



How Three Great Tools Can Work Together

by Randy Eckhoff, IBM Corp.
Software Engineer, Lotus BeanMachine

Abstract

VisualAge for Java... BeanMachine... NetObjects Fusion... I am sure you have heard of these tools before, usually as individual articles. But have you heard how you can use these three tools together? This article will describe to you how you can create a Java bean using VisualAge for Java. You will then take this bean and use it in BeanMachine to create a cool multimedia applet. Lastly, you will create a web site using NetObjects Fusion that uses this applet and publish it out to a web server.

Quick Overview

Before you get started, lets think about web applications a bit and who develops them. A sophisticated web application will consist of a snazzy, well designed user interface that provides access to some data that resides somewhere. The user of this application does not care where the data sits or how it is retrieved. It could be a CICS transaction to a DB/2 database sitting on an AS/400 box. These are terms that the typical user does not comprehend, or care to. This breakout also applies to how the web application is built.

A visual designer's eyes will glaze over when they hear things like CICS, DB/2 and AS/400. Their area of expertise is user interface and visual design. They always want to do the impossible and ask Herculean tasks of software engineers who always come through for them. Developing a web application is getting more and more complicated. Before, it was the visual designer who developed the application which was usually just some multimedia pizzazz. Now, web users are demanding more sophisticated applications. They want to shop on the web, or do their banking online. The visual designer knows how to build a web site and add really cool multimedia to these sites. And they are great at it. But ask them to add a database query so that someone can pay off their credit card balance, and they just look at you.

This is where the software engineer comes in. They know how to do this stuff and this is their area of expertise. They'll have that database query done in no time. (Yes, there are people who can do both, but they are in the minority). This is where VisualAge for Java, BeanMachine, and NetObjects Fusion come in.

With VisualAge for Java, you write all your access to data code. It can create Java applets, applications, and beans. Everything you need to develop a business application.

With Lotus BeanMachine, you create cool multimedia applets. There is a small amount of overlap with VisualAge for Java in that both can create applets. But BeanMachine is for glitzy, multimedia applets that make the data look good instead of just a boring table.

With NetObjects Fusion, you create your web site using html and applets.

It is that simple. All three tools are visual builders- you select an item from a palette and place it on your visual layout area. What could be easier!

This article will give you a tour of all three tools. You will use VisualAge for Java to create a simple bean. It will illustrate many of the points to build a bean. A more complex bean could have been built but that complexity is dependent on the different VisualAge parts that are used and the function they have to offer. The bean will take the contents of an entry field and add them to a listbox. Once the bean is created, you will import it into BeanMachine to create a cool multimedia applet. This applet will take the item selected in the listbox and put it up in "lights". After the applet has been created, you will use NetObjects Fusion to create a web site that will use this applet on one of it's web pages.

Before you get started, it would be good to know what versions of the software were used for this article. A Windows 95 box was used running VisualAge for Java 1.0, BeanMachine 1.1, and NetObjects Fusion 2.0. Also, this article will not be a comprehensive, function by function overview of all three tools. One could write many different articles and still not be done. There are other articles written about the individual tools out there. Just search this database or search the web. Ready? Lets jump right in!

VisualAge for Java

Start VisualAge. The **Quick Start** window is displayed. Select **Go to workbench** and then **OK**. This displays the **Workbench**. Think of this as home base. The tabs (Project, Packages, etc) across the top of the window are different views of your work. Projects are exactly that, a way to group the work that you do. Packages is a programming thing that lets you group your code. Class and interfaces is basically your code. For example, if you were writing code that handled mortgage calculations like fixed and variables rates, you might have a Fixed30Year class and an ARM class. These two classes would do the actual work to calculate a mortgage. The package might be named Mortgage and the project might be named OnlineBanking. In Java, basically everything is a class, regardless of what it does. Applets, applications, beans are all Java classes. It is the specific classes and who they inherit from that give them unique behavior.

Select **Add Project** from the **Selected** pulldown and type in *ExampleBeans*. Select **Finish**. You just created a project. If it is not selected, select ExampleBean and then select **Add Package** from the **Selected** pulldown. Type in *ExampleForArticle* for the new package name and select **Finish**.

