Understanding how TEST STUDIO adds to VISUAL STUDIO'S Testing Capabilities

Telerik
TEST STUDIO

www.telerik.com/test-studio   www.twitter.com/teleriktesting

This document has been designed to help clarify the way Telerik Test Studio goes beyond Visual Studio 2010 /2012 testing products (Visual Studio 2010 Test Professional and Ultimate, as well as Visual Studio 2012 Ultimate, Premium and Test Professional) to offer richer user experience and enhanced productivity. Its purpose is also to help visitors understand the key challenges with test automation that we are trying to solve with our solution beyond just the "Hello World" test case.

With the release of Visual Studio 2010, Microsoft has introduced the capability to automate the user interface (UI) of applications on the Windows platform. Visual Studio 2010 introduces the CodedUI test framework and the MTM (Microsoft Test Manager) as solutions for Testers and QAs to help manage test creation and test automation. We believe this to be a great step forward and a win for our customers especially in the areas of Test and Lab management. The Visual Studio / .NET platform has traditionally been exclusive to developers but now with VS2010 it also encompasses testers and QA professionals.

BENEFITS	Test Studio	Visual Studio 2010	Visual Studio 2012
Level of complexity	Technical and non-technical	Development skills required	Development skills required
Point-and-click UI	Yes	Limited	Limited
Easy to maintain tests	Yes	No	No
Silverlight automation	Yes (full record/play support; no automation peers are required)	Only for Silverlight 4	Yes
Test step keyword view	Yes	No	Limited
Browser execution support	IE, FF, Safari, Chrome	IE, FF	IE, FF, Chrome (requires Selenium)
Native support for Telerik UI components and Kendo UI widgets	Yes	No	No
Web-specific testing features	Yes	Yes	Yes
Data-driven testing	Automated	Manual (coding experience required)	Manual (coding experience required)
Licensing model	Standalone app and a plugin that integrates with VS Professional and above	Visual Studio Premium/ Ultimate	Visual Studio Premium/ Ultimate
Product updates	2 major, 4 minor/ year	Unknown	Unknown
Premier support (response within 24h)	Yes	No	No

The specialized vs. "One-Size-Fits-All" solution: how Test Studio caters to your testing needs

The Visual Studio approach to test automation is to offer a broad solution that is a good entry point for developers looking to begin test automation. As your automation needs grow, you may run into challenges that arise from Coded UI's single approach for covering multiple technologies.

With Test Studio we address the specific automation needs of web and desktop applications. We studied the workflow of testers when automating web and installable apps, and their respective testing life-cycles which helped us identify areas that take the longest to automate or are very tedious. We've built many specialized features that cater to these needs that you will not find in Visual Studio; here are a few worth noting:

DOM Explorer

A full DOM explorer for your web application that supports nested frames and allows you to do rich search across the entire DOM. You can also navigate directly to specific nested Frame or iFrame with one right click and directly craft verification from it. This feature reduces your reliance on external tools and reduces the context switching between our tool and others.

Watch a short video on Test Studio's DOM Explorer

Web Verifications

Build computed or in-line style verifications directly from the UI in addition to the attribute/content and CSS visibility verifications. Such scenarios are important when validating web element positioning and layering.

