QUANTUMART

QP7 Backend Explorer 2 User Guide

v. 2.0

http://www.quantumart.com

Contents

Glossary

- Add-In – see QP7 Backend Explorer
- Backend – website administration system based on QP7.Framework.
- CMS – Content Management System – Content Administration System – general name for systems that allow users to change data (text, graphical, etc.) on a website separate from the website's actual presentation
- QP7 Backend Explorer – Add-in for Microsoft Visual Studio 2005 and 2008 that allows interaction with templates, pages, objects and formats of a website based on QP7.Framework 7. QP7.Framework 7.6 and higher requires QP7 Backend Explorer 2. (Previous QP7.Framework's versions use QP7 Backend Explorer of version 1.6.)
- QP7.Framework –Quantum Art's CMS.
- IntelliSense – system of clues while writing code as part of Visual Studio
- Panel – area of Visual Studio containing a list of tools combined in a group
- User – person that works with QP7 Backend Explorer
- Site – collection of web pages located within one domain name as used in QP7.Framework based CMS.

1 General Overview

QP7. Framework Visual Studio 2005 and 2008 Add-In (Add-In) – is a programming tool that allows interaction with templates, pages, objects and formats located inside QP7.Framework CMS and leveraging various tools and technologies of Microsoft Visual Studio.

By working with the Add-in web developer can access the following tools that are unavailable via backend's web interface:

- Interactive syntax checking
- IntelliSense – highlight of keywords, appearance of clues and auto-complete feature
- Drag-n-Drop – ability to insert pre-generated code by dragging elements from the tree onto the code windows (presentation and code-behind).
- Ability to edit properties of various elements via "Properties" panel (as part of Visual Studio)
- Work with multiple backend elements at a time (check-in, check-out, page assembly)
- Ability to work with multiple connections within one solution
- Ability to work in a multi-user environment enabled by a locking (check-in/check-out) system
- Ability to perform local assembling of WEB-site and use all advantages of the Microsoft Visual Studio debug monitor.
- Ability to use for local assembling (Build) and debugging both built in WEB-server of the Microsoft Visual Studio or, for example Internet Information Services (IIS).

2 Working Scheme

Figure 2-1 describes the interaction of QP7 Backend Explorer with QP7.Framework system. The Add-In does not directly access the database nor the QP7.Framework CMS, instead, it interacts with web services layer, which communicates with the database or dispatches calls to appropriate QP7.Framework CMS functions.

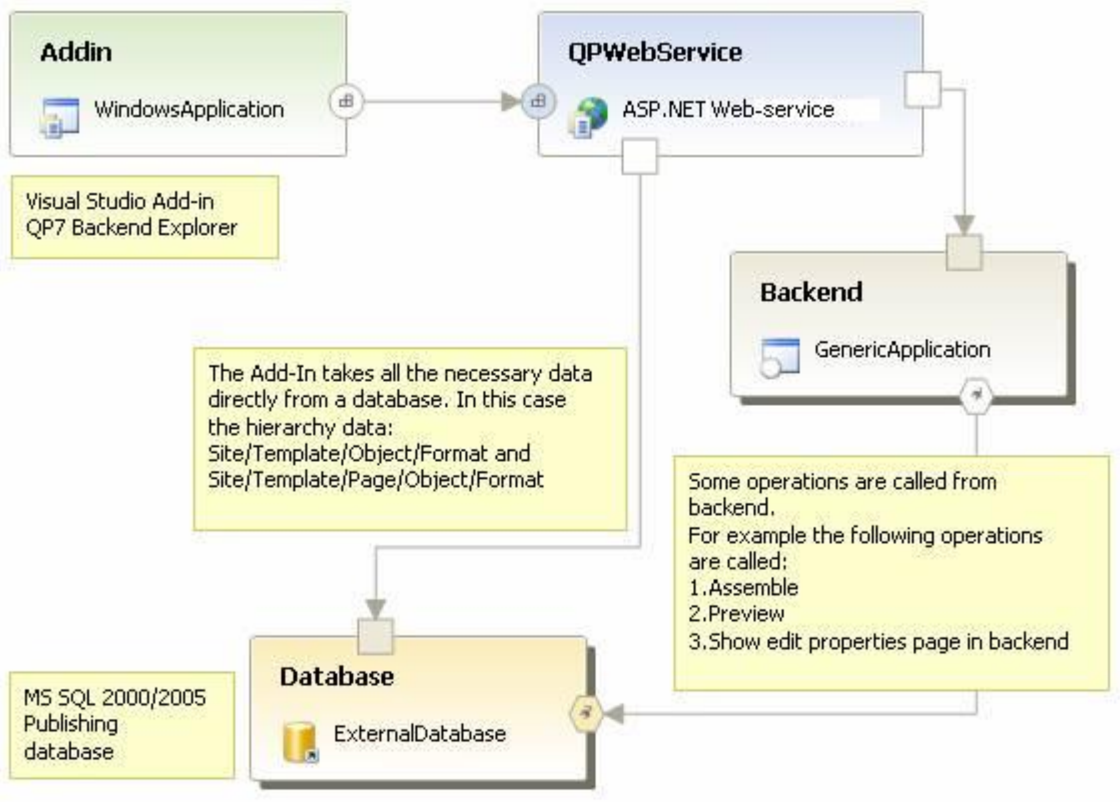

Addin
WindowsApplication
Visual Studio Add-in
QP7 Backend Explorer
QPWebService
ASP.NET Web-service
Backend
GenericApplication
The Add-In takes all the necessary data directly from a database. In this case the hierarchy data:
Site/Template/Object/Format and
Site/Template/Page/Object/Format
Some operations are called from backend.
For example the following operations are called:
1.Assemble
2.Preview
3.Show edit properties page in backend
MS SQL 2000/2005
Publishing
database
Database
ExternalDatabase

Figure 2-1

3 Add-In Requirements and Installation

Installation of the Add-In itself is done via an installer.
Removing the Add-In can be done in the following two ways:

- By running the installer. If Add-In is already set up, the installer offers to uninstall it.
- Using “Start | Control Panel | Add or Remove Programs” dialog.

During the removal process it’s possible to specify whether to keep or discard the connections files (see Local File Structure). If the connections files are saved, working with them can be continued once the Add-In is reinstalled.

In order to use the Add-In it’s necessary to have access to an installed and configured QPWebService as well as an installed instance of Visual Studio.

In order to use the Debug mode it is required to obtain a license file for using quantumart.dll as this library will reside on the local hard drive. The license server is http://licence7.quantumart.com.

4 Add-In Basics

4.1 Start Working

After installing Add-In inside “Tools” menu of Visual Studio there will be a new item: “QP7 Backend Explorer 2” (Figure 4-1):

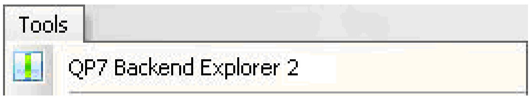


Figure 4-1

If "QP7 Backend Explorer 2" is not visible inside "Tools" menu see "Known issues.txt", located at: "Start | Programs | QP7 Backend Explorer 2 | Known issues".

Once "QP7 Backend Explorer 2" is clicked on, Visual Studio opens a new panel (Figure 4-2):

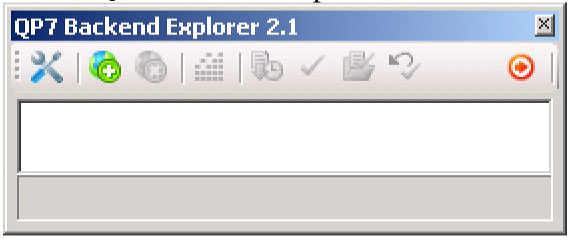


Figure 4-2

Using this panel is analogous to using other Visual Studio panels (for example: Toolbox, Server explorer, etc.). It can be fixed, moved, closed.

Closing of the panel does not mean that the Add-In is closed, instead, it is just hidden to free up screen space. (to exit click on "Exit Add-In" ⊙ button). The panel can always be accessed from "Visual Studio | Tools | QP7 Backend Explorer 2" menu if it appears to be closed.

When working with only one connection, that connection will be loaded once QP7 Backend Explorer panel loads. When working with multiple connections only connection headers will be loaded; the data can be loaded by a user request (Fig. 4-3)

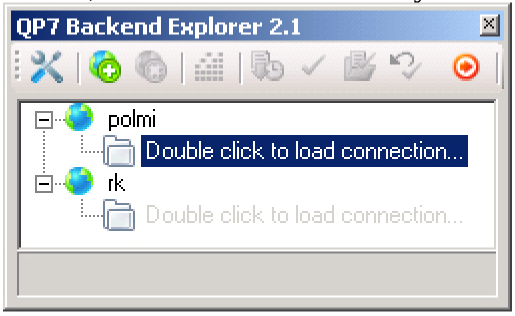

QP7 Backend Explorer 2.1
polmi
Double click to load connection...
rk
Double click to load connection...

