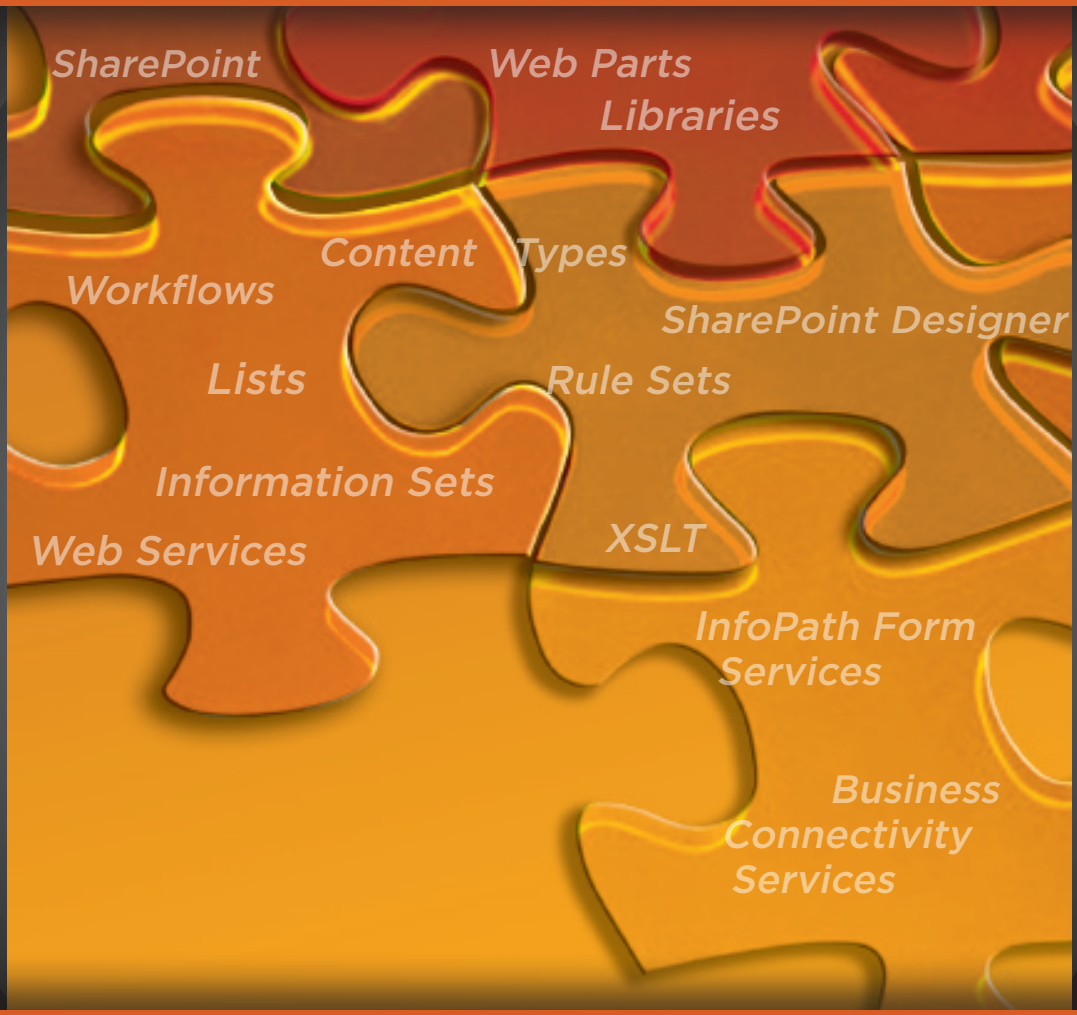
Enterprise Application Development in SharePoint 2010

— Creating an End-to-End Application without Code —



Ira Fuchs

ihf PUBLISHING

About the SharePoint 2010 Development Platform

While you could implement enterprise functionality using the 2007 versions of *SharePoint, InfoPath,* and *SharePoint Designer,* these tools were not sufficiently mature and integrated to do so in an efficient and elegant way. Working in the 2007 version of *SharePoint* required developers to construct compensatory workarounds and write custom procedural code to implement the enterprise application functions that *SharePoint 2010* can implement intrinsically and easily today. The developer experience in *SharePoint 2007* was adequate but far from optimal. This has changed dramatically in *SharePoint 2010*. The tighter integration between *SharePoint, InfoPath* and *SharePoint Designer* provides a more coherent developer experience and the most significant development limitations that prevented *SharePoint 2007* from being a first class development platform have been addressed in *SharePoint 2010*.

SharePoint 2010 is actually a multi-functional environment that incorporates a number of valuable product offerings that have been combined and integrated together. The *SharePoint 2010 Enterprise Edition* is comprised of the following platform services:

- **Portal Collaboration Services**
- **Web Content Publishing Services**
- **Enterprise Content Management Services**
- **PerformancePoint Services**
- **Search Services**
- **InfoPath Form Services**
- **Excel Services**
- **Access Services**
- **Visio Services**
- **Business Data Connectivity Services**
- **Managed Metadata Services**
- **Content Type Publishing**
- **Workflow Services**
- **Profile Services**
- **Office Web Applications**

This veritable cornucopia of enterprise functional value easily makes *SharePoint 2010* one of the most useful and leveraged products that any organization can deploy. It provides the broad capabilities to address almost any collaborative scenarios and operational requirements that an organization will encounter. In addition to these intrinsic services, the 2010 versions of *SharePoint Designer* and *InfoPath* complement and complete *SharePoint* as a development platform. *SharePoint Designer* is available as a free download and *InfoPath* is part of the *Office 2010 Professional Plus* suite. *InfoPath Form Services* provides the ability to render form templates created in the *InfoPath* design-time tool in a browser window and in Web Parts.

The *SharePoint 2010* tools and features that we will be primarily working with are *InfoPath, Business Data Connectivity Services* through *SharePoint Designer*, and *SharePoint Designer* workflows. The following is a description of these tools and their functions.