DOM Captures on Failure

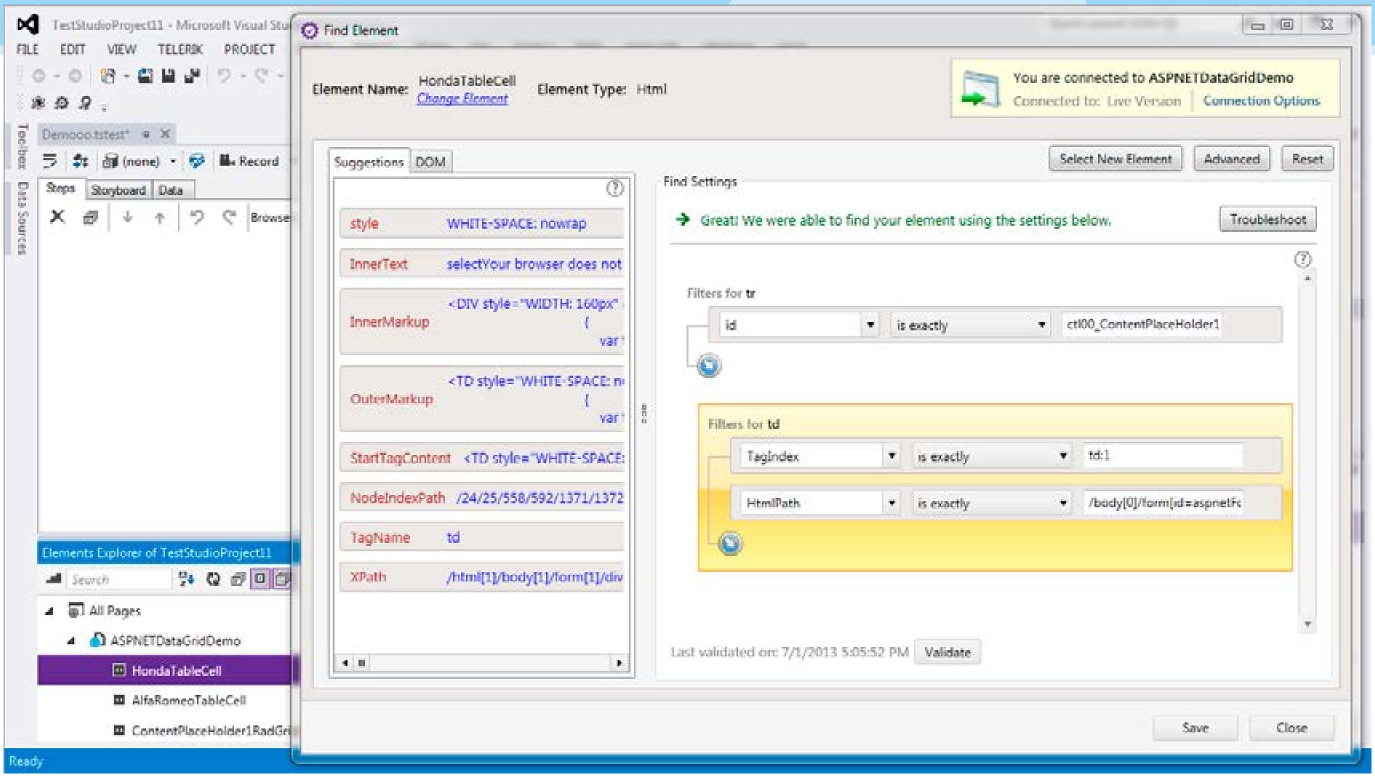
When your test fails, we capture the DOM of the page at the state of failure and display it for you to help you resolve your failure easier.

Execution Intelligence

The execution engine is built to handle AJAX solutions from the ground-up and if you are executing a web test, the engine will intrinsically ensure the element exists and is visible on the page before running each step. The check for visibility uses the actual HTML/CSS visibility checks rather than UI Automation, which is much more reliable. For Silverlight and WPF scenarios we also ensure the element is not moving by doing a motion check on the actual visual element. This feature enhances the reliability of your execution dramatically so that you don't need to clutter your tests with sleeps and waits.

Find Expressions & Live Validation

As many experienced testers will tell you, tests frequently fail due to element changes in the application as it evolves. The Find Expression is a UI editor that allows you to easily resolve element find failures with a suggest feature to help you understand why the find failed. It also comes with an integrated DOM so that you can validate right there and then that you are finding the correct element instead of having to execute the test with each edit. A great time saver! ↓


TestStudioProject11 - Microsoft Visual Stu
FILE EDIT VIEW TELERIK PROJECT
Demooo.tstest*
Record
Steps Storyboard Data
Elements Explorer of TestStudioProject11
All Pages
ASPNETDataGridDemo
HondaTableCell
AlfaRomeoTableCell
ContentPlaceHolder1RadGri
Ready
Find Element
Suggestions DOM
Select New Element Advanced Reset
Troubleshoot
Save Close

Element Name: HondaTableCell *Change Element* Element Type: Html

You are connected to ASPNETDataGridDemo
Connected to: Live Version | Connection Options

style	WHITE-SPACE: nowrap
InnerText	selectYour browser does not
InnerMarkup	<DIV style="WIDTH: 160px" { var
OuterMarkup	<TD style="WHITE-SPACE: n { var
StartTagContent	<TD style="WHITE-SPACE:
NodeIndexPath	/24/25/558/592/1371/1372
TagName	td
XPath	/html[1]/body[1]/form[1]/div

Find Settings

Great! We were able to find your element using the settings below.

