



The crime scene: SharePoint forms, the death of InfoPath, and emerging of skybow Rich Forms

by Adis Jugo

| Director of Product Technology at sybow

| Microsoft MVP for Office Servers and Services and Microsoft MVP for Office Development

| Email: adis.jugo@skybow.com, twitter: @adisjugo

Abstract

One of the main intended uses of SharePoint Server was, from its very beginnings, development of custom solutions based on SharePoint platform. Let it be simple data entry forms, or workflow interactions and endpoints, or complex business solutions solely based on SharePoint, there always was a need for designing forms for data entry, editing and viewing. Standard SharePoint list forms are adequate only for very basic scenarios: layouts cannot be changed without too much tweaking under the hood, control fields do not have any dynamic behavior (cascading lookups, for example), and fields and forms data validation is present only on a very basic level. Different approaches have appeared during the time. on how to overcome this problem. One of the most common approaches was developing own web parts and application pages, and hosting it in SharePoint, to overcome those issues. This was a legitimate, and supported approach, but it did bring a few issues with it. For one, we were back to classical web development: developing own SharePoint artifacts, like web parts, meant that some of the SharePoint advantages – fast building, fast adoption – were going away. Furthermore, a lot has been said and written about SharePoint server side development: if it is not done properly, it can, and it will, decrease overall SharePoint performance. And, upgrade and migration were inevitably bringing issues on its own.

27.06.2016

skybow AG | <http://www.skybow.com/de/>



On the other hand, Microsoft has offered InfoPath (Forms Services), a proprietary server-side solution, as a part of the SharePoint's enterprise edition, to mitigate some of those problems. InfoPath has offered a decent forms designer, control behaviors and control validation to a degree, and a rudimental code backend (server side) for implementing actions on data changes. InfoPath was great for simple to medium-complex solutions which required forms, but it was lacking advanced data control and data processing features for anything more complex. It was often a case to ditch the InfoPath project in the middle, and to revert to developing custom web parts and application pages to replace it, after the developers realize the limitations of InfoPath. And then we were in the middle of good old classical development again. This, among some other things, has made SharePoint development so expensive.

When Microsoft ditched InfoPath, in January 2014, nobody was really surprised. InfoPath was one of the "legacy" technologies in SharePoint, fully server side, difficult to maintain, and even more difficult to migrate. With all of its limitations, and in the same time inability to develop server side code for fast emerging SharePoint Online from Office 365, it was clear that the replacement was needed. And it was needed fast.

In the same time Microsoft started working on the next generation forms solution, code-named FOSL (Forms On SharePoint Lists). This all has been happening in the middle of tectonic changes in the IT industry – more cloud, more client side, more mobile – which made FOSL obsolete even before it appeared. The planned FOSL release with SharePoint 2016 has been canceled, and InfoPath support has been extended until 2026. Does that mean that InfoPath is alive again? No. There is no active development of InfoPath, of any kind. With the extended support, Microsoft is telling the existing InfoPath customers, that they will not let them down, which is a good thing. But that shouldn't be understood, by any means, as a message that you



should continue developing with InfoPath. You shouldn't: it's a dead and hopelessly outdated technology.

So that does not leave us with many options there. Classical ASP.NET development (web parts, application pages) is not supported on Office 365, InfoPath is dead, and FOSL has been cancelled. Microsoft has left the field to the partners. And, sure, we have in the meantime seen the partner forms solutions emerging, some with server-side processing, some purely client side. Big BPM companies, like Nintex and K2, presented their own forms solutions, which are more meant to serve as endpoints for their workflows. Some client-side solutions focus on forms design, but completely neglect data processing, control behaviors and acting on data.

Until now, that is. skybow Rich Forms is an award-winning, feature rich, fully client side solution enabling solution developers to easily create forms, perform data bindings – including master-slave relations - precisely define control behavior, perform controls and forms validation, and create automated data-driven actions. Since it is completely developed with client side technologies like JavaScript, it can be used both on SharePoint Online and SharePoint Server (on premises), and the forms and form definitions can be easily migrated between online and on premises, between development, staging and production. skybow Rich Forms are the present, and the future, of SharePoint Forms development.

