Recall from reading the previous section "What Is the XSP Client Side JavaScript Object?" that there is a hierarchy and extension tree for the XSP Client Side JavaScript object, depending on the running platform. Therefore, Tables 4.2 and 4.3 list all public and private XSP Client Side JavaScript object functions from the XSP web, mobile, Notes, Composite Application, IBM Mashup Center, and Debug extensions (IBM Mashup Center is not part of the Notes/Domino product family, but is an IBM WebSphere offering). The first column of each table gives specifics about availability within each platform. Where a function is available within all platforms, it is specified within square brackets as **All**. Otherwise, the available platforms are listed individually. Supported platforms are **Web**, **Mobile**, **Notes**, **CA** (denoting composite application), and **MU** (denoting IBM Mashup Center). The description column also includes the name of the source JavaScript file, for your convenience.

Table 4.2 Summary of the Public XSP Client Side JavaScript Object Functions

Function Name	**Description**
XSP.alert(message) [All]	Displays a generic alert dialog. Standard dialog in a web browser/mobile device, and RCP dialog in the Notes client. This is an overridable extension point. [xspClientDojo.js]
XSP.confirm(message) [All]	Displays a generic confirm dialog, with OK and Cancel buttons. Standard dialog in a web browser/mobile device, and RCP dialog in the Notes client. This is an overridable extension point. [xspClientDojo.js]
XSP.error(message) [All]	Displays a generic error dialog. This is an overridable extension point. [xspClientDojo.js]
XSP.prompt(message, defaultCaption) [All]	Displays a generic prompt dialog containing the given message, along with the `defaultCaption`, if specified. The `defaultCaption` value appears within the prompt dialog's edit box, enabling the user to input a value that returns when the dialog is closed. This is an overridable extension point. [xspClientDojo.js]
XSP.djRequire(name) [All]	Loads a Dojo module into the context of the current page. Use this only when you do not want to include the required module in aggregated resources. Otherwise, use `dojo.require()`. [xspClientDojo.js]

Function Name	**Description**
XSP.addPreSubmit Listener(formId, listener, clientId, scriptId) [All]	This function adds a custom Client Side JavaScript function, termed the listener, to a queue of zero or more other listeners. This queue of listeners gets executed just before the page is submitted to the server side. The `formId` parameter must specify the ID of the current form the listener will be triggered against on submission—this is a mandatory parameter. The `listener` parameter is a function reference to a custom Client Side JavaScript function—this is a mandatory parameter. The `clientId` parameter can specify the client-side fully namespaced ID of any single `eventHandler` or button control within the current page. This causes the presubmit listener to trigger only when the specified `clientId eventHandler` or button is invoked. This parameter is optional but should at least be specified as `null`. When set to `null`, the presubmit listener is triggered when any `eventHandler` or button tries to submit the current page. Also note that the custom client-side listener function does not need to return any result because this is ignored in the processing of the presubmit listener queue. [xspClientDojo.js]
XSP.addQuerySubmit Listener(formId, listener, clientId, scriptId) [All]	Adds a custom Client Side JavaScript function, termed the listener, to a queue of zero or more other listeners. This queue of listeners is executed when the page tries to submit to the server side. Based on the return results from any `querysubmit` listener functions in the queue, submission can proceed or be stopped. The `formId` parameter must specify the ID of the current form the listener will be triggered against on submission—this is a mandatory parameter. The `listener` parameter is a function reference to a custom Client Side JavaScript function—this is a mandatory parameter. The `clientId` parameter can specify the client-side fully namespaced ID of any single `eventHandler` or button control within the current page. This causes the `querysubmit` listener to trigger only when the specified `clientId eventHandler` or button is invoked. This parameter is optional but should at least be specified as `null`. When set to `null`, the `querysubmit` listener is triggered when any `eventHandler` or button tries to submit the current page. Also note that the custom client-side listener function should return a Boolean result because this is used in the processing of the `querysubmit` listener queue to either allow or stop page submission. [xspClientDojo.js]

Table 4.2 Summary of the Public XSP Client Side JavaScript Object Functions (cont'd)

