

Visual Mining, a division of Tervela, Inc.

# **Building a Charting EJB With NetCharts Pro 7.2**

**A Programmers Guide to Building Chart-Enabled  
Applications with NetCharts® Pro 7.2  
Version 7.2  
Summer 2015**

---

## Table of Contents

<b>1.0</b>	<b>SCOPE.....</b>	<b>3</b>
	PREREQUISITES FOR CREATING EJB'S WITH NETCHARTS PRO.....	3
<b>2.0</b>	<b>CREATING AN EJB THAT USES NETCHARTS PRO .....</b>	<b>4</b>
	CREATING THE EJB REMOTE INTERFACE.....	4
	CREATING THE EJB HOME INTERFACE .....	5
	CREATING THE EJB BEAN CLASS .....	5
<b>3.0</b>	<b>DEPLOYING THE EJB .....</b>	<b>8</b>
<b>4.0</b>	<b>TESTING THE EJB.....</b>	<b>9</b>
<b>5.0</b>	<b>WHERE TO GET ADDITIONAL HELP .....</b>	<b>12</b>
	APPENDIX A – SAMPLE PROEJB REMOTE INTERFACE .....	13
	APPENDIX B – SAMPLE PROEJBHOME HOME INTERFACE .....	14
	APPENDIX C – SAMPLE PROEJBBEAN EJB BEAN .....	15
	APPENDIX D – SAMPLE PROEJB TEST SERVLET.....	17
	APPENDIX E – SAMPLE WEB.XML.....	21
	GENERAL INFORMATION.....	22

## 1.0 Scope

This document provides NetCharts Pro users information and instructions for wrapping NetCharts Pro functionality within an Enterprise Java Bean implementation. This process will require experience with Java programming and Enterprise Java Bean configuration. The example describes creating an EJB 2.x Stateless Session Bean and anyone comfortable with these skills should be able to perform the procedures described in this white paper.

### ***Prerequisites for creating EJB's with NetCharts Pro***

Before beginning, you should ensure the following:

- NetCharts Pro Version 7.2 or later has been downloaded.
- You are familiar with Java programming and syntax.
- You have an EJB enabled Application Server and are familiar with deploying an EJB Session Bean and Servlets (via a WAR and/or EAR) to the Application Server.

## 2.0 Creating an EJB that uses NetCharts Pro

NetCharts Pro is a set of graphical components intended for Java developers. NetCharts Pro enables complete control over graph creation, presentation and behavior. To encapsulate the NetCharts Pro API behind an EJB implementation, the first decision to make is what portion of the NetCharts Pro API to expose via the EJB. In this example, we will expose the basic functionalities of generating a chart object, returning the client-side JavaScript used for active labels and adding a chart image into a Java Servlet.

There are three class files (two interfaces and a class) we will need to create for the EJB. The EJB remote interface, the EJB home interface and the EJB bean implementation class. For this example, the EJB will be named `ProEJB`.

### *Creating the EJB Remote Interface*

The remote interface defines the methods that will be exposed to the client. The following are the requirements for the remote interface:

- The remote interface must extend `java.ejb.EJBObject`.
- The methods must be valid RMI methods. Their return values and arguments must be serializable types and the `throws` clause must include `java.rmi.RemoteException`.
- Each method in the remote interface must have a matching method signature and implementation in the EJB class. The implementations in the EJB class must throw the same business logic exceptions that the remote interface does.

The methods we will be exposing are `getGraphFromTemplate` and `getPage`. Since we are generating an interface, no method implementation will be included. Here is the sample `ProEJB` remote interface:

```
package ncpro.examples.ejb;

import java.rmi.RemoteException;
import java.util.Hashtable;

import javax.ejb.EJBObject;
import javax.servlet.http.HttpServletRequest;

import netcharts.pro.common.NFGraph;
import netcharts.pro.common.page.NFPageParams;

public interface ProEJB extends EJBObject {
    public NFGraph getGraphFromTemplate(
        String template,
        Hashtable<String, String> variables)
        throws RemoteException, Exception;
    public String getPage(
        HttpServletRequest request,
        NFGraph chart,
        NFPageParams params)
        throws RemoteException, Exception;
}
```