Form design and data binding

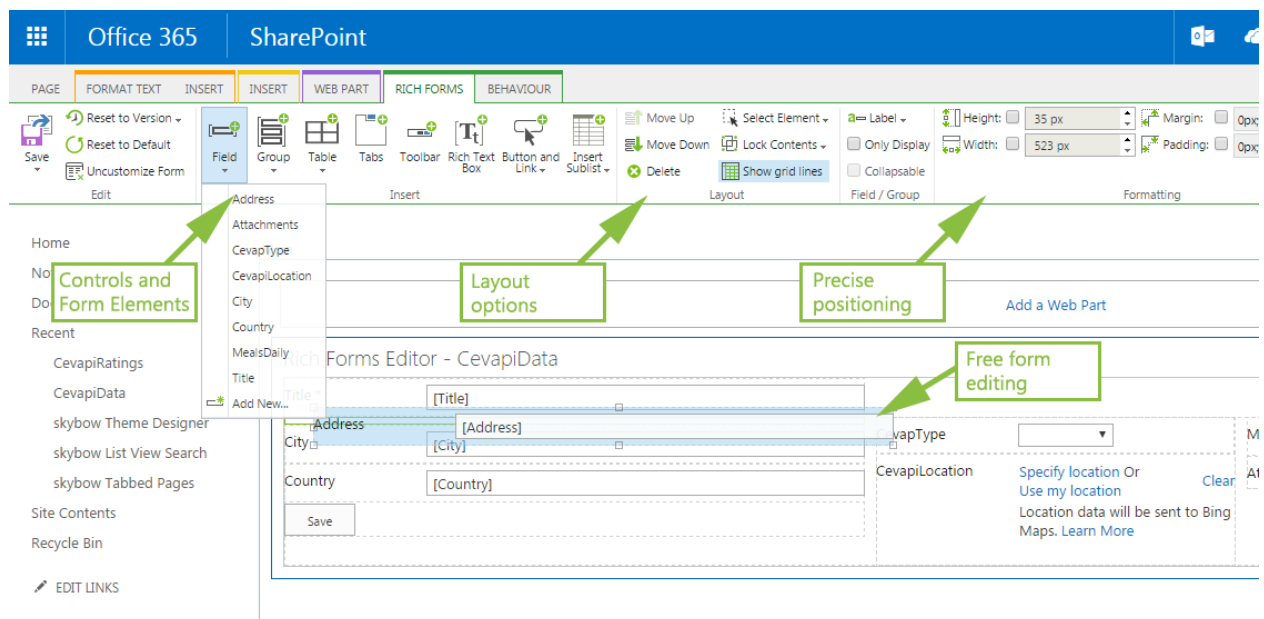
A modern forms solution needs to have a proper forms designer. In SharePoint world, that is even more important: for every list and library, we can modify three default forms – Add New, Edit and Display forms, and, furthermore, we might want to add a list-based form on a normal SharePoint page.

skybow Rich Forms gives us, through its landing page, an overview of the existing lists and libraries, and forms present on them. The forms which have already been customized are displayed in somewhat stronger colors, like the display form in this test list.

The screenshot shows the 'Rich Forms' interface. On the left, there's an 'Instructions & Support' section with 'Documentation' and 'Support' links. The central 'Configure Rich Forms' section displays a table with columns 'Recent', 'All', 'Lists', and 'Forms'. A list named 'CevapiData' is visible, with a 'New' button highlighted in green. A box labeled 'available forms' points to the 'New' button, and another box labeled 'available lists' points to the 'Lists' column. A 'Customize Form...' button is at the bottom. The right section, 'About this App', includes a 'Rate this App' button, a description of the app, and a testimonial from Daniel Wyss, the 'skybow App Builder'.