Figure 4-3

4.2 Finish Working

To exit out of the Add-In click on the "Exit Add-In" ⊙ button. If you're planning to exit out of Visual Studio itself, first exit out "QP7 Backend Explorer 2" Add-In and then close Visual Studio.

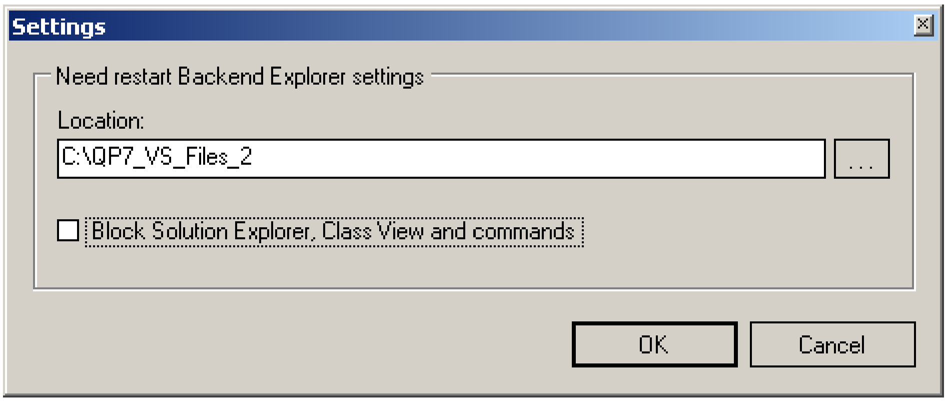
The next time you load the Add-In all connections will be restored from the locally saved files. If "Save Data Offline" was not checked during the creation of a connection then files are not saved locally. (see Create New Connection).

If there are some files left open with unsaved changes then there will be a dialog window with the list of such files. You could choose there to save these changes or just to resume work with Add-In.

4.3 Add-In Configuration

Clicking on the “Settings” button invokes the Settings window (Figure 4-4):

- “Location” – specifies path to a folder where Add-In files will be stored;
- “Block Solution Explorer, Class View and commands” parameter allows to block appearance of the corresponding Microsoft Visual Studio windows on the screen. Default value of this parameter is off.



Settings

Need restart Backend Explorer settings

Location:
C:\QP7_VS_Files_2

□ Block Solution Explorer, Class View and commands

Figure 4-4

To apply new settings Visual Studio needs to be restarted.

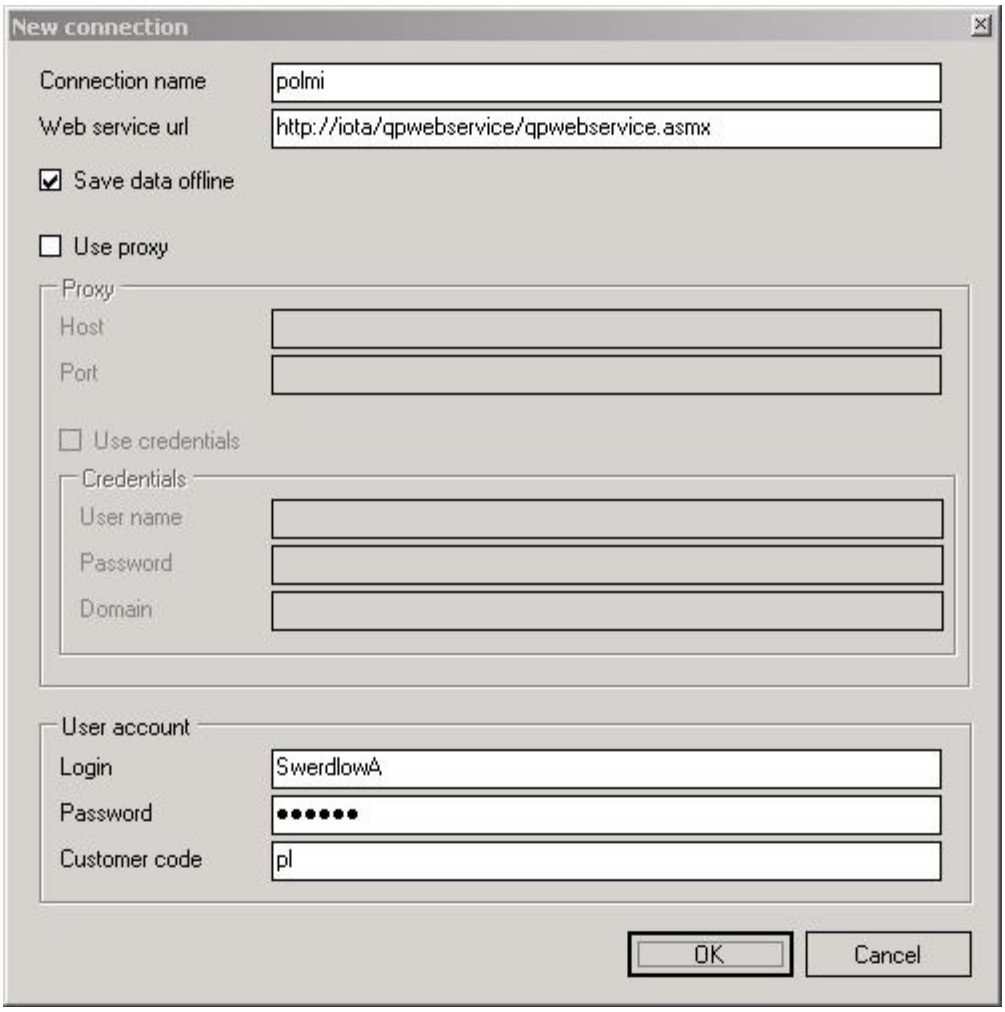
4.4 Working with Connections

QP7 Backend Explorer is capable of managing unlimited number of connections to QP7 CMS, but only one connection can be open at a time.

4.4.1 Create New Connection

Before creating a connection Add-In can be configured, otherwise default settings will be used. (see Add-In Configuration). The settings can also be changed at a future time.

To create new connection to QP7.Framework Backend click on the “Add new connection” button, a dialog window will appear (Figure 4-5):



New connection

Connection name: polmi

Web service url: http://iota/qpwebservice/qpwebservice.asmx

☑ Save data offline

☐ Use proxy

Proxy

Host:

Port:

☐ Use credentials

Credentials

User name:

Password:

Domain:

User account

Login: SwerdlowA

Password: ●●●●●●

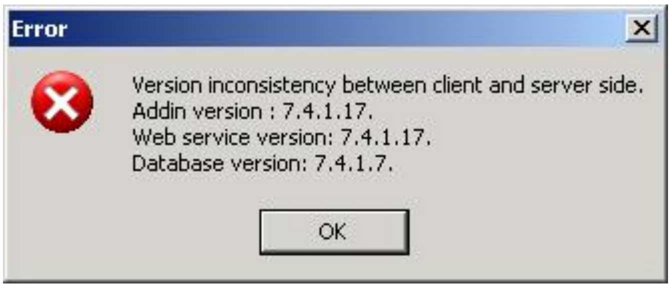
Customer code: pl

Figure 4-5

The panel contains the following fields:

- "Connection name" – connection name that will appear on QP7 Backend Explorer
- "Web service url" – url path to web-services
- "Save data offline" – whether or not to save files locally; if files are saved then the connection will be restored upon its next launch.
- "Use Proxy" – activates proxy fields; "Use credentials" – activates proxy authentication parameters.
- "User Account" – defines parameters for QP7.Framework access (same as backend login information).

After the "OK" button is clicked, the Add-In will import various data from QP7.Framework into Visual Studio (might take a few minutes).
To create a connection it is necessary for the Add-In version not to exceed the QPWebservice version. And the web service version should not exceed the database version. The error message will be shown (Fig. 4-6) if the versions are out of sync:

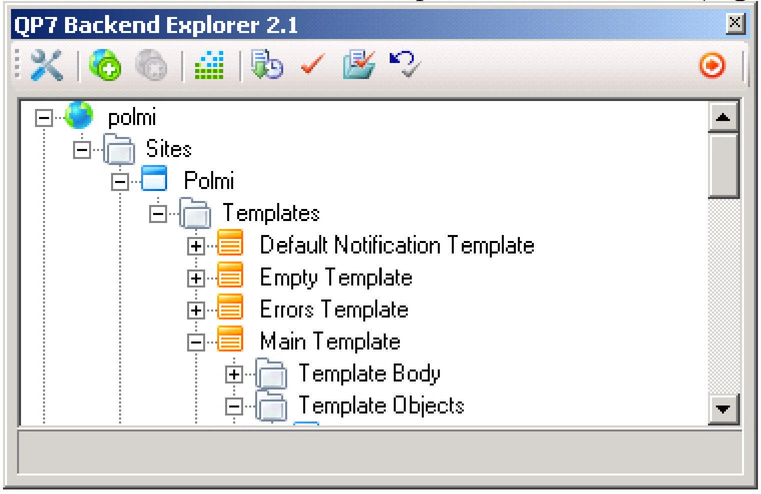


Error

Version inconsistency between client and server side.
Addin version : 7.4.1.17.
Web service version: 7.4.1.17.
Database version: 7.4.1.7.