## Creating the EJB Home Interface

The home interface provides the client with methods to create an instance of the EJB Bean and retrieve a remote reference. The following are the requirements for the home interface:

- The home interface must extend `javax.ejb.EJBHome`.
- The home interface must define at least one `create` method, which must be called `create`. The return type for the `create` methods must be the remote interface. In this example, a `ProEJB` interface is returned.
- The methods must be valid RMI methods and must throw `java.rmi.RemoteException` and `javax.ejb.CreateException`.
- The home interface `create` methods must match the counterpart methods defined in the EJB class (named `ejbCreate`). The return types of the methods will be different. The home interface `create` methods will return the remote interface (in this case `ProEJB`) while the EJB Bean class will return `void` from the `ejbCreate` methods.

Here is the sample `ProEJBHome` home interface:

```
package ncpro.examples.ejb;

import java.rmi.RemoteException;

import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface ProEJBHome extends EJBHome {
    public ProEJB create() throws RemoteException, CreateException;
}
```

## Creating the EJB Bean Class

Next, we will create the Bean class that contains the Java implementation using NetCharts Pro. The EJB Bean class contains the implementation of the methods that clients can call. The requirements for the bean implementation class are:

- The EJB Bean class must implement the `javax.ejb.SessionBean` interface.
- The EJB Bean class cannot be abstract.
- The EJB Bean class must have methods and implementations that match the `create` methods within the home interface. The methods should be `public` and have a return type of `void`. They should also be named according to the naming convention `ejbCreate`.
- The EJB Bean class methods must match the method declared in the remote interface.
- The methods must be valid RMI methods and must throw `java.rmi.RemoteException` and `javax.ejb.CreateException`.

Here is the sample `ProEJBBean` EJB Bean class:

```
package ncpro.examples.ejb;

import java.rmi.RemoteException;
import java.util.Hashtable;

import javax.ejb.CreateException;
```

```
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.servlet.http.HttpServletRequest;

import netcharts.pro.common.NFGraph;
import netcharts.pro.util.NFServletUtil;
import netcharts.util.NFContext;
import netcharts.util.NFGlobal;

public class ProEJBBean implements SessionBean {
```

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

The following set of methods must be implemented when implementing the `javax.ejb.SessionBean` interface.

```
    public void setSessionContext(SessionContext arg0)
        throws EJBException, RemoteException {
    }

    public void ejbRemove() throws EJBException, RemoteException {
        System.out.println("ProEJBBean.ejbRemove: called "+this);
    }

    public void ejbActivate() throws EJBException, RemoteException {
        System.out.println("ProEJBBean.ejbActivate: called "+this);
    }

    public void ejbCreate() throws CreateException {
        System.out.println("ProEJBBean.ejbCreate: called "+this);
    }

    public void ejbPassivate() throws EJBException, RemoteException {
        System.out.println("ProEJBBean.ejbPassivate: called "+this);
    }
```

Calling `NFProUtil.enableServerContext` instructs NetCharts Pro to disable any client side logic that is unnecessary during pure server side chart generation. By including it in a static code block the context will be set before any charts are generated.

```
    static {
        NFProUtil.enableServerContext();
    }
```

The next set of API methods is the implementation of the methods defined in the EJB remote interface (`ProEJB`) and a helper method `performSubstitution`.

```
    public String getPage(HttpServletRequest request,
        NFGraph chart,
        NFPageParams params)
        throws RemoteException, Exception {
        return NFServletUtil.getPage(request, chart, params);
    }
```

```

public NFGraph getGraphFromTemplate(String template,
                                     Hashtable<String, String> variables)
    throws RemoteException, Exception {
    return NFGraph.getGraphFromTemplate(
        performSubstitution(template, variables));
}

protected String performSubstitution(String templateCDL,
                                     Hashtable <String, String> variables) {
    if (templateCDL == null) {
        return (templateCDL);
    }

    if (variables == null) {
        return (templateCDL);
    }

    if (variables.size() == 0) {
        return (templateCDL);
    }

    Enumeration<String> keys = variables.keys();
    while (keys.hasMoreElements()) {
        String variableName = keys.nextElement();
        templateCDL = templateCDL.replace(variableName,
            variables.get(variableName));
    }

    return templateCDL;
}

```

## 3.0 Deploying the EJB

Next, the EJB needs to be deployed to the J2EE Application Server. The EJB JAR file will need to contain the class files for the home and remote interfaces and the EJB Bean implementation. It will also need to contain NetCharts Pro and a deployment descriptor (**ejb-jar.xml** in the **META-INF** directory).