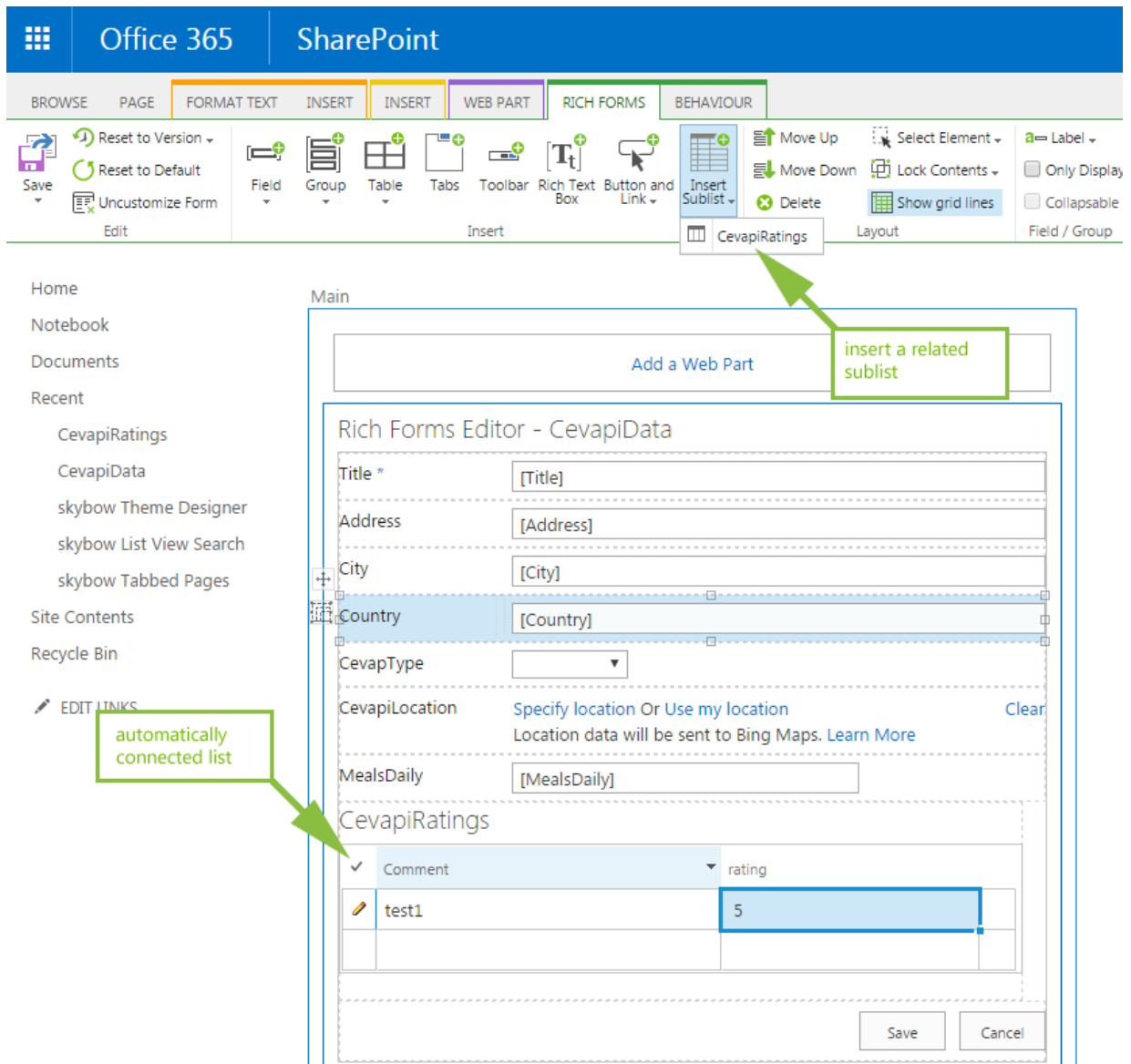
When we start creating a new form with skybow Rich Forms from the scratch – in this case the Add New form - we see a myriad of options. We can freely drag and drop and rearrange fields. We can even delete fields we don't need, like the "attachments" field in this case.

Using layout options and canvas containers – freely resizable horizontal and vertical groups, tables and tab pages - we can rearrange controls any way we want: in multiple columns, groups with resizing, accordion groups, etc. In comparison to InfoPath, here we have much more options to make the forms looking modern and being high performant.



Furthermore, creating master-slave forms with skybow Rich Forms is as simple as inserting related sublists from the ribbon bar. The sublist, in a form of quick-edit grid, will be placed into the selected container, and ready for further customization – resizing grid, fields, etc.

