

USING THE SAVVION FORM BUILDER

John Sadd
Fellow and OpenEdge Evangelist
Document Version 1.0
July 2011

Building Business Process Applications Using OpenEdge BPM


Using the Savvion Form Builder

BUSINESS MAKING PROGRESS™

John Sadd

Progress. | OpenEdge.

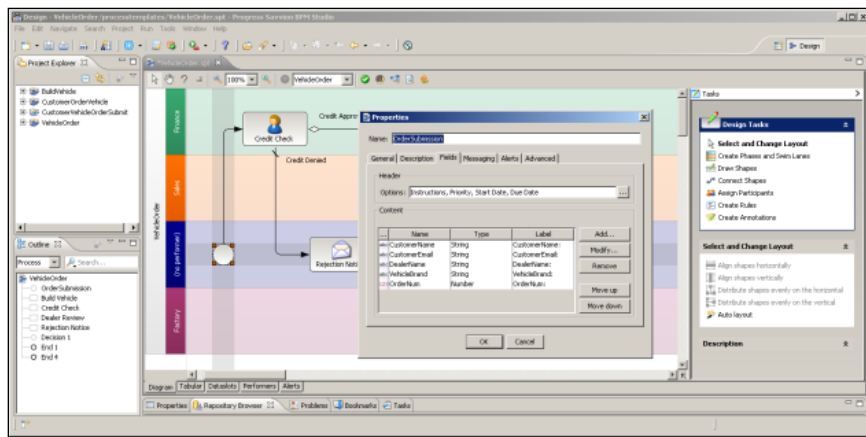
Progress. | Savvion.

PROGRESS
software 

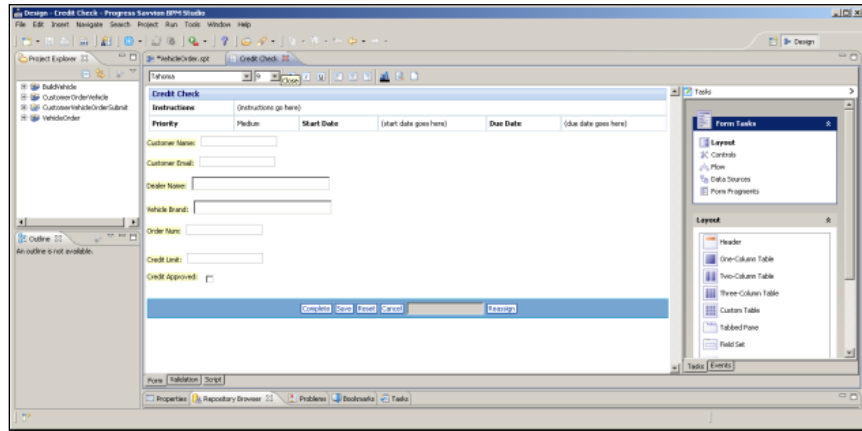
DISCLAIMER

Certain portions of this document contain information about Progress Software Corporation's plans for future product development and overall business strategies. Such information is proprietary and confidential to Progress Software Corporation and may be used by you solely in accordance with the terms and conditions specified in the PSDN Online (<http://www.psdn.com>) Terms of Use (<http://psdn.progress.com/terms/index.ssp>). Progress Software Corporation reserves the right, in its sole discretion, to modify or abandon without notice any of the plans described herein pertaining to future development and/or business development strategies. Any reference to third party software and/or features is intended for illustration purposes only. Progress Software Corporation does not endorse or sponsor such third parties or software.

Having walked you through the basic aspects of building a process model in Progress Savvion, in the video series and white paper ***Building Your First BPM*** Process, in this paper I'll start drilling down into some of the product's features in more detail, and get into a discussion of how you can use some of these features to integrate with an OpenEdge application. In this paper I introduce more material about creating forms in Savvion. First, a quick review of a few of the things covered in the first project. The **VehicleBuild Start** step uses an auto-generated form. I just selected some Dataslots to display as fields in the form, and I was able to re-order them from top to bottom, and that's all I specified:

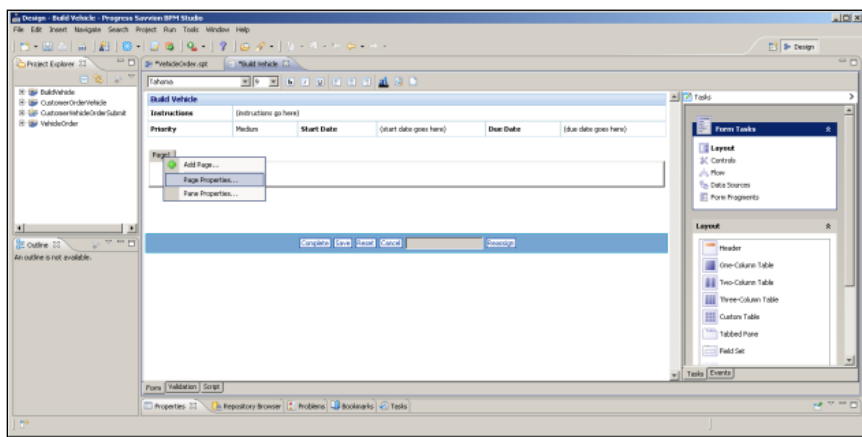


On the other hand, Activity worksteps like **CreditCheck** use a form that I was able to lay out in the form designer, like this one:

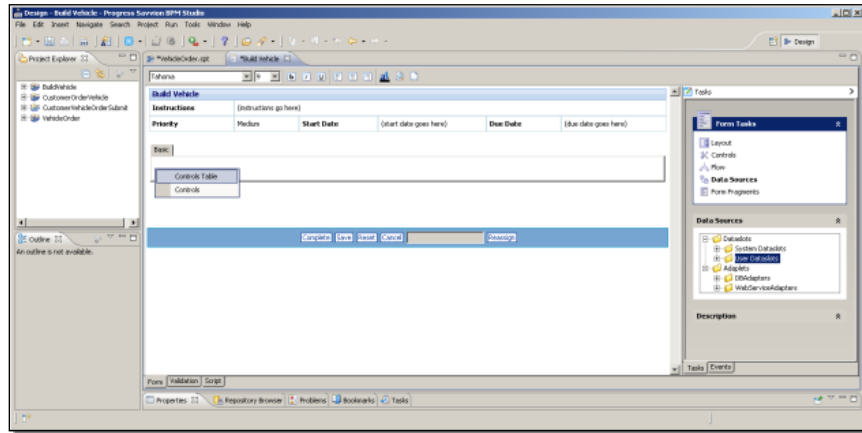


Let me show you some more of the things you can do on a form that you lay out yourself. You'll remember that I put in a **BuildVehicle** step just as a placeholder along the path the process follows if a vehicle the customer orders isn't in stock. Now I'll add some things to this form just to show you the possibilities. First I want to move the footer out of the way a bit, because of course I want what I add to be between header and footer.

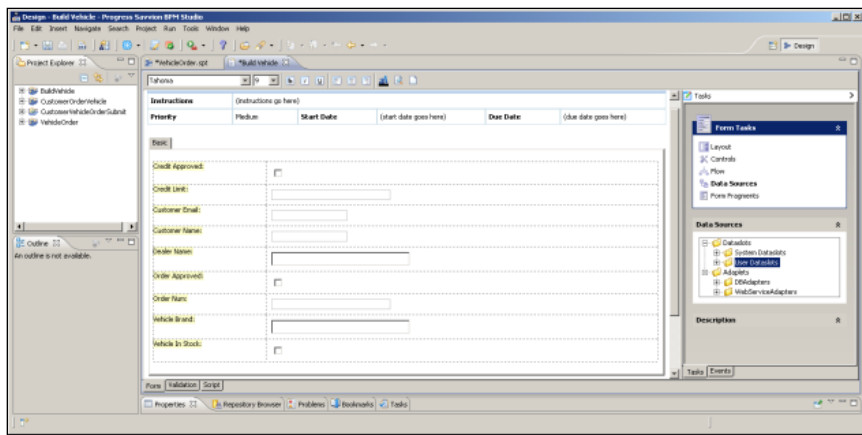
One important thing to know is that you can easily create a paged form, using the **Tabbed Pane** layout option. Notice also that controls that affect the overall layout, like this one and the tables, are listed under the **Layout** section of the form tasks. So I'll drag a Tabbed Pane onto the form, which gives me a default first page. Right-clicking on the page tab I can bring up page properties:



In the property sheet I can give it a label, such as Basic, since I'll drop some basic controls onto this page. Next I get some values to drop onto the form. Remember that the process Dataslots are available under the **Data Sources** section of the **Form Tasks**. If I expand the Dataslots, I can show you one easy trick, which is to grab the **User** dataslots folder itself, and drag that onto the form, in this case onto Page 1.



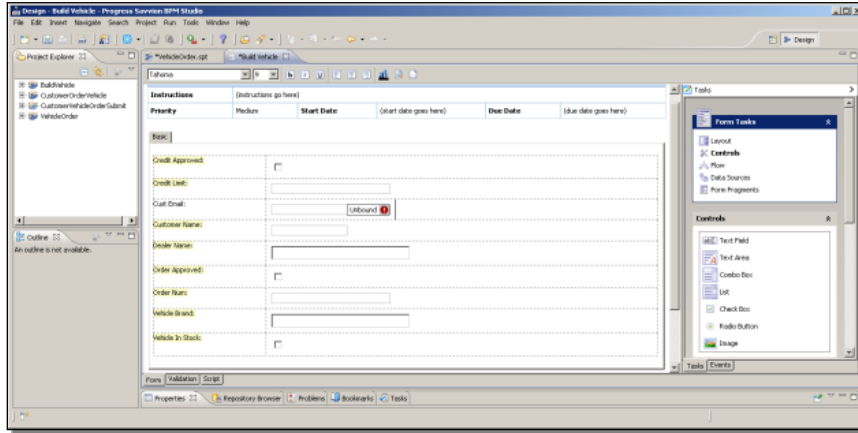
When I drop it, I get a choice between just getting the controls and their labels together side by side, or creating a two-column table with the labels in column 1 and the controls in column two, which is what I choose. Here is the form with all the fields on Page 1:



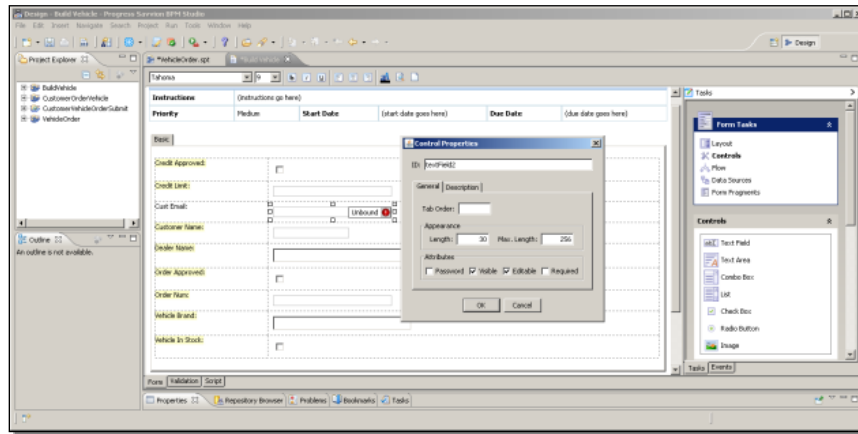
Remember that you can assign more categories to Dataslots than just one User category, which would make selecting a set of Dataslots in one go more practical. Next I make some changes to some of the fields.