The deployment descriptor provides information to the Application Server about how the EJB should be deployed. It also contains configuration information describing the EJB such as the EJB name, the names of the class files that contain the home and remote interfaces as well as the EJB Bean implementation and other EJB container specific properties. A sample **ejb-jar.xml** is included:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC
    "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN"
    "http://java.sun.com/dtd/ejb-jar_1_1.dtd">

<ejb-jar>
    <enterprise-beans>
        <session>
            <ejb-name>ProEJB</ejb-name>
            <home>ncpro.examples.ejb.ProEJBHome</home>
            <remote>ncpro.examples.ejb.ProEJB</remote>
            <ejb-class>ncpro.examples.ejb.ProEJBBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Bean</transaction-type>
        </session>
    </enterprise-beans>
    <assembly-descriptor>
        <container-transaction>
            <method>
                <ejb-name>ProEJB</ejb-name>
                <method-name>*</method-name>
            </method>
            <trans-attribute>NotSupported</trans-attribute>
        </container-transaction>
    </assembly-descriptor>
</ejb-jar>
```

**NOTE:** Deploying the EJB will be different depending on the J2EE Application Server.

**NOTE:** If you are deploying the NetCharts Pro based EJB within an UNIX environment, Visual Mining's products are written in Java and contain code that creates graphical objects (charts). This can introduce a dependency on the host system's native graphics environment. On UNIX platforms, Java's graphics libraries (the Java AWT) are implemented on top of XWindows/Motif graphics libraries and by default depend on access to an XServer in order to obtain font metrics and other graphics information.

NetCharts Pro is commonly run on Unix machines that do not have XServers. Java code that performs server-side chart image generation are often deployed on production machines that are headless (i.e. a machine without a monitor/mouse/keyboard attached) and do not have a physical XServer installed.

Sun introduced a headless processing mode in starting with JVM 1.4. Running NetCharts Pro 7.2 with a JVM 1.6 or greater using the *-Djava.awt.headless=true* flag eliminates the need for an XServer.



## 4.0 Testing the EJB

After the EJB has been deployed, it is available for requests from clients. For this example, we will create a Servlet to be run within the J2EE Application Server. To properly server chart images using NetCharts Pro within a web application a set of Servlet configurations should be configured with the web application web.xml. See the companion document *Programming with NetCharts Pro* for more information on how to configure your web application to use NetCharts Pro. The sample web.xml found in [Appendix E – Sample web.xml](#) includes the Servlet configurations as a reference.

A sample Servlet included will serve a web page that contains a form allowing the user to enter data to chart. The Servlet includes a base chart template that will be used to generate a chart object via the EJB `getChartFromTemplate()` method. The data entered by the user will also be included with this call and substituted into the template during processing. The return from the `getPage()` method will be used to include the chart within the web page.

The sample Servlet includes a method named `getEJB` to retrieve the EJB instance. This method includes code that uses two environment variables, `EjbContextFactory` and `EjbJNDIName`. Because J2EE Application Servers sometimes have different mechanisms for EJB naming and JNDI lookups these variables are used to specify Application Server specific properties. The variable `EjbJNDIName` specifies the names of the EJB to use when performing a JNDI lookup. The variable `EjbContextFactory` can be used to specify a class name if the Application Server uses a custom class for JNDI lookups. These variables can be configured in the web.xml of the WAR containing the sample Servlet. A sample web.xml can be found in [Appendix E – Sample web.xml](#).

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
    // If the theData parameter exists the user
    // has posted some data to use.
    Hashtable<String, String> variables = new
        Hashtable<String, String>();
    String theData = request.getParameter("theData");
    if (theData == null) {
        theData = DEFAULT_DATA;
    }
    variables.put("DATASET_VARIABLE", theData);

    // Retrieve the chart template.
    // The chart template could be stored on disk or in
    // an external database.
    String template = getDefaultChartTemplate();

    // Generate the chart object.
    NFGraph graph = getChartObject(template, variables, response);

    // Start outputting the page.
    ServletOutputStream os = response.getOutputStream();
    os.println("<html>");
    os.println("<head>");
    os.println("</head>");
    os.println("<body>");
    os.println("<div align=\"center\">");
    if (graph != null) {
```

```
        try {
            // Generate the chart on the page.
            os.print(getChartImageCode(graph,
                request, response));
        } catch (RemoteException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // Add the form so the user can chart their own data.
    os.println("<form method=\"POST\" name=\"DataForm\">");
    os.println("<p>Enter the data to chart in the format <i>"
        + DEFAULT_DATA + "</i>: ");
    os.println("<input type=\"text\" name=\"theData\" value=\""
        + theData + "\" size=\"30\">");
    os.println("<input type=\"submit\" value=\"Submit\" "
        + name=\"SubmitButton\"></p>");
    os.println("</form>");
    os.println("</div>");
    os.println("</body>");
    os.println("</html>");
}
```

```
protected String getChartImageCode(NFGraph chart,
    HttpServletRequest request,
    HttpServletResponse response) {

    try {
        ProEJB proEjb = getEjb();
        if (proEjb == null) return null;

        String ret = proEjb.getPage(request, chart,
            new NFRasterPageParams(
                NFRasterPageParams.MIME_TYPE_JPG));
        proEjb.remove();
        return(ret);
    } catch (Exception exc) {
        return "Error: " + exc.getMessage();
    }
}
```

```
protected NFGraph getChartObject(String template,
    Hashtable<String, String> variables,
    HttpServletResponse response) {

    try {
        ProEJB proEjb = getEjb();
        if (proEjb == null) return null;

        NFGraph graph = proEjb.getGraphFromTemplate(template,
            variables);
        proEjb.remove();
        return(graph);
    } catch (Exception exc) {
        System.out.println(exc.getMessage());
    }
}
```

```
        return null;  
    }  
}
```