Function Name	**Description**
XSP.canSubmit() [All]	Should be called before any page submission, to prevent page resubmission, either accidentally when a user double-clicks on a link or if the user is impatient compared to the expected time for a page refresh. If the page has recently been submitted, it returns `false` and the submission should be abandoned. Otherwise, it assumes that the submission will occur and updates the `lastSubmit` value, the date stamp of the last submission. If the page is not submitted after this is called, `XSP.allowSubmit()` should be invoked so that further user actions can submit the page. The time allowed between submissions is configured through the `submitLatency` variable. Avoid using this function—rare use cases exist when you need to use this function, such as if Client Side JavaScript performs page submission, as with `document.forms[0].submit()`. Therefore, controlling page submission using `XSP.canSubmit()` can provide more robust page behavior. [xspClientDojo.js]
XSP.allowSubmit() [All]	If the page is not submitted after a call to `canSubmit()`, this should be invoked to re-enable page submission. Avoid using this function—rare use cases exist when you need to use this function, such as if Client Side JavaScript performs page submission, as with `document.forms[0].submit()`. Therefore, controlling page submission using `XSP.allowSubmit()` can provide more robust page behavior. [xspClientDojo.js]
XSP.setSubmitValue (submitValue) [All]	The `SubmitValue` property is the value sent to the XPages server and is available from the context object. This function usually can be called while processing a client-side event, right before the event is submitted to the server. It can be used, for example, for passing component-related data. [xspClientDojo.js]
XSP.getSubmitValue() [All]	The `SubmitValue` property is the value sent to the XPages server and is available from the context object. This function can be called while processing a client-side event, usually right before the event is submitted to the server. [xspClientDojo.js]

Function Name	**Description**
XSP.validateAll(formId, valmode, execId) [All]	Runs client-side converters and validators before page submission to the server. The `formId` parameter must specify the form to validate. This is useful in a multiform page. The `valmode` integer parameter must specify a value of `0`, `1`, or `2`. `0` = No validation. `1` = Run converters only. `2` = Run Converters and Validators. The `execId` string parameter is optional and can be used to specify a single control within the current page to validate. If `null`, the whole page is validated. [xspClientDojo.js]
XSP.getFieldValue(node) [All]	Returns the value of the given HTML DOM node parameter. A string value is returned for a single value node, or a comma-separated string is returned for a multiple-value node (such as an option control). [xspClientDojo.js]
XSP.getDijitFieldValue(dj) [All]	Returns the value of the given Dijit instance, based on the existence of the `dijit.getValue()` function. [xspClientDojo.js]
XSP.validationError(clientId, message) [All]	An overridable extension point similar to the `XSP.alert()` and other dialog-based functions. This gives developers an opportunity to provide a custom error display to the end user. The default behavior is to display an error message dialog. The `clientId` string parameter must specify the ID of the failing control. The `message` string parameter is used to relay the failing message or warning. [xspClientDojo.js]
XSP.scrollWindow(x, y) [All]	Scrolls the current window contents to the specified x and y integer coordinates. [xspClientDojo.js]

Table 4.2 Summary of the Public XSP Client Side JavaScript Object Functions (cont'd)

Function Name	**Description**
XSP. partialRefreshGet(refreshId, options) [All]	Programmatically executes `GET`-based partial refresh requests to the XPages server-side runtime. The `refreshId` string parameter value must specify a fully namespaced HTML DOM element ID. This is the target receiver of the partial refresh response content. The `options` parameter is optional and used to send custom parameters to the server side as `GET` request parameters. It can also be used to specify `onStart`, `onError`, and `onComplete` event function callbacks. These are triggered accordingly during the request lifecycle. [xspClientDojo.js]
XSP. partialRefreshPost(refreshId, options) [All]	Programmatically executes `POST`-based partial refresh requests to the XPages server-side runtime. The `refreshId` string parameter value must specify a fully namespaced HTML DOM element ID. This is the target receiver of the partial refresh response content. The `options` parameter is optional and used to send custom parameters to the server side as `POST` request parameters. It can also be used to specify `onStart`, `onError`, and `onComplete` event function callbacks. These are triggered accordingly during the request lifecycle. An `immediate` request parameter can also be included in the options content to control validation execution. [xspClientDojo.js]
XSP.attachClientFunction (targetClientId, _event, clientSideScriptName) [All]	Connects a client-side function to an event on an XPages control. [xspClientDojo.js]
XSP.attachClientScript(target ClientId, _event, clientScript) [All]	This function is used to connect a client-side script call to an event on an XPages control. [xspClientDojo.js]
XSP.addOnLoad(listener) [All]	Attaches a Client Side JavaScript function to the current page's `onLoad` event. The `listener` function-reference parameter is used to specify the Client Side JavaScript function. [xspClientDojo.js]
XSP.showSection(sectionId, show) [All]	Toggles the expanded state of the specified section control using the client-side `sectionId` string parameter value and `show` Boolean parameter value. [xspClientDojo.js]

