

Chapter 6

SharePoint and InfoPath

In this chapter:

- Inspecting logic and checking design of InfoPath forms
- Deploying InfoPath forms
- Populating multi-select list box in InfoPath 2007 web forms programmatically
- Working with InfoPath radio buttons and check boxes
- Copying file attachments from one section to another in InfoPath
- Managing older versions of InfoPath forms
- Creating a filtered column in InfoPath (data from a SharePoint list)
- Submitting InfoPath data to a SharePoint list
- Creating web enabled InfoPath forms
- Publishing InfoPath web form using stsadm command line utility
- Programming InfoPath web forms
- Populating InfoPath drop down with SharePoint list data
- Retrieving data from a SharePoint list
- Retrieving data from a SharePoint list using SPQuery
- Showing filtered items in InfoPath drop down

Inspecting Logic and Checking Design of InfoPath Form

It is easier to inspect the logic manually in simple forms but companies use InfoPath not to create simple forms but to create forms in an easy and simple way. These forms can get quite complex and large. Once developed and deployed, there are different techniques that developers can use to track down errors but even at design time inspecting and fixing errors in the form can be a tedious job. Forms can have hundreds of nodes or may be 20-30 fields in a single view. It's not easy to find problems with the data sources without the help of a tool. Unfortunately, logic

inspector and design checker, tools that come with InfoPath are usually overlooked and not taken advantage of by the developers.

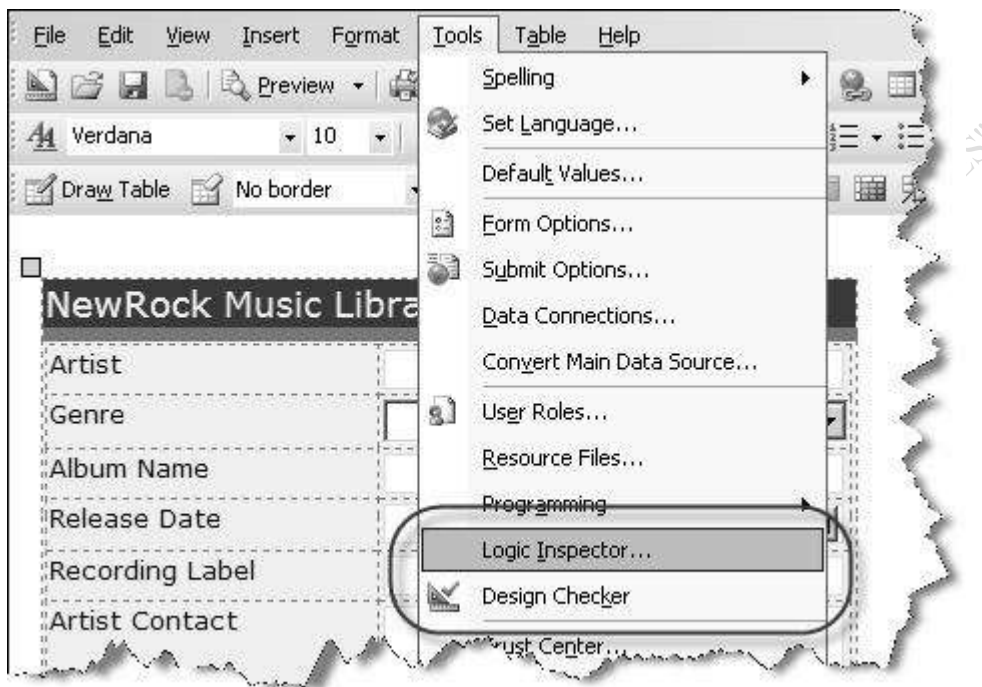


Figure 6.1: Logic Selector can be selected from the Tools menu

There are some errors that become evident only after when a developer tries to deploy the form. Large number of data sources or fields is not the only problem, no form is complete without the data validation checks, rules and business logic. Logic Inspector enables developers to find problems with the form before they try and deploy it on the destination server. It allows them to view the data validation checks, rules, calculated default values, and business logic, all in one place. Design Checker is another tool that can be used to check the form design for performance related issues before deploying the form. I admit, even I used to underestimate the power of this tool but It really helped me once to resolve performance issues in a large form that comprised of hundreds of nodes, validation checks, rules and thousands of lines of business logic code. Let's take a look at the Logic Inspector first before moving on towards the Design Checker.



