Exploring .Net "Orcas"

By Punit Ganshani

Punit Ganshani, employed by Cognizant Technology Solutions India Pvt. Ltd (NASDAQ: CTSH), a global IT services provider headquartered in Teaneck, N.J., is an author of a book titled 'Journey to C.' He has delivered various sessions on System Security, Ajax.Net, ASP.NET and Web application designing and has authored more than 18 white papers on various topics for national and international journals.

Language:	.Net 3.5
Platform:	Independent
Level:	Beginners/Intermediate

Contents

Released on 19th April 2007, this package ships with some great features that should be useful to programmers. With looks as compelling and as professional as Windows Vista, features of Visual Studio 2005, improved interfaces, LINQ, and a powerful intellisense, Microsoft® Visual Studio® 2008 (or Orcas) enables developers to rapidly create connected applications that deliver the highest quality, rich user experiences.

For a business analyst, capturing and analyzing information becomes easier and effective. For a developer, there are many such features that could change the way developers breathe and live. It provides a very good interface to communicate with Office 2007 applications, making applications more manageable, reliable and secure.

Let us have a brief look what Orcas has in store for people like us – the developers!

First the evolution!

There's not much to write on the evolution. I guess, the **Figure 1** would do the justice.

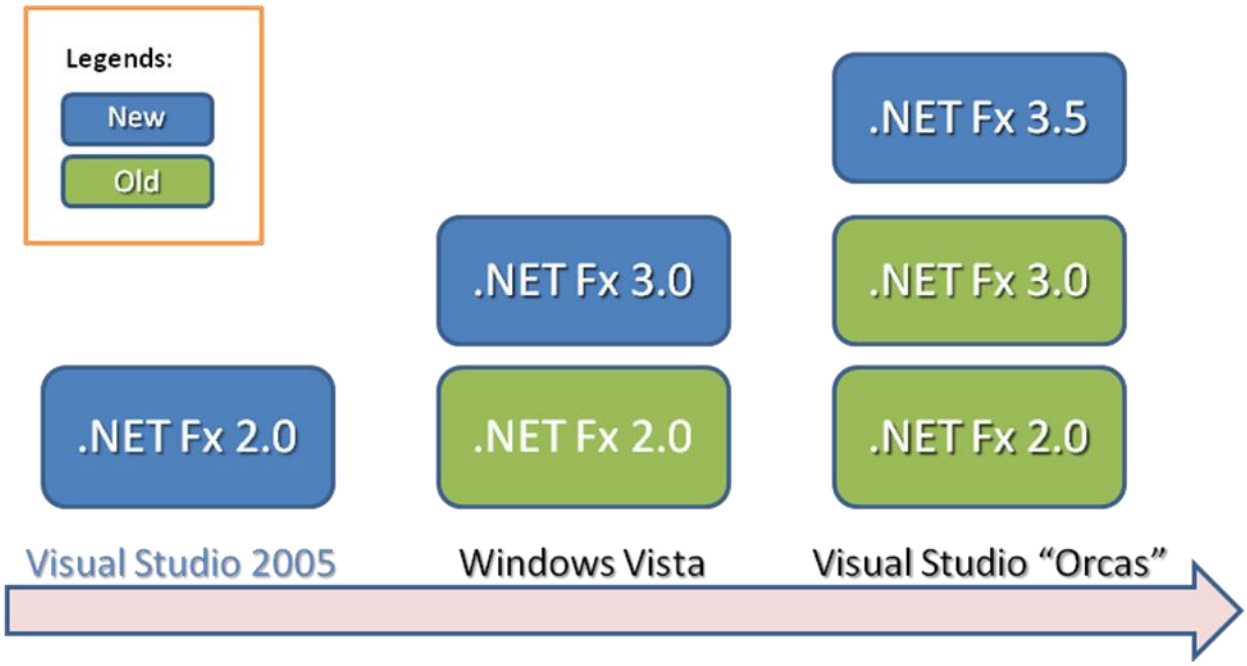

Figure 1: Evolution of Visual Studio

Backward Compatibility

Orcas (.Net 3.5 framework) is a combination of .Net 3.0 (introduced earlier this year) and improved versions of C#, VB.NET, ADO.NET, ASP.NET AJAX and CLR. Developers now have the freedom to choose their platform (.Net 2.0, 3.0-Vista or 3.5-Orcas) using the same VS 2008 IDE. Once a framework version has been selected, Visual Studio Orcas will enable the reference features that are appropriate for that version of the framework. (See **Figure 2**)

