

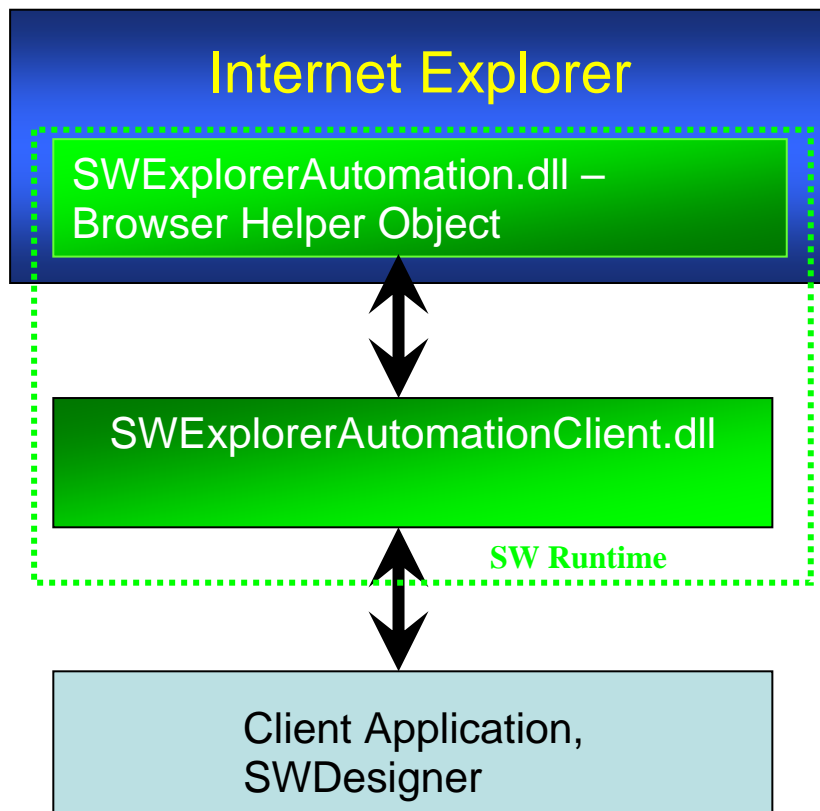
SW Explorer Automation Developers Guide

1 Introduction

SW Explorer Automation (SWEA) creates an object model (automation interface) for any Web application running in Internet Explorer. The automation interface consists of pages (scenes) and controls. The page consists of controls. The following controls are supported: HtmlContent, HtmlAnchor, HtmlImage, HtmlInputButton, HtmlInputCheckBox, HtmlInputRadioButton, HtmlInputText, HtmlSelect, HtmlTextArea. The object model is defined visually by SWEA designer. The designer allows to record scripts (C# and VB) based on the defined application object model.

2 Architecture

The system consists of SWExplorerAutomation runtime and SWDesigner. The runtime provides an object (programming) model for client applications.



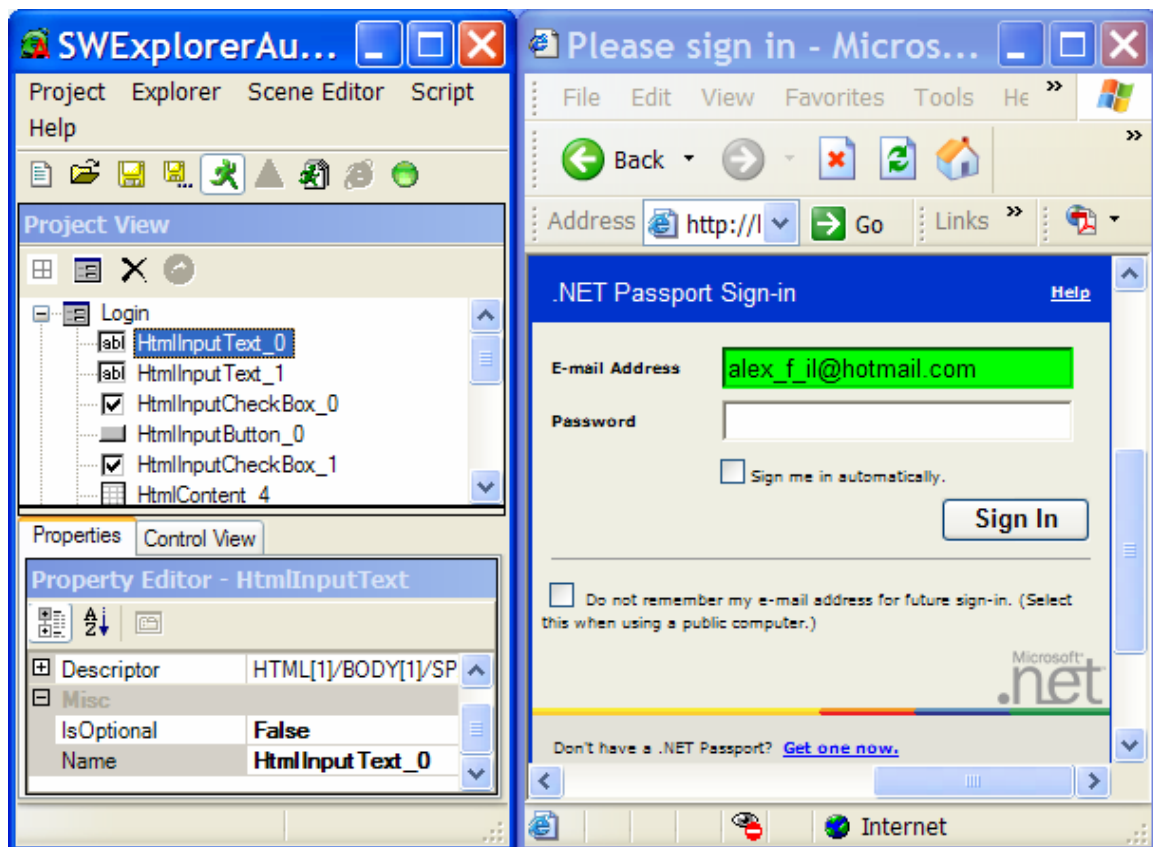
The object model can be visually created using SWDesigner. The object model consists of Scene (Page) and Control definitions. For every Web application page we want to work with, a scene should be created. The scene contains all controls defined for the page. The SWEA will activate the scene if it matches a Web page. When the scene is

activated all controls on the scene become available for scripting - control properties can be set or get.

The SWEA runtime uses scene and control descriptors to identify (match) scene to Web page. One scene can match many Web pages by using regular expressions in the scene and control descriptors.