Figure 6.2: *Logic Inspector inspects Data validation, Calculated Default Values, Rules and Programming*

Logic Inspector and Design Checker can be selected from the Tools menu. Design Checker is also available in the Design Tasks pane. When you work on a large form, it is very common to delete fields and/or nodes from the form and at the same time adding new nodes or fields in the form. This can easily inject problems in your form because many of the fields are interdependent on each other. One field may be dependent on a value in another field. Now suppose you remove the field that had data that was being used in another field. You will not get any error message but when you will try to deploy the form, it will fail. Now imagine you have to find the problem manually in the form that houses hundreds of nodes and 30-40 fields in each view and there may be more than one view in your form. It will be a tedious task. Logic Inspector comes to your rescue at this point.

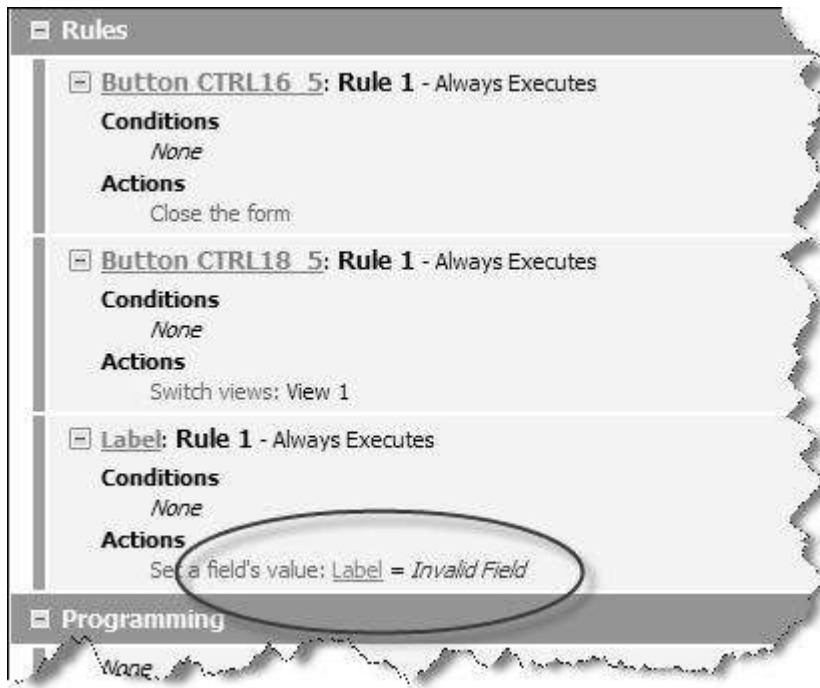


Figure 6.3: Logic Inspector catches broken rules

See Figure 6.3 above. "Invalid Field" message shown in red tells you that a field that was being used in a rule has been deleted. You will have to re-add the field, remove the rule or edit the rule in order to fix the problem. Your form will not publish unless you fix this problem.

You can run Logic Inspector to view the logic in the complete form or you can also invoke it to inspect logic in a single field.