If you have multiple sublists for the list you are editing, of course you can place any, or all of them - beside each other, above each other, in the tabs, or however you might want. It is all about dropping a sublist into an appropriate container. Now, when you compare this with InfoPath repeating sections, this is much easier and faster, and much more straight forward to do.



Then again, creating a form does not stop with designing its layout and setting the control data bindings. In modern looking forms you would want to place some other controls or design elements. skybow Rich Forms enables you to place any available web part into any skybow Rich Forms container, and so to enrich the look and feel of your form. The possibilities there are countless. In the example in screenshot below, we are adding an Image Viewer, but on the same way you can add a Content Editor webpart and inject your own HTML or JavaScript, just the way you want it to be.

Office 365 SharePoint

BROWSE PAGE **FORMAT TEXT** INSERT INSERT WEB PART RICH FORMS BEHAVIOUR

Text Image Video and Audio Office 365 Video App Part Web Part

Categories Parts About the part

Media and Content

Image Viewer

Displays a specified image.

Add part to: Main

skybow Tabbed Pages

Site Contents

Recycle Bin

EDIT LINKS

Country [Country]

CevapType [dropdown]

CevapiLocation Specify location Or Use my location Clear

MealsDaily [MealsDaily]

Image Viewer

To link to an image, open the tool pane and then type a URL in the **Image Link** text box.

CevapiRatings

Comment rating

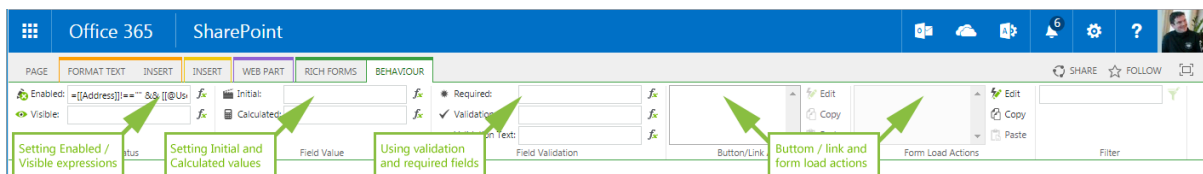
insert any webpart into the rich form

Forms controls behaviors

After the form layout is created, and databindings are set, you will probably want to create behaviors of the single form fields, and of the form as a whole. This was always an issue with InfoPath, since it's behavior customization capabilities did not go far enough for any more complex use.

This is where skybow Rich Forms excel. Clicking on the "Behavior" ribbon bar, there is a whole new world of controlling controls and form behaviors. You can use famous skybow expression language (based on JavaScript, but SharePoint context aware), to implement all required actions.

So you can, for example, set the specific form fields, or container controls, to be visible, or enabled, if the condition is met. Then – especially important for add new and edit forms – you can set the default and calculated value of the fields. Mind you, those are not SharePoint calculated fields we are talking about, this is fully-fledged JavaScript, where you can write the whole code blocks, including methods, and reference your own libraries. Still, all of that is context aware, what means you can use all SharePoint field data, and context information, like user-specific, security-specific, or site/sitecollection specific information. The same way, using the full power of JavaScript, you can validate fields, and set the proper validation messages for the users.



For example, in this screenshot below, we are setting a field to be visible only if the value of the "Address" field (in the same list item) is not empty, and the current user is



member of "skybow Demo Members" group. Of course, even the group name can be calculated, based on the values of the other fields. And so on: you see the power you can put into the field behaviors with skybow Rich Forms.

The skybow expression builder helps you to get the expressions faster and with less errors: on the right side, you have an easy access to the list fields and context variables, so it is, even for not seasoned SharePoint and JavaScript developers, relatively easy to put together meaningful expressions. Clicking on a "Test" button will evaluate the expression, and show you possible errors. In the design phase you cannot count with the real field values, that's why skybow Rich Forms allow you to set the test field values to test your expressions with.