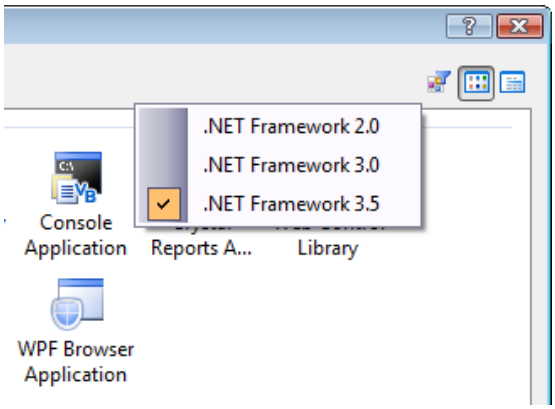

Figure 2: Multi-targeting support.

We can also achieve this by using Project Properties or Property Pages. (see **Figure 3**)

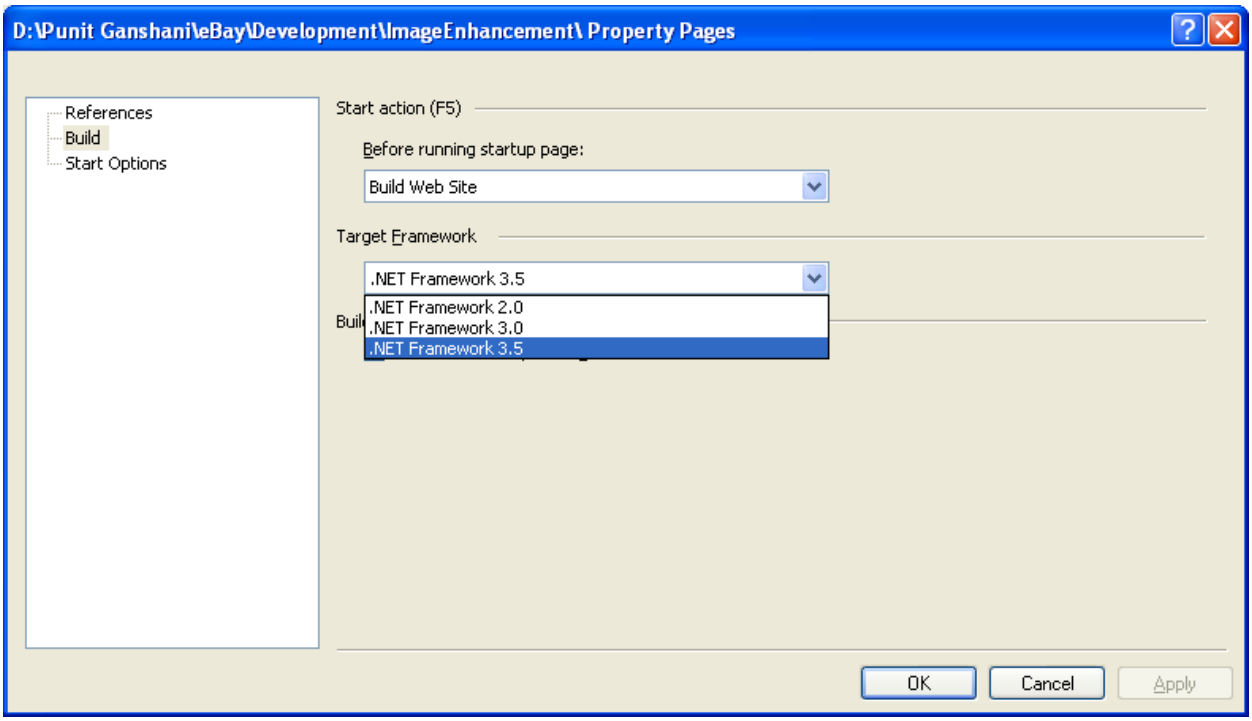

Figure 3: Multi-targeting through Property Pages

For example, features such as an already enabled Ajax website or LINQ (Language Integrated Query) will not be visible and functional on a .Net 2.0 Web application.

This is also called as Multi-targeting support.