The full servlet code can be found in [Appendix D – Sample ProEJB Test Servlet](#).

## 5.0 Where to get additional help

The NetCharts Pro distribution includes API documentation for more information on the various API calls available. The distribution also includes sample code showing various API calls, including how to include images within a servlet.

More information on Enterprise JavaBeans can be found at <http://java.sun.com/products/ejb/> and J2EE at <http://java.sun.com/j2ee/>.

For more information about NetCharts Pro please contact [support@visualmining.com](mailto:support@visualmining.com).

## Appendix A – Sample ProEJB Remote Interface

```
package ncpro.examples.ejb;

import java.rmi.RemoteException;
import java.util.Hashtable;

import javax.ejb.EJBObject;
import javax.servlet.http.HttpServletRequest;

import netcharts.pro.common.NFGraph;
import netcharts.pro.common.page.NFPageParams;

public interface ProEJB extends EJBObject {

    public NFGraph getGraphFromTemplate(String template,
                                         Hashtable<String, String> variables)
        throws RemoteException, Exception;

    public String getPage(HttpServletRequest request, NFGraph chart,
                          NFPageParams params)
        throws RemoteException, Exception;
}
```

## Appendix B – Sample ProEJBHome Home Interface

```
package ncpro.examples.ejb;

import java.rmi.RemoteException;

import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface ProEJBHome extends EJBHome {
    public ProEJB create() throws RemoteException, CreateException;
}
```