Now we come to the good stuff. With the project and package out of the way, you can now create your class. Yes, earlier in the article, it was stated that you will create a bean. Remember that beans are classes. You'll get around to the bean stuff later. With

ExampleForArticle selected, select **New Class/Interface** from the **Selected** pulldown. For the class name, type in *NameInLights*. At this point, you need to make a decision. You need to ask yourself "What Java class do I need to inherit from". The answer to this question is important because it determines the overall behavior your class will have. Since you are building something that needs to be able to contain user interface elements, you want to create a visual class.

How does one learn what class to inherit from? This is where you start to see one of the separations between software engineer and visual designer. If you are using VisualAge for Java, you are a software engineer and will need to know the ins and outs of the Java language. Also, you will consult the Java reference documentation to figure out what you can do in Java. Remember that VisualAge for Java is a fully integrated development environment.

So where does that leave us? After looking in the Java API reference, you learned that the Panel class provides a visual space in which you can add other components. For the superclass, type in *java.awt.Panel*. Panel is the name of the class and java.awt is what package the Panel class can be found in. And since it is so much nicer to do things visually, make certain that **Design the class visually** is selected. When you are finished, select **Finish**.

The NameInLights class is created and up comes the **Visual Composition Editor**. Look familiar? It should if you used other VisualAge products. This is where you lay out your visual components from your Parts Palette. The parts are broken into different categories and can be found on the left side of the window. The tabs across the top provide access to the actual source code that makes up this class, where the class resides in the overall class hierarchy, different editions in the library, and information about the class that will be exposed when the class is used as a bean.

Lay out the following parts: an entry field, listbox, and push button. Select the button and then select **actionPerformed** under **Connect** from the button's popup menu. Move the mouse to the listbox and click. Select **addItem** from the resulting popup menu. You just made a connection! This is how you wire parts together without writing any code. However, the connection is not yet complete. When the button is clicked, something will be added to the listbox. That "something" has not yet been specified. Since the contents of the entry field will be added, select **Text** from the entry field's **Connect** menu. Move the mouse to the green connection line and click. Select **item** from the resulting popup menu. The connection is now complete.

You've been kept in the dark long enough. A bean is what is needed in order to be used in BeanMachine. So how do I make this class accessible as a bean? Easy. Switch to the BeanInfo tab and select **New BeanInfo class** from the **Features** pulldown. Type in *NameInLights* and then select **Finish**. The class is now accessible as a bean. A quick 4 sentence summary of a bean follows. JavaBeans is a specification, defined by Sun Microsystems, for building reusable Java components. These components have actions, properties, and events that can be wired together to build more Java components, be it beans, applets, applications or other entities. The BeanInfo is what tells other tools about the public interface of the bean. For example, BeanMachine will read the BeanInfo to determine what actions, properties, and events to expose when the bean is imported into BeanMachine.

So what actions, properties, and events do you want to surface for your bean? For your sample, only the listbox's selected item property will be exposed. Select **New Property Feature** from the **Features** pulldown. Type in *nameToHilite*. Notice the property type.

This is where you specify what type of property nameToHilite is. Is it a string, an integer, a boolean, etc? Since the listbox contains strings, you want this field to be java.lang.String. Select **Finish**.

Since you want to expose what is selected in the listbox, you need to somehow place that value into nameToHilite everytime it changes. Sound complicated? Not at all. Switch back to the Visual Composition Editor and connect the listbox's selectedItem property to the bean's nameToHilite property. You know how to do the first part of the connection. To connect to nameToHilite, move the mouse to the white area outside the dotted box. In the resulting popup, select **All Features**. Then select **nameToHilite** and select **OK**. This completes your property to property connection in the **Composition Editor**.

There is one little quirk about the basic user interface controls that are provided by Java. The property to property connection you just created will not automatically run when the listbox's selected item changes. The listbox was implemented such that when one of its properties changes, it will not notify anybody who is interested in knowing that it has changed. However, separate from properties, the listbox has events that you can monitor. By attaching one of these events to the connection, you can force the connection to run. To do this, double click on the connection. In the **Source event** dropdown list, select **itemStateChanged**. Then select **OK**.

You are almost done with VisualAge for Java. You just need to save it and place the bean into a file that BeanMachine can use to import it. From the **File** pulldown, select **Save Bean**. Then close the window and go back to the **Workbench**. Select **ExampleBeans** and from the **File** pulldown, select **Export**. Select the **JAR File** radio button and then select **Next**. Specify where you want to store the JAR file and name it something like *Example.jar*. In the listbox, select **ExampleForArticle.NameInLights** to be marked as a bean. This means that the JAR file will store which classes it contains are beans. BeanMachine will ask the JAR file which classes are beans and then use the BeanInfo to find out the specific details about the individual beans themselves. Select **Finish** to save the JAR file.