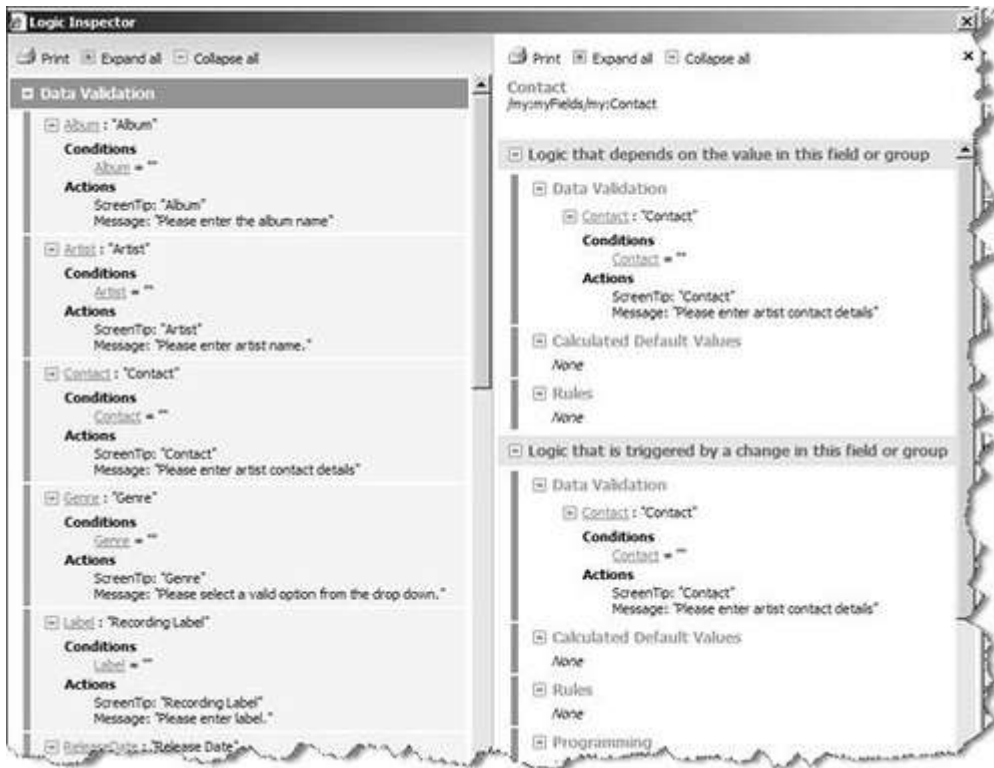


Figure 6.4: *Logic Inspector can be invoked for a single field*

To inspect a single field, right click the field and select "Logic Inspector". It will show you the complete details, for example, there will be a section called "Logic that depends on the value in this field group". There will be another one called "Logic that is triggered by a change in this field or group". (See Figure 6.4). You can also view logic for a single field by clicking the "Field Name" hyperlink. It will open another pane to the right that will show you the details.

Now let's turn towards the Design Checker. Design Checker is another very important tool that you should not miss when working on a complex form. Performance may not matter in case of small forms but it really matters when one is developing a complex form. Creating nested sections, nested tables, nested nodes is a very common scenario in complex forms. This is where developers usually make a mistake without thinking about the implications that will affect the form's performance. Especially if it's a web enabled form that will run in a browser, then of course you can not compromise on the form's performance. No one will know the difference when opening the form in the InfoPath client application but loading the same form in a web browser will show you why it is important to take care of the performance issues. Complex forms load very slow in browsers. It can take a complex form 2 to 20 seconds or may be even more to load in a browser. Well, that's a lot of time and you won't believe how annoying it becomes when the

form takes so much time to load. You can reduce this loading time by fine tuning your form and resolving the performance issues. One mistake that developers usually make is that they don't take care of the hierarchy in which they insert the nodes in the data source. This can have serious implications. For example, if you have added **fields** to a **section** then in the nodes' hierarchy, **fields'** nodes should come under the **section's** node. Look at the figure below to get a better understanding of what I mean.