Filters for tr

id	is exactly	ctl00_ContentPlaceHolder1

Filters for td

TagIndex	is exactly	td:1
HtmlPath	is exactly	/body[0]/form[id=aspnetFc

Last validated on: 7/1/2013 5:05:52 PM Validate

Coded vs. Code-less (script-less) Automation

Do I always have to write code to automate my test? What if my tester is not proficient in C# or VB.NET? Am I stuck with manual testing?

We don't believe you always need to write code to build test automation. In Visual Studio, to properly perform test automation of real line-of-business applications you always have to revert to the CodedUI test which is a pure C# or VB.NET class that leverages the CodedUI framework for automation. The element mapping in Visual Studio - also known as the UIMap - generates a complex hierarchy of classes to represent the elements as strongly-typed objects in code. This approach is not very convenient when it comes to test maintenance. For example, a simple 15-step test in Visual Studio generates around 1000 lines of code for the test method.

Yes, there will always be cases when you will need to revert to code but that doesn't mean your entire test has to be created and / or maintained in code.

Test Studio has been designed to allow most QA/Test professionals to automate without having to revert to code in most scenarios. Even complex conditional logic, element extraction, and data driven tests can be created and maintained without writing a single line of code.

When complex logic is needed, a step or two out of your entire test can be converted and written in C# or VB.NET code without having to convert the entire test. Such approach not only simplifies test automation for non-developer QA professionals but also makes test maintenance a lot easier. You can update your test by changing properties rather than editing code and doing code refactoring. Many of our customers that use Test Studio end up writing 90-95% of their test automation without having to code anything. Minimizing complex code will increase your QA Teams productivity, allowing more time to create a larger regression suite resulting in better software, delivered on time and on budget.

UIAutomation vs. Full Application Access

UIAutomation technology is the evolution of the Accessibility API which is the mechanism by which screen readers and other aided technologies enable visually-impaired users to interact with a computer. Since 2005 Microsoft have been pushing UIAutomation as a solution not only for accessibility tools but also for test automation.

UIAutomation depends on application level hooks that give access to external programs to invoke or retrieve information from the application. UIAutomation is 100% dependent on the developer of the application giving access to these parts of the application; else those parts can't be automated. With UIAutomation we found ourselves constantly hitting the wall of "not exposed by developer" argument and we ended up having to run tests manually because developers didn't have enough cycles to expose needed functionality.

Furthermore, UIAutomation does not offer real-user simulation. When calling a UIAutomation API you are actually invoking a method call within your application. The user experience could actually be a lot different for many applications.

When building Test Studio, we wanted to build a solution for testers that gives them freedom and full access to the application under test without limitations. Therefore we completely avoided any heavy reliance on UIAutomation and used it only in certain scenarios. Our architecture relies on having full access to the application whether it is a XAML, MVC application or an HTML page. You can access any and every element in the application and perform real-user actions against it.

The Visual Studio automation solution is almost completely architected on UIAutomation.

Cross-Browser Support

Visual Studio 2010 added support for Firefox thus allowing users to execute tests against two major browsers – Internet Explorer and Firefox. Visual Studio 2012 was finally able to include Safari and Google Chrome by leveraging the Selenium framework and its support for these browsers.

Not only does Test Studio offer test execution support for IE, FF, Safari and Chrome out of the box, it's the first tool to add cross-browser test recording capabilities with its R1 2013 product update.