3 Working with SWDesigner

The SWDesigner allows visually define scenes and controls for Web application pages. It transforms an Internet Explorer instance into a visual editor. Using mouse it is possible to select controls (HTML elements) or text fragments and define for them the programmable objects.




3.1 Getting Started

3.1.1 Defining a Scene



To define a scene for the Hotmail login page:

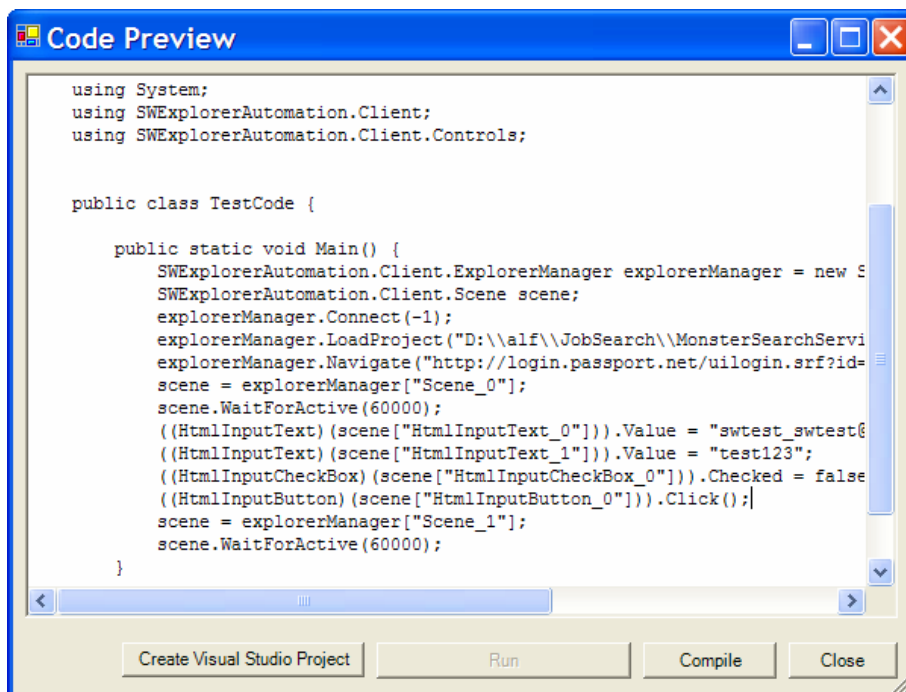
- From the **Explorer** menu, select **Run** .

- Navigate IE to <http://www.hotmail.com>.
- From the **Scene Editor** menu, select **Start** .
- Using mouse highlight (green) “E-Mail address” control.
- Click **SceneEditor\Add Selected Control Form** from the context menu or press ALT key. All the form controls will be added to the project.

3.1.2 Generating Script

To record a script for the scene:

- From the **Script** menu, select **Activate Recorder** .
- From the scene Context menu in Project View select **Navigate** .
- Select control in the Project View.
- Enter the control value into Control View.
- Repeat for all data entry controls (e-mail and password are required).
- Select **HtmlInputButton**. Press the simulated button in the Control View.
- From the Script menu, select **Generate C# Code**.
- Click on **Compile, Run** or **Create Visual Studio Project** buttons.



```

using System;
using SWExplorerAutomation.Client;
using SWExplorerAutomation.Client.Controls;

public class TestCode {

    public static void Main() {
        SWExplorerAutomation.Client.ExplorerManager explorerManager = new SWExplorerAutomation.Client.ExplorerManager();
        explorerManager.Connect(-1);
        explorerManager.LoadProject("D:\\alf\\JobSearch\\MonsterSearchService");
        explorerManager.Navigate("http://login.passport.net/ui/login.srf?id=scene=explorerManager[\"Scene_0\"]");
        scene.WaitForActive(60000);
        ((HtmlInputText)(scene["HtmlInputText_0"])).Value = "swtest_swtest@";
        ((HtmlInputText)(scene["HtmlInputText_1"])).Value = "test123";
        ((HtmlInputCheckBox)(scene["HtmlInputCheckBox_0"])).Checked = false;
        ((HtmlInputButton)(scene["HtmlInputButton_0"])).Click();
        scene = explorerManager["Scene_1"];
        scene.WaitForActive(60000);
    }
}

```

Buttons at the bottom: Create Visual Studio Project, Run, Compile, Close

The generated script:

```

namespace Test {
    using System;
    using SWExplorerAutomation.Client;
    using SWExplorerAutomation.Client.Controls;
}



```



```

public class TestCode {
    public static void Main() {
        SWExplorerAutomation.Client.ExplorerManager explorerManager =
            new SWExplorerAutomation.Client.ExplorerManager();
        SWExplorerAutomation.Client.Scene scene;
        explorerManager.Connect(-1);
        explorerManager.LoadProject("hotmail.htm");
        explorerManager.Navigate("http://login.passport.net/ui/login.srf?id=2");
        scene = explorerManager["Scene_0"];
        scene.WaitForActive(60000);
        ((HtmlInputText)(scene["HtmlInputText_0"])).Value = "someaddress @hotmail.com";
        ((HtmlInputText)(scene["HtmlInputText_1"])).Value = "password";
        ((HtmlInputButton)(scene["HtmlInputButton_0"])).Click();
        scene = explorerManager["Scene_1"];
        scene.WaitForActive(60000);
    }
}

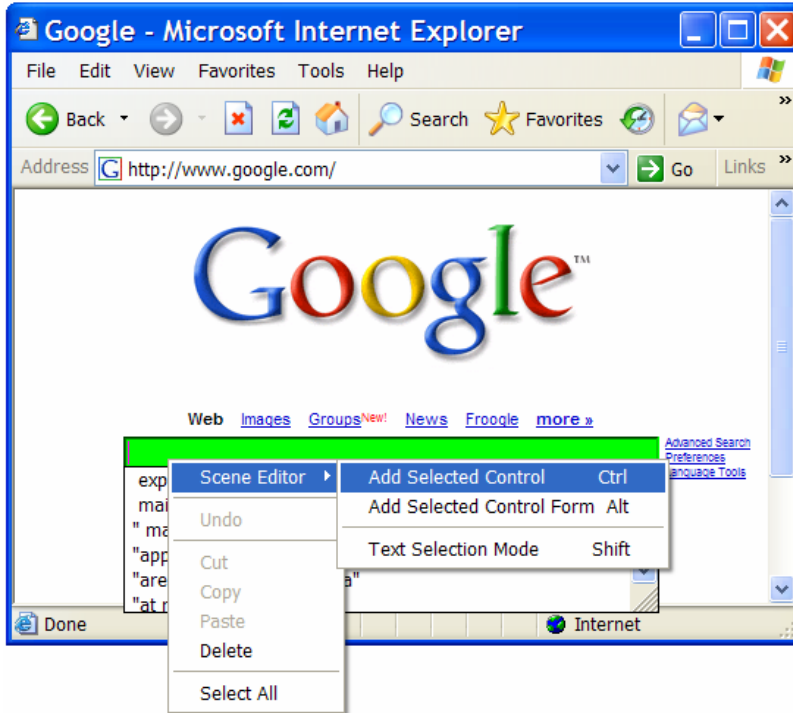
```