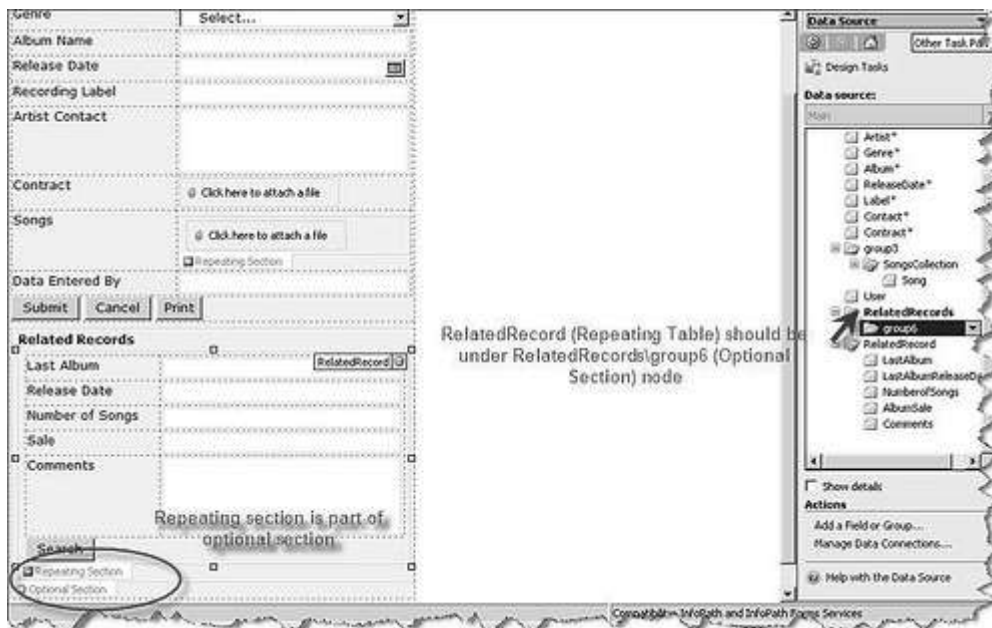


Figure 6.5: Locate node hierarchy inconsistencies with Design Checker

"RelatedRecord" is a repeating table and is a part of "RelatedRecords" which is a repeating section and we can see this in the design view of the form but in the data source, if you notice, the "RelatedRecord" comes under "myFields" and not under the "RelatedRecords" node. This will work. Preview the form and everything will work as expected. This is a very simple example. In real life scenarios, some times we have to use nested tables that can go up to 5 levels deep and in such cases if a developer does not take care of the hierarchy in which the fields are added to the data source then this can have serious implications as far as form's performance is concerned. Form will load very slow. If you want to read more about the performance related issues and tips on how to fine tune InfoPath forms, then there are a couple of very good articles available on Microsoft site:

Improving the Performance of InfoPath 2007 Forms (<http://msdn2.microsoft.com/en-us/library/bb380251.aspx>)

InfoPath Forms Services best practices (<http://technet2.microsoft.com/Office/en-us/library/0e9966df-e374-4df5-b3be-9848c78f9ca71033.msp?mfr=true>)

If your form has hundreds of data nodes then use Design Checker to check hierarchy related inconsistencies.

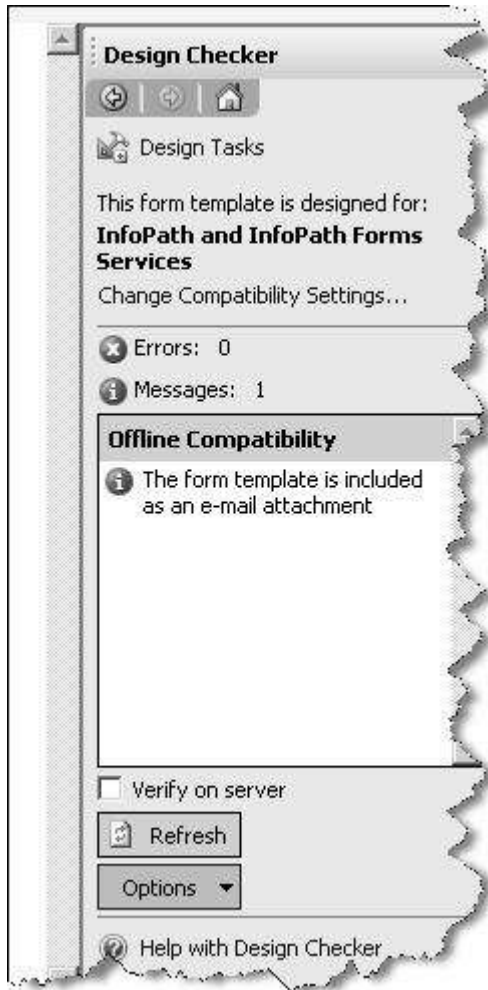


Figure 6.6: Design Checker Pane

All inconsistencies are shown in the "Offline Compatibility" section. To view a detailed message, click the message in this section. It will pop-up a message box that will show you the detailed message. Developers should always run Design Checker before publishing the form and should fix all issues reported by the Design Checker. Design Checker also helps in pinpointing "Online Compatibility" issues. Check the "Verify on server" option and click