Figure 4-4

When the import process is completed the “QP7 Backend Explorer 2” panel will display a tree of sites and child elements for this particular connection (Figure 4-7):



QP7 Backend Explorer 2.1

```
polmi
  Sites
    Polmi
      Templates
        Default Notification Template
        Empty Template
        Errors Template
        Main Template
          Template Body
          Template Objects
```

Figure 4-7

4.4.2 Delete Connection

To delete a connection click on the connection name m_a and then click on the “Delete connection” button at the top of the “QP7 Backend Explorer 2” panel. This will also delete al child elements and locally stored files.

5 *Working with Elements of the Hierarchical Tree*

5.1 Project Tree

The Project Tree is a collection of hierarchical child elements and their containers. Element-Containers are indicated by the folder symbol (“Sites”, “Templates”, “Template body”, “Template pages” and others are considered to be Element-Containers). The Element-Containers have a limited contextual menu (right-click menu) and cannot be edited or removed (for example, the contextual menu does not contain “Open” item). Most types of Element-Containers can create child elements via the “create new element” item of the contextual menu.
Child Elements can be added, deleted or edited (except for auto-generated ones like “object fields”.

The general tree structure consists of the following:

- The root of the tree contains connections
- The next level is consisted of sites for a given connection
- The following level is the “Templates” Element-Container, containing site templates. “Default Notification Template” template is always present as one of the templates (it’s auto-generated). Every template contains three Element-Containers:
    - Template Body – contains template code (starting point of every page that is based on this template):
        - Presentation – pseudo-HTML code as used by .NET specifications
        - Code Behind – code-behind as used by .NET specifications
    - Template Objects – objects that can be shared or overridden by all template pages
    - Template Pages – pages that are based on this template (once assembled they become actual .aspx files, may contain their own objects)
- Every object can contain one or more formats, which allow access to code:
    - Presentation
    - Code Behind
- Objects of type “Publishing Container” contain an additional read-only Element-Container: “Template Object Fields” or “Page Object Fields” (for template objects and page objects respectively). It contains all fields (including system fields) of underlying Content Table.

5.1.1 Add and Delete Tree Elements

New elements can be added to Element-Containers (indicated by the folder 📁 symbol) by selecting an Element-Container, right clicking on it and selecting “Create new element” item from the contextual (right-click) menu. The type of the dialog window that appears next will depend on the type of Element-Container, but in most cases it will only allow setting of basic parameters. Other parameters can be added or edited in the “Properties” panel for a given element (selected in the tree) or via QP7.Framework Backend web interface.

When object is created via Add-In a default format is automatically created with pre-populated code templates for “Presentation” and “Code behind”. The default code may be different for each object type and for each programming language. Parent’s template object format can be copied for the overriding page object by checking “Copy format” checkbox.

To delete a tree element – select it in the tree, right click and select “Delete” from the contextual menu (you can also select multiple elements from the same level of the navigation tree). Most elements can be deleted this way except for site (can be removed via the Backend’s web interface), Element-Containers and “Presentation” / “Code Behind” (both are removed when their format is deleted). All child elements are removed when a parent element is deleted.

5.1.2 Promote Page Object to Template Object Level

The Add-In also allows promoting a Page Object to a Template Object in cases where Page Objects are generic enough to be shared among all Template Pages. To promote Page Object, select it in the tree, right click and choose “Promote to template level” from the contextual menu. A page object will become a template object.

The promoted object’s invocations by the sibling Page Objects will continue to function correctly.

5.1.3 Elements Create Like

The Add-In allows to easily create copies of Objects, Pages or Templates. To copy one of these elements, select "Create Like" from the contextual menu. The "Create Like" dialog window similar to the one shown during the process of new object creation will follow.
Only the current element is copied (with its default format if it has one). The other child elements (Template Pages, when a Template is copied or Object Formats, when an Object is copied) are not copied.

5.2 Multi-User Environment (check-in / check-out)

The Add-In allows multiple developers to work on the same project by utilizing check-in / check-out (locking).

The limitations of the current version require developers to be part of the "Administrators" group (set in the QP7.Framework's Backend)

The Multi-User Visual Studio environment is very similar to "Visual Source Safe" from Microsoft. Developers have currently four options (available from the contextual menu on the right mouse click): "Get latest version", Check out", "Undo check out", "Check in". The options can be applied to one or more elements belonging to the same level and to all of their child elements if "recursive" option is selected ("recursive" selection can be set afterwards once the options dialog window is displayed). Each menu option will cause the tree elements to be renewed.

5.2.1 Get Latest Version

This option allows developers to get the latest version for any element or to refresh the project's sections. It's invoked by selecting an element, performing right mouse click and selecting "Get latest version" (Figure 5-1):

Figure 5-1

If the "recursive" flag is left blank in the subsequent window then only the current element will be renewed; otherwise all child elements will be renewed (Figure 5-2):

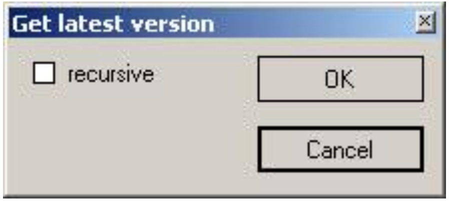


Figure 5-2

The other way to get latest version of a selected element is to press the "Get latest version" button on the Add-In toolbox. In that case the recursive "Get latest version" action will be performed for a chosen tree element analogous in behavior to the option in the context menu.

5.2.2 Check Out

Initially all elements are opened as read-only. In order to edit them and their properties it's necessary to first lock them by invoking "Check-Out" from the contextual menu (Figure 5-3):

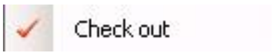


Figure 5-3

Formats if opened as read-only, Presentation or Code Behind, are checked out automatically once the code is attempted to be modified.

The “recursive” flag is analogous to the “Get latest version” option (Figure 5-4):

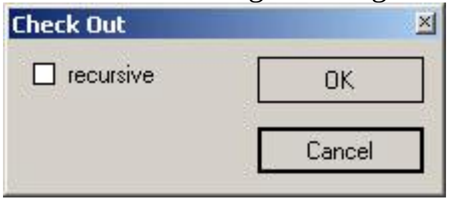


Figure 5-4

Invoking “Check Out” causes the Add-In to load the latest version for the elements of the tree. After an element is checked-out its tree image is grayed-out and lock icon is added. (Figure 5-5):

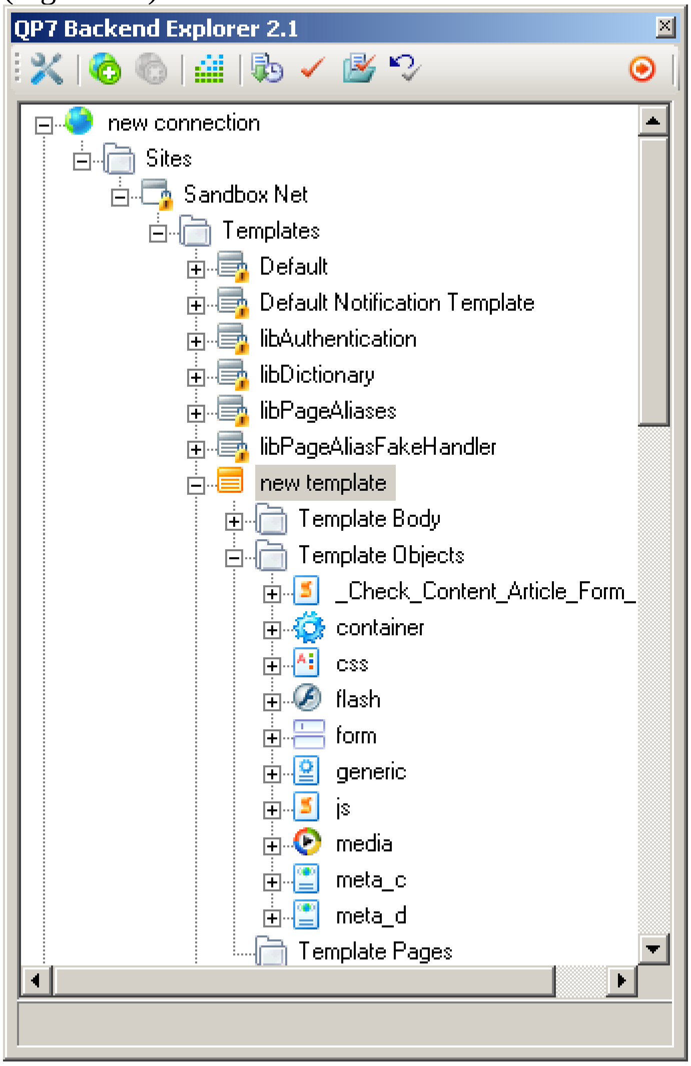


Figure 5-5

Elements that are locked by other users appear grayed-out as well, but the user icon is displayed instead of the lock icon. (Figure 5-6):