InfoPath and Forms Server

InfoPath is a tool for creating form applications that generate structured XML information. *InfoPath 2010* actualizes its enterprise deployment potential as a result of enhancements to the capabilities of *InfoPath Form Server* and by being more tightly integrated with *SharePoint 2010*. InfoPath has many capabilities that complement and enhance *SharePoint*, but there are three distinguishing attributes of *InfoPath* that stand out:

One, *InfoPath* is a model for rules based development exemplifying one of the best implementations of the declarative development paradigm. In *InfoPath* application logic is expressed as field operations and conditional statements which in turn drive validation, formatting and action events. Both information in the form and the behavior of the form itself can be manipulated using rules. As will be seen shortly, *InfoPath* rule sets are an elegant and efficient way to implement complex application logic.

Illustration 5 below shows the *InfoPath Rule Inspector*, used to view all the rule logic in a form. The fields used in rule operations and events are hyperlinked, providing a comprehensive view of all the rules that a field participates in.

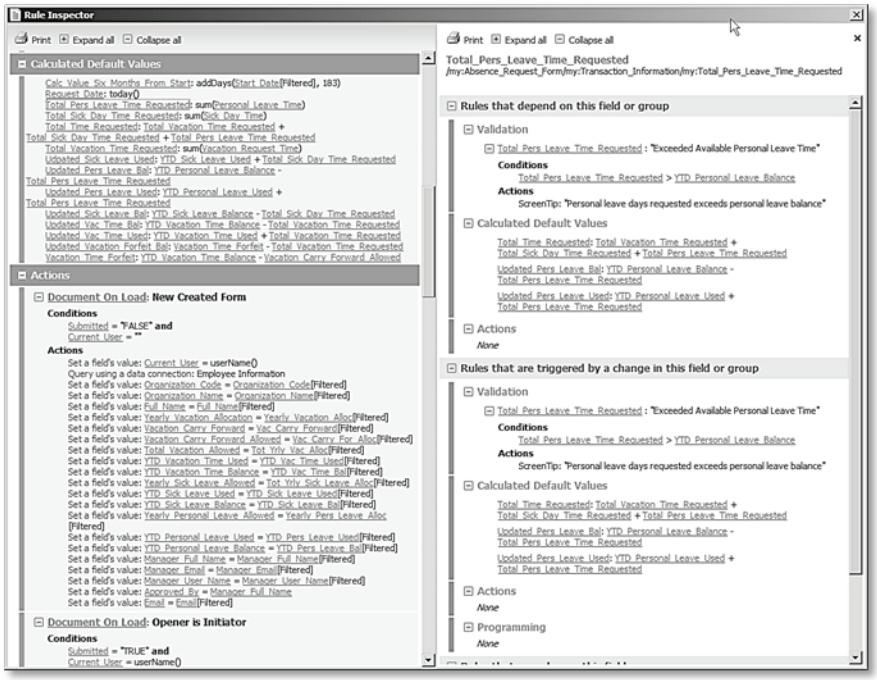


Illustration 5 - InfoPath Rule Inspector

Two, *InfoPath* forms can access an arbitrary number of data sources simultaneously and dynamically. The data sources can be Line of Business (LOB) applications, databases, Web Services and *SharePoint* lists and libraries. The information accessed from these data sources can interact with each other in the form and be manipulated in almost any way. For example, an *InfoPath* form can contain controls that consume financial

market data feeds that are published as web services. Rule based logic applied to the information captured by one form control generates the analytical output values that are used to query other data sources. In this manner the *InfoPath* form functions as an interactive analytical engine with sophisticated data gathering and presentation capabilities as shown in **Illustration 6** below.

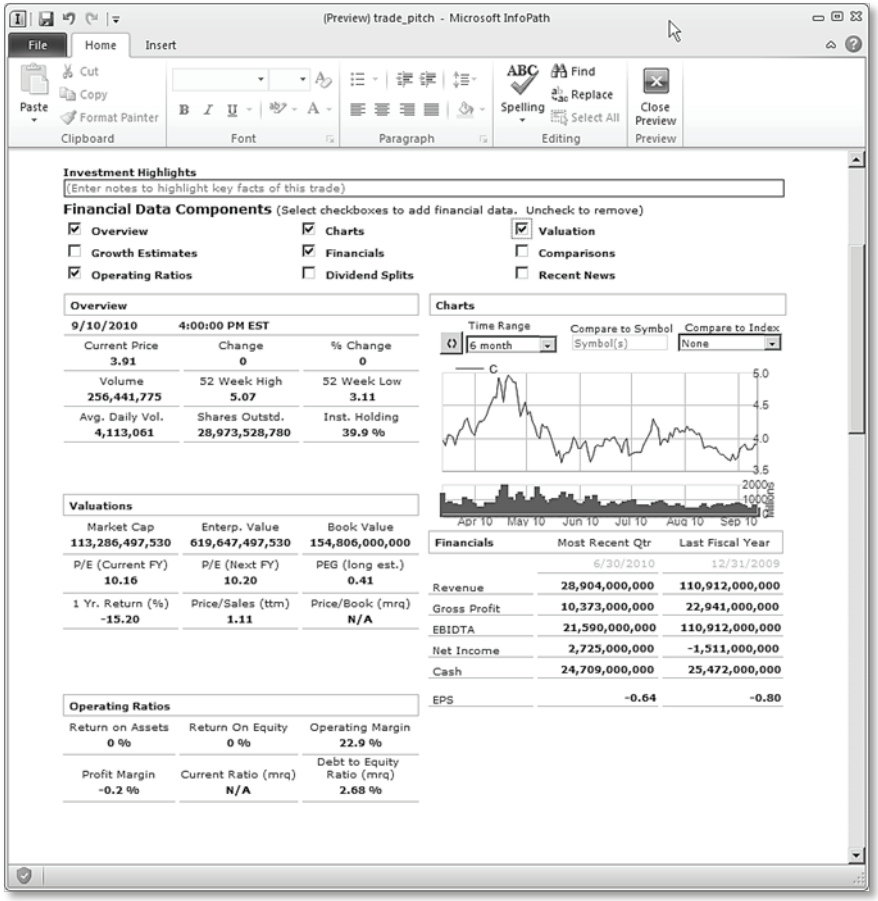


Illustration 6 - InfoPath form used for financial data gathering and presentation