XAML Automation

Visual Studio 2012 added support of Silverlight 5 applications. Test Studio has been automating against Silverlight since version 2 (2009) and has made many refinements in our record/playback compatibility with rich specialized features for Silverlight and WPF applications automation including:

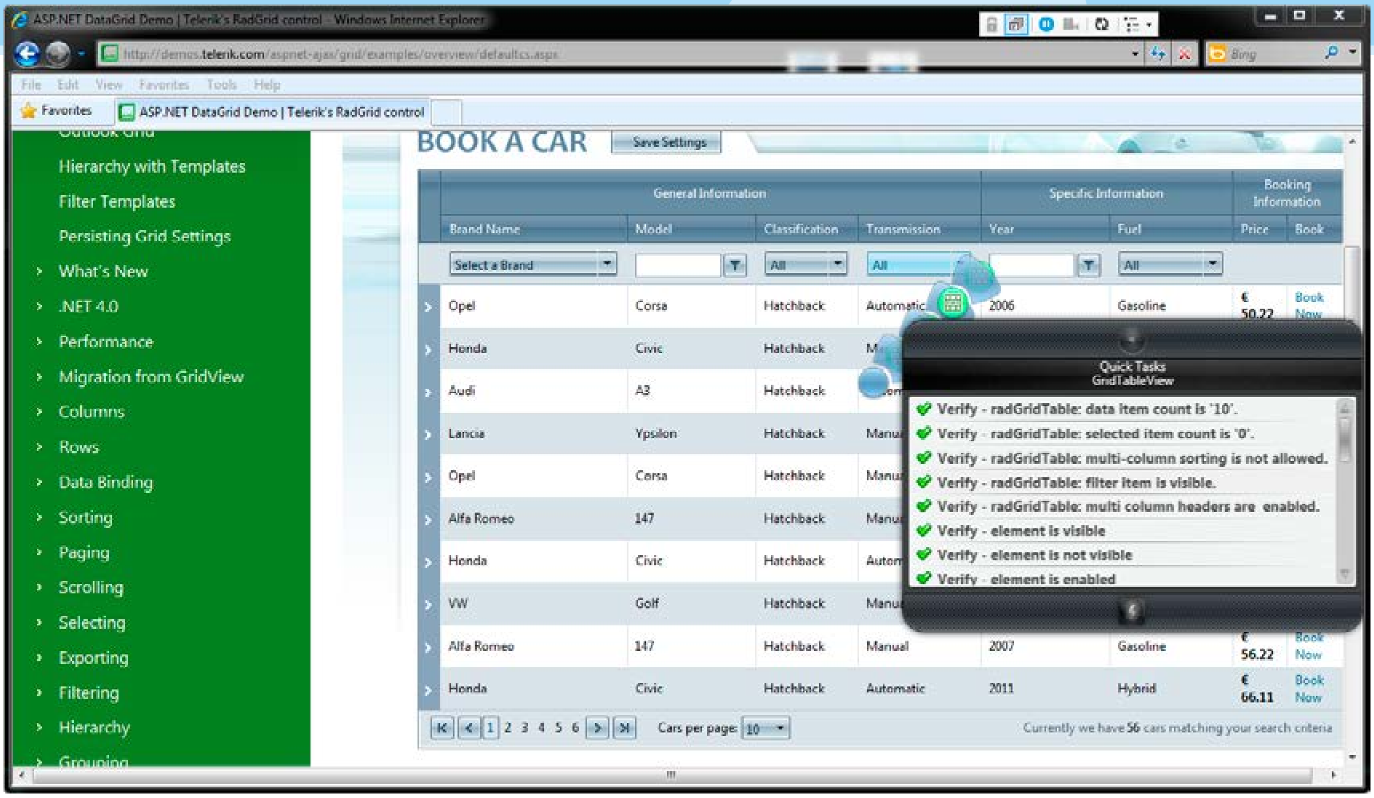
- **Visual Tree inspection:** View the entire visual tree of your Silverlight application and target specific elements for verifications.
- **Rich support for out-of-box controls:** All the out-of-box controls that ship with Silverlight and WPF are supported by Test Studio with specialized translators for each control. For example, when using the Silverlight/WPF Calendar control and selecting a date, our translators understand the control being used and can determine the action you are performing. So instead of recording a click against a low-level element, it records a select for a specific date. This makes the test easy to understand, and easy to update.
- **Custom controls support:** From experience with real-business applications that customers are trying to automate, we found that many applications use custom controls built in house by the development team. Some inherit from Panels/Calendars/Checkboxes...etc. Our tool provides a smart control matching feature that allows the tool to drop to the base control type and offer the common tasks for that control instead of dropping to just a raw base element with only generic tasks associated with it.