## Appendix C – Sample ProEJBBean EJB Bean

```
package ncpro.examples.ejb;

import java.rmi.RemoteException;
import java.util.Enumeration;
import java.util.Hashtable;

import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.servlet.http.HttpServletRequest;

import netcharts.pro.common.NFGraph;
import netcharts.pro.common.page.NFPageParams;
import netcharts.pro.util.NFProUtil;
import netcharts.pro.util.NFServletUtil;

@SuppressWarnings("serial")
public class ProEJBBean implements SessionBean {
    static {
        NFProUtil.enableServerContext();
    }

    public ProEJBBean() {
        super();
    }

    public void setSessionContext(SessionContext arg0)
        throws EJBException, RemoteException {
    }

    public void ejbRemove() throws EJBException, RemoteException {
        System.out.println("ProEJBBean.ejbRemove: called "+this);
    }

    public void ejbActivate() throws EJBException, RemoteException {
        System.out.println("ProEJBBean.ejbActivate: called "+this);
    }

    public void ejbCreate() throws CreateException {
        System.out.println("ProEJBBean.ejbCreate: called "+this);
    }

    public void ejbPassivate() throws EJBException, RemoteException {
        System.out.println("ProEJBBean.ejbPassivate: called "+this);
    }

    public String getPage(HttpServletRequest request,
        NFGraph chart, NFPageParams params)
        throws RemoteException, Exception {
        return NFServletUtil.getPage(request, chart, params);
    }

    public NFGraph getGraphFromTemplate(String template,
        Hashtable<String, String> variables)
        throws RemoteException, Exception {
        return NFGraph.getGraphFromTemplate(
            performSubstitution(template, variables));
    }
}
```

```
protected String performSubstitution(String templateCDL,
                                     Hashtable<String, String> variables) {

    if (templateCDL == null) {
        return(templateCDL);
    }

    if (variables == null) {
        return(templateCDL);
    }

    if(variables.size() == 0) {
        return(templateCDL);
    }

    Enumeration<String> keys = variables.keys();
    while (keys.hasMoreElements()) {
        String variableName = keys.nextElement();
        templateCDL = templateCDL.replace(variableName,
                                           variables.get(variableName));
    }

    return templateCDL;
}
```