Figure 5-6

If an element is locked by another user then the contextual menu will have only two options (Figure 5-7):

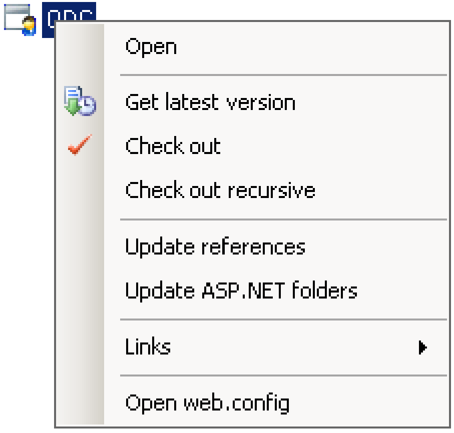


Figure 5-7

In this case there will be only two active buttons on the toolbox: "Get latest version" and "Check Out".
"Check Out" will only renew elements (analogous to "Get Latest Version") if they're locked by another user. If since the last renewal elements become unlocked then they will be locked (checked-out) by the current user (regular "Check Out" will be performed).
The other way to check out a selected element (or elements) is to press the "Check out" button on the Add-In toolbox ✓ It works analogously to the "Get Latest Version" button on this panel: performs one-click "Check Out" of an element, similar to the same one in the context menu.

5.2.3 Check Out Recursive

This context menu option (Fig. 5-8) supplements the "Check Out" option, and allows performing "Check Out" for selected elements and all their child elements, bypassing the additional "Check Out" dialog.

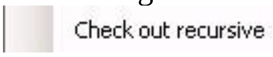


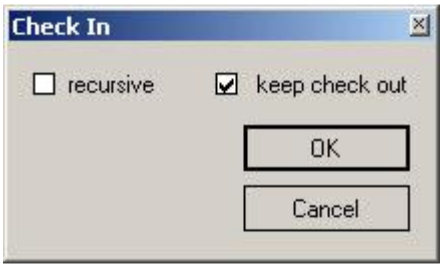
Figure 5-8

5.2.4 Undo Check Out

Releases the lock on elements as well as renews the elements.
"Undo check out" dialog works analogously to the previous ones. The toolbox button ↺ works in a similar way.
Once invoked the elements on the server are unlocked and changes that were made by the user on his/her local machine (in QP7 Backend Explorer) are rolled-back.

5.2.5 Check In

Saves changes to the database ("assembly" is not performed).



☐ recursive ☑ keep check out

Figure 5-9

The “recursive” flag is analogous to the one used by “Check out”, “Get latest version” and “Undo check out”. The “keep checked out” flag (Figure 4-8) will keep the object locked after the changes are saved, otherwise the locks will be released.
The toolbox button has the same function as “Check in” in the context menu.
If there are unsaved files that are attempted to be checked in then a dialog window will appear offering to save them.

5.3 Working with a Group of Elements

It is possible to apply any of the following actions: “Check In”, “Check Out”, “Undo Check Out”, “Get Latest Version”, and “Delete” simultaneously to a group of elements that are located on the same level of the tree: pages of the same template, objects of the same page or template etc. To work with a group of elements select a group of elements in the QP7 Backend window in a way that is common for Windows applications: select with “Ctrl” or “Shift” buttons pressed, and then choose the desired action from the toolbox or the context menu then.

All the operations that are made with a group of elements are performed recursively.

5.4 Edit Element Properties

Any element except for Container-Elements has properties which can be edited (via Visual Studio Properties Panel) once the element is checked-out (or locked) by choosing the “Open” item from the contextual menu (Figure 5-10):

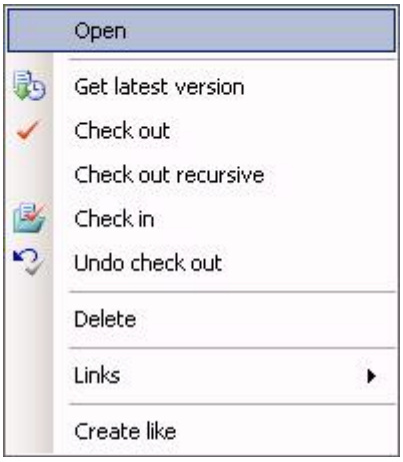
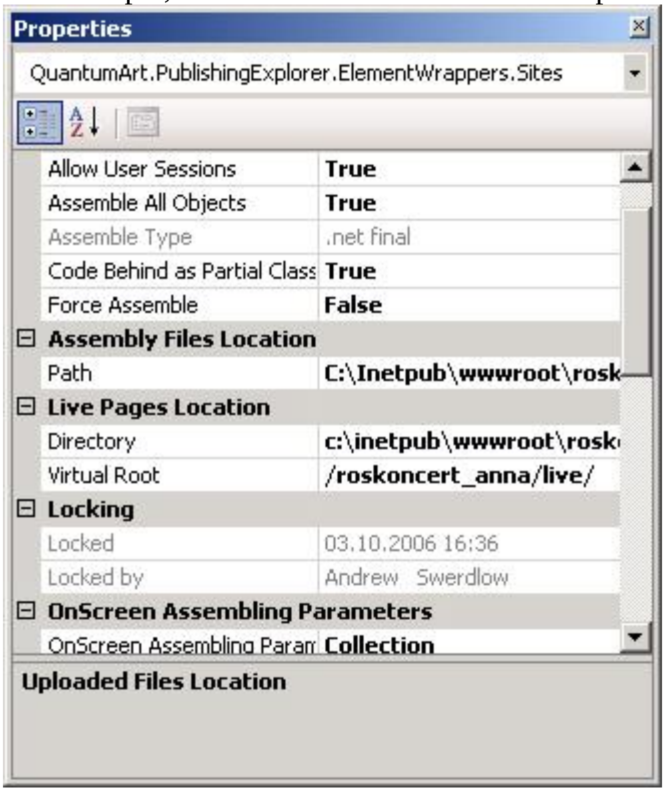


Figure 5-10

If the current element properties were not shown, the required element should be selected once again.

For example, below is a screen shot of the Properties Panel for a site (Figure 5-11):

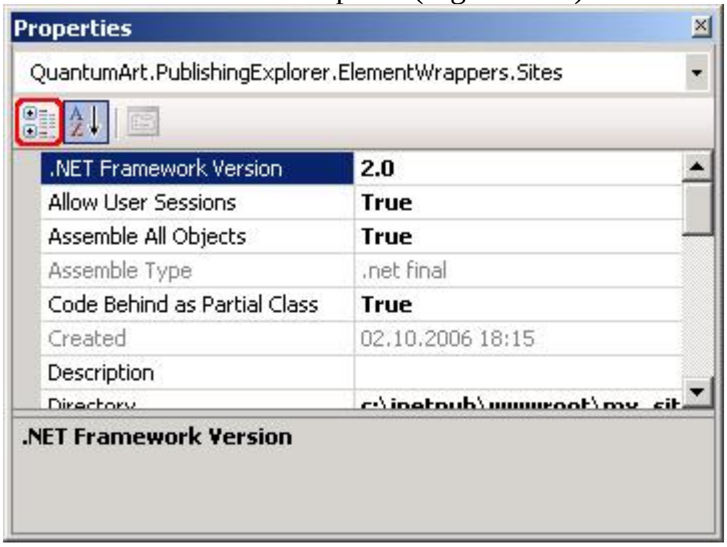


Allow User Sessions	True
Assemble All Objects	True
Assemble Type	.net final
Code Behind as Partial Class	True
Force Assemble	False
⊟ Assembly Files Location	
Path	C:\Inetpub\wwwroot\rosk
⊟ Live Pages Location	
Directory	c:\inetpub\wwwroot\rosk
Virtual Root	/roskoncert_anna/live/
⊟ Locking	
Locked	03.10.2006 16:36
Locked by	Andrew Swerdlow
⊟ OnScreen Assembling Parameters	
OnScreen Assembling Param	Collection