Native Support for RadControls for AJAX, Silverlight, WPF and Kendo UI Web widgets

If you use Telerik UI components (ASP.NET AJAX, Silverlight and WPF) or Kendo UI Web widgets in the application you are trying to automate, then you won't find a more suited test automation tool for your application than Test Studio.

Test Studio comes with rich support for each component:

- The components are automatically detected on your page visually.
- When performing an action within the control, the action is routed through a special translator built for each control to help identify the action at higher contextual level than a raw click, for example: TreeView=>Expand, Calendar=>SelectDate, DropDown=>Select Item
- Each control has a set of specialized verification and synchronization tasks that are built dynamically for your application in the desired state you want verified:



General Information				Specific Information		Booking Information	
Brand Name	Model	Classification	Transmission	Year	Fuel	Price	Book
Select a Brand		All	All		All		
Opel	Corsa	Hatchback	Automatic	2006	Gasoline	€ 50.22	Book Now
Honda	Civic	Hatchback	M				
Audi	A3	Hatchback					
Lancia	Ypsilon	Hatchback	Manu				
Opel	Corsa	Hatchback	Manu				
Alfa Romeo	147	Hatchback	Manu				
Honda	Civic	Hatchback	Autom				
VW	Golf	Hatchback	Manu				
Alfa Romeo	147	Hatchback	Manual	2007	Gasoline	€ 56.22	Book Now
Honda	Civic	Hatchback	Automatic	2011	Hybrid	€ 66.11	Book Now

1 2 3 4 5 6 Cars per page: 10 Currently we have 56 cars matching your search criteria

Quick Tasks
GridTableView

- ✓ Verify - radGridTable: data item count is '10'.
- ✓ Verify - radGridTable: selected item count is '0'.
- ✓ Verify - radGridTable: multi-column sorting is not allowed.
- ✓ Verify - radGridTable: filter item is visible.
- ✓ Verify - radGridTable: multi column headers are enabled.
- ✓ Verify - element is visible
- ✓ Verify - element is not visible
- ✓ Verify - element is enabled

TELERIK RADCONTROLS FOR
ASP.NET AJAX

TELERIK RADCONTROLS FOR
Silverlight

TELERIK RADCONTROLS FOR
WPF

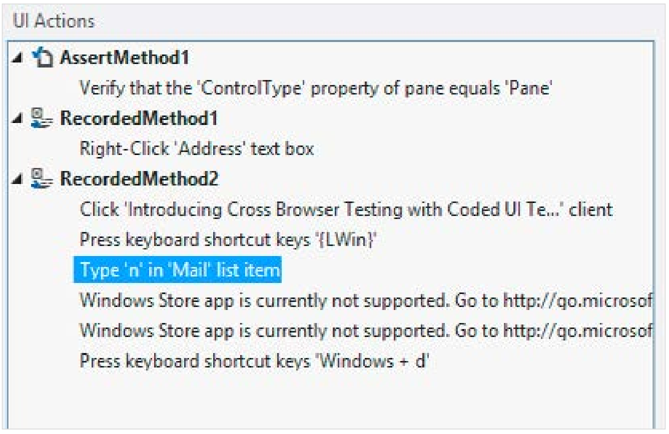
Kendo UI
THE WAY OF HTML5

With Test Studio you will have the ability to:

- Spend more time building test scenarios to ensure the quality of your application rather than trying to figure out how to automate an action on the control and understanding the complexity of its rendering.
- You don't need to worry about backward compatibility. With each Telerik product release, the translators are updated so your test continues to work without having to touch any of your tests.

UI Actions view vs. Test Studio's Test Explorer

Visual Studio 2012 introduced a UI Actions view where you can see all your recorded steps. The view is very basic and offers little flexibility in editing, maintaining and visualizing your tests.