3.2 Internet Explorer Control Selector

The Scene Editor allows visually define controls for the automation model. The Scene Editor can be activated by selecting **Start** from the **Scene Editor** menu or by clicking on the  tool bar button. If the editor was activated the button icon is changed to , showing the active status of the editor.

The controls can be defined using HTML Tag Selection Mode (default) or Text Selection Mode. HTML Tag Selection Mode allows creating a control for highlighted (green) HTML tag. The editor can be switched to a Text Selection Mode using Scene Editor Context menu or by pressing **Shift** key. The mode allows creating a control for a tag which is a parent of the selected IE text. When the editor is switched to the Text Selection Mode the mouse cursor is changed from  to . In both modes the control is created by selecting Add Selected Control from the Scene Editor Menu or by pressing **Ctrl** key.

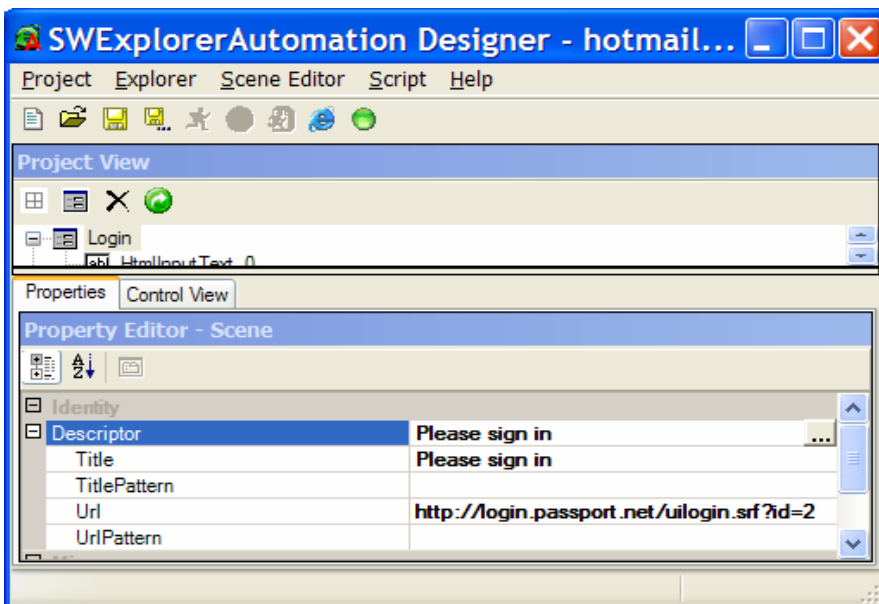
For the HTML forms, the editor allows to create controls for all HTML elements of a form in one step: by selecting Add Selected Control Form from the Scene Editor Context menu or by pressing **Alt** key.



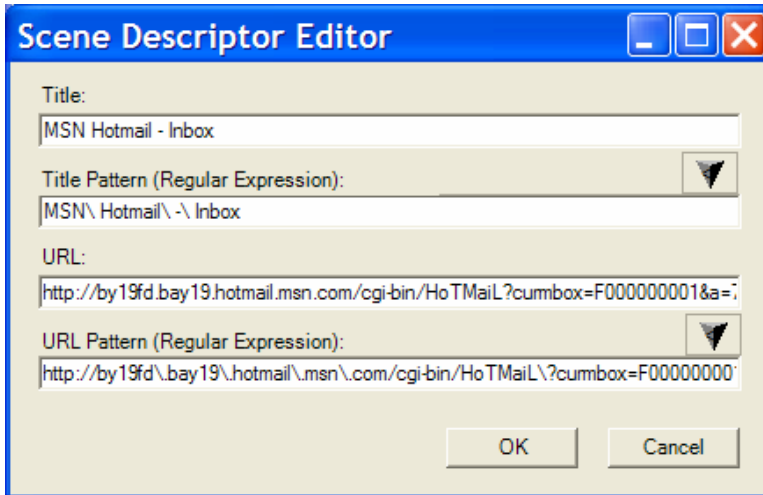
3.3 Scene Descriptor Editor

The Scene Descriptor editor is used to set up the Scene Descriptor properties. The properties are used in the Scene identification process.

The scene descriptor properties can be edited using Property Editor or Scene Descriptor Editor Dialog.



The properties are matched at runtime to the Web page data. Title and URL regular expression patterns allow runtime match one scene to many similar Web pages. It is possible to create a regular expression template – the arrow buttons create the default regular expressions. The following example shows how to use the regular expression patterns for Hotmail Web pages.



Hotmail URL:

<http://by19fd.bay19.hotmail.msn.com/cgi-bin/HoTMaiL?cumbox=F000000001&a=7552bcc6172c25f0c9e8258573d09cff&fti=yes>

Hotmail URL pattern:

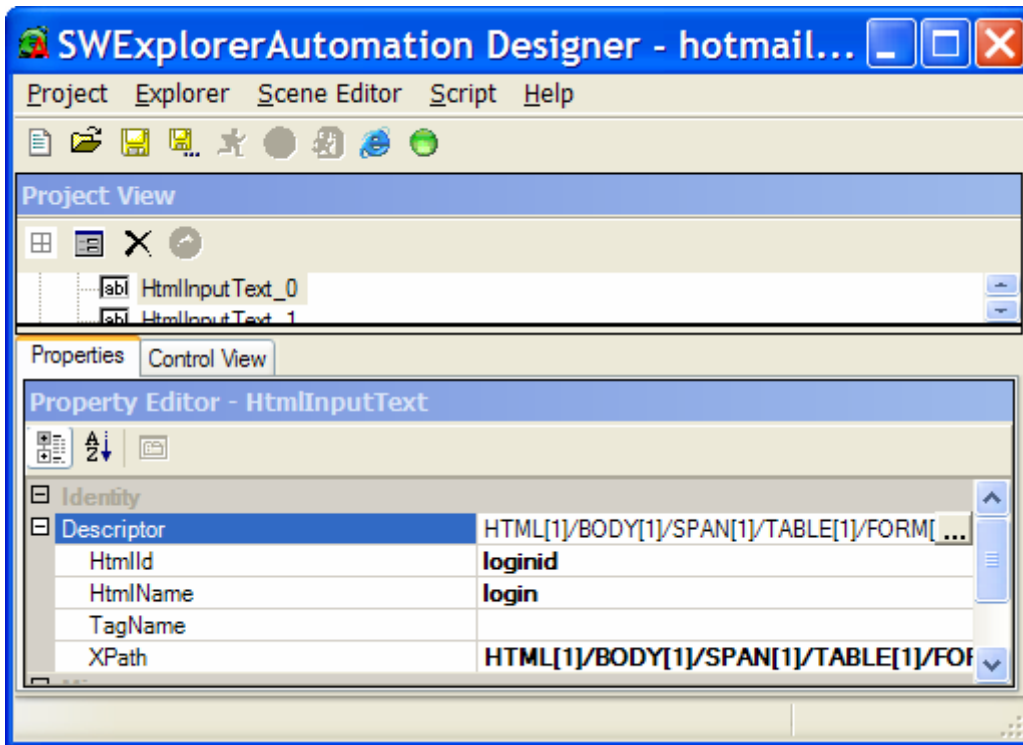
[http://by19fd\,bay19\,hotmail\,msn\,com/cgi-bin/HoTMaiL\?cumbox=\(.*\) &a=\(.*\)&fti=yes](http://by19fd\,bay19\,hotmail\,msn\,com/cgi-bin/HoTMaiL\?cumbox=(.*) &a=(.*)&fti=yes)

3.4 Control Descriptor Editor

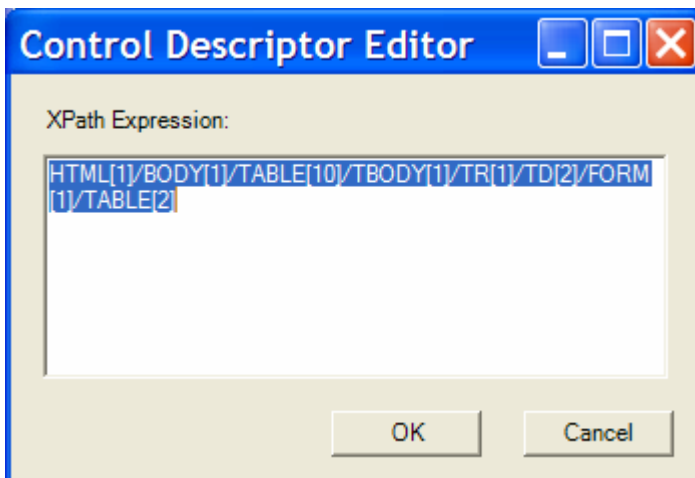
The control descriptor is used by the SWEA runtime to identify the control on the Web page. A scene will be activated only if all non optional controls on it are activated.

A control descriptor has the following properties: HtmlId, HtmlName and XPath. The properties are listed in the order in which the runtime will use them to identify a control.

The control descriptor properties can be edited using Property Editor or Control Descriptor Editor Dialog.



Control Descriptor Editor Dialog:

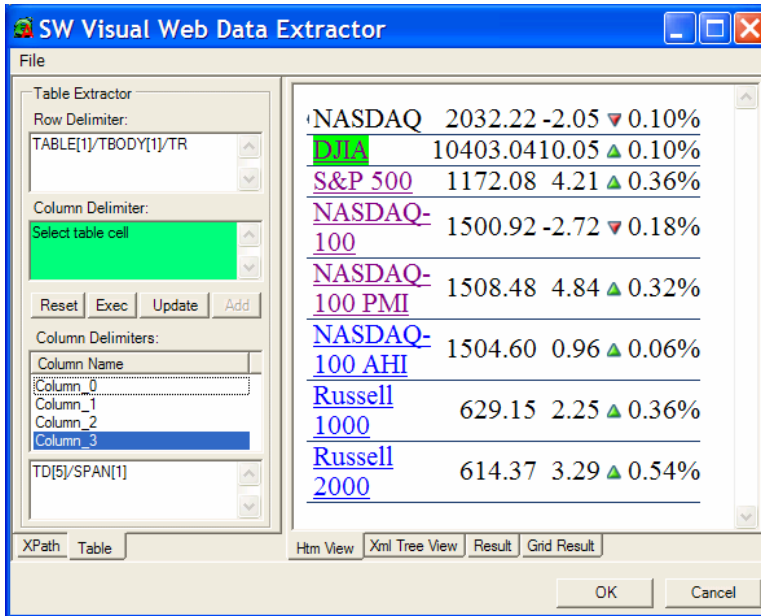


3.5 Visual Data Extractors

Data extractors allow extracting information from the Web pages. The extractors implemented as properties of **HtmlContent** control and are accessible in runtime from the script. There are **TableDataExtractor** and **XPathDataExtractor** data extractors.

3.5.1 Table Data Extractor

The Table Data Extractor extracts tabular data from the Web pages. If a Web page contains repeating information patterns than the data can be transformed into **ADO.NET DataTable** object.



To define the Table:

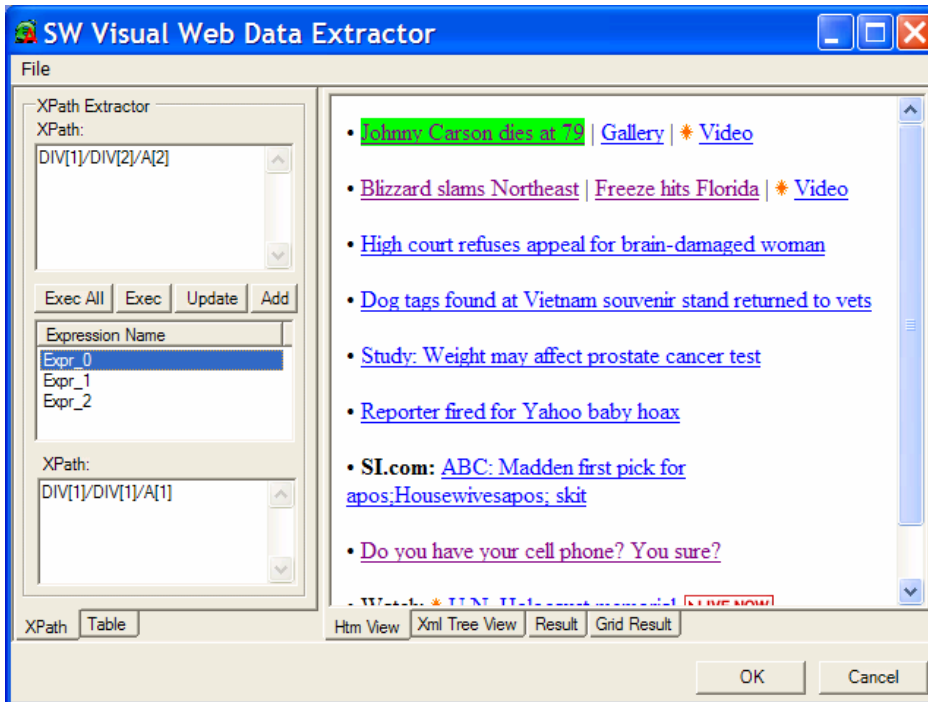
- Create **HtmlContent** control (can be done using Text Selection Mode) which covers the area from which the data should be extracted.
- Run custom **TableDataExtractor** property editor.
- Select (highlight and left mouse click) on the cell from the table. Select another cell from the same column.
- Click on the **Add** button to generate XPath expressions for column and row.
- Repeat for all columns.
- Click on **Exec** button to preview the extraction result in the Grid Result tab.
- Click on **OK** button to save the table definition.

To update the Table definition:

- Select (highlight and left mouse click) on the cell from the table. Select another cell from the same column.
- Click on **Update** button.

3.5.2 XPath Data Extractor

XPathDataExtractor allows visually define XPath expressions for the data extraction.



To define XPath expression:

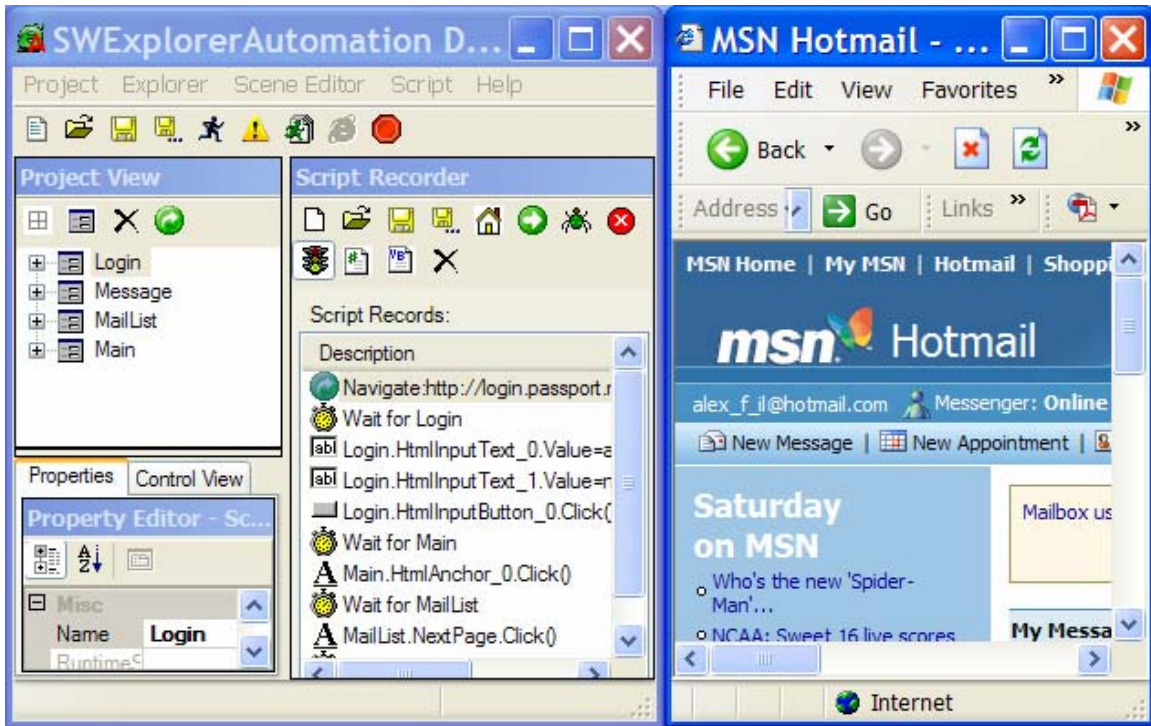
- Create **HtmlContent** control which covers the area from which the data should be extracted.
- Run custom **XPathDataExtractor** property editor.
- Select (highlight and left mouse click) on the data. Click on the **Add** button to save the named XPath expression.
- Click on **Exec** button to preview the extraction result in the **Result** tab.

To update the XPath expression:



- Select (highlight and left mouse click) on the data.
- Click on **Update** button.

3.6 Working with Script Recorder

The script recorder allows recording of automation scripts. The SW Designer simulates controls in Control View tab and records all actions with them into the Script Recorder project. The C# or VB.NET code can be generated from the Script Project. The Script Project consists of Navigation, Activation and Property Assignment records. The records properties can be updated by selecting the record in Script Record list.



3.6.1 Recording a script

The Script recorder records user actions with simulated control views. The recorder is activated by selecting **Activate Recorder** from the **Script** menu or by clicking  toolbar button. The recording can be suspended and resumed by toggling  toolbar button.




To record a script:

- From a scene context menu select **Navigate**.
- Sequentially select the controls of the scene. If the scene is active the **Control View** will show a simulated control for the selected scene control.
- Enter a data into the simulated control.
- Repeat for all data entry controls. If the control triggers IE navigation, wait until a scene will be activated.
- Click on the simulation of the navigation control (HtmlAnchor, HtmlButton).
- Wait for a scene to be activated.
- Repeat the previous actions for the scene.

3.6.2 Updating Script

The script records can be removed from the project and properties of any script record can be updated.