Uploaded Files Location

Figure 5-11

Standard Visual Studio methods are used to work with elements’ properties.
The properties can be shown by categories or in an alphabetical order. To switch between them use the left button on this panel (Figure 5-12):

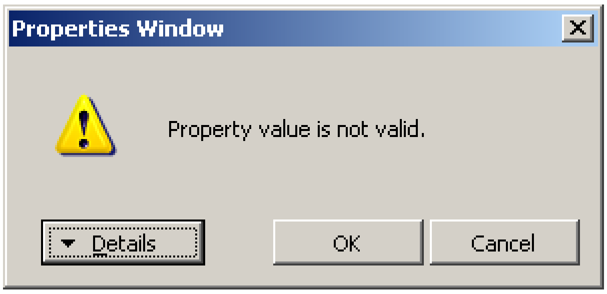


.NET Framework Version	2.0
Allow User Sessions	True
Assemble All Objects	True
Assemble Type	.net final
Code Behind as Partial Class	True
Created	02.10.2006 18:15
Description	
Directory	c:\inetpub\wwwroot\my_sit

.NET Framework Version

Figure 5-9

After a property is changed (it changes when Enter button is pressed or when the field loses focus with Tab button or mouse action) the validity of the value is checked. If it is not valid, then the following alert message will show up (Fig. 5-13):



Properties Window

Property value is not valid.

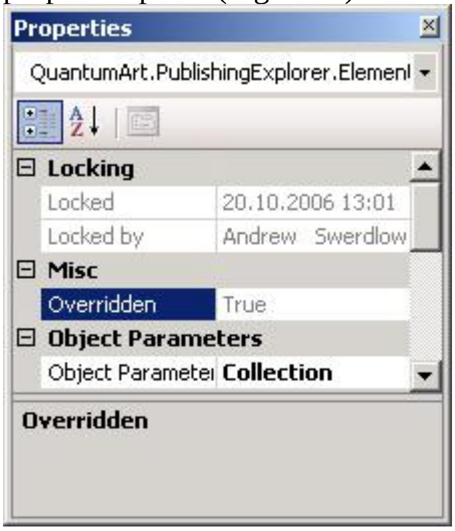
Figure 5-10

5.4.1 Change Element Name

The names of elements can be changed by changing "Format Name", "Object Name", "Page Name", "Template Name", "Site Name" properties (for formats, objects, pages, templates or sites respectively). When the name properties are changed the local tree is refreshed to reflect these changes. In order to post changes to the database it's still necessary to perform "Check In".

5.4.2 The "Overridden / Overrides" field

The Add-In allows defining whether or not a template object is overridden by any of the page objects. If it is overridden then the "True" value will appear in the "Overridden" field of the properties panel (Fig. 5-14).



Properties

QuantumArt.PublishingExplorer.Element

Locking

Locked 20.10.2006 13:01

Locked by Andrew Swerdlow

Misc

Overridden True

Object Parameters

Object Paramete Collection

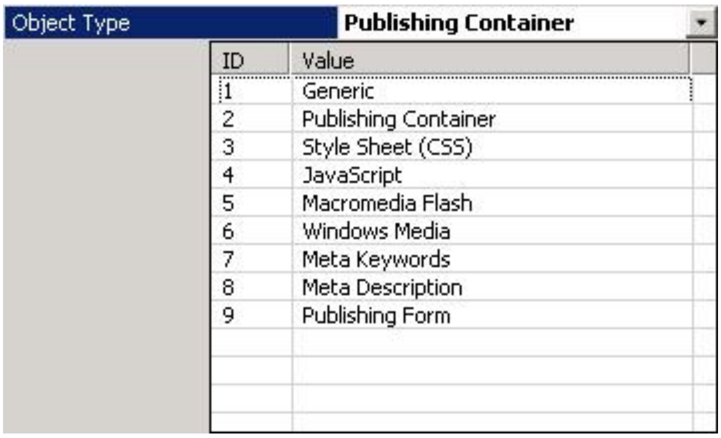
Overridden

Figure 5-11

Page Object Parameters have a similar field: "Overrides". It shows whether or not the page object overrides a template object.

5.4.3 Change Object Type

The Add-In allows changing object types via "Object Type" field (Fig. 5-15).



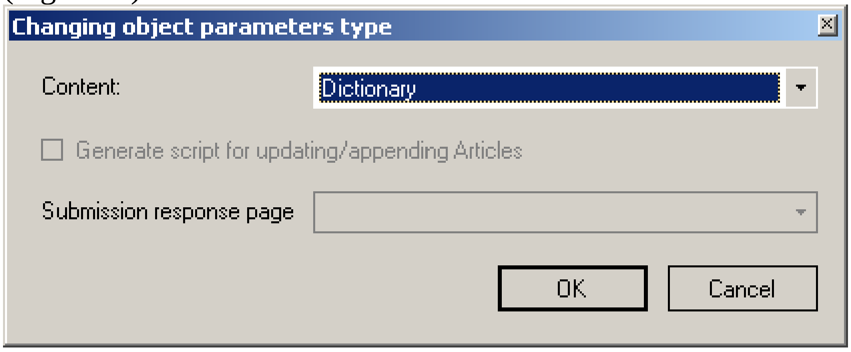
Object Type: Publishing Container

ID	Value
1	Generic
2	Publishing Container
3	Style Sheet (CSS)
4	JavaScript
5	Macromedia Flash
6	Windows Media
7	Meta Keywords
8	Meta Description
9	Publishing Form

Figure 5-12

This operation can not be rolled back using the “Undo Check Out” operation.

When object type is changed to “Publishing Container” or “Publishing Form”, an additional dialog window appears where the user can choose a Content Type that will be used by this object (Fig. 5-16).



Changing object parameters type

Content: Dictionary

□ Generate script for updating/appending Articles

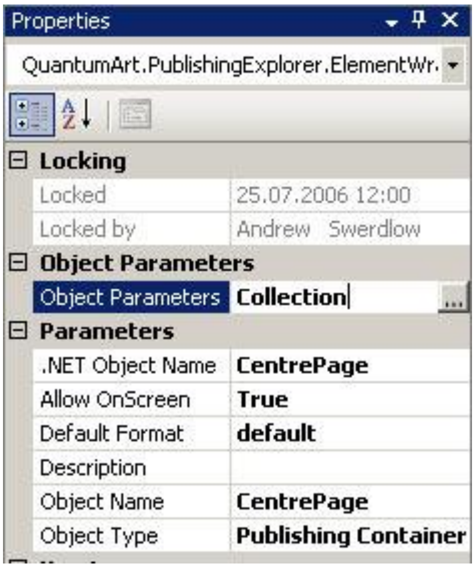
Submission response page

Figure 5-13

For an object of type “Publishing Form” there will be additional options present, offering to choose a redirection page once form is submitted (“Submission response page” field), and to generate code for the resulting html form with update/create new scripts (“Generate script for updating/appending Articles” flag).

5.5 Edit Object Parameters

Properties panel for every object contains the information about its name, default format, type and other parameters (Fig. 5-17):



QuantumArt.PublishingExplorer.ElementWr.

Locking	
Locked	25.07.2006 12:00
Locked by	Andrew Swerdlow
Object Parameters	
Object Parameters	Collection
Parameters	
.NET Object Name	CentrePage
Allow OnScreen	True
Default Format	default
Description	
Object Name	CentrePage
Object Type	Publishing Container

Figure 5-14

Objects with complex structure, like “Publishing Container” and “Publishing Form”, have additional parameters. To get access to these properties the “Collection” button should be pressed.

For object of any type, except of “Publishing Container” and “Publishing Form” pressing on the “Collection” button will not result in any action.

The dialog fields are analogous to the corresponding fields in QP7.Framework web interface.

Working with Default Values and Default Order fields (“Publishing Container” parameters editing dialog) is analogous to working with any Grid control elements. To paste a value it is enough just to enter a string inside the asterisk-marked field. To delete an item left-click on the left column of the item and press Delete.

5.6 Using Links

To interact with other backend elements there are built-in links in the Add-In context menu (Fig. 5-18).

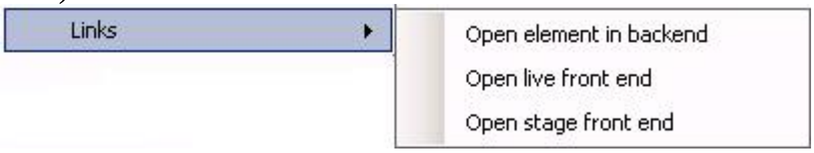


Figure 5-15

The “Open element in backend” link allows editing of an element through QP7.Framework web interface. Access parameters that are set by user in the connection properties and link to the Backend that is placed in QPWebService’s configuration file are used to determine the path to the Backend.