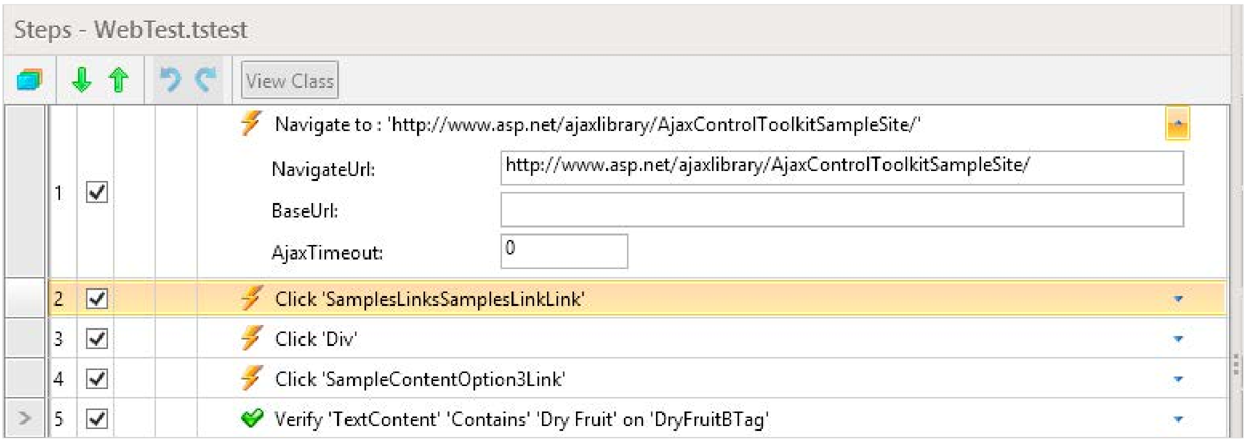

UI Actions

- AssertMethod1
  - Verify that the 'ControlType' property of pane equals 'Pane'
- RecordedMethod1
  - Right-Click 'Address' text box
- RecordedMethod2
  - Click 'Introducing Cross Browser Testing with Coded UI Te...' client
  - Press keyboard shortcut keys '{LWin}'
  - Type 'n' in 'Mail' list item
  - Windows Store app is currently not supported. Go to http://go.microsof
  - Windows Store app is currently not supported. Go to http://go.microsof
  - Press keyboard shortcut keys 'Windows + d'

VS 2012

In Test Studio, the Test Explorer provides the user with an extension and rich experience that is currently not offered in Visual Studio. For example, in Test Studio's Test Explorer you can:

1. Enable/Disable specific steps.
2. Edit steps directly in the explorer without having to switch windows.
3. Recorder steps by doing drag/drop.
4. Convert verifications into waits or vice-versa
5. Componentize your tests and refactor them using Test As Steps feature without having to revert to code refactoring.
6. Place visual break-points for execution
7. User for-loops, if-else...etc. without the need to write code.
8. Extra dynamic values from applications and bind them to subsequent steps.


Steps - WebTest.tstest

View Class

1	✓	Navigate to : 'http://www.asp.net/ajaxlibrary/AjaxControlToolkitSampleSite/'
		NavigateUrl: http://www.asp.net/ajaxlibrary/AjaxControlToolkitSampleSite/
		BaseUrl:
		AjaxTimeout: 0
2	✓	Click 'SamplesLinksSamplesLinkLink'
3	✓	Click 'Div'
4	✓	Click 'SampleContentOption3Link'
5	✓	Verify 'TextContent' 'Contains' 'Dry Fruit' on 'DryFruitBTag'

Test Studio

At the end of the day, our Test Explorer is not meant to be a tool for visual refactoring of code behind which is what Visual Studio's UI Action view is.