You are done with VisualAge for Java. Pretty heavy stuff when you consider everything that you have done. It is not as intense from here. Close VisualAge, and let us move on to BeanMachine.

BeanMachine

This tool is a dream come true for the visual designer. It is all about making complex tasks easy and simple. With BeanMachine, you will be using the bean you created with VisualAge for Java and a few other parts provided by BeanMachine. First, you need to import the bean into BeanMachine. There are 2 ways to do this. One way is to add it through the BeanMachine **Bean Wizard**. However, this requires you to restart BeanMachine after it has been added. A quicker way is to copy the JAR file into the BeanMachine\Beans directory. Do that now. When BeanMachine starts, it will automatically import any new beans it finds in the Beans directory. When beans are automatically imported, they are placed in a category on the palette named the same name as the JAR file. So if you named the JAR file Example.jar, the name of the category will be Example. Start BeanMachine and you will see a progress dialog showing that BeanMachine is importing new beans. When it is complete, the **Composer** window and the **New Applet Wizard** come up. Close the wizard since you will be working straight from the Palette.

Go to the **Palette** and select the **Example** category. The **Palette** is refreshed to show all the parts in this category. You see only one which is the `NameInLights` bean that you created with VisualAge for Java. Select it and drop it out in the Composer. Look familiar? It should look exactly like it did in VisualAge for Java. Next, go to the **Controls** category and drop out a push button. Then, go to the **Multimedia** category and drop out an image and a nervous text part. What is the applet going to do? First, the user will enter data into the listbox. Once all the data is there that they want, the user will click on another button. This will start a special effect on the image. Once this special effect is finished, what ever is selected in the listbox will be displayed as the nervous text. Sound complicated? Hardly.

Now that you have laid out all the parts that are needed, you just need to set a few properties and make a few connections. First, select the image and go to the **Details** window. The **Details** window is where you set the properties and connections for the selected part. Let us get all the properties done first. Set the **picture** property to `BeanMachine\samples\effects\world.gif`. Then, set the **transition** property to some special effect. Ripple is always a pretty cool one. Lastly, set the **auto start transition** property to no. Auto start, when set to yes means that when the applet starts, the transition will automatically start. Other parts have similar auto start values. There is one more property to set. Select the button you just added and set the **label** property to something like *Click me to start transition*. BeanMachine should look something like [this](#).

Now for the connections. With the button still selected, switch to the **Connections** tab. BeanMachine does connections a little differently than VisualAge for Java. For a different audience, BeanMachine decided to go with more of a sentence style. The source part is already selected- the button. So all you need to do is specify which event you want to connect from and what part you want to do something to. On the first row, in the **When** column, select the **clicked** event. In the **Part** column, select **Image1**. In the **Do** column, select **start transition**. You have just created an event to action connection. This connection takes care of starting the special effect on the image when the button is clicked. The last thing that needs to be done is to display the selected item from the listbox in the nervous text.

Select the image. You are going to now create an event to property connection. Again, the source part is already selected so in the **When** column, select **transition ended**. In the **Part** column, select **Nervous Text1**. In the **Do** column, select **Set text**. Two new columns appear. In the **Using** column, select **NameInLights1**. This is the bean from VisualAge for Java. In the **Value** column, select **nameToHilite**. This is the property that you created on the **BeanInfo** tab in VisualAge for Java. [Here](#) is what the connection should look like.

That's it! You just wired together a bean you created with another bean. Test it by selecting the play button on the toolbar. Once the compile step is complete, up comes Applet Viewer. Add some items into the list. When you are done, select an item in the list. Notice that the nervous text started playing when the applet started but the image did not do its special effect. That is because auto start is on for nervous text and off for the image. Select the **Click me** button. The transition starts. When it finishes, the item selected in the list is automatically displayed in the nervous text.

You are almost done with BeanMachine. Close Applet Viewer and select the publish button. Save the applet and give it a name like *ExampleApplet*. In the **Publish Wizard**, select the **Local** tab. You can publish just about anywhere like a local file system or some remote FTP site. You can also publish out a NetObjects Fusion component. By

creating these components, you can add BeanMachine applets into NetObjects Fusion in one step and have all dependent files copied into their proper place for NetObjects Fusion to use. This is important because NetObjects Fusion has a very specific directory structure. Here is the **Publish** wizard.