“Open live/stage front end” links allow seeing site’s front end.

For objects of type “Publishing Container” there is an additional link in the list (Fig. 5-19)

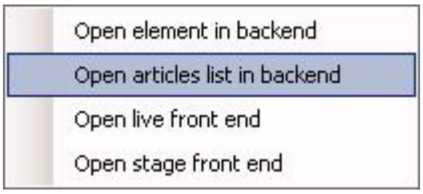


Figure 5-16

The “Open articles list in backend” link opens backend at the “Articles” tab for the content that is connected with this object. Links are opened in the default browser.

5.7 Edit Formats

One of the most common actions in a site creating process is format editing. The superior convenience of formats editing is one of the most important features of the Add-In.
To edit a format in should be opened with a double click, or using the “Open” item in the context menu.

If a user starts to perform editing of format then dialog appears on the screen. This dialog allows to approve or decline the operation.
In the case of approving nonrecursive “Checkout” operation performs automatically for a present format.

5.7.1 Turn on IntelliSense (“Update references”)

If the IntelliSense feature of Visual Studio does not seem to work for various object types defined in the project’s DLL’s (located in the “bin” folder), select the site in the tree, right mouse click and click on “Update References” of the contextual menu (Figure 5-20):

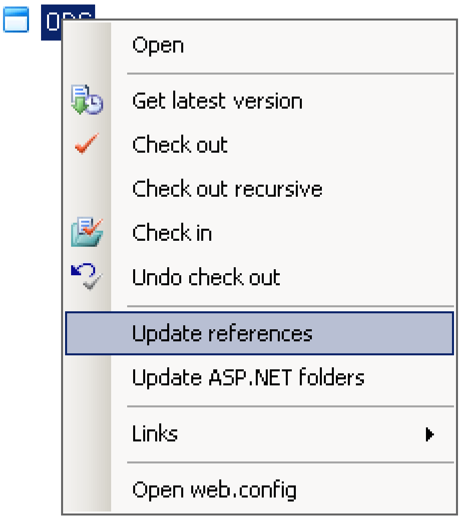


Figure 5-20

The process of “Update References” downloads all .dll assemblies located on the server in the folder “bin” and references them in the project.
When the Create New Connection operation is performed this action is executed automatically.
“Update References” is also useful if dll’s are modified on the server.

5.7.2 Drag-n-drop

Drag-n-drop can be used to drag objects, formats and field names *(for publishing container objects)* into the code windows (presentation and code behind). The result of drag-n-drop operations is insertion of code.

Drag-n-drop allows inserting of objects only in accordance with the QP7.Framework rules. Drag-n-dropping of page objects onto the template objects' code is not allowed.

Inserted code is different for presentation and code-behind.
For Presentation: `<qp:placeholder calls="Object name" runat="server"/>`
For Code Behind: `ShowObject("Object name", this);`

Drag-n-drop inserts object calls in the following formats:

- For Template Objects or Page Objects within the same template or page *"Object_Name"*
- For Template Objects from other templates *"Template_Name.Object_Name"*
- For Template Object Formats and Page Object Formats within the same template *"Object_Name.Format_Name"*
- For Template Object Formats from other templates *"Temlate_Name.Object_Name.Format_Name"*

5.7.3 Set Default Code

The Add-In allows replacing the current object format code with the default one.
To replace Presentation or Code Behind default code, first check out the format and then select the "Set Default" option in the context menu (Fig. 5-21):

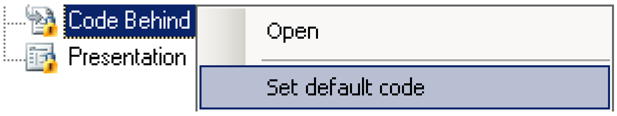


Figure 5-17

The default code is different for Presentation and Code Behind; it is also distinguished for objects of type "Publishing Container" and the other types. The default code also depends on format language.

5.8 Preview

Objects (Formats) can be previewed by choosing "preview" from contextual (right-click) menu. Preview works the same as the format preview in the backend.

Preview is implemented as a direct invocation of backend preview. In case of errors check the same functionality in the backend

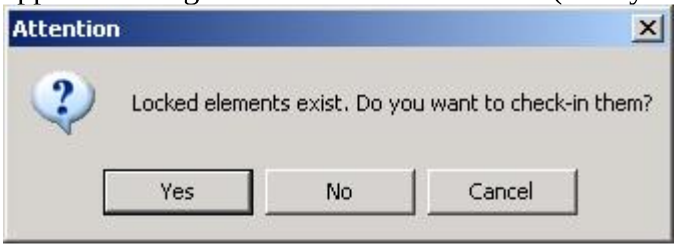
5.9 Page Assembly

In order to assemble a page or multiple pages, choose a tree element (or several elements) in the Add-in and click assemble icon on the toolbar at the top ▦ .
If press "Assemble" button while element of hierarchy is chosen and this element is belongs to the template but doesn't belong to any page (for example object of template) then only this element will be assembled.

During the assembling of separate page of template reassembling of only one page will be performed. Reassembling of all pages of the template (as it was in previous versions) will not be performed. For reassembling of all pages this operation should be performed evidently for the corresponding template.

- Choosing a Template will result in the assembly of all Template Pages within the template.
- Choosing a Template Object or Format will result in the assembly of the Object or Format only (new in this Add-In's version).
- Choosing a Page will result in the assembly of the current page and all Objects that's enclosed.
- Choosing a Page Object or Page Format will result in the assembly of the Object or Format only (new in this Add-In's version).
- Choosing site will result in the assembly of all pages of all templates for this site.
- Choosing connection will result in the assembly of all sites (and thus all pages for all sites)

When elements are locked by a developer and the assembly is invoked, a message box will appear offering to check these elements in (if they were modified) (Fig 5-22):



Attention

Locked elements exist. Do you want to check-in them?

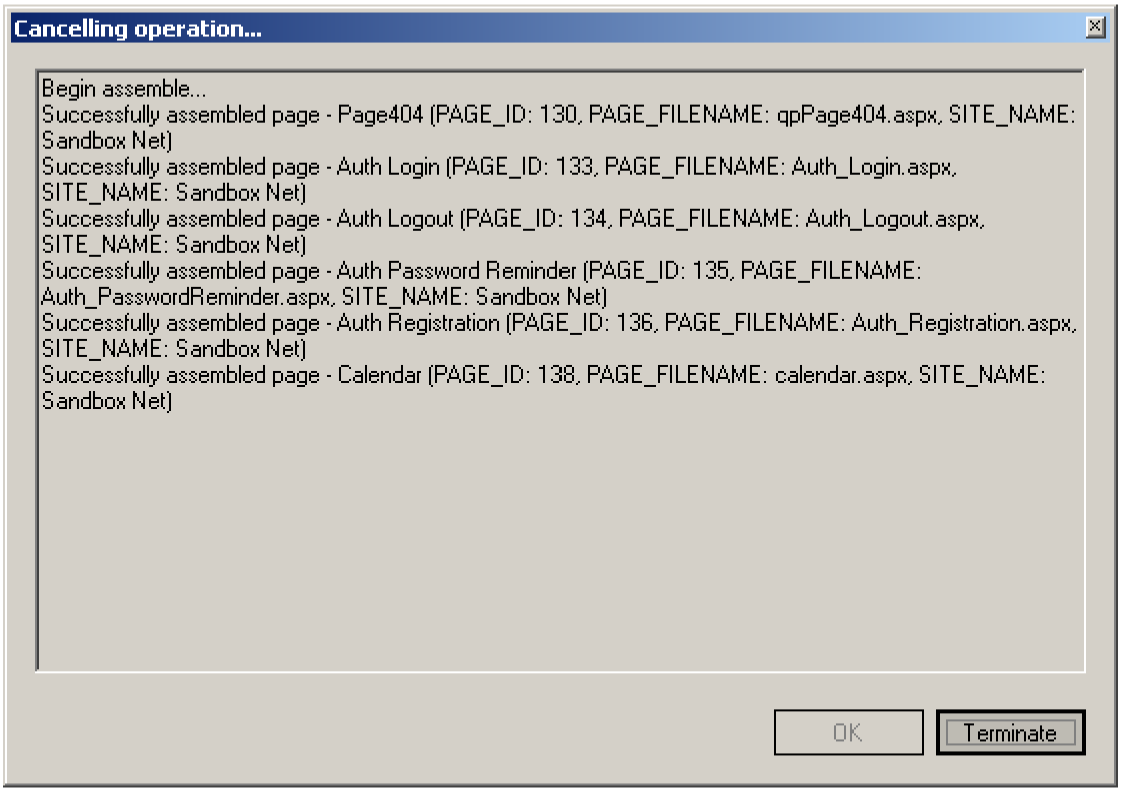
Figure 5-18

- If the element is a Template Format or Template Object, the whole locked template is checked for existing locked elements.
- If the element is a Page Object or a Page Format, the whole page is checked for existing locked elements.
- In other cases the current element is checked.
- If unsaved files exist a message box will appear offering to save them.