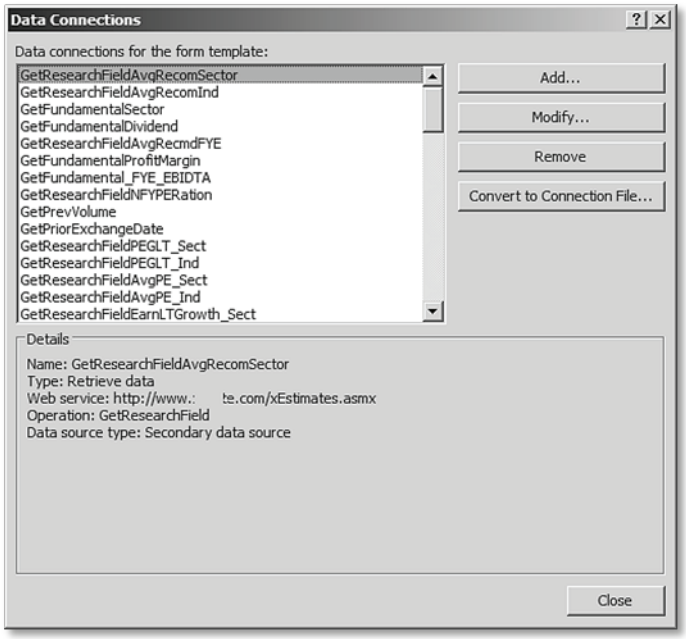


Illustration 7 - Web Services Data Connections for the financial data form

Illustration 7 at left shows the dozens of web service based data sources that this form accesses depending upon the criteria specified by the user or generated from rule logic.

Three, *InfoPath* uses *XML Schema* to represent the form's information set, and the controls used in the form to capture information are bound to the schema field nodes. The form generates an XML instance document of the captured information that is based on the organization and structure of the XML schema. **Illustration 8** below shows a section of the XML schema form fields on the right, and the corresponding form section and controls bound to the fields on the left.

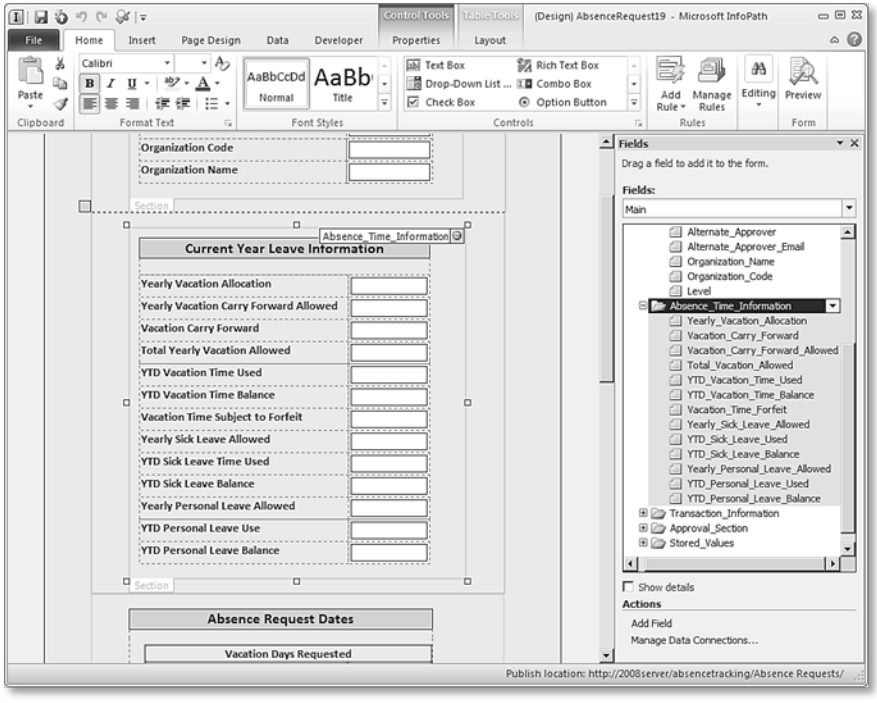


Illustration 8 - InfoPath schema segment bound to a form section

The value of creating XML documents that conform to a schema is that the meaning, function and use of the information in the document is comprehensible to and operable by any XML enabled application that can access the underlying schema for the document. Today, every industry specific initiative to develop a common vocabulary and set of procedures for the exchange and processing of information is based on *XML Schema*, as are the *Web Services* Protocols themselves. Using *InfoPath*'s *Submit* function it is easy to send XML data from the form to any number of receiving applications.

These three capabilities allow *InfoPath* browser forms to function as smart "client" applications capable of integrating, processing, and exchanging information with diverse and distributed data sources. This smart client aspect of *InfoPath* is what makes it a uniquely versatile and leveraged tool for building sophisticated enterprise applications.

As a form development tool *InfoPath* provides a built-in library of form controls such as drop-down lists, scrollable list boxes, calendar controls, check boxes, radio buttons, repeating tables, buttons, and numerous other controls, all of which can be customized through property attributes. *InfoPath* also supports repeating sections, optional sections, and dynamic conditional formatting, all of which facilitate a very high level of design and functional flexibility. Declarative rule sets can be bound to XML nodes in the schema or to controls and sections in the form that governs the form's behavior. This behavior is manifested in the following ways:

- Displaying multiple form views
- Constraining and validating the information that can be entered in form controls
- Auto-populating the form
- Displaying and populating controls and sections based on contingencies and dependencies
- Generating automatic, derived and computed values
- Invoking events, prompts, and instructions
- Multi-conditional cascading filters
- Accessing multiple data sources
- Submitting the form
- Opening and closing form behavior

The *Employee Absence Tracking* application that we will create will take full advantage of and demonstrate all of these capabilities and features.

InfoPath Forms Server is a *SharePoint Enterprise Edition* feature that allows *InfoPath* form templates (the form application that is created with *InfoPath*) to reside and run on a *SharePoint* server and renders *InfoPath* forms in a browser. *Forms Server* delivers two benefits that make *InfoPath* a viable platform for enterprise applications: one, the *InfoPath* program does not have to be installed on an end user's computer, and two, because the application logic runs server-side rather than on the client, the lifecycle management of an InfoPath form template is simplified. Prior to the arrival of *Forms Server* in *SharePoint 2007* the only way that *InfoPath* could be used widely in an organization was to distribute copies of the program and disseminate the form templates to users. If the form template was modified, it had to be re-distributed again. This was a deal breaker and accounted to a great degree for the general lack of *InfoPath* deployment in large organizations.