Guess what you're going to do. That's right. Select **Yes** and also select **Publish NetObjects Fusion Component**. Then select **Finish**. BeanMachine will publish to the BeanMachine\publish directory, creating all the files you need. You are finished with BeanMachine so close it.

NetObjects Fusion

Unlike other tools that create just web pages, NetObjects Fusion creates an entire web site. Start NetObjects Fusion. If this is the first time you are running it, it will prompt you to enter the name of the web site you want to create. Call the web site something like *Example*. You are immediately placed into a tree view of the web site. The only web page you see is your **Home** page. Across the top of the window, you see various buttons that let you go to a specific page, manage the overall look of the web site, manage what file and other assets are used, and publish the site when it is finished. You can also preview the site while you are constructing it.

Another button is the **New Page** button that will add a new web page as a child to the page that is currently selected. Select it now and you will see a new web page called **Untitled** added below **Home**. Select **Home** and select **New Page** again. Now you have two **Untitled** pages under **Home**. Adding web pages is that easy. Change the names of the two **Untitled** pages to something like *page2* and *page3* by single clicking on the **Untitled** text.

Double click on the **Home** page and now you can lay out your home page however you want. The **Tools** palette contains various user interface elements that you can use. The **Properties** window lets you specify various properties of the specific elements on your page as well as the page itself. Lets add a simple element. Select the **Text** tool and place it on your page. Type in something like *I'm home!*.

The master border that you see defined around the layout area applies to all child pages and can be modified through the use of styles. A more in-depth look of NetObjects Fusion including how to create your own styles is available elsewhere. Lets go to **Page1** by clicking on the **Site** button and then double clicking on **Page2**. Notice that the layout is the same as for the **Home** page. This is the page that will contain our BeanMachine applet.

On the **Tools** palette, select the **NFX** tool and place it on your web page. An **Installed Components** window is displayed that shows all NetObjects Fusion Components that are already installed. You need to install the one you created so select the **Add** button and in the resulting File dialog, navigate to the directory where you published **ExampleApplet**. Select **ExampleAppletComp.nfx** and then **Open**. The **Installed Components** list is updated to show the new component. Select **ExampleApplet - BeanMachine** and then **OK**. You will not see the actual applet render itself within NetObjects Fusion because this would involve running the Java VM within NetObjects Fusion which is not currently supported. Instead, you will **see** a thin black frame sized to the size of the applet with a picture in the middle. Trust me, this is the applet you built. Also, the applet used other files like world.gif. These files have been added to the Asset list automatically but will be hidden from view. This is so that you don't have to

worry about managing them specifically which is one of the benefits of using NetObjects Fusion Components. You could have added this applet by using the **Applet** tool from the palette but then you would have to manually modify the assets list.

Nothing will be added to **Page3** so select the **Publish** button. You are now ready to publish your web site. You have two choices here. You can stage your web site to a test server for testing. Or you can publish it to your external site. And like many good tools, you can customize various settings for each step. Select **Settings** and then select the **Publish** tab. You can publish to a local file system or a remote FTP location. Select the **Local** radio button and specify a local directory to publish to and then select **OK**. Select the **Publish** button and the web site will be published.

You are done!

Close NetObjects Fusion and start up a web browser that is Java 1.1 enabled. Examples would be Navigator 4.02, Internet Explorer 4.0, HotJava, or Lotus Notes 4.6. Open up index.html (it was renamed from Home by NetObjects Fusion when you published) and navigate through your web site. Go to **Page2** and see your applet run!

Summary

I hope this has given you a good overview of what you can do with these tools. Yes, we could have built a more elaborate bean. Instead of calling addItem on the listbox, it could have been a DB/2 query. Yes, the applet could have done more and the web site could have been more complex. The point is that the possibilities with these 3 tools are endless.

Websites You Might Want To Visit

IBM VisualAge for Java is at <http://www.software.ibm.com/ad/vajava>

Lotus BeanMachine is at <http://www2.lotus.com/beanmachine>

NetObjects Fusion is at <http://www.netobjects.com>

Java is a trademark of Sun Microsystems, Inc.

Other companies, products, and service names may be trademarks or service marks of others.

Copyright Trademark

Java Feature

Java Home

▶ IBM HOME

▶ ORDER

▶ EMPLOYMENT

▶ CONTACT IBM

▶ LEGAL