Function Name	**Description**
XSP.findForm(nodeOrId) [All]	Returns the parent form element for the given node or client-side element ID. [xspClientDojo.js]
XSP. findParentByTag(nodeOrId, tag) [All]	Returns the nearest parent element matching the specified `tag` string parameter value. [xspClientDojo.js]
XSP. getElementById(elementId) [All]	Retrieves an element from the current HTML DOM based on the specified `elementId` string parameter value. Note that the `elementId` represents the fully namespaced client-side element ID. [xspClientDojo.js]
XSP.hasDijit() [All]	Determines the presence of any Dijit objects and the `dijit.byId()` function within the current HTML page. [xspClientDojo.js]
XSP.trim(s) [All]	Returns the `s` string parameter value, trimmed of leading and trailing whitespace. [xspClientDojo.js]
XSP.startsWith(s, prefix) [All]	Returns a Boolean value indicating whether the given `s` string begins with the specified prefix string value. [xspClientDojo.js]
XSP.endsWith(s, suffix) [All]	Returns a Boolean value indicating whether the given `s` string ends with the specified suffix string value. [xspClientDojo.js]
XSP.toJson(o) [All]	Converts an object to a String serialization of that object. [xspClientDojo.js]
XSP.fromJson(s) [All]	Parses a JSON string to return a JavaScript object. [xspClientDojo.js]
XSP.log(message) [Web/MU]	Opens a new browser window, with the given message parameter value written into the window contents. [xspClientDojo.js]

Table 4.2 Summary of the Public XSP Client Side JavaScript Object Functions (cont'd)

Function Name	**Description**
XSP.dumpObject(obj, options) [Web/MU]	Returns a list of property/value pairs available on the given `obj` parameter. [xspClientDebug.js]

Table 4.3 Summary of the Private XSP Client Side JavaScript Object Functions

Function Name	**Description**
XSP.getMessage(key) [All]	Retrieves a translated string from the localized XSP client-side locale bundles. [xspClientDojo.js]
XSP._pushListener(listeners, formId, clientId, scriptId, listener) [All]	Used internally by the `XSP.addPreSubmit Listener` and `addQuerySubmitListener` functions and others. [xspClientDojo.js]
XSP._SubmitListener(formId, listener, clientId, scriptId) [All]	Used internally by the `XSP.addPreSubmit Listener` and `addQuerySubmitListener` functions and others. [xspClientDojo.js]
XSP._processListeners(listeners, formId, clientId) [All]	Processes an array of listeners, either the `querySubmit` or `preSubmit` listeners. When processing the `querySubmit` listeners, it stops at the first listener that returns `false`. [xspClientDojo.js]
XSP.attachValidator(clientId, required, converter, validator1, ..., multipleValueSeparatorString)	Connects a control with any converter and validation objects specified for that control. Calls to this function are automatically generated by the XPages runtime. This is a private function. [xspClientDojo.js]
XSP._Validator(clientId, required, converter, validatorList, multiSep) [All]	Used internally by the `XSP.attachValidator()` function. This is a private function. [xspClientDojo.js]

Function Name	**Description**
XSP.DateConverter(dateFormat, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP.TimeConverter(timeFormat, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP. DateTimeConverter(dateFormat, timeFormat, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP.IntConverter(message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, then there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]

Table 4.3 Summary of the Private XSP Client Side JavaScript Object Functions (cont'd)

Function Name	**Description**
XSP.NumberConverter(dot, tho, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP.RequiredValidator (message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP.DateTimeRangeValidator (minTime, maxTime, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP.LengthValidator(min, max, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, then there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]