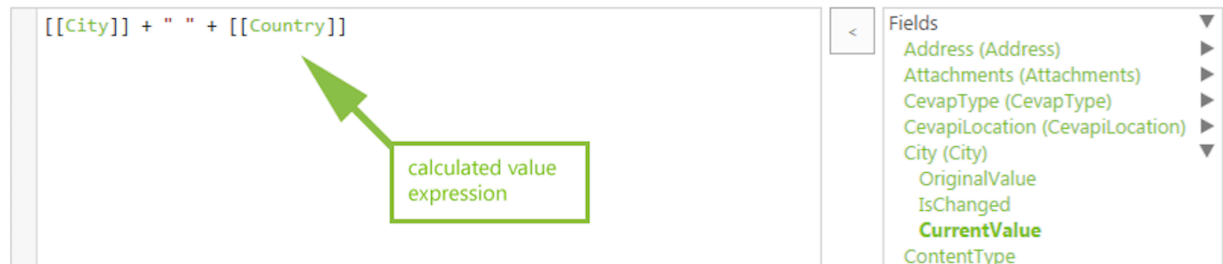
As mentioned, even if one-liners are the most often use cases in those expressions, nothing really stops you to paste and evaluate the whole JavaScript code blocs, or even use the methods from your own referenced JavaScript libraries. You can pass SharePoint fields values and context values as parameters to those methods. The possibilities are countless, really. By placing the controls into containers, and showing/hiding the whole containers based on expressions, you can create complex user-tailored views, which would show different portions of forms to different users, or user roles. It's that powerfull.

false"'. Below this is a form with fields: Title (Test 1), Address (Test 2), City, Country, CevapType (dropdown), CevapLocation (with a 'Specify location Or Use my location' link and a 'Clear' button), and MealsDaily. On the right, a 'Fields' list includes Address, Attachments, CevapType, CevapLocation, City, ContentType, Country, ID, MealsDaily, Title, and Context Objects (User, Email, Id, IsMemberOfGroup, LoginName, Profile, Title, Web, Site, Form, Page). Green callout boxes point to the expression, the toolbar, the evaluation banner, the context builder fields, and the fields list." data-bbox="144 105 911 575"/>

How simple it all is, shows the next example: The following expression is used as a calculated value behavior for the "Address" field. Simply by putting this line of code, the "Address" field will always have the value of connected fields "City" and "Country". So user never actually has to enter the "Address" field – it will be calculated. Now, put to that same field the value "false" for "Enabled" expression, so that user cannot even select it, you get how quickly you can implement powerful scenarios with skybow Rich Forms.

Expression Builder - Calculated Value

The field value will be calculated using the following Template expression:

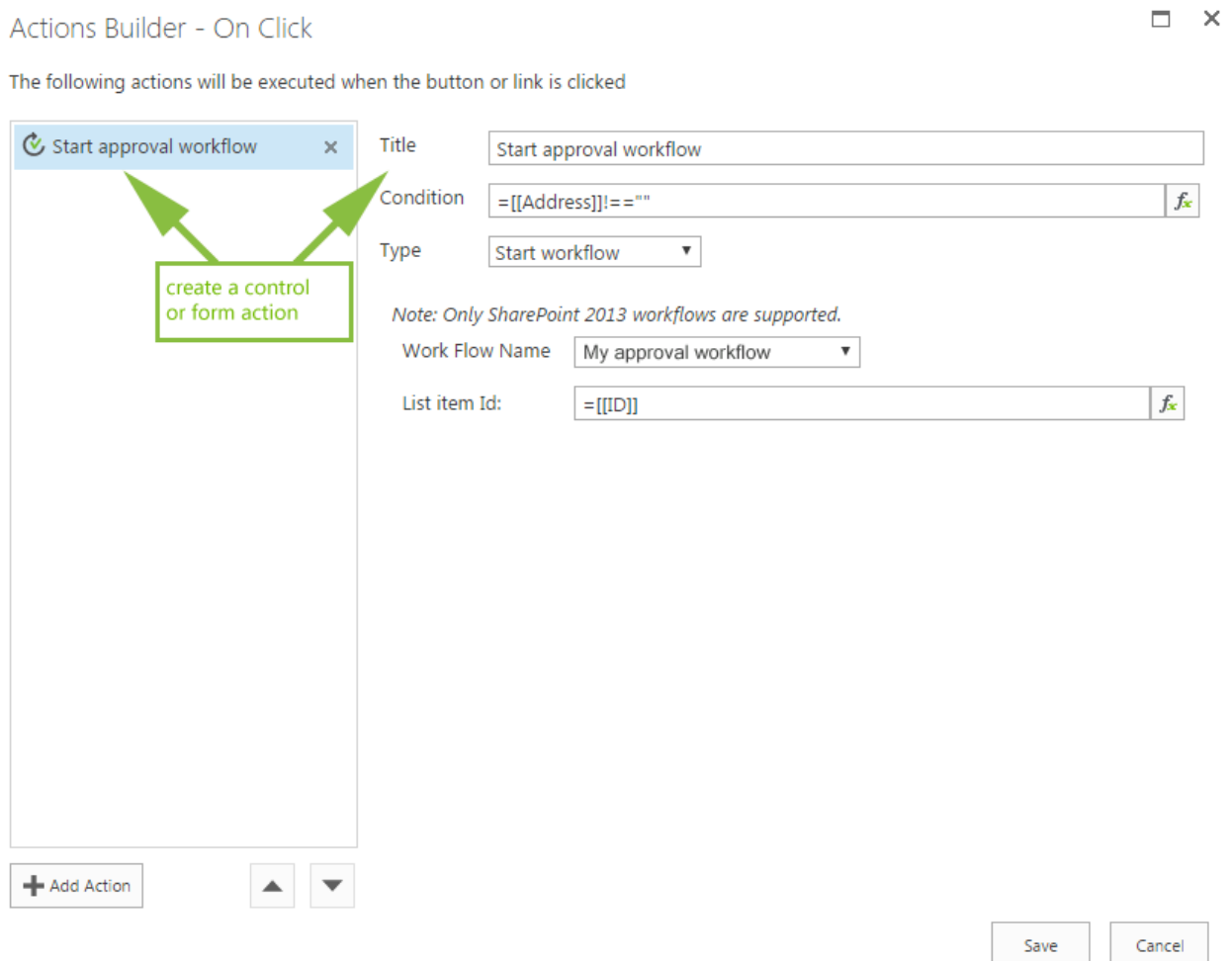


Acting on data changes

The next thing where skybow Rich Forms excel is acting on data. While in InfoPath you were limited to a small set of background processing options, which caused the forms to behave slow and unresponsive in the most of the cases, skybow Rich Forms offer a rich set of preconfigured actions that can be triggered. And if that is not enough, with making web services call and executing custom JavaScript code blocs, there are actually no limits on the customization capabilities.

So you can, after adding a button or link, define the actions behind those buttons. You can “silently” add, edit or delete a SharePoint List Item, without user ever even noticing it. You can open a List Form or any other page from your SharePoint, you can redirect the user to a custom Url, or reload page, which is very useful after saving a form. Yes, there is also an action to save the current form, if the default save button is not enough. You can set the form field values, send email messages using built-in SMTP support, show any message, start a workflow. If all if that is not enough, you can call web services by executing HTTP requests (for example, SharePoint REST Services), or even execute a custom JavaScript code block, where you can use JSOM – the JavaScript flavor of SharePoint Client Side Object Model. It is all possible.

With all of that, you can use skybow expression language to further define the action. In the following example, clicking on a button will start a workflow “My approval workflow” on the current list item, but only if the “Address” field is not empty. This expression for evaluating the condition can contain field values or SharePoint context information, like user or security information. So we can make the conditions like – start this workflow only if the current user is in the group X, and the field Y has value Z. This sort of things.



Actions Builder - On Click

The following actions will be executed when the button or link is clicked

- Start approval workflow

create a control or form action

Title: Start approval workflow

Condition: =[Address]!=""

Type: Start workflow

Note: Only SharePoint 2013 workflows are supported.