InfoPath Forms Server 2007 was typical of an initial version 1.0 product; not all of the *InfoPath* client functionality was supported in browser forms. Some of the missing capabilities were the very ones required to implement the sophisticated use case requirements of Enterprise applications. While it was possible to devise workarounds around these deficiencies, it created additional work for developers and necessitated using awkward methods. These feature gaps have been eliminated with *Forms Server 2010* and it is now possible to implement the same complex form behavior in a browser form that was previously available only in the *InfoPath* client.

Furthermore, *InfoPath* now provides a valuable service to *SharePoint* that is highly complementary and enabling: the default ASP forms that SharePoint generates for lists and workflows can now be replaced with *InfoPath* forms that take advantage of all the sophisticated form behavior capabilities itemized above. *SharePoint* automatically generates Active Server Pages (ASP) for adding, editing and deleting items in lists. Additional columns can be added to a list or library with the ability to assign a datatype (single line of text, number, date) or function (a look-up from another list or library, or calculation based on other columns in the form) to the column that in turn generates an ASP.NET control on the form page. However, there are no facilities for implementing complex field and form behavior, such as validation and auto-population in *SharePoint* ASP forms without writing code. This can now be accomplished using *InfoPath*'s declarative methods.

Illustration 9 below shows the design view of an *InfoPath* form for a *SharePoint* list used for issue tracking. The rich functionality that *InfoPath* brings to *SharePoint* lists provides capabilities for manipulating, processing and displaying list information that would have previously required a substantial amount of procedural coding to accomplish.

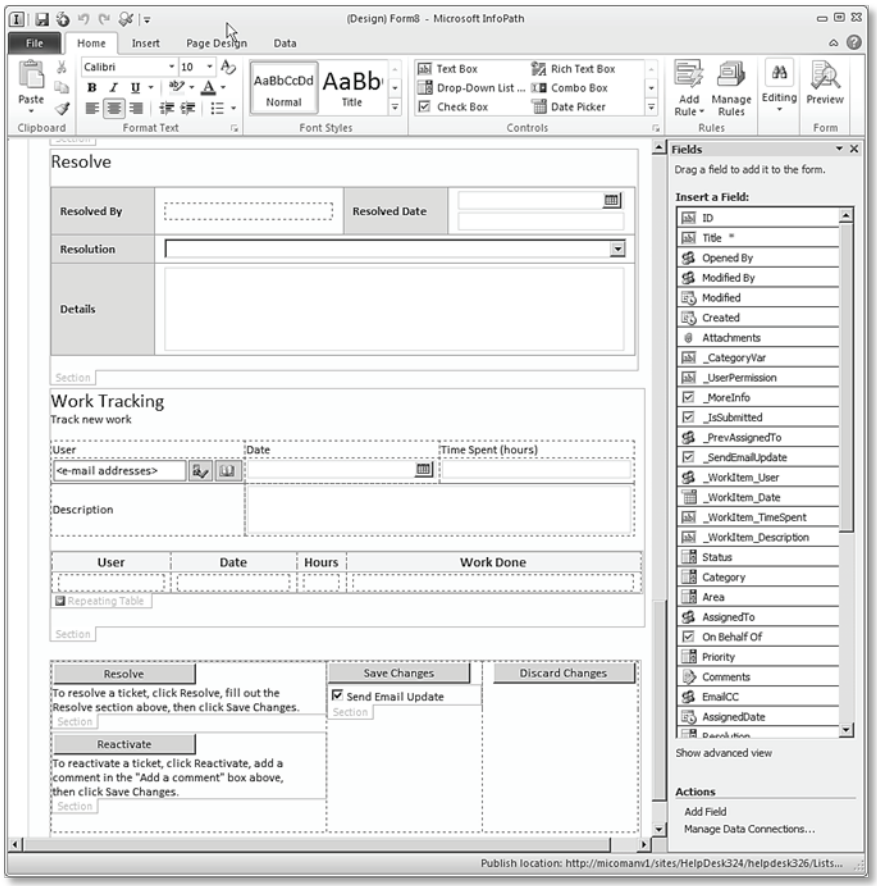


Illustration 9 - Design view of an InfoPath form for a SharePoint list

In addition, *SharePoint 2010* now has an *InfoPath Form Web Part*. You can simply insert one or more browser enabled *InfoPath* forms on any *SharePoint* page as Web Parts and interactively connect them together. This capability provides numerous options for accessing and displaying complex information sets on any *SharePoint* Web Part page. **Illustration 10** below shows an *InfoPath Form Web Part* used to display leave entitlement and usage information from an *SQL Server* database exposed through *Business Data Connectivity Services*.