Function Name	**Description**
XSP. NumberRangeValidator(min, max, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP.RegExpValidator(expr, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP.ExpressionValidator(expr, message) [All]	Instances of this object are generated by the server-side converter. You should never need to explicitly invoke this method in Client Side JavaScript yourself. If you found a use case, then there would be cross-site scripting vulnerabilities because all client-side validation should have matching server-side validation, to prevent invalid data from being accepted when the browser is compromised. This is a private function. [xspClientDojo.js]
XSP.attachEvent(clientId, targetClientId, _event, clientSideScriptName, submit, valmode, execId) [All]	Connects an `eventHandler` to a control in the client side. Therefore, when the specified event is triggered in the client side against the `targetClientId` control, a request is issued to the server-side component to trigger any corresponding server-side `eventHandler` code or simple actions. This is a private function. [xspClientDojo.js]
XSP._getEventData(targetNode, targetId, eventName) [All]	Used internally by the `XSP.attachEvent()` function and others. This is a private function. [xspClientDojo.js]

Table 4.3 Summary of the Private XSP Client Side JavaScript Object Functions (cont'd)

Function Name	**Description**
XSP.fireEvent(evt, clientId, targetId, clientSideScriptName, submit, valmode, execId) [All]	Used internally by the `XSP.attachEvent()` function and others. This is a private function. [xspClientDojo.js]
XSP._doFireEvent(evt, form, clientId, clientSideScriptName, submit, valmode, execId) [All]	Used internally by the `XSP.attachEvent()` function and others. This is a private function. [xspClientDojo.js]
XSP._scrollPosition() [All]	Used internally by the `XSP.scrollWindow()` function and others. This is a private function. [xspClientDojo.js]
XSP._setAllowDirtySubmit(flag) [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModified-Flag`. This is a private function. [xspClientDojo.js]
XSP._isAllowDirtySubmit() [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModifiedFlag`. This is a private function. [xspClientDojo.js]
XSP._setDirty(flag, formId) [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModifiedFlag`. This is a private function. [xspClientDojo.js]
XSP._isDirty() [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModifiedFlag`. This is a private function. [xspClientDojo.js]
XSP._getDirtyFormId() [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModifiedFlag`. This is a private function. [xspClientDojo.js]

Function Name	**Description**
XSP.attachDirtyListener(clientId) [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModifiedFlag`. This is a private function. [xspClientDojo.js]
XSP.attachDirtyUnloadListener (saveMessage) [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModified-Flag` and `modifiedMessage`. This is a private function. [xspClientDojo.js]
XSP._validateDirtyForm(formId, clientId) [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModi-fiedFlag`. This is a private function. [xspClientDojo.js]
XSP._saveDirtyForm(evt, cli-entId, targetId, clientSideScript-Name, submit, valmode, execId) [All]	Used internally by the Dirty Save feature—see the `<xp:view>` properties for `enableModified-Flag`. This is a private function. [xspClientDojo.js]
XSP._doFireSaveEvent(evt, form, clientId, clientSideScript-Name, submit, valmode, execId) [All]	Used internally by the Dirty Save feature to save any data sources on the current page—see the`<xp:view>` properties for `enableModified-Flag`. This is a private function. [xspClientDojo.js]
XSP.attachPartial(clientId, targetId, execId, eventName, scriptName, valmode, refreshId, onStart, onComplete, onError) [All]	Used to connect a partial refresh–enabled `eventHandler` to a control in the client side. Therefore, when the specified event is triggered in the client side against the `targetId` control, a partial refresh request is issued to the server-side component to trigger any corresponding server-side `eventHandler` code or simple actions. This is a private function. [xspClientDojo.js]
XSP.firePartial(evt, clientId, targetId, execId, scriptName, val-mode, refreshId, onStart, onCom-plete, onError) [All]	Used internally by the `XSP.attachPartial()` `/partialRefreshGet/partialRefreshPost` functions. This is a private function. [xspClientDojo.js]

Table 4.3 Summary of the Private XSP Client Side JavaScript Object Functions (cont'd)