3.6.3 Playing Script

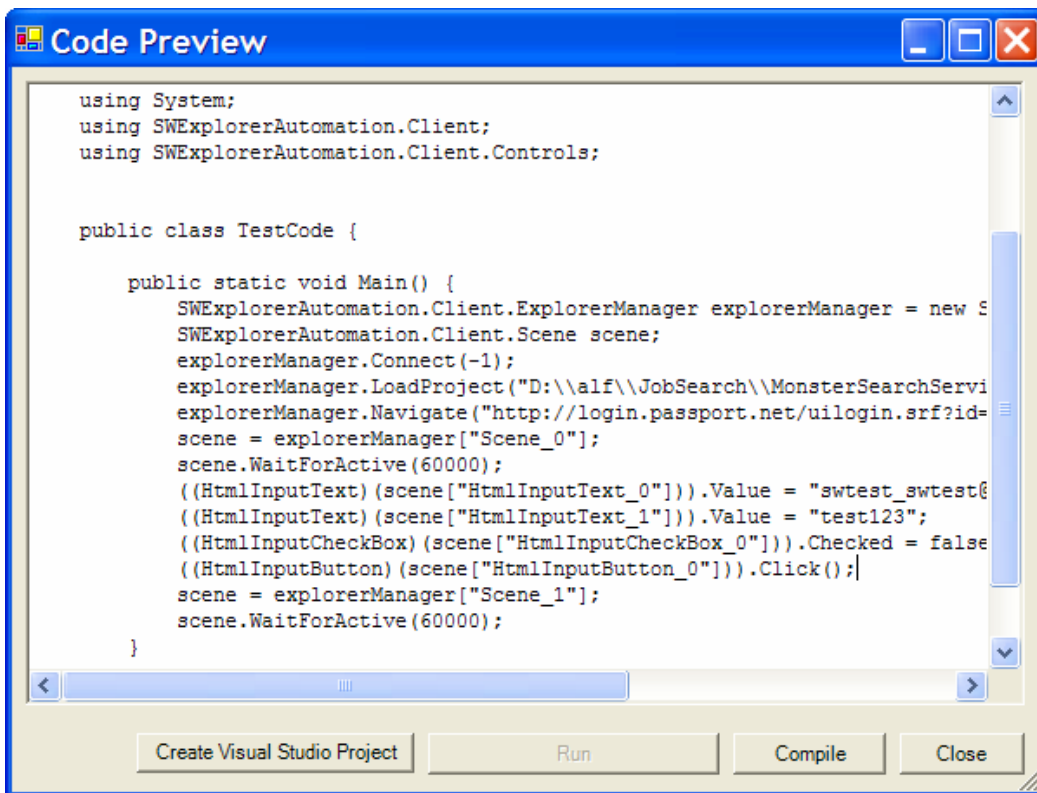
It is possible to play all script records at once  or to play them sequentially . To replay the script from the first script record, click on the Home  toolbar button.

3.6.4 Generating C# or VB.NET code

The Designer generates C# or VB.NET code from the Script Project.

To generate the code:

- From the **Script** menu, select **Generate C# code**; the **Code Preview** form will be popped up.
- To compile the code, click on **Compile** button.
- To create Visual Studio project, click on **Create Visual Studio Project**.



```
using System;
using SWExplorerAutomation.Client;
using SWExplorerAutomation.Client.Controls;

public class TestCode {

    public static void Main() {
        SWExplorerAutomation.Client.ExplorerManager explorerManager = new S
        SWExplorerAutomation.Client.Scene scene;
        explorerManager.Connect(-1);
        explorerManager.LoadProject("D:\\alf\\JobSearch\\MonsterSearchServi
        explorerManager.Navigate("http://login.passport.net/ui/login.srf?id=
        scene = explorerManager["Scene_0"];
        scene.WaitForActive(60000);
        ((HtmlInputText) (scene["HtmlInputText_0"])).Value = "swtest_swtest@
        ((HtmlInputText) (scene["HtmlInputText_1"])).Value = "test123";
        ((HtmlInputCheckBox) (scene["HtmlInputCheckBox_0"])).Checked = false
        ((HtmlInputButton) (scene["HtmlInputButton_0"])).Click();
        scene = explorerManager["Scene_1"];
        scene.WaitForActive(60000);
    }
}
```

3.6.5 Loading and saving script

The script project can be saved to or loaded from a file. The saved script project contains a reference to the screen definitions project. Loading of the script project also loads the screen definitions project.

4 Programming Model

The programming model consists of scenes and controls. The objects are defined visually using SWDesigner. When a scene matches a Web page all controls on it will become active and ready for scripting. See Table 1-1 for a description of mapping between the controls and HTML tags.

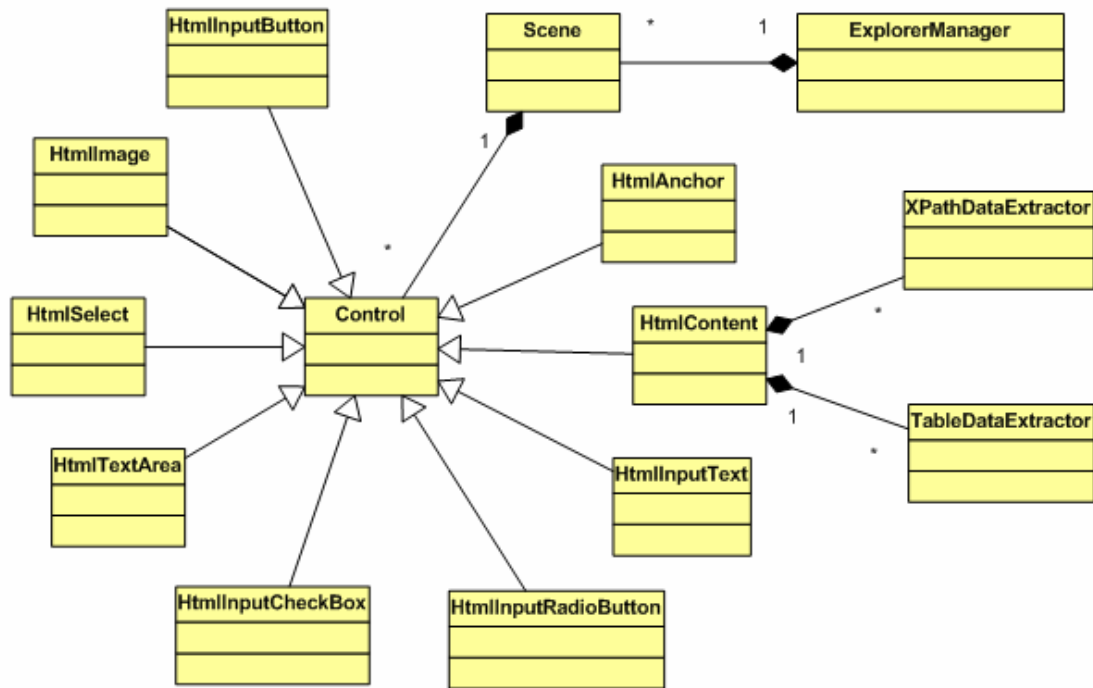


Table 1-1 Controls to HTML tags mapping.