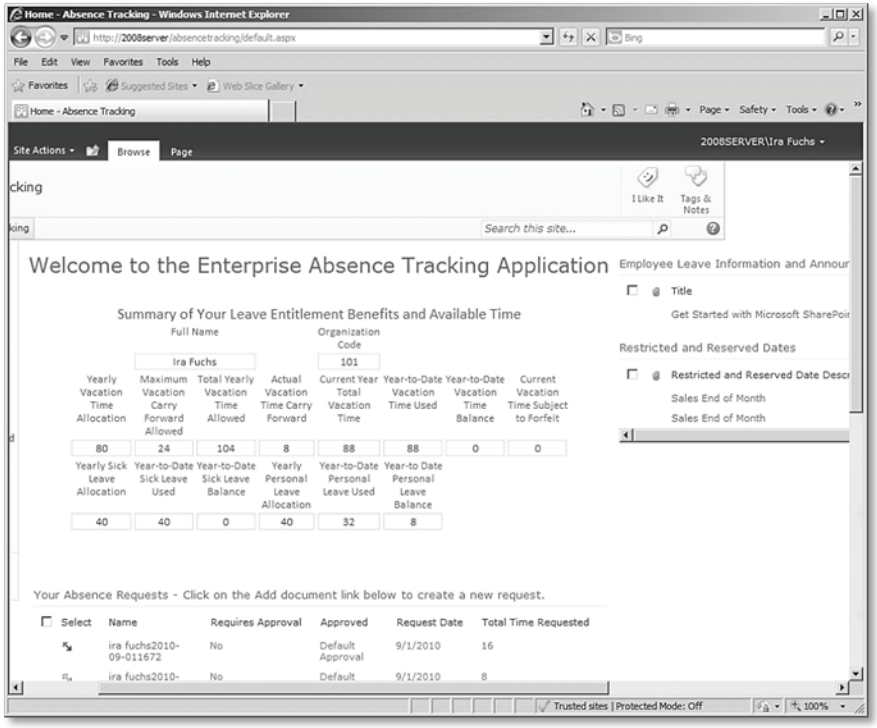


Illustration 10 - InfoPath Form Web Part accessing Line of Business information

Business Data Connectivity Services

Business Data Connectivity Services (BCS) is the new version of the *Business Data Catalog* that was originally introduced in *Microsoft Office SharePoint Server 2007*. It is now a feature of *Microsoft SharePoint Foundation 2010* with extended capabilities exposed in the *SharePoint 2010 Enterprise Edition*. BCS, through the *SharePoint Designer External Content Type* wizard, enables information from external Line of Business applications to be accessed by *SharePoint* and the *Microsoft Office* applications.

Business Data Connectivity Services uses two primary object definitions: *External Data Sources* and *External Content Types* (ECTs). *External Data Sources* are the Web Services, databases and other applications that BCS can connect to. *External Content Types* are the information sets accessed in the data sources as well as the operations that can be executed on those information sets, such as Create, Read, Update and Delete (CRUD) methods.

External Content Types are physically represented and displayed in *SharePoint 2010* as an "external list". An external list functions just like any other *SharePoint* list with all the attendant views and settings that *SharePoint* provides. The individual columns in an ECT list can also be incorporated in other *SharePoint* lists or libraries. In addition *External Content Types* can be defined so that the information set in an external source is directly accessible to *Microsoft Outlook, Word* and *Access* by mapping the ECT fields to *Microsoft Office* metadata objects.

SharePoint 2007 did not include tools for creating *Business Data Catalog* data source and ECT definitions. In *SharePoint 2010* both *SharePoint Designer* and *Visual Studio* include capabilities for creating and publishing BCS *Application Model* definitions. **Illustration 11** below shows the *SharePoint Designer* summary page for an *External Content Type*.

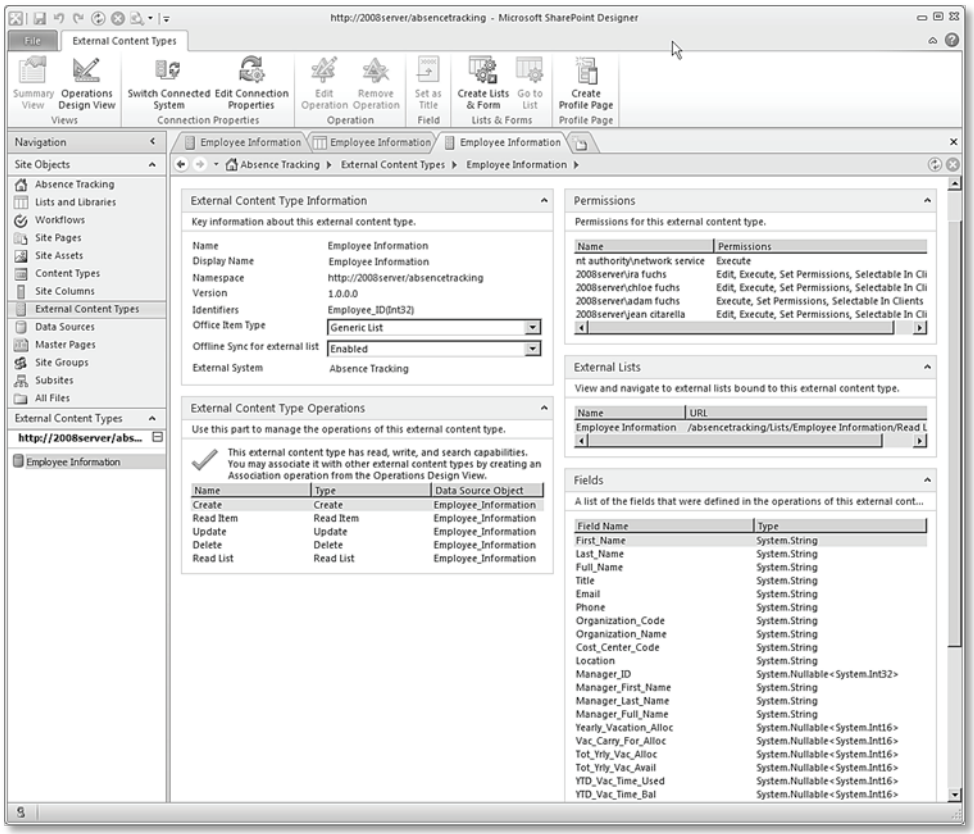


Illustration 11 - SharePoint Designer summary page for an External Content Type

Because of the substantial improvements in tools and functionality, *Business Data Connectivity Services* and *SharePoint Designer* now play a key role in the design and development of SharePoint enterprise applications. As we will demonstrate later in creating the *Employee Absence Tracking* application, *SharePoint Designer* not only provides significant efficiencies for connecting to and exposing the information and functions of external applications, but also provides capabilities for utilizing that information and functions in numerous ways.

SharePoint Designer

SharePoint Designer 2010 is a radical departure from its predecessor, *SharePoint Designer 2007*. For all intents and purposes *SharePoint Designer 2010* a completely new application with significantly enhanced capabilities to create, extend and manage *SharePoint* artifacts, as well as develop composite applications declaratively. It is the tool that ties everything together in the *SharePoint* platform, providing a development experience that is integrated, cohesive and conforms to development lifecycle best practices. There is an embarrassment of riches in *SharePoint Designer 2010* that makes it an indispensible tool for creating Enterprise applications in *SharePoint*. The user interface of *SharePoint Designer 2010* has been completely redesigned. Its governing design principle is to organize all *SharePoint* artifacts, as well as the tools to create, manipulate and integrate those artifacts, so that their relationships to each other are clearly presented and made accessible as efficiently as possible.