Function Name	**Description**
XSP._partialRefresh(method, form, refreshId, options) [All]	Used internally by the `XSP.attachPartial()` /`partialRefreshGet`/`partialRefreshPost` functions. This is a private function. [xspClientDojo.js]
XSP._replaceNode(refreshId, content) [All]	Used internally by the `XSP.firePartial()` function and others. This is a private function. [xspClientDojo.js]
XSP.processScripts(s, ex) [All]	Used internally by the XSP object function and others. This is a private function. [xspClientDojo.js]
XSP.execScripts(a) [All]	Used internally by the XSP object function and others. This is a private function. [xspClientDojo.js]
XSP.parseDojo(node) [All]	Used internally by the XSP object function and others. This is a private function. [xspClientDojo.js]
XSP.attachSimpleConfirm Submit(clientId, targetClientId, _event, message) [All]	Used to connect a confirm handler to the current page. Generated by the Confirm Simple Action. This is a private function. [xspClientDojo.js]
XSP.tagCloudSliderOnChange (sliderValue, sliderId) [All]	Used by the Tag Cloud Custom Control in the Discussion Template to control the visualization of the expanding/contracting values. This is a private function. [xspClientDojo.js]
XSP._loaded() [All]	Used internally by the XSP Object when a page loads. It is used in several page load and submission functions. This is a private function. [xspClientDojo.js]
XSP.attachViewColumn CheckboxToggler(viewId, colId) [All]	Used internally by the XSP Object to attach a column header check box toggler for the `View` control. This is a private function. [xspClientDojo.js]
XSP._toggleViewColumn CheckBoxes(viewId, colId) [All]	Used internally by the XSP Object to process column header check box toggling for the `View` control. This is a private function. [xspClientDojo.js]

Function Name	**Description**
XSP.isViewPanelRowSelected (viewId, ckId) [All]	Used internally by the XSP Object to identify any checked rows within a `View` control. This is a private function. [xspClientDojo.js]
XSP.initSectionScript (targetSectionId, sectionId, expand) [All]	Used internally by the XSP Object to initialize expand/collapse behavior for the `Section` control. This is a private function. [xspClientDojo.js]
XSP._moveAttr (fromNode, toNode, attrName) [All]	Used internally by the XSP Object to move an element attribute from one element to another by the `Section` control. This is a private function. [xspClientDojo.js]
XSP.serialize(o) [All]	Deprecated in favor of the `XSP.toJson()` function. It is being preserved to avoid breakage of existing scripts. [xspClientDojo.js]
XSP.logw(message) [Web/MU]	Used by the `XSP.log()` function. This is a private function. [xspClientDebug.js]
XSP._dumpObject(obj, name, indent, depth, options) [Web/MU]	Used by the `XSP.dumpObject()` function. This is a private function. [xspClientDebug.js]
XSP.publishEvent (eventName, payload, payloadType) [CA/MU]	Empty implementation in `xspClientDojo.js`. This is overridden by `xspClientCA` and `xspClientMashup.js` implementations. This function is automatically output by the XPages runtime when `ComponentPublish*Action` simple actions exist on the XPage. This is a private function. [xspClientDojo.js/xspClientCA.js/xspClientMU.js]
XSP.dispatchEvent(source, name, value, event) [All]	Automatically generated by the XPages runtime for cross-communication with the XPages View Part in the Notes client. This is a private function. [xspClientDojo.js]

Table 4.3 Summary of the Private XSP Client Side JavaScript Object Functions (cont'd)

Function Name	**Description**
XSP.setComponentMode(mode, params) [MU]	Empty implementation in `xspClientDojo.js`. This is overridden by the `xspClientMashup.js` implementation. This function is automatically output for Set Component Mode simple actions that exist on the XPage. This is a private function. [xspClientDojo.js -> xspClientMashup.js]
XSP.dispatchJSONEvent(source, name, value, event) [CA]	Automatically generated by the XPages runtime for cross-communication with the Composite Application container. This is a private function. [xspClientCA.js]
XSP.onComponentLoaded() [MU]	Used by the XSP Object within a mashup to initialize the widget when the page loads. This is a private function. [xspClientMashup.js]
XSP.callJavaAction(actionId, params, needReturn) [Notes]	Used internally by the XSP Object in XPages in the Notes client application. This is a private function. [xspClientRCP.js]
XSP._embedControl(id, handle) [Notes]	Used internally by the XSP Object in XPages in the Notes client application. This is a private function. [xspClientRCP.js]
XSP._resize() [Notes]	Used internally by the XSP Object in XPages in the Notes client application. This is a private function. [xspClientRCP.js]

The Public XSP Client Side JavaScript Object Functions

This section provides you with details on each of the publicly scoped XSP Client Side JavaScript object functions listed in Table 4.2. Note that it does not provide any detail on the private functions listed in Table 4.3. Notes/Domino 8.5.3 has 33 public and 58 private XSP Client Side JavaScript object functions. You can also refer to the **PCGCH04.nsf** sample application, where you will find the **publicXSPFunctions** XPage. This XPage contains working examples of the 33 public XSP Client Side JavaScript object functions. Figure 4.3 shows this XPage in a browser.