The form editor gives me a label field based on the Dataslot label. I can't move it around, but I can replace it if I need to. Just backspacing over it deletes the whole label, and then I can type a label of my own, just as I could type text anywhere in the form. As for the control itself, I can delete that as well just to show you something else.

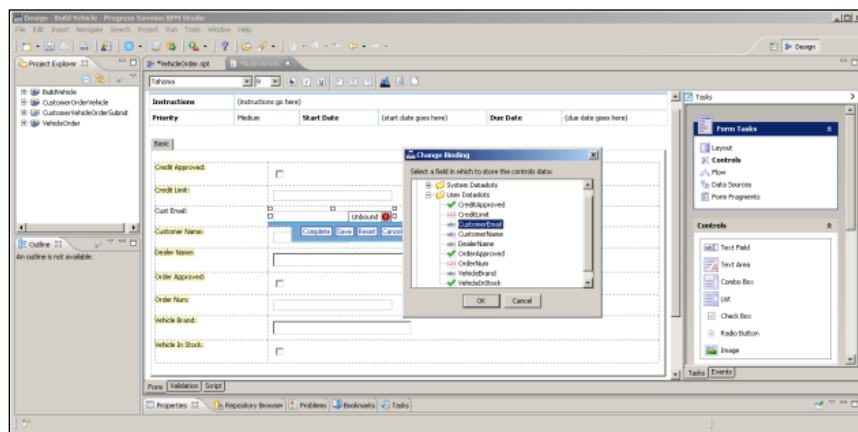
In addition to dragging dataslots onto a form and getting bound controls for them, that is, controls bound to the value of the Dataslot, I can also just select **Controls** to bring onto the form. For instance I can grab a **Text Field**, and drag that on in place of the bound control I just deleted. When I drop it, I see a warning symbol that the control is **Unbound**. This is OK; I could use unbound controls to hold form values that I populate myself, but this is just notifying me that this control isn't automatically bound to a value at runtime:



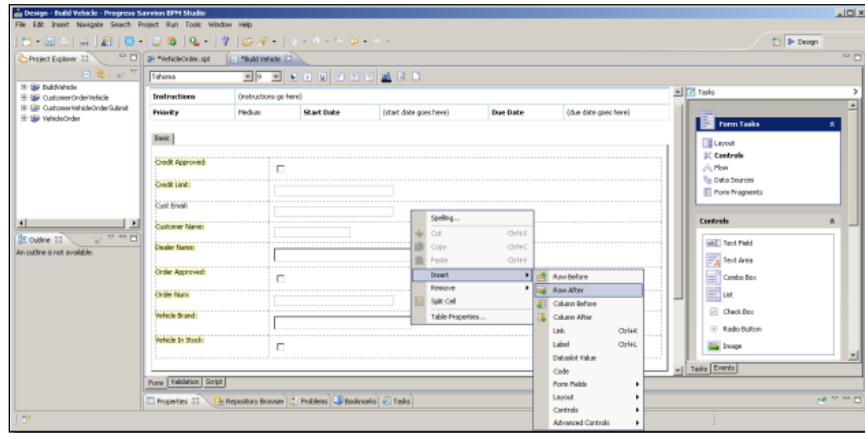
If I double-click on a control, bound or otherwise, its property sheet comes up as a dialog box. Here you can see all the basic properties you can set for the control, including whether it's editable in this form, whether it's required, and whether it's visible. You might define non-visible controls to hold values a form uses but doesn't need to display, for instance.



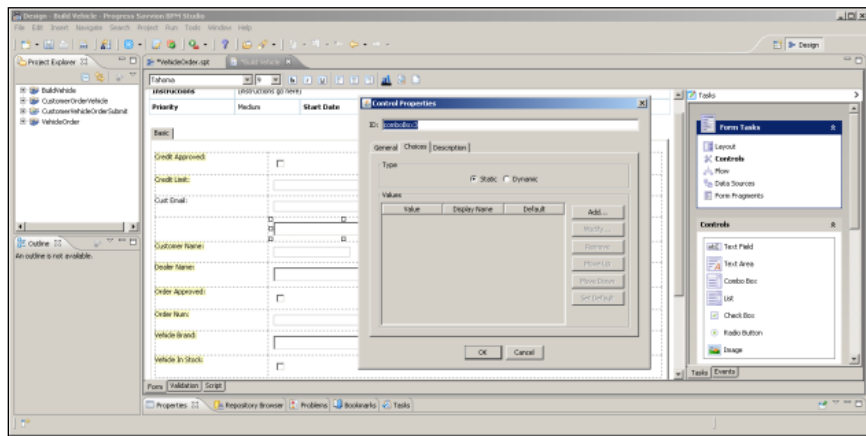
If you right-click in the control, you can select the Binding option to set or change the control's dataslot binding. In this case I'll rebind it to the **CustomerEmail** dataslot that I started out with:



You might start with unbound controls and then bind them afterwards to dataslots if you wanted to use a different type of user interface control than the default that's specified in the Dataslot definition, for instance. Now I have the CustomerEmail field back, and the form confirms that this control is now bound to the CustomerEmail dataslot. If I right-click in the row, I can add another one:



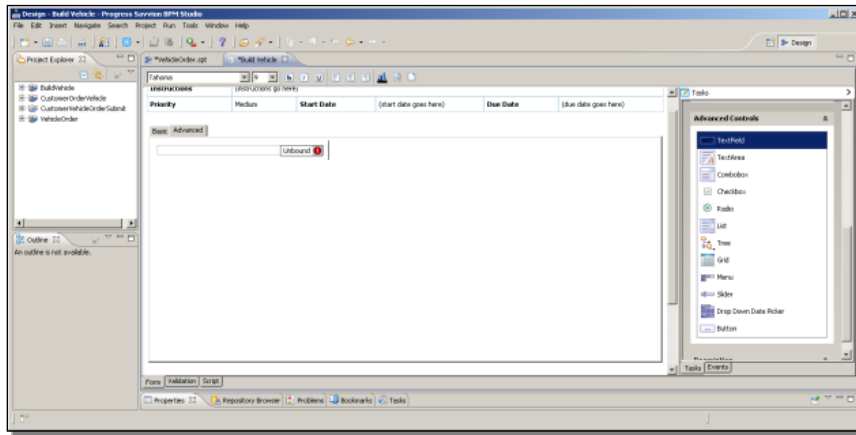
I bring a combo box onto the form, just to show you something about its properties. Again I double-click to bring up the basic control properties. Under the **Choices** tab, you can see that I have two different ways to populate the combo box. The **Static** option is the one I used when I created the **DealerName** and **VehicleBrand** dataslots. The **Add** button would let me add hard-coded values to display at runtime.



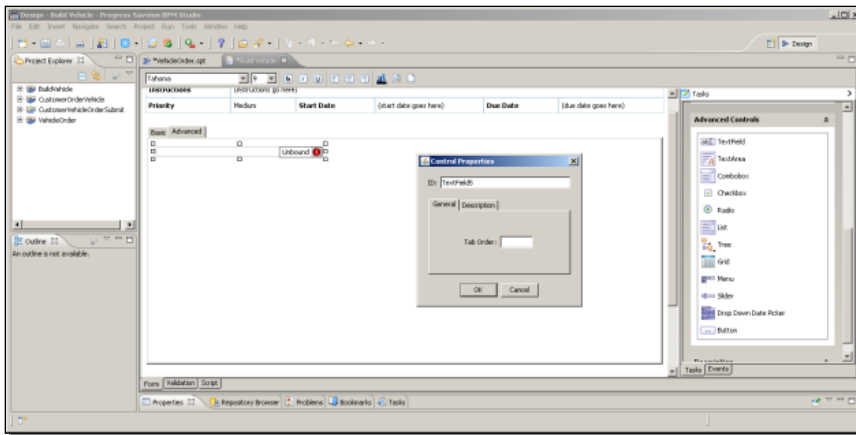
The **Dynamic** option, however, lets me specify a Data Source that will populate the combo box at runtime from the runtime value of one of my Dataslots. This is a choice for many types of controls, and one that I'll be showing you more about in a later presentation on how to populate the user interface with values at runtime. For now, I just wanted to make sure you know where to find that option. I can get rid of the extra row I added here, and then add a second page to the form to drop some different kinds of controls onto.

I name this folder tab **Advanced**, because I'm going to show you some **Advanced** controls. The controls I've been using so far are basic controls. If I scroll down in the Controls section, you can see that there's another whole group of Advanced controls.

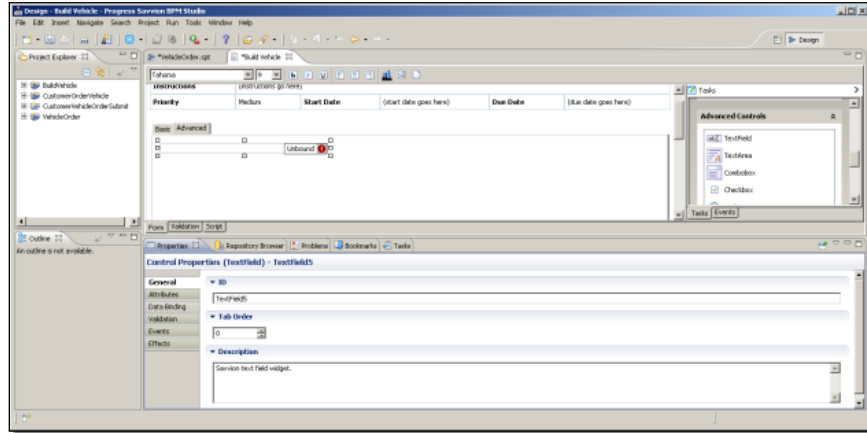
Some of these are the same types as the basic controls, such as the Text Field. I can drag this onto the form, drop it, and it looks just the basic text field:



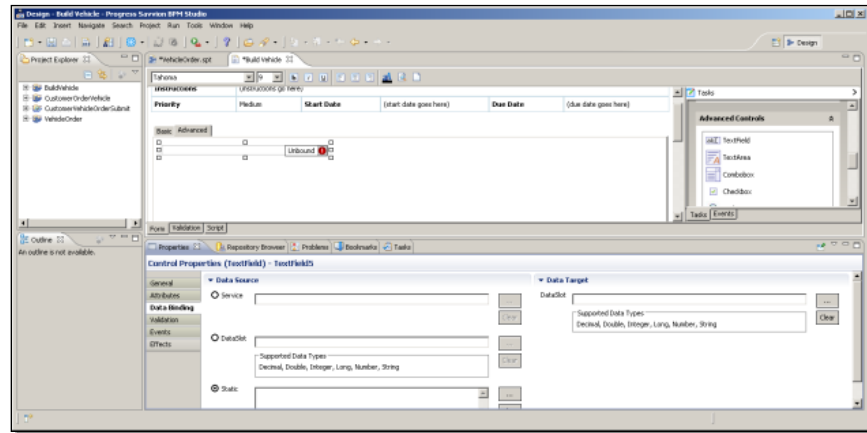
But if I double-click to bring up its properties, I get nothing in the dialog box but a few skeleton properties:



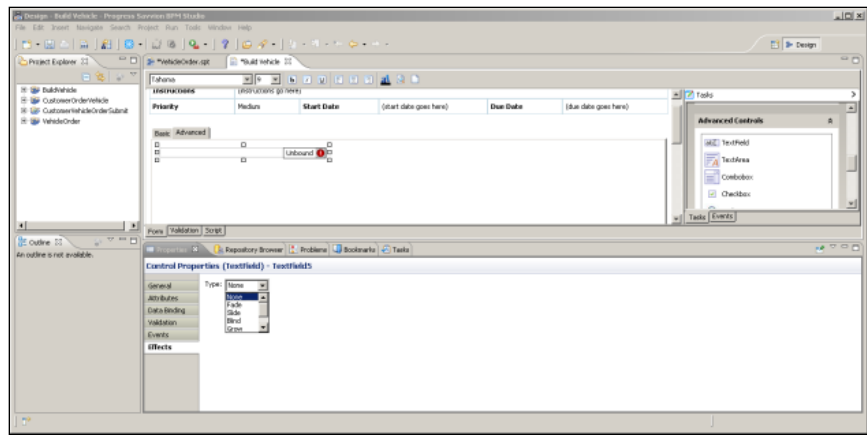
The reason for this is that the Advanced controls generally have more extensive properties than their basic counterparts. So rather than organizing them all under the dialog box, you have to use the properties view that in my layout is part of this pane that I have mostly had hidden at the bottom of the design tool. If I drag that pane up and make sure the **Properties** tab is selected, I see all the Advanced control's properties, organized under different sub-tabs. Here are the **General** properties like those you would see for basic controls:



The **Binding** properties include an option to bind the control to the output from a service:

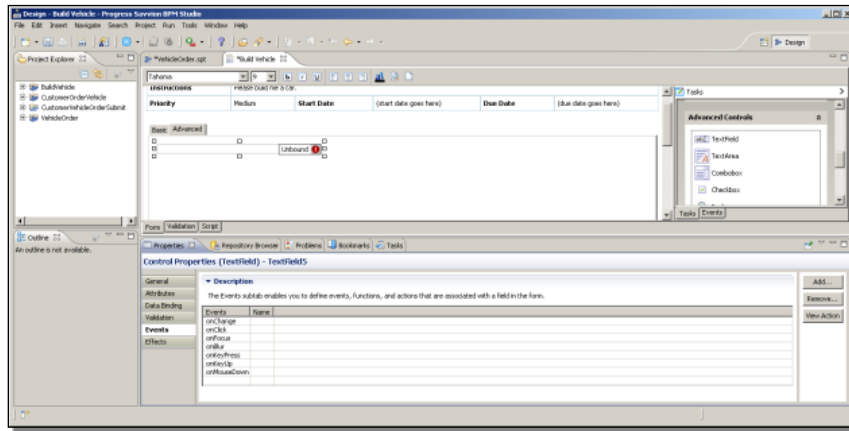


Advanced controls can also use visual effects to let you draw attention to them, such as fading or growing:

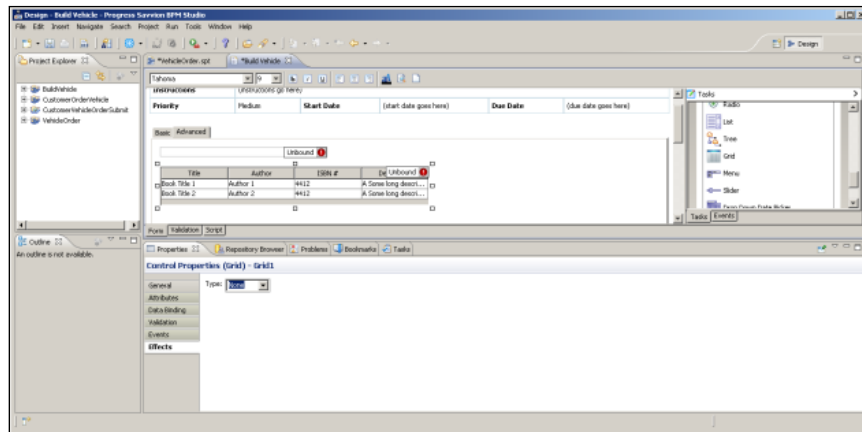


Another important group of properties is in the **Events** tab. These are all the kinds of events that you can associate with a control, on click, on change, and so on. Basic controls can have events, too, and you need to open up the control properties in this