Every *SharePoint* artifact (site, list, library, workflow, content type, ECT) has a summary page that provides an aggregated view of all the subcomponents used by or related to that artifact, and allows the user to navigate to any summary pages for those objects. The summary page for a list will display the settings, views, forms, content type, custom actions, and workflows for that list. The summary page for a content type will reveal the forms and workflows for that content type. Any of the *SharePoint* operations and settings that can be applied to an artifact are contextually activated and made available on the summary page *Ribbon*, providing a highly efficient and flexible way of creating and managing SharePoint objects. *SharePoint Designer 2010* provides full artifact creation and editing capabilities; any function that can be executed using the SharePoint user interface can also be accomplished in *SharePoint Designer* without having to navigate between pages when using the *SharePoint* user interface. **Illustration 12** below shows how *SharePoint Designer* displays all of the lists and libraries in a site.

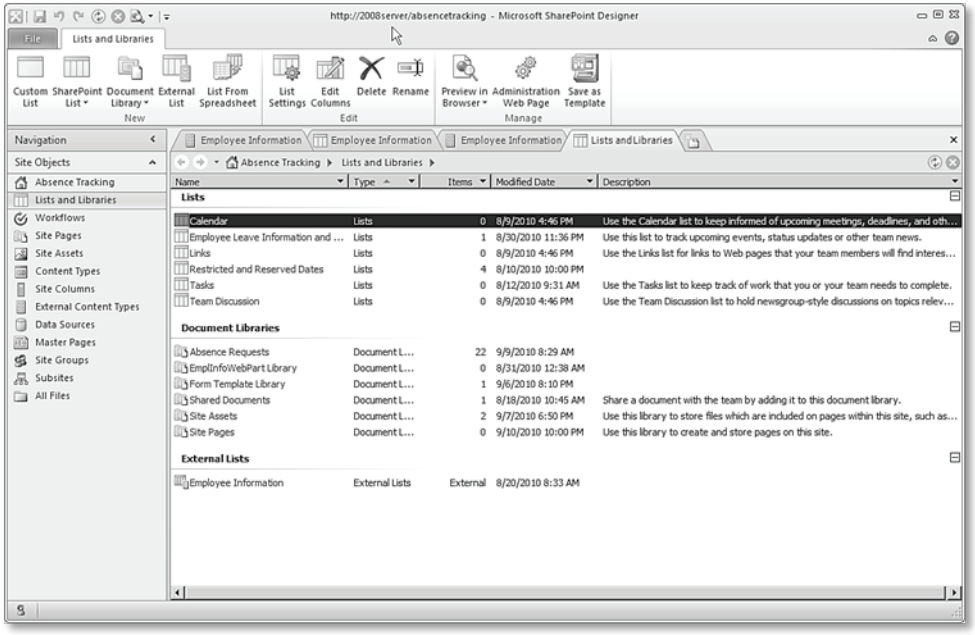


Illustration 12 – How SharePoint Designer displays all the lists and libraries in a Site

The *SharePoint Designer Navigation* pane organizes and presents the artifact categories for a *SharePoint* site collection or site. Clicking on an artifact category on the *Navigation* pane displays all the respective artifacts

for that category. Clicking on a specific artifact will display its summary page. **Illustration 13** below shows the *SharePoint Designer* summary page for a form library.

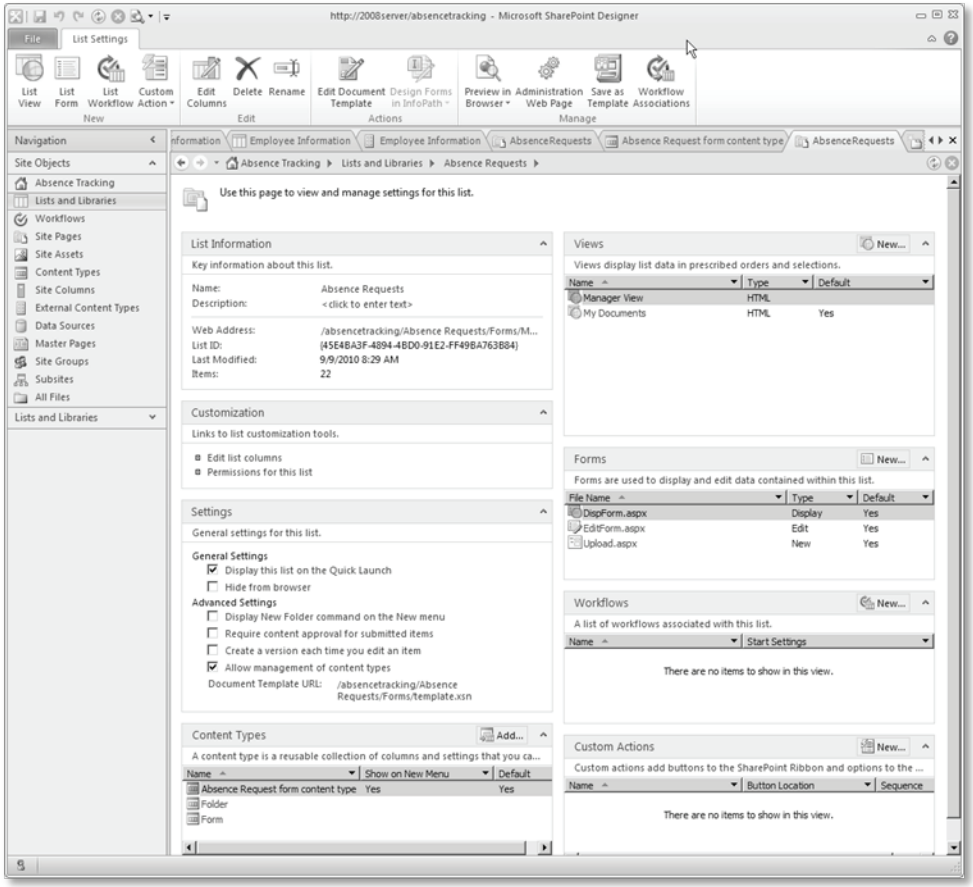


Illustration 13 - SharePoint Designer summary page for a library

SharePoint Designer now has full contextual *Ribbon* support. The *Ribbon* will display and activate the operational functions available at any given time based on the task being executed. The *Ribbon* significantly reduces the complexity of working with the numerous and rich capabilities of *SharePoint Designer*. **Illustration 14** below shows how the *Ribbon* menu in *SharePoint Designer* surfaces list and library functions when *Lists and Libraries* are selected on the *Navigation* pane.