Data Driving Tests

In Visual Studio 2012 data driving tests is still a manual experience that requires code level changes to your tests. For example, this is how a data-driven tests looks like in Visual Studio:

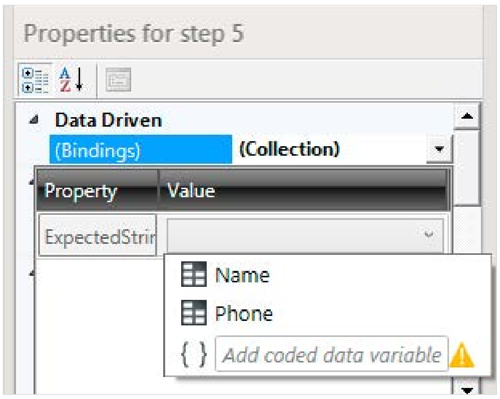
Ref: http://msdn.microsoft.com/en-us/library/ee624082.aspx

```
[DeploymentItem("DataDriven.csv"),
    DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",
        "|DataDirectory|\\DataDriven.csv", "DataDriven#csv",
        DataAccessMethod.Sequential),
    TestMethod]
public void CodedUITestMethod1()
{
    // To generate code for this test, select "Generate Code for
    // Coded UI Test" from the shortcut menu and select one of
    // the menu items.
    this.UIMap.AddTwoNumbersParams.TextInput1EditText =
        TestContext.DataRow["Input1"].ToString();
    this.UIMap.AddTwoNumbersParams.TextInput2EditText =
        TestContext.DataRow["Input2"].ToString();
    this.UIMap.AddTwoNumbers();

    this.UIMap.AssertforAddExpectedValues.TextAnswerEditText =
        TestContext.DataRow["ExpectedResult"].ToString();
    this.UIMap.AssertforAdd();
}
```

All these changes are needed, and needed to be done by hand to enable all actions to be data-driven. In Test Studio, we have a visual data-binding experience that allows you to bind, edit and remove bindings without touching any code.

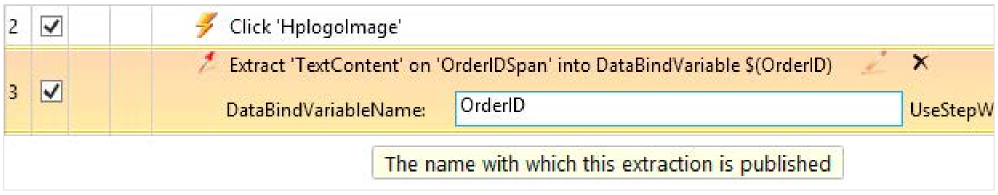
The binder auto-detects all the columns from the data source and you simply pick the column name from a list provided to you.



Data Value Extractions

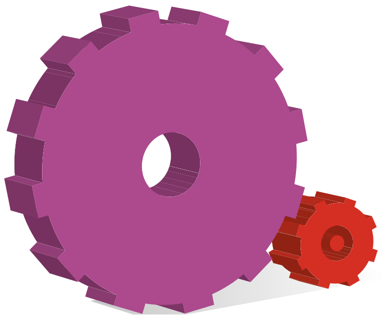
There are many scenarios in test automation when you are working with dynamically generated data that the application under test produces. The classic example of this is an OrderID that a system generates on initial execution of a test that you want the test to carry that value throughout the remainder of the test for validation or other data entry automation.

In Visual Stduio, you pretty much are on your own here and you simply need to hand-code all this in your test methods. In Test Studio, you can solve these automation scenarios by using the "ExactValue" feature the enables you to extract any type of value from any property on the page and bind that to any future step where needed.


2 Click 'HplogoImage'
3 Extract 'TextContent' on 'OrderIDSpan' into DataBindVariable $(OrderID)
DataBindVariableName: OrderID
UseStepW
The name with which this extraction is published

Customer Support & Release Agility

At Telerik we have world-class support that starts the day you download your trial. Each trial user gets the ability to not only use the community forums to ask questions but also gets to submit support tickets that our team of support engineers address promptly, with friendly service and knowledgeable responses.



Each license comes with 1 year of support and updates. Each ticket submitted is responded to within 24 hours. If you are still having problems, we schedule GoToMeetings to help guide you through you automation scenario.

With two major releases a year and weekly public internal build releases, your feature requests and bug fixes are accessible with a faster turnaround unmatched by any other product.

Read more about Test Studio's plugin for Visual Studio

Download 30-day trial with priority support

Telerik
TEST STUDIO