## Appendix D – Sample ProEJB Test Servlet

```
package ncpro.examples.examples;

import java.io.IOException;
import java.rmi.RemoteException;
import java.util.Hashtable;
import java.util.Properties;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import ncpro.examples.ejb.ProEJB;
import ncpro.examples.ejb.ProEJBHome;
import netcharts.pro.common.NFGraph;
import netcharts.pro.common.page.raster.NFRasterPageParams;

public class ProEJBExampleServlet extends HttpServlet {
    protected static String DEFAULT_DATA = "100.0,200.0,300.00,200.0,100.0";

    protected static String     ejbContextFactory = null;
    protected static String     ejbJNDIName = null;
    protected static String     examplePathPrefix = null;

    private String defaultTemplate =
        "ChartType      = BARCHART;" +
        "ChartName       = \"chart\";" +
        "ChartSize        = (500,300);" +
        "ColorTable       = " +
        "    \"x284b53,x005699,xb8bc9c,x271651,xaa0036,xecf0b9,x999966,\" +
        "    \"x333366,xc3c3e6,x594330,xa0bdc4,x005699,x999966,x213321,\" +
        "    \"x998300;\" +
        "Background       = (white,NONE,1,\"null\",TILE,black);" +
        "AntiAlias         = ON;" +
        "Legend            = (\"OFF\",black,\"SansSerif\",10,0);" +
        "LegendItems       = ;" +
        "Header            = (\"Dynamic Data Via EJB\",black,\"" +
        "    \"SansSerif\",12,0,,CENTER);" +
        "RightTitle        = (\" \",black,\"SansSerif\",12,0,,CENTER);" +
        "RightTitleBox      = (null,NONE,2,\"null\",TILE,black);" +
        "DwellLabel        = (ON,white,\"SansSerif\",10,0);" +
        "DwellLabelBox      = (grey,RAISED,1,\"null\",TILE,black);" +
        "BackgroundFillPattern = (NONE,null,null);" +
        "Grid              = (lightgray,null,black);" +
        "GridLine           = (BOTH,SOLID,1);" +
        "GridAxis           = (BOTTOM,LEFT);" +
        "RightScale         = ;" +
        "RightTics          = (OFF,,\"SansSerif\",10,0);" +
        "RightTicLayout      = (AUTO,0,2);" +
        "TopScale           = ;" +
        "TopTics            = (OFF,,\"SansSerif\",10,0);" +
        "TopTicLayout        = (AUTO,0,2);" +
        "BottomScale        = ;" +
        "BottomTics         = (ON,black,\"SansSerif\",10,0);" +
        "BottomTicLayout     = (AUTO,0,2);" +
        "LeftScale1         = (0,,);" +
```

```
"LeftTics1      = (ON,,\"SansSerif\",10,0);"+
"LeftTicLayout1  = (AUTO,0,2);"+
"LeftAxesGaps = ;"+
"LeftAxesLayout  = 0;"+
"Grid3DDepth = -1;"+
"BarLabels       = \"Jan\", \"Feb\", \"Mar\", \"Apr\", \"May\", "+
                  "\"June\", \"July\", \"Aug\", \"Sept\", "+
                  "\"Oct\", \"Nov\", \"Dec\""+
"DataSets        = (\"BarSet1\",null,BAR);"+
"BarSymbol       = (BAR,null);"+
"BarValueLabelStyle = NONE;"+
"DataSet1        = DATASETVARIABLE;"+
"BarBorder       = (NONE,1,black);"+
"Bar3DDepth      = 5;"+
"BarWidth        = 61;"+
"GraphLayout     = VERTICAL;"+
"GraphType       = GROUP;";

public String getDefaultChartTemplate() {
    return defaultTemplate;
}

public void doPost(HttpServletRequest request,
                   HttpServletResponse response) throws
    ServletException, IOException {
    doGet(request, response);
}

public void doGet(HttpServletRequest request,
                   HttpServletResponse response) throws
    ServletException, IOException {
    if (NFServletUtil.isSecondPass(request)) {
        NFServletUtil.writeImage(request, response);
        return;
    }

    // If the theData parameter exists the user
    // has posted some data to use.
    Hashtable<String, String> variables =
        new Hashtable<String, String>();
    String theData = request.getParameter("theData");
    if (theData == null) {
        theData = DEFAULT_DATA;
    }
    variables.put("DATASETVARIABLE", theData);

    // Retrieve the chart template. The chart
    // template could be stored on disk or in
    // an external database.
    String template = getDefaultChartTemplate();

    // Generate the chart object.
    NFGraph graph = getChartObject(template, variables, response);

    // Start outputting the page.
    ServletOutputStream os = response.getOutputStream();
    os.println("<html>");
    os.println("<head>");
    os.println("</head>");
    os.println("<body>");
    os.println("<div align=\"center\">");
    if (graph != null) {
        try {
```

```
        // Generate the chart on the page.
        os.print(getChartImageCode(graph,
            request, response));
    } catch (RemoteException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// Add the form so the user can chart their own data.
os.println("<form method=\"POST\" name=\"DataForm\">");
os.println("<p>Enter the data to chart in the format <i>" +
    DEFAULT_DATA + "</i>: ");
os.println("<input type=\"text\" name=\"theData\" value=\"" +
    theData + "\" size=\"30\">");
os.println("<input type=\"submit\" value=\"Submit\"
    name=\"SubmitButton\"></p>");
os.println("</form>");
os.println("</div>");
os.println("</body>");
os.println("</html>");
}

protected String getChartImageCode(NFGraph chart,
    HttpServletRequest request,
    HttpServletResponse response) {
    try {
        ProEJB proEjb = getEjb();
        if (proEjb == null) return null;

        String ret = proEjb.getPage(request, chart,
            new NFRasterPageParams(
                NFRasterPageParams.MIME_TYPE_JPG));
        proEjb.remove();
        return(ret);
    } catch (Exception exc) {
        return "Error: " + exc.getMessage();
    }
}

protected NFGraph getChartObject(String template,
    Hashtable <String, String> variables,
    HttpServletResponse response) {
    try {
        ProEJB proEjb = getEjb();
        if (proEjb == null) return null;

        NFGraph graph = proEjb.getGraphFromTemplate(template,
            variables);
        proEjb.remove();
        return(graph);
    } catch (Exception exc) {
        System.out.println(exc.getMessage());
        return null;
    }
}

protected ProEJB getEjb() {
    Properties props = System.getProperties();
```

```
// Check to see if this servlet engine uses a custom
// class for doing JNDI lookups. For example, JRun
// uses a custom class "allaire.ejpt.ContextFactory".
// If the property EjbContextFactory exists and is
// non-null, load it into the properties object before
// doing the lookup.

String nejbContextFactory = getEjbContextFactory(this);
String nejbJNDIName = getEjbJNDIName(this);

if(nejbContextFactory != null) {
    props.put(Context.INITIAL_CONTEXT_FACTORY,
               ejbContextFactory);
}

try {
    Context ctx = new InitialContext(props);
    ProEJBHome proEjbHome =
        (ProEJBHome)ctx.lookup(nejbJNDIName);
    return proEjbHome.create();
} catch(Exception exc) {
    System.out.println(exc.getMessage());
    return null;
}

}

protected static synchronized String getEjbContextFactory(
    HttpServlet httpServlet) {
    if(ejbContextFactory == null) {
        ejbContextFactory = getProperty(httpServlet,
                                         "EjbContextFactory");
    }
    return(ejbContextFactory);
}

protected static synchronized String getEjbJNDIName(
    HttpServlet httpServlet) {
    if(ejbJNDIName == null) {
        ejbJNDIName = getProperty(httpServlet, "EjbJNDIName");
    }
    return(ejbJNDIName);
}

protected static String getProperty(HttpServlet httpServlet,
    String name) {
    String result = null;

    try {
        Context c = new InitialContext();
        result = (String)c.lookup("java:comp/env/"+name);
    } catch(Exception exc) {
    }

    // For WebLogic 5.1 support.
    if(result == null) {
        result = httpServlet.getInitParameter(name);
    }

    System.out.println("[getProperty] looking up "+name+
        ",received "+result);
    return(result);
}

}
```