Assembly is implemented as a direct invocation of backend assembly. In case of errors check the same functionality in the backend.

During the process of assembly a dialog window appears. The assembly can be cancelled or terminated.

- "Cancel" button will cause the assembly operation to stop once the assembly for the current page is finished.
- "Terminate" button (Fig. 5-23) causes the Add-In to stop interacting with QPWebService without waiting for any kind of response.



```
Begin assemble...
Successfully assembled page - Page404 (PAGE_ID: 130, PAGE_FILENAME: qpPage404.aspx, SITE_NAME: Sandbox Net)
Successfully assembled page - Auth Login (PAGE_ID: 133, PAGE_FILENAME: Auth_Login.aspx, SITE_NAME: Sandbox Net)
Successfully assembled page - Auth Logout (PAGE_ID: 134, PAGE_FILENAME: Auth_Logout.aspx, SITE_NAME: Sandbox Net)
Successfully assembled page - Auth Password Reminder (PAGE_ID: 135, PAGE_FILENAME: Auth_PasswordReminder.aspx, SITE_NAME: Sandbox Net)
Successfully assembled page - Auth Registration (PAGE_ID: 136, PAGE_FILENAME: Auth_Registration.aspx, SITE_NAME: Sandbox Net)
Successfully assembled page - Calendar (PAGE_ID: 138, PAGE_FILENAME: calendar.aspx, SITE_NAME: Sandbox Net)
```

Figure 5-19

6 *Local compilation and debugging of site*

The main difference between the QP7 Backend Explorer ver.2 and older versions is creating file structure of the site on the local disc of developer absolutely identical to the structure on the WEB-server of the client. Every time when source code is changing it is also changing in the same way on the local disc. It allows to perform local assembling (Build) and debugging of the site. At the same time all the opportunities of the Microsoft Visual Studio debugger, such as – breakepoints, call stack windows, watch and other are used.

Important! In order to use the Debug mode it is required to obtain a license file for using quantumart.dll as this library will reside on the local hard drive. The license server is http://licence7.quantumart.com.

6.1 ASP.NET special folders

The whole web content downloads when the Create New Connection operation is performed. If the following folders App_Code, App_Data, App_GlobalResources, App_LocalResources, App_WebReferences, App_Browsers contains any files then this folders will be created locally and all files will be downloaded to the corresponding folders. If you want to update folders content, then select «Update ASP.NET folders» in context site menu.

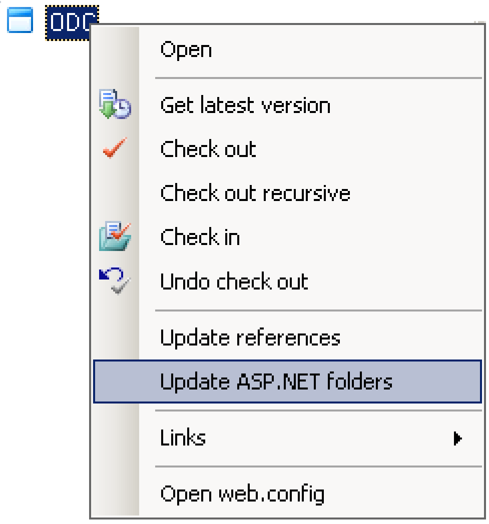


Figure 6-14

App_code is necessary for compiling site locally when it not empty on WEB-server.

6.2 Web.Config file

During the compilation and performing code of the page of the site is calling for data to Quantum Publishing database. That is why it is so important to give correct connection parameters to this database. Connection parameters and other site parameters are setting in web.config file. Open to edit this file in the Visual Studio is possible either by pointing it in the Solution Explorer window or by choosing “Open web.config” point in the context menu of the site unit in the Add-in tree.

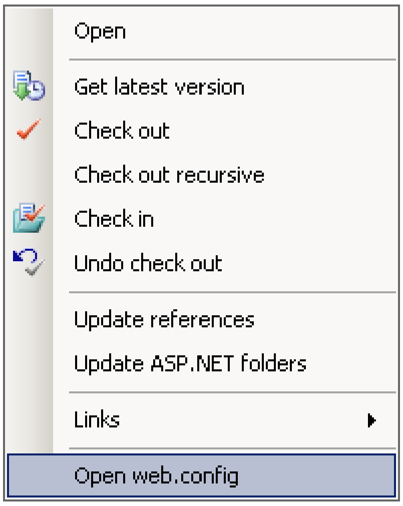


Figure 6-25

Connection parameters should be filled in connectionStrings/qp_database key.
There are also MailHost, MainLogin, MailPassword, RelNotifyUrl keys of appSettings section should be filled if needed.

On the first assembling InternalExpirationTime key is inserts in the appSettings. It manages catching of internal data of quantumart.dll and its default value is 0, i.e. catching is off. This value shouldn’t be changed without necessity because it could effect results of the site assembling.

6.3 Setting parameters for local starting of site

On the settings page of site – Add –in tree element path, to upload-sources which will be used for local site assembling and debugging could be indicated. That is why so important to give the value to parameters of Local Debug Parameters" group.

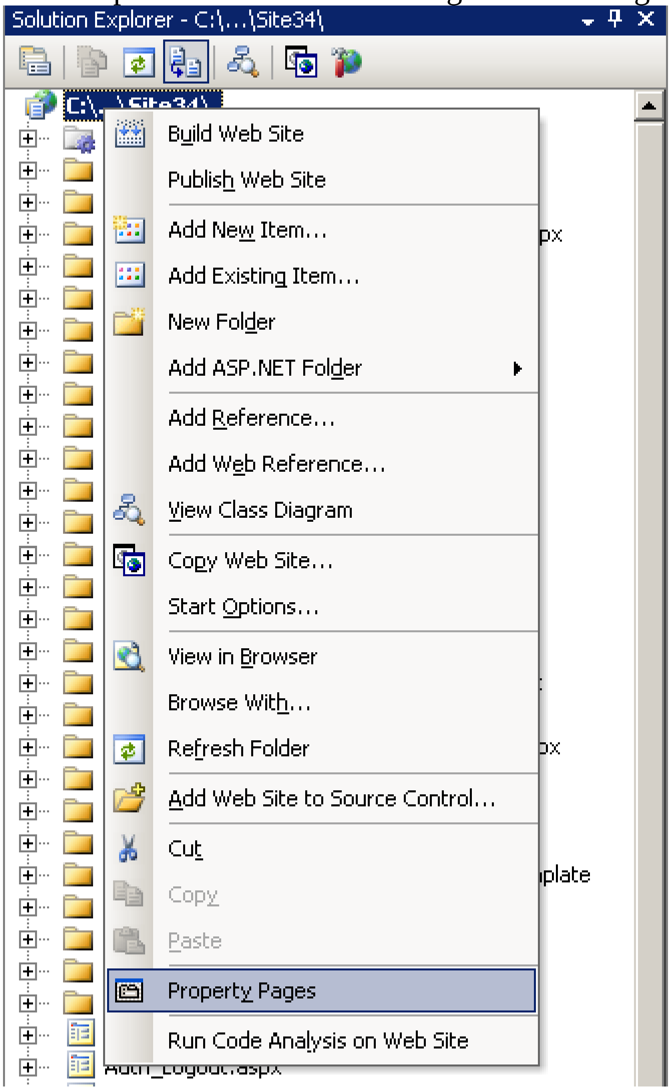


Figure 6-36

Their roles are identical to standard parameters "Uploaded Files Location" in the QP7 Backend. To make site pages use mentioned values "Use Local Upload Variables" value should be turned on. Since these parameters are stored in the QP7 database but not on the local disc of developer, it’s not needed to perform CheckOut of site to change them. For using these values regenerating of source code is not needed too.

6.4 Local starting of site

6.4.1 Starting using WEB-server of the Microsoft Visual Studio

For starting using Web-Server of the Microsoft Visual Studio is not needed to show any extra settings. Only thing is needed – to show connection parameters to the QPublishing database right (6.2) and click the F5 button (or perform the Debug/start Debugging command). Compilation of all pages will be performed. In the case of errors information about them will be represented by standard methods of Microsoft Visual Studio – in the Error List window. In the case of successful compilation browser window with the Start page of site will open.

