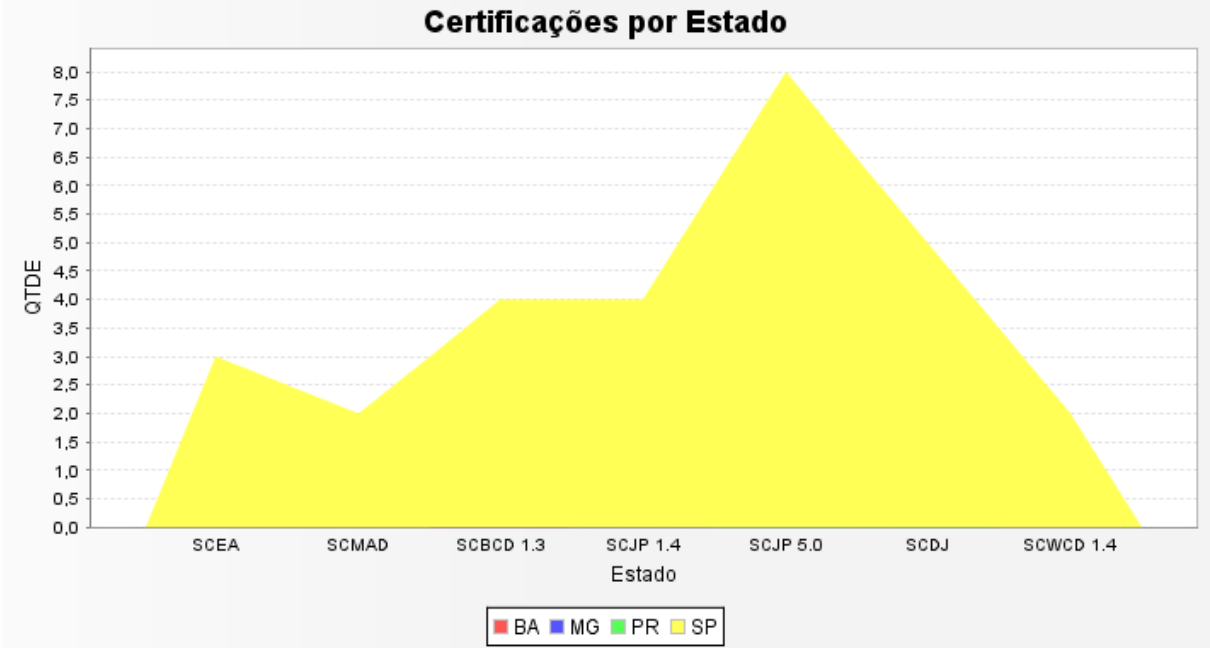
```
                    <cewolf:data>
                        <cewolf:producer id="graficoDS" />
                       </cewolf:data>
                </cewolf:chart>
                <cewolf:img chartid="grafico5" renderer="/cewolf" width="710" height="380"/>

</body>
</html>
```

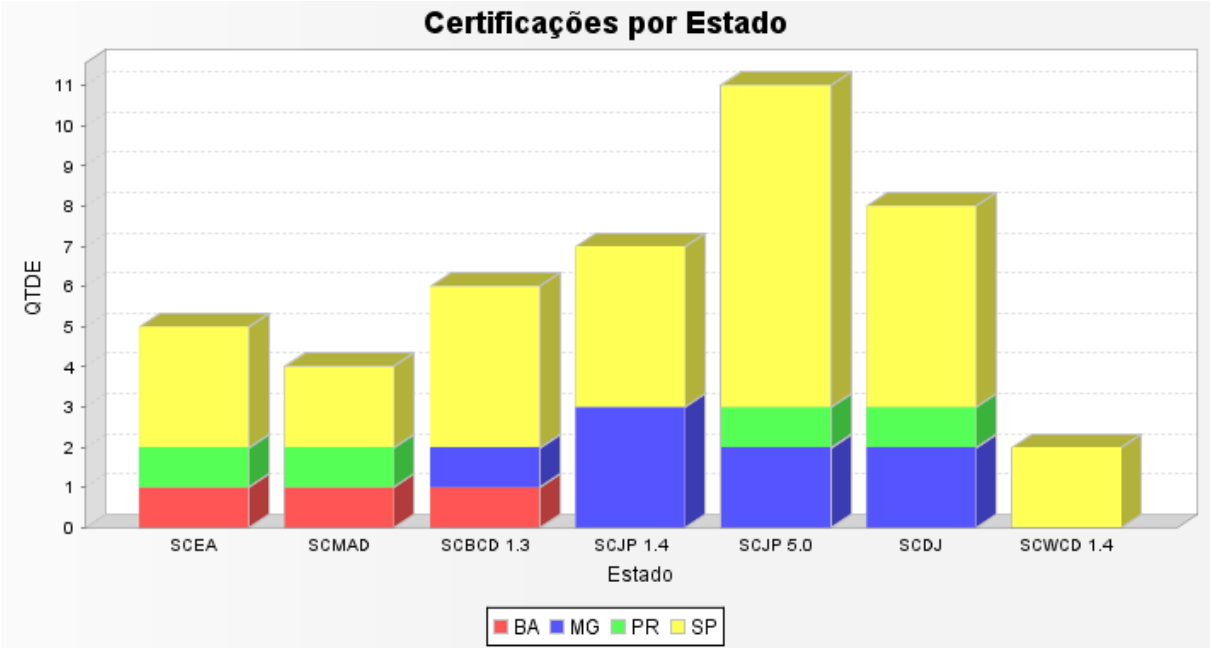
É incrível a quantidade de gráficos diferentes que podemos criar com o DefaultCategoryDataset, veja nas figuras abaixo.




Certificações por Estado
QTDE
0,0 0,5 1,0 1,5 2,0 2,5 3,0 3,5 4,0 4,5 5,0 5,5 6,0 6,5 7,0 7,5 8,0
Estado
SCEA
SCMAD
SCBCD 1.3
SCJP 1.4
SCJP 5.0
SCDJ
SCWCD 1.4
BA MG PR SP


Certificações por Estado
QTDE
0 1 2 3 4 5 6 7 8
SCEA
SCMAD
SCBCD 1.3
SCJP 1.4
SCJP 5.0
SCDJ
SCWCD 1.4
Estado
BA MG PR SP





4. Conclusão

O Cewolf é uma ótima biblioteca para criar gráficos na web, oferece facilidade e uma grande variedade de estilos de gráficos.

5. Referências

Site oficial, http://cewolf.sourceforge.net/new/index.html

Tomcat, http://tomcat.apache.org/

Eclipse, http://www.eclipse.org/

JfreeChart, http://www.jfree.org/jfreechart/

GUJ, http://www.guj.com.br

Fonte, http://www.furutani.eti.br/tutoriais/Graficos.zip

Atualizado: sábado, 26 de novembro de 2005 as 20:49:31