## Appendix E – Sample web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
    <display-name>NetCharts Pro Example EJB Servlet</display-name>
    <servlet>
        <servlet-name>ResourceServlet</servlet-name>
        <display-name>ResourceServlet</display-name>
        <servlet-class>netcharts.pro.util.NFResourceServlet</servlet-class>
        <init-param>
            <param-name>RetainImageInSession</param-name>
            <param-value>true</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>ProEJBEx</servlet-name>
        <display-name>ProEJBEx</display-name>
        <servlet-class>ncpro.examples.ejb.examples.ProEJBExampleServlet</servlet-
class>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>ResourceServlet</servlet-name>
        <url-pattern>/getresource</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ResourceServlet</servlet-name>
        <url-pattern>/NFLicense.dat</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ResourceServlet</servlet-name>
        <url-pattern>/netcharts.jar</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ProEJBEx</servlet-name>
        <url-pattern>/ProEJBEx</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
    <env-entry>
        <env-entry-name>EjbJNDIName</env-entry-name>
        <env-entry-value>ProEJB</env-entry-value>
        <env-entry-type>java.lang.String</env-entry-type>
    </env-entry>
</web-app>
```

---

## General Information

NetCharts, NetCharts Pro, NetCharts Server, NetCharts Designer, NetCharts Performance Dashboards, Chart Definition Language and Visual Mining are trademarks of Visual Mining, a division of Tervela, Inc.

Other product names used in this document are trademarks of their respective owners.

© 1996-2015 Visual Mining, a division of Tervela, Inc. All rights reserved.

**Visual Mining, a division of Tervela, Inc.**

2301 Research Blvd.  
Suite 201  
Rockville, MD 20850

**Inquiries**

<b>General Phone</b>	800.308.0731
<b>International</b>	+1.301.795.2200
<b>Fax</b>	301.947.8293
<b>General Inquiries</b>	<a href="mailto:info@visualmining.com">info@visualmining.com</a>
<b>Customer Support</b>	<a href="mailto:support@visualmining.com">support@visualmining.com</a>
<b>Sales</b>	<a href="mailto:sales@visualmining.com">sales@visualmining.com</a>
<b>Press and Media Inquiries</b>	<a href="mailto:marketing@visualmining.com">marketing@visualmining.com</a>

**Web**

<http://www.visualmining.com>