Control Name	HTML tag
HtmlInputButton	<BUTTON> or <INPUT type='BUTTON INPUT_SUBMIT IMAGE'>
HtmlImage	
HtmlSelect	<SELECT>
HtmlTextArea	<TEXTAREA>
HtmlInputCheckbox	<INPUT type='CHECKBOX'>
HtmlInputRadioButton	<INPUT type='RADIO'>
HtmlInputText	<INPUT type='TEXT'>
HtmlAnchor	<A>
HtmlContent	All other tags

4.1 Working with objects

The scenes and controls are created dynamically by **ExplorerManager** from a project file, visually created in **SWDesigner**.

4.1.1 Loading project file

The project files are created by **SWDesigner**. The project file contains a scene and control definitions for an automation project.

```
// Create ExplorerManager Instance
ExplorerManager explorerManager = new ExplorerManager();
// Launch Internet Explorer
explorerManager.Connect();
// Load a scene and a control definitions
explorerManager.LoadProject("hotmail.htm");
```

4.1.2 Navigating Internet Explorer

The SWEA runtime matches scenes defined in the automation project to Web pages loaded by Internet Explorer. To start the matching Internet Explorer should be navigated to a Web application page.

```
SWExplorerAutomation.Client.Scene scene;
scene = explorerManager["Scene_0"];

// Navigate to Hotmail login page
explorerManager.Navigate("http://login.passport.net/ui/login.srf?id=2");
// Wait for the [Scene_0 ] to be activated. Throw exception in the case of timeout (60 seconds).
scene.WaitForActive(60000);
```

4.1.3 Waiting for Scene Activation

SWEA runtime runs the scene identification whenever a Web page is loaded into Internet Explorer. The **WaitForActive** function waits for a scene to reach the active state. There are two implementation of the function, one waits for one scene and one waits for multiple scenes to be activated.

```
scene.WaitForActive(60000);

String[] activatedScenes = explorerManager.WaitForActive(new String[] { "Scene_1", "Scene_2" }, 30000);
```

4.1.4 Forcing the scene identification

The SWEA runs the scene identification according to IE events. Unfortunately in some cases the IE doesn't fire the events correctly. It is possible to force the scene identification by calling **RunIdentification** function.

```

if (!scene.WaitForActive(60000)) {
    explorerManager.RunIdentification();
    if (!scene.WaitForActive(60000)) {
        throw new Exception("Scene was not activated")
    }
}

```

4.1.5 Working with controls

When a scene matches a Web page it will be activated - all controls on the scene became available for the scripting.

```

((HtmlInputText)(scene["HtmlInputText_0"])).Value = "someaddress @hotmail.com";
((HtmlInputText)(scene["HtmlInputText_1"])).Value = "password";
((HtmlInputButton)(scene["HtmlInputButton_0"])).Click();




```

If the control property **IsOptional** is **true**, the control can be inactive while the scene which contains it is active. The optional controls help to solve problems with a paged data. Usually Web pages with the data have previous and next links, but previous is not present on a first page and next is not present on a last page. Making the controls optional solves the problem, without creating additional scenes for the Web pages. It is possible to verify status of a control by calling control's **IsActive()** function.

4.1.6 Working with Events

The SWEA supports both synchronous and asynchronous programming models. In most situations synchronous model is sufficient. The asynchronous model can be used mostly for error processing and tracing. The asynchronous model is provided by **ExplorerManager** class events.

The events of the **ExplorerManager** class are listed here. For a complete list of **ExplorerManager** class members, see the [ExplorerManager Members](#) topic.

 SceneWaitStatus	Occurs repeatedly while runtime wait for scene activation. Not implemented
 SceneActivated	Occurs when SWEA runtime matches the scene to a Web page
 SceneDeactivated	Occurs when the matched Web page is

	unloaded from the Internet Explorer
⚡ ServerError	Occurs when an error happens
⚡ ApplicationDisconnected	Occurs when Internet Explorer is closed
⚡ DialogActivated	Occurs when Dialog Window is popped up
⚡ DialogDeactivated	Occurs when Dialog Window is closed
⚡ FileDownload	Occurs before IE downloads file

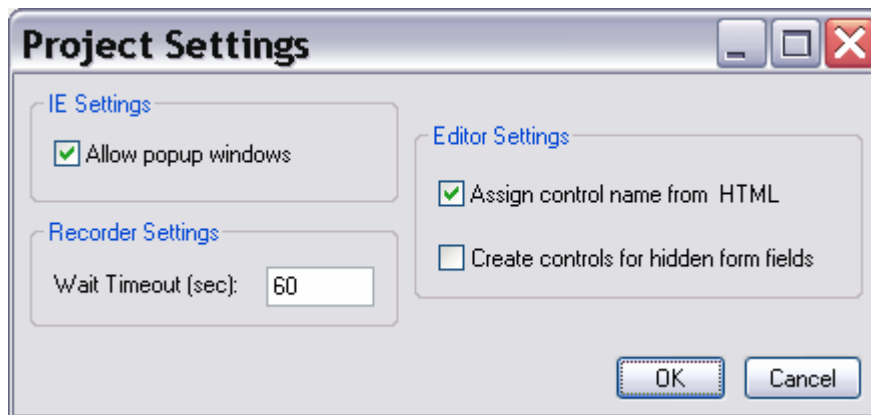
5 Deployment

Any **SWExplorerAutomation** applications requires **SWExplorerAutomation.dll**, **SWExplorerAutomationClient.dll** and project (*.http) files to be distributed to the target machines. The files may be deployed to target computers in two ways. One way requires running of SWExplorerAutomation setup. Either way is to simply copy **SWExplorerAutomation.dll** and **SWExplorerAutomationClient.dll** and manually register the **SWExplorerAutomation.dll** file on the target machine.