"Refresh". If the publishing location is SharePoint then the Design Checker will verify if the destination site or library still exists or not. This can also help if there is an issue with any of your promoted fields. If you change the data type of a field in your form that was previously promoted to be used in a SharePoint forms library, and you forget to remove the previously promoted field and re-add it, your form will not publish and will give you an error. Again, as I said, if your form contains many fields then locating such issues becomes easier with Design Checker. Design Checker will show you the source of the error which you can then easily fix before trying to publish the form again. All in all, Design Checker is a handy tool and lots of headaches can be avoided if developers use these tools to find and fix the problems at design time.

We just saw why Logic Inspector and Design Checker are important tools for any professional developer working on complex InfoPath forms. I would also like to discuss two more tools very briefly that can be helpful in finding issues with complex forms. These may not offer help in finding the business logic issues but you can locate the javascript errors with these tools. One is Fiddler. Fiddler can be used to log all HTTP traffic between computer and Internet. Fiddler, when used with a browser, can be very useful in finding javascript issues with forms. Fiddler uses its own built-in inspector to analyze HTTP traffic.

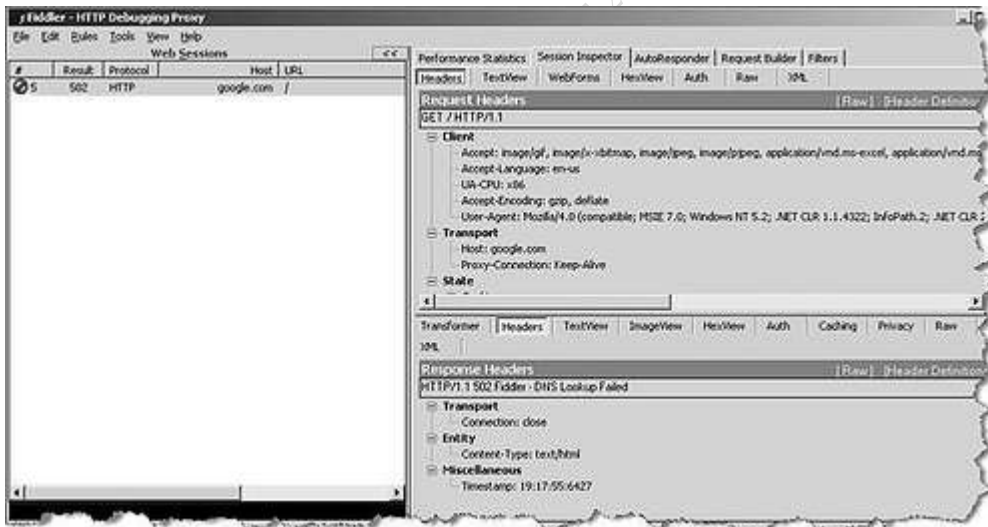


Figure 6.7: Catch HTTP and javascript errors with Fiddler

Another important tool is Internet Explorer Developer Toolbar (<http://www.microsoft.com/downloads/details.aspx?familyid=e59c3964-672d-4511-bb3e-2d5e1db91038&displaylang=en>). IE Developer Toolbar is a must have for every web developer. It is not meant for finding issues with InfoPath forms but because web enabled

InfoPath forms load and run in a browser, developers can use it to find any javascript related errors. It allows users to explore Document Object Model (DOM) of a web page.

And finally, if you developed a form using Visual Studio then you don't need any other debugger. Built-in Visual Studio debugger is enough to find business logic and javascript related errors in your web form. Javascript error will show you the filename that contains the source of the error. You can open that file in Visual Studio and put a break point in the function that you think contains the source of the error. Visual Studio will show the exact source and the error message that you can then fix easily.

SAMPLE CHAPTER FROM BOOK "SHAREPOINT 2007 TIPS, TRICKS AND TECHNIQUES"