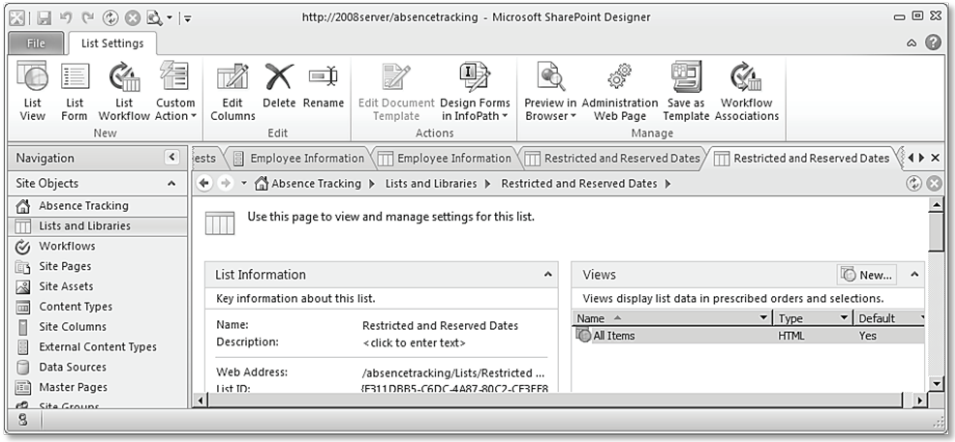


Illustration 14 – The Ribbon menu displaying the functions available for working with lists

As stated in the *InfoPath* overview earlier, the New, Display and Edit forms for list items, as well as any custom forms for a list, can now be created and modified in *InfoPath*. This can be accomplished directly from *SharePoint Designer 2010* from a list's summary page. The *Ribbon* on the summary page will activate a *Design Forms in InfoPath Ribbon* button.

By centralizing and integrating the declarative programming and configuration capabilities of *SharePoint* and *InfoPath* within *SharePoint Designer*, it now becomes a comprehensive and cohesive development console for creating *SharePoint* artifacts and applications. To further this design objective, *SharePoint Designer* has been enhanced with the following additional declarative development capabilities:

- Custom actions for list and libraries can now be created in *SharePoint Designer*. By simply clicking on the *New Custom Action* button on the Custom Actions section of the list or library's summary page a custom action can be defined to invoke opening of a form, initiate a workflow, or navigate to URL that can invoke any URL enabled function. These custom actions are represented as menu items on the *Ribbon* for the list or library or respective Web Part for the list or library. Custom actions make *SharePoint* applications easier to use by explicitly exposing the appropriate actions that can be executed for a list or library in the right context.

- *SharePoint Designer* now also includes the authoring capabilities for creating BCS artifacts, providing the facilities to connect to external data sources create external content types and publish them to the BCS instance running on a *SharePoint* server, as well as auto-generate the external list representation of the external content types and associated forms.

In addition to being able to create and modify list forms using *InfoPath*, *List Views* in *SharePoint 2010* have been reengineered as *Data Views*, which are essentially XSLT List View Web Parts or XLVs for short. XLVs

provide numerous features for interacting with list information, such as paging, filtering and sorting on column headers, and inline editing. And because standard *List Views* are now *Data Views* they can be completely customized with *SharePoint Designer*, using all of the *Data View* features for working with external data sources, conditional formatting, custom styles, and related item views that display parent-child relationships between multiple lists. What's more, after customizing an XLV in *SharePoint Designer*, you can still modify that view using all of the options in the browser, such as adding or removing columns. An XLV is fully customizable both in *SharePoint Designer* and in the browser.

Illustration 15 below shows a form library in *Edit* mode from the *SharePoint* user interface with a *Current User Filter Web Part* inserted on the page. This Web Part is used to create a *Managers View* of the library items that will display only those items where the logged-in user is the manager identified in the *Manager User Name* column. Because *List Views* are now *Data View* Web Parts there is unlimited possibilities for the way in which list or library information can be manipulated, presented or made to interact with any other information in a site.

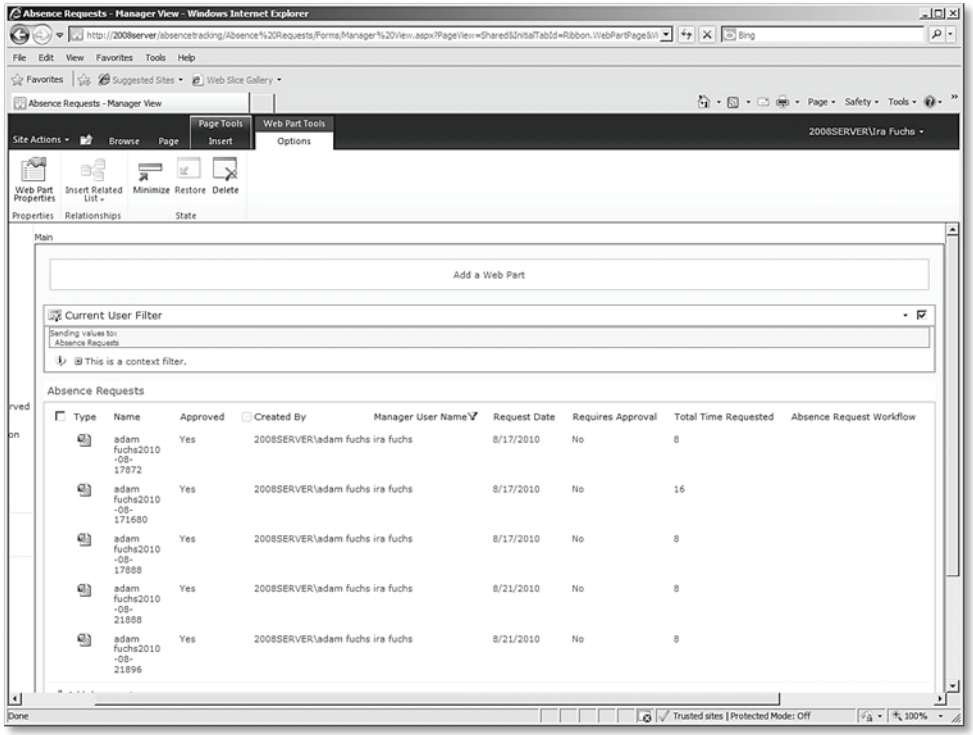


Illustration 15 - Form Library page in Edit mode showing a Current User Filter Web used to filter items for the current logged-In user

SharePoint Designer 2010 still has page editing capabilities but its primary emphasis is on creating and configuring *SharePoint* artifacts such as lists, libraries, workflows and ECTs. The experience of editing *SharePoint* Web Part pages now reflects this emphasis. Any of these artifacts can be placed and linked together inside the *main content placeholder* of a Web Part page, which is the only part of the Web Part page that is unlocked and editable in the default editing mode of *SharePoint Designer 2010*, making it significantly easier and straightforward to work with.