6 Advanced

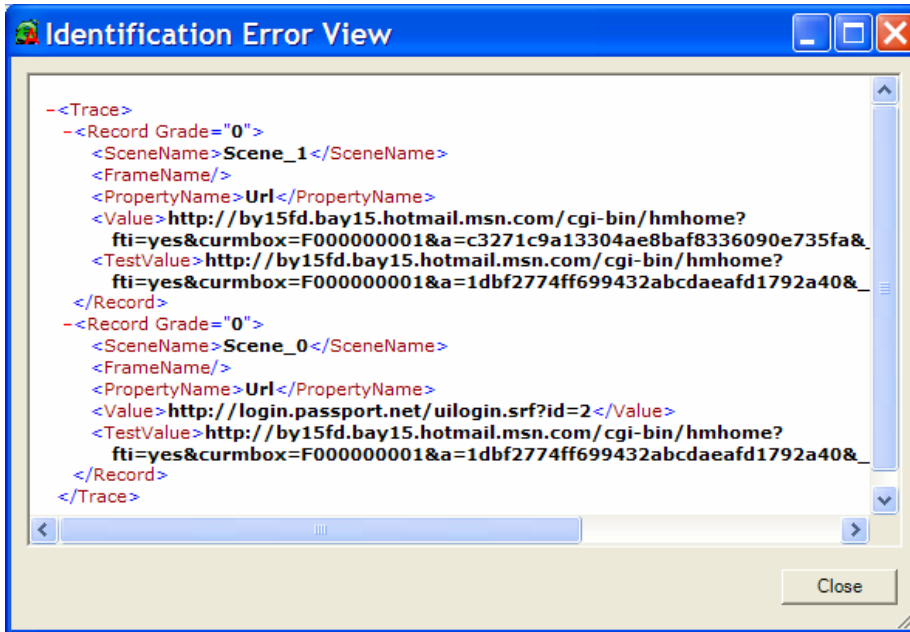
This chapter describes advanced techniques that may be used in a SWEA client development.

6.1 Advanced Settings



6.2 Understanding Identification Log

If SWDesigner toolbar  shows  it means that none of the project scenes was identified. The toolbar icon activates **Identification Error View** dialog. The dialog shows the reasons why the scenes were not identified.



The example view shows that Scene_1 was not identified because **Url** (http://by15fd.bay15.hotmail.msn.com/cgi-bin/hmhome?fti=yes&curmbox=F000000001&a=c3271c9a13304ae8baf8336090e735fa&c3271c9a13304ae8baf8336090e735fa&_lang=EN&country=US) of the Web page doesn't match **Url** property (http://by15fd.bay15.hotmail.msn.com/cgi-bin/hmhome?fti=yes&curmbox=F000000001&a=1dbf2774ff699432abcdaeafd1792a40&_lang=EN&country=US) of the Scene Descriptor. The identification problem can be solved using URL Regular Expression Pattern of the Scene Descriptor. Setting the property to [http://by15fd.bay15.hotmail.msn.com/cgi-bin/hmhome?fti=yes&curmbox=F000000001&a=\(.*\)&_lang=EN&country=US](http://by15fd.bay15.hotmail.msn.com/cgi-bin/hmhome?fti=yes&curmbox=F000000001&a=(.*)&_lang=EN&country=US) will solve the identification problem.

6.3 Popup dialog boxes

The SWEA allows managing Dialog boxes popped up by IE. The **Dialog Scene** is automatically created and populated with controls whenever Internet Explorer pops up a dialog window. The following code simulates click on [OK] button of any popped up dialog window.

```
explorerManager.DialogActivated +=new DialogActivatedEventHandler(explorerManager_DialogActivated);

static public void explorerManager_DialogActivated(object o, DialogScene s) {
    if (s.ControlExists("OK")) {
        (s["OK"] as DialogButton).Click();
    }
}
```

6.4 Downloading files

Web pages can contain file (PDF, ZIP, etc) references. There are two ways to download a referenced file: synchronous and asynchronous.

FileDownload event is used to download file asynchronously. The event is fired when Internet Explorer is about to download file.

```
static void Progress(object sender, Int64 size) {
    Console.WriteLine("Downloaded {0}", size);
}

static void explorerManager_FileDownload(object sender, FileDownloader fd) {
    FileDownloader f = fd;
    f.DownloadProgress += new DownloadProgressEventHandler(Progress);
    f.DownloadFile(Path.GetTempFileName(), true, true);
    Console.WriteLine("\n Press Enter to exit");
}

explorerManager.FileDownload += new FileDownloadEventHandler(explorerManager_FileDownload);
```

The following example shows how to download file referenced from active scene synchronously.

```
scene.DownloadFile(downloadUrl, fileName, true);
```

6.5 Converting Web Pages into images

Any Web page loaded into Internet Explorer can be converted into an image.

```
explorerManager.GetExplorerWindowImage(scene.RuntimeSceneInfo.HWND).Save(@"c:\temp\image.bmp");
```

6.6 Unhandled popup dialog boxes

If the popup dialog is shown SWEA but not handled by client, SWEA runtime will throw an exception and will close the pop-up window.

6.7 Advanced Control Descriptor XPath expressions

The Control Descriptor has XPath property. The property is used to identify a control on the Web page. There are two types of the XPath expressions, simple and complex. The simple expression is an absolute XPath path, starting from the root (HTML tag). It should not contain any predicates. The complex expression can be any XPath expression, without limitations.

The SWDesigner creates simple expressions in most cases. The complex expressions can be created only manually, by editing the XPath property.

The complex expression allows defining scenes which are more tolerant to a Web page changes.

6.8 Invoking any Html element function or property

The SWEA defines and binds a SWEA control for the HTML element. It is possible to invoke any property or function of the HTML element. Any SWEA control has Invoke function. It is possible to call function or property of a HTML element. The first parameter of the function is the action to be invoked. There are three prefixes of the action “Get_”, “Let_” and “Method_”.

To set property of Html element:

```
control.Invoke(“Let_Value”, “NewValue”);
```

To get property of Html element:

```
String html = (String) Control.Invoke(“Get_InnerHtml”, “”);
```

To invoke method of HTML element:

```
String html = (String) Control.Invoke(“Method_Click”, “”);
```

There is a more advanced technique of invoking HTML element by using HtmlContent control. It is possible to invoke any HTML element which is located under the base tag of the HtmlContent control. It is possible to invoke HTML element using XPath expression or unique element identifier.

The XPath expression should be able to select the element from XML returned by OuterXml property of HtmlContent control.

```
HtmlContent content;  
.....  
string val = content.InvokeHtmlElementByXPath (“TABLE/TBODE/TR/TD/INPUT”, “Get_Value”, “”);
```

Every XML Element of XML returned by OuterXml property has [id] attribute. The attribute value is generated by SWEA runtime and unique.

```
HtmlContent content;  
.....  
string xml = content.OuterXml;  
string elementId = ..... // get id from the XML  
string val = content.InvokeHtmlElement(elementId, “Get_Value”, “”);
```

7 Frequently Asked Questions