Code and Reference Cleaning

Orcas helps us to remove all references that are not in use. By default, every ASP.NET page has some references added. After coding, we can right click the class name and click on 'Remove Unused Usings' as shown in **Figure 4.**

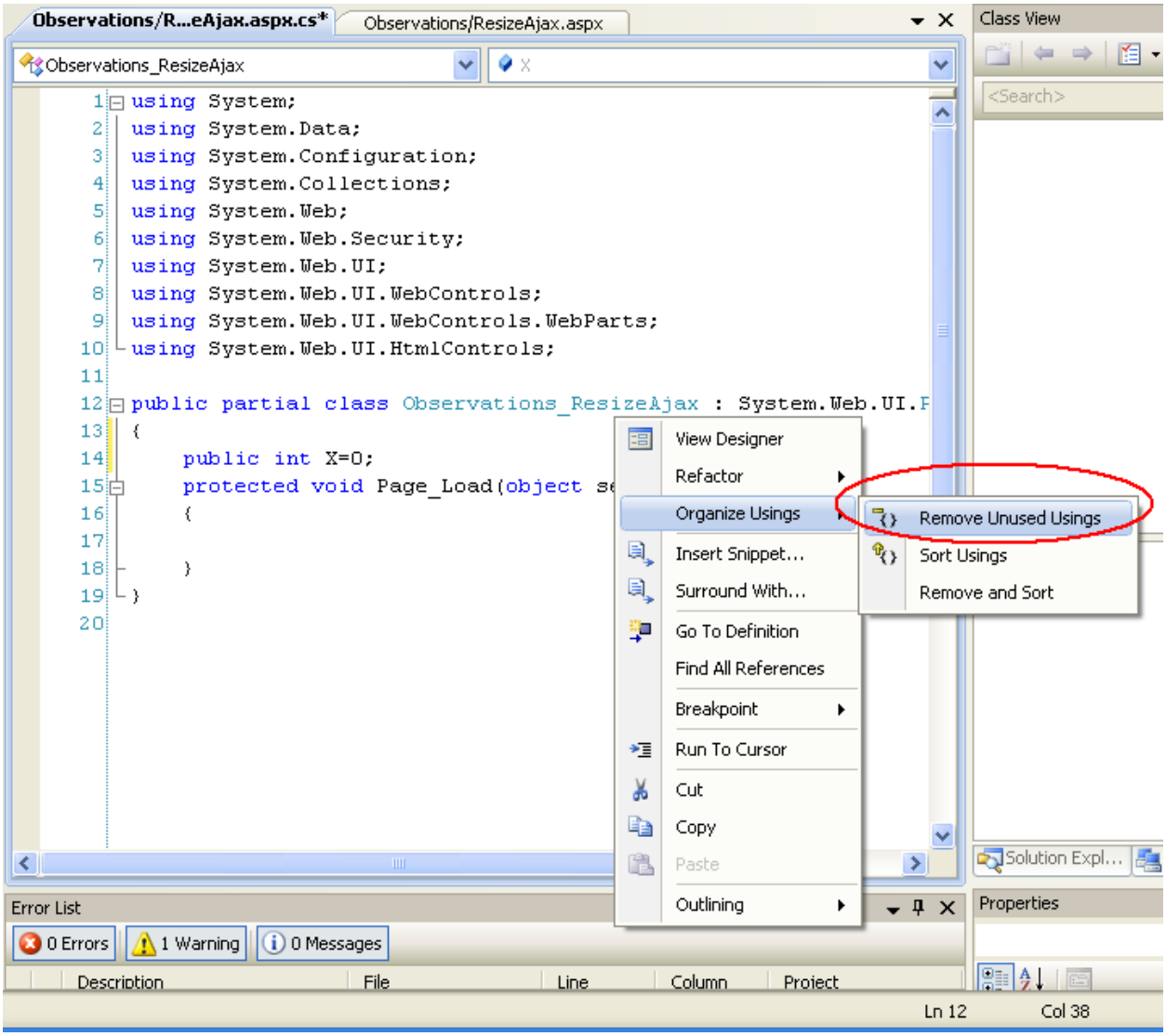

Figure 4: Code Cleaning

If we somehow, delete some useful references (say for example: System.Collections), we wish to use some classes of the missed out reference (say ArrayList), we can type-in ArrayList and right click and Click on 'Resolve' as shown in **Figure 5**