6.4.2 Starting using external WEB-server

Let's view the situation of site starting using external WEB-server by the example of IIS.
First of all virtual directory should be created for the site. For the created from the virtual directory of site rigging management of IIS should be indicated:

- Read and scripts performing permissions.
- Turn Net Framework ver.2 using on and choose corresponding pool of supplements.
- Turn the Integrated Windows Authentication for the site on.

After that change parameters of the site in the Solution Explorer window of the Microsoft Visual Studio.

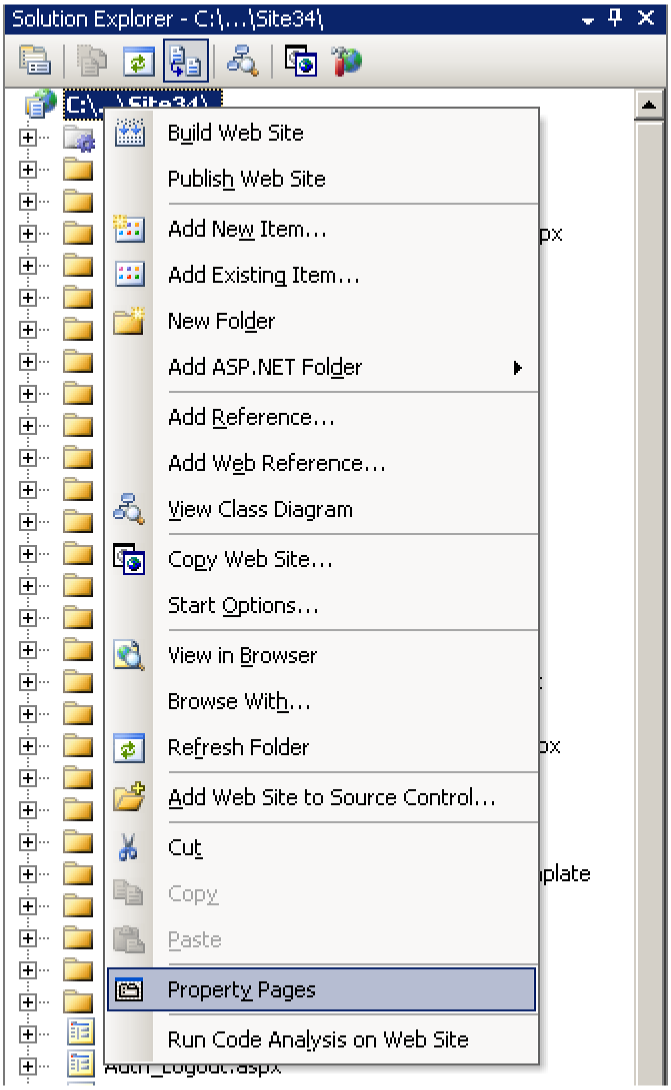
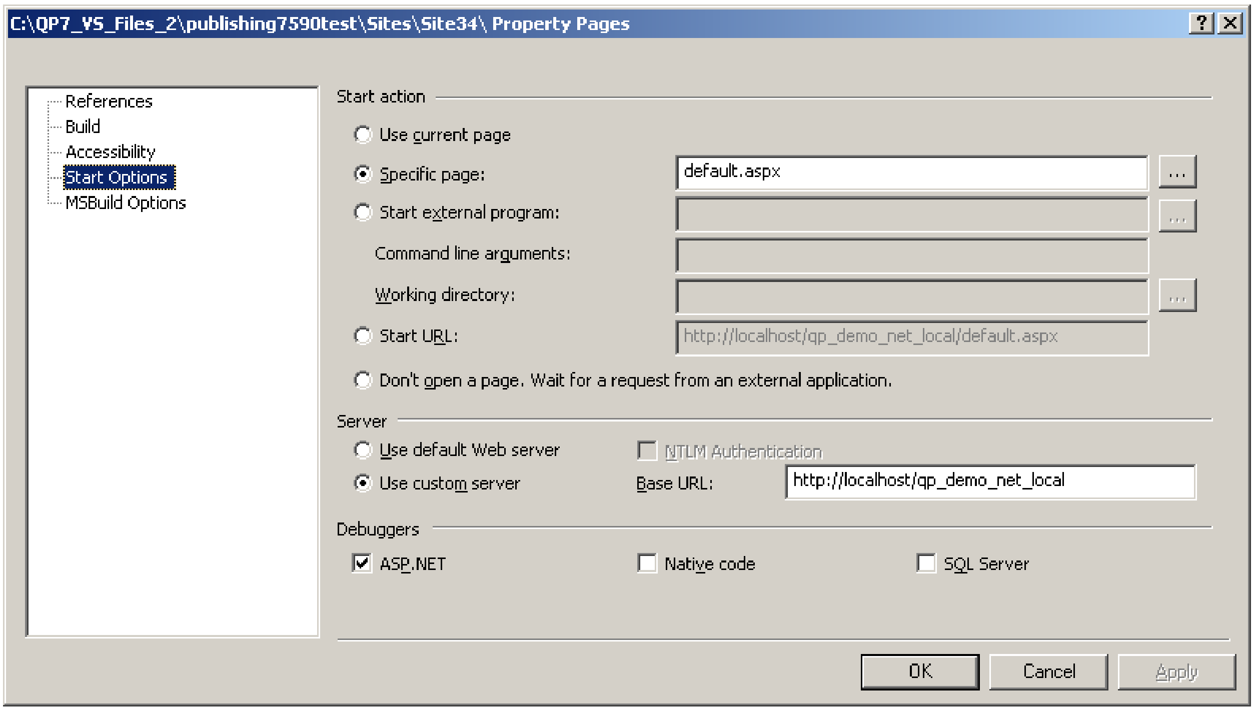


Figure 6-47

Give the WEB-server address and the name of the start page inserting your values as it's shown on the dialog.



C:\QP7_VS_Files_2\publishing7590test\Sites\Site34\ Property Pages

Start action

○ Use current page

● Specific page: default.aspx

○ Start external program:

Command line arguments:

Working directory:

○ Start URL: http://localhost/qp_demo_net_local/default.aspx

○ Don't open a page. Wait for a request from an external application.

Server

○ Use default Web server □ NTLM Authentication

● Use custom server Base URL: http://localhost/qp_demo_net_local

Debuggers

☑ ASP.NET □ Native code □ SQL Server

Figure 6-58

Since the address of local site is putting into the source code of page, regeneration of the source code of the whole site should be performed after finishing all parameters. To perform this "Get Latest Version" recursive operation is performing for the site. This operation should be performed every time when using WEB-server is changed.

After finishing all these steps the F5 button could be pressed (or perform Debug/Start Debugging command). Compilation of all pages will be performed. . In the case of errors information about them will be represented by standard methods of Microsoft Visual Studio – in the Error List window. In the case of successful compilation browser window with the Start page of site will open.

7 Appendix

7.1 File structure

All files for all connections are stored in the folder specified in the configuration settings of QP7.Backend Explorer (see Add-In Configuration). The structure of this folder is described below.

The root of the folder contains folder names that match the connection names.

Also this folder contains RecentList.xml, which contains information about the connections.

If the "Save Data Offline" flag was not set at the connection creation, the connection data will be deleted from disk after the connection is closed.

Each connection folder contains the following files:

- "connection_name.sln" – Microsoft Visual Studio Solution File.
- "connection_name.xml", "connection_name-schema.xml" – contains data for populating Add-in tree.

- “connection_name-webServiceParams.xml” – contains settings for accessing the web-services layer.

This file contains login/password/customer code for web-services access, therefore connection files should be kept in a safe place.

The structure of the rest of the folders matches the hierarchical structure of the tree where hierarchical tree elements are represented by hierarchical folders with an exception of presentation and code-behind elements which are represented by “Presentation.ascx” and “Code Behind.cs”.
Further file structure is repeating file structure of the WEB-site. This is the main difference from previous versions of the QP7 Backend Explorer.

7.2 Data Recovery

Some files may become damaged if an unrecoverable error happens while working with QP7 Backend Explorer.

If QP7.Backend Explorer cannot load the connection and the data on the local machine has not yet been uploaded to the server then follow the next steps:

1. Find out what file caused the error.
2. Copy “Sites” folder inside a connection to some other location on the hard drive.
3. Delete the connection via Add-In
4. Create new connection with the same name and the same parameters.
5. Exit out of Add-In
6. Replace the folder Sites inside the newly created connection folder with the previously copied folder “Sites”
7. Launch Add-In