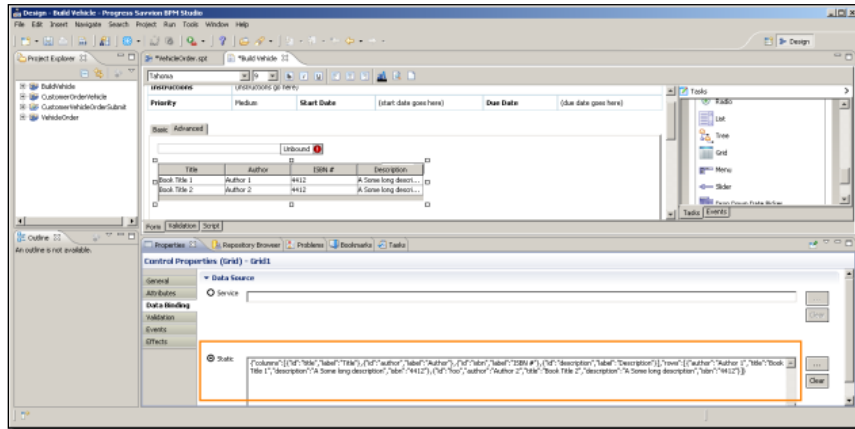
Properties View to see and use those events for basic controls, since they're not included in the basic property dialog.



Generally, it's good to get into the habit of using the Advanced controls even when their basic appearance is the same as the basic controls, because they're more flexible and full-featured. But there are also useful controls that only have an Advanced version. The grid is just one of those, and I'll quickly take a look at that. Its General properties look about the same as for other controls:

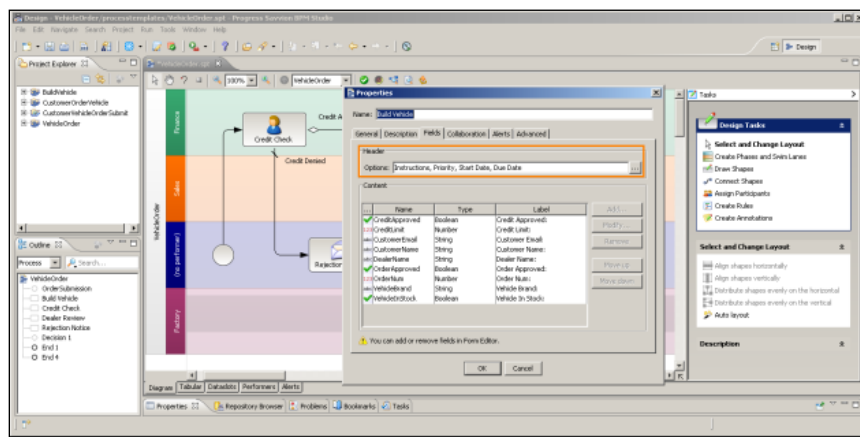


But under Data Binding, you can see a block of JSON text as a way to provide a static display in the grid, or you can specify a service whose output will populate the grid:

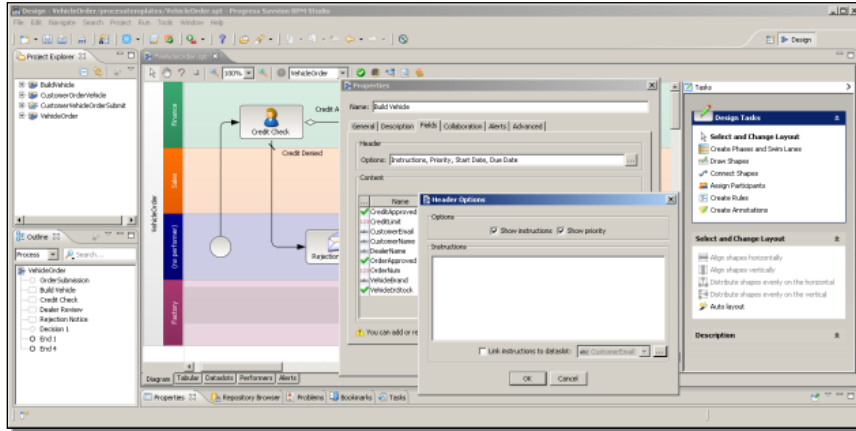


This was a very quick look at some of the controls. Obviously you should get to know them by taking your own tour of their various properties and the effects that they have. My goal here is just to make sure you know where to look for them. I clean up a bit by removing the second page on this form. Then the final thing I want to show you is how to modify the header and footer sections of a form.

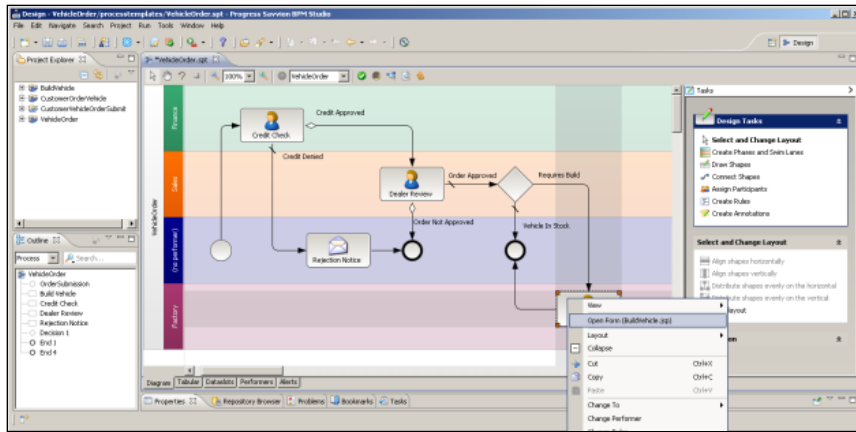
There are property sheets for the header and footer that you can get at from within the form, but in most ways it's better to adjust them from the workstep itself. Let me show you what I mean. If I right-click in the Build Vehicle workstep, and select properties again, and then the Fields tab, you'll note that in addition to the field list that I've manipulated before, there's a section at the top called **Header**. The **Options** are the special field values that are displayed in the form header by default:



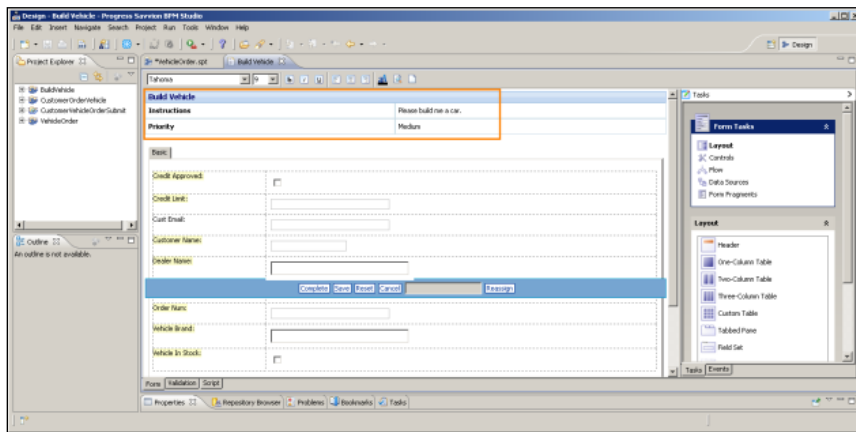
If I select the ellipsis, I can set the **Instructions** field to be some specific text I type in:



Alternatively, if I check this **Link to dataslot** option on, then I can instead select a Dataslot whose value will be displayed in the Instructions field at runtime. You can do one or the other but not both. Let's see what happens if I get out of here, and back into the form itself.

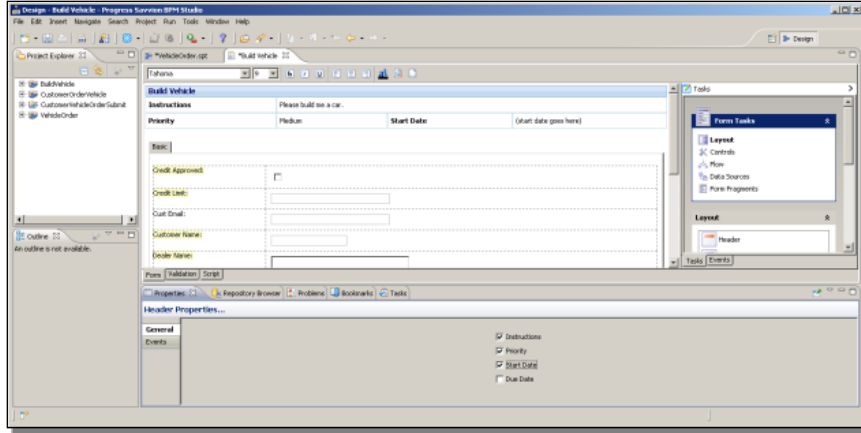


Here's the header with the Instructions text in it:

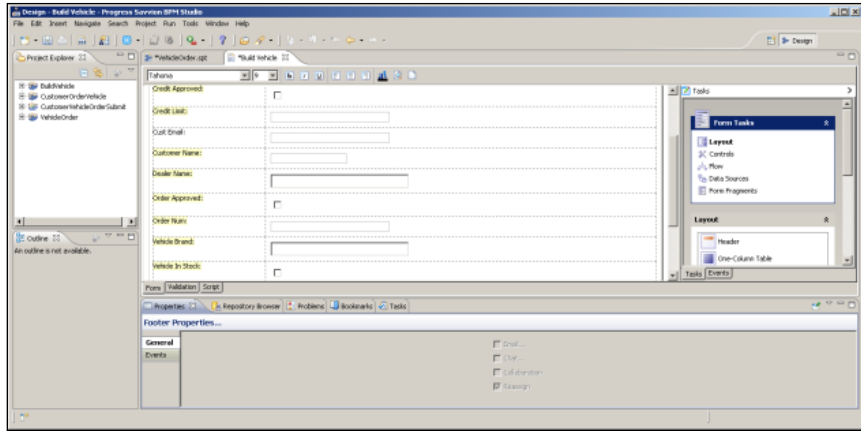


One thing you'll notice is that it may happen that if you make a change to the header properties in the property sheet you access from the Fields tab in the workstep

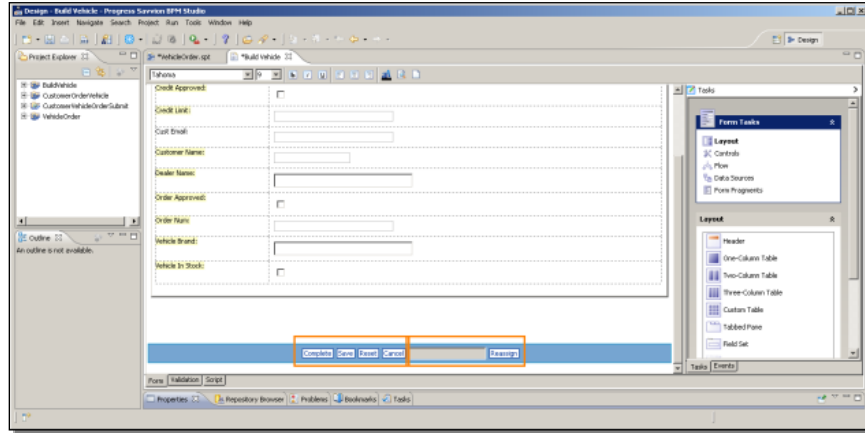
properties, the start date and due date may go away, so it's worth showing you that there is a separate header property list you can access from within the form. If I bring up the Properties View again, then you can see that you can check **Start** and **Due** dates back on, though you can't assign the Instructions text from here:



Now what about the footer? The footer has a basic property sheet here as well that you access by right-clicking on it, but as you can see in the Properties view, its basic properties are shown, but they're read only:

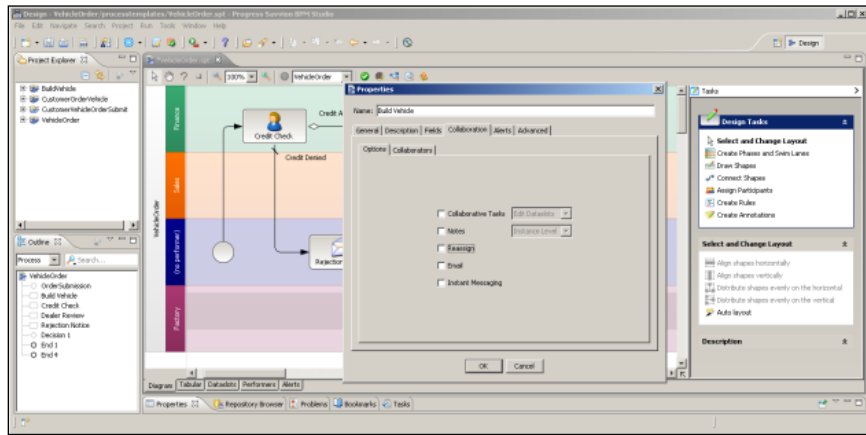


Once again the more complete property sheet is available through the workstep properties. Take a look at the default footer here:

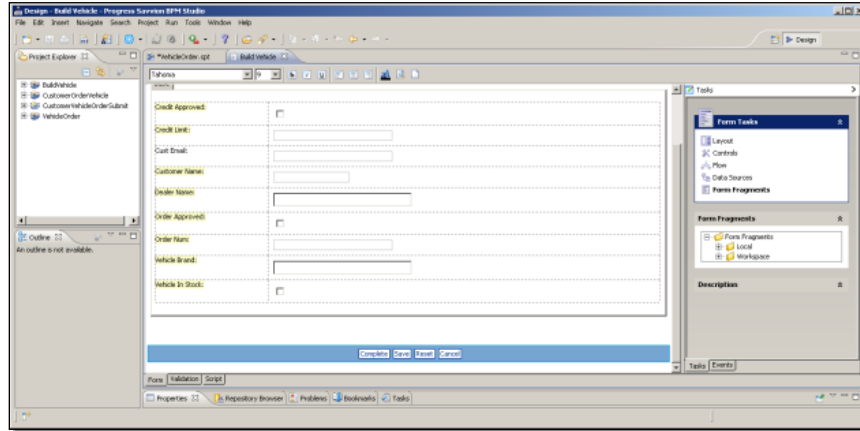


There are standard buttons to allow the user to deal with the form: to **Complete** it and signal the process to move on to the next step, to **Save** data entered so far without completing the step, to **Reset** displayed values that were already entered, or to **Cancel** altogether. The other elements you can put into the footer are identified as **Collaboration** options, ways that you can communicate with other Performers about what you're doing, or not doing, and how they should respond to that. As you can see, the default is a **Reassign** button with a text field to allow you to reassign this task step to another Performer. Let's take a look at how to set that.

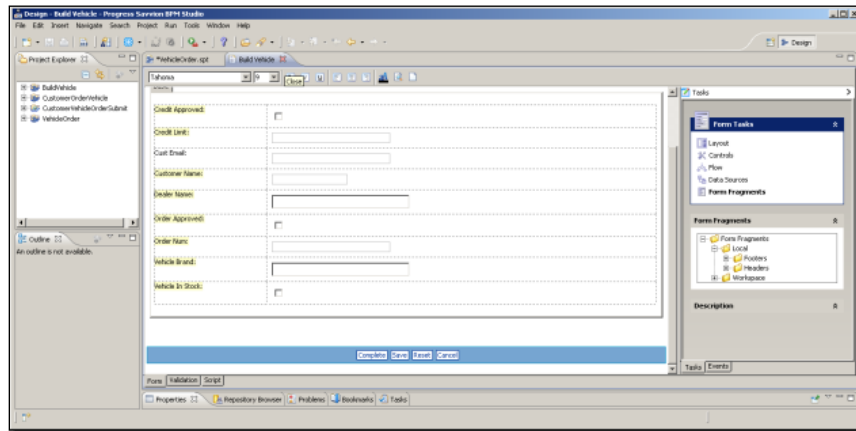
I'll save what I've got, and go back into the workstep properties. Next to the Fields tab there's a **Collaboration** tab. This lets you configure the footer contents. Here you can see several options for collaborating with other Performers, including email and instant messaging. I can check off the **Reassign** option, and go back and see what this looks like in the form:



I'll scroll down to see the footer, and you can see that the Reassign option is gone:



That's the basic way you can adjust the contents of the footer. There's a more flexible option for customizing both header and footer, though, which I'll just briefly point out to you. Under the **Form Tasks**, there's an entry named **Form Fragments**. If I just expand the **Local** group, you can see that it points to options for headers and footers. I won't go into how you configure this here, which you can find in the documentation, but once again, I just wanted to make sure you know what to look for.



This completes a brief tour of the principle features of the Savvion form builder. In the next video and paper, I show you how to use dynamic values in forms, including how to populate form values such as radio sets or combo boxes at runtime, with values retrieved from an OpenEdge application.