Illustration 16 below shows the same form library in Illustration 15 above, being edited in *SharePoint Designer*. Both the object and code view of the page are available for editing. With *SharePoint Designer* you can further customize the presentation or functional attributes of any *SharePoint* artifact.

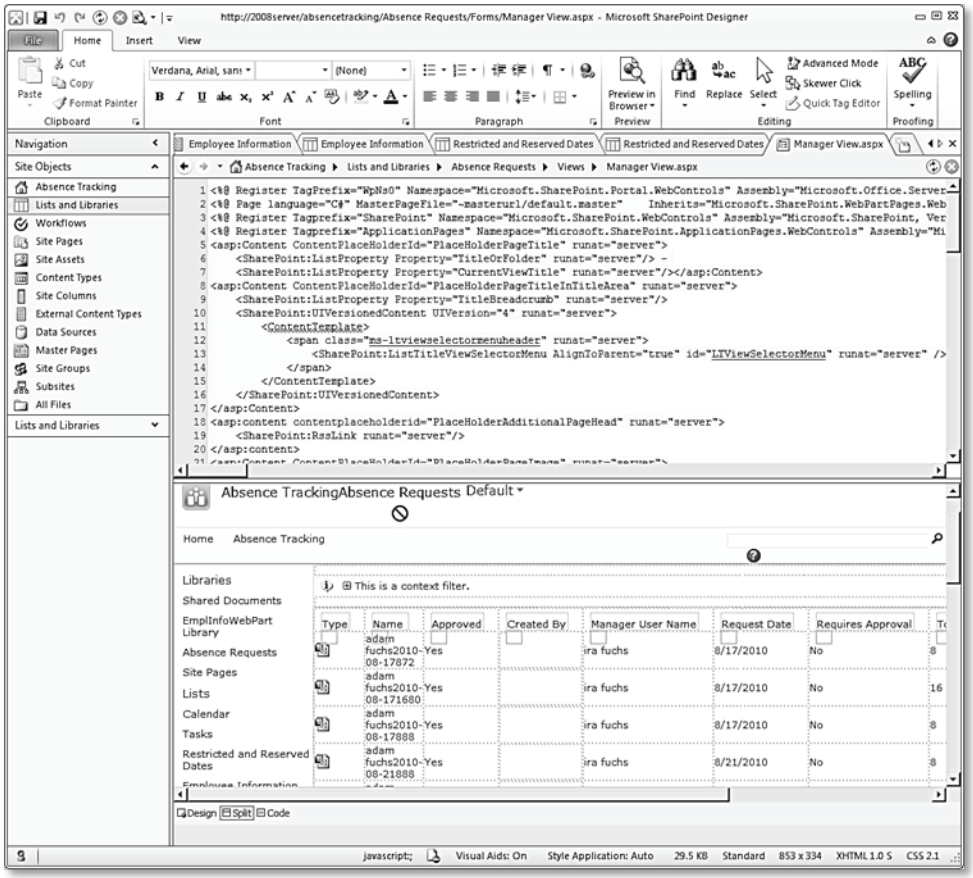


Illustration 16 - Form library page being edited in SharePoint Designer

SharePoint Workflows

Workflows are the mechanisms for executing formal processes. There are two general categories of workflow: human and document. Human workflow is the operational management of required responses by people to events in a process. Document workflow is the routing and processing of information in a process, and many organizational processes incorporate both.

SharePoint 2007 introduced an embedded workflow runtime engine, based on the *Windows Workflow Foundation* as well as several out-of-the-box common workflows (Approval, Collect Feedback, and Collect Signatures) which could be used with any list, library or content type. However these workflows could be not customized and had to be used as is, including their accompanying workflow forms. For more complex workflow requirements *SharePoint Designer* provided a workflow editor capable of creating workflows with multiple steps, with each step containing multiple conditions, actions and variables.

While the workflows that could be created in *SharePoint Designer 2007* were sophisticated enough to address a variety of use cases they were limited by being entirely bound to a specific list or library and not could not be reused. A *SharePoint Designer* workflow could not be saved and reapplied to another list, library or content type. This limitation was compounded by the fact that *SharePoint Designer* workflows could not be bound to content types either. Both of these limitations precluded flexible component reuse which is a fundamental tenet of enterprise application development.

The workflow development capabilities of *SharePoint Designer 2010* address these limitations and also include additional functional improvements that make it capable of addressing a wider range of human and document centric workflow scenarios. Any workflow created in *SharePoint Designer 2010* can be easily reused. A workflow created in the top-level site of a site collection is globally reusable; it can be bound to any list, library, or content type in the site collection. A workflow created in any site of the site collection can be reused in that particular site or a sub-site below it. *SharePoint Designer* workflows can now be bound to content types, either at the site collection level where any content types that inherit from the parent content type will include the workflow, or to just specific instances of a content type.

SharePoint Designer 2010 also introduces the ability to create a site workflow that is associated with the site itself not with a list, library or content type. In addition, a reusable *SharePoint Designer* workflow can be saved and exported as a *SharePoint Solution Package* (WSP). The workflow WSP file can be opened in *Visual Studio 2010* where it can be edited. A workflow WSP file can also be uploaded to a site collection on a different *SharePoint* farm where it can then be opened and modified by *SharePoint Designer*.

The workflow editor of *SharePoint Designer 2010* has also been improved with additional capabilities for sub steps, parallel actions, and a much broader set of workflow conditions and actions. The workflow editor is also context sensitive, displaying actions specific to a workflow association. If the workflow is bound to a document library the editor will activate a set of document library centric actions. Another feature of *SharePoint Designer* workflows is the ability to apply a workflow to a set of documents, so that a workflow action will iterate on all items in the document set.