Work Flow Name: My approval workflow

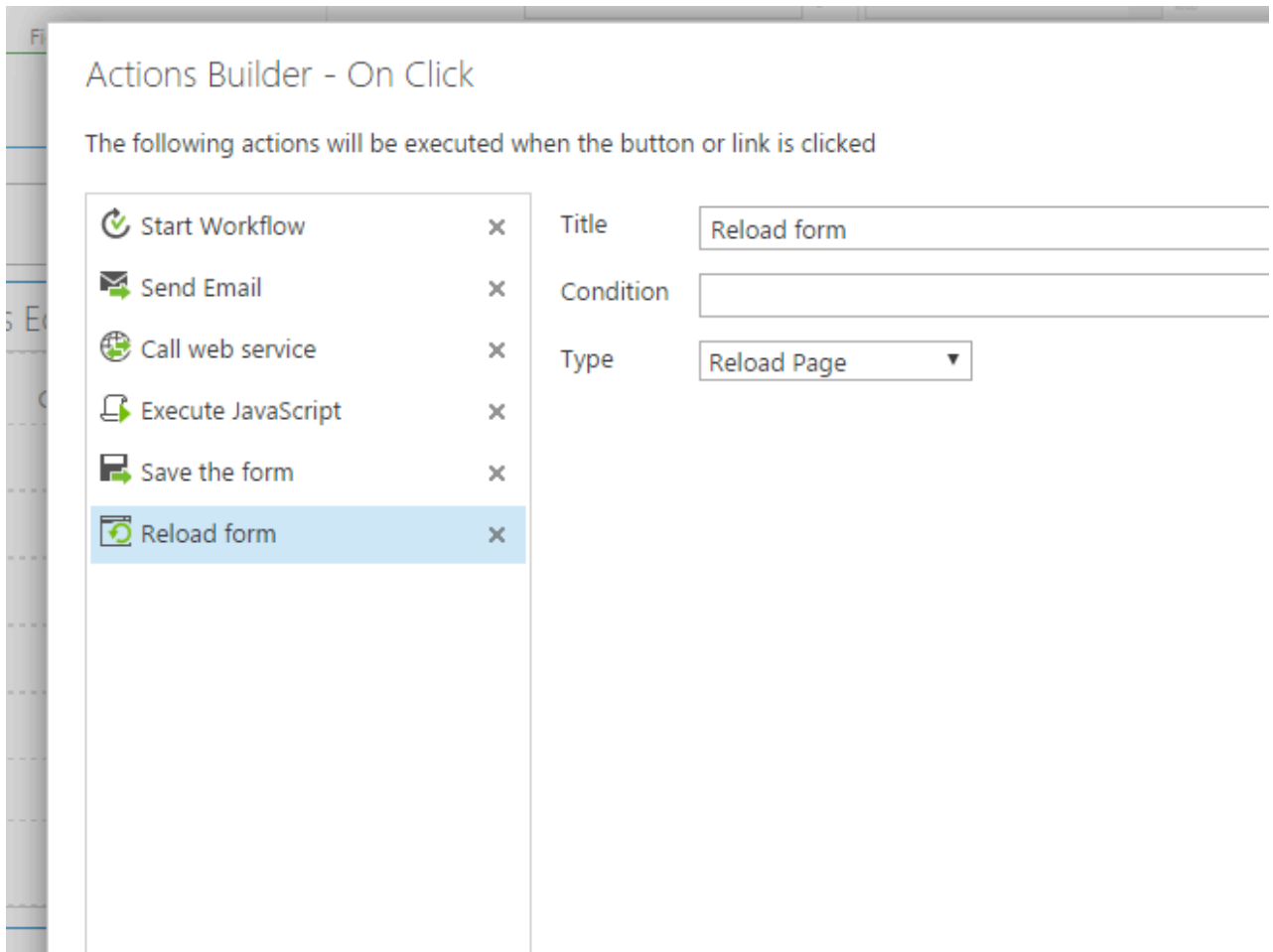
List item Id: =[ID]

+ Add Action

Save Cancel

You can staple actions in a single event. So clicking on a button will trigger multiple actions, in the order they are sorted. You can always reorder the actions, of course. In

this case below, clicking on a button will first start a workflow, then send an Email, call a web service, execute JavaScript, save the current form, and on the end reload the whole form.



Very important feature is also executing the same actions on the form load. We have got the same list of actions there, and they can also be stapled. So you can, for example, each time a form is loaded, check the data which are in form, or in the list, and based on that data start a workflow, or send an email. Or execute a JavaScript. This brings a whole new value to the page lifecycle, and opens the possibilities which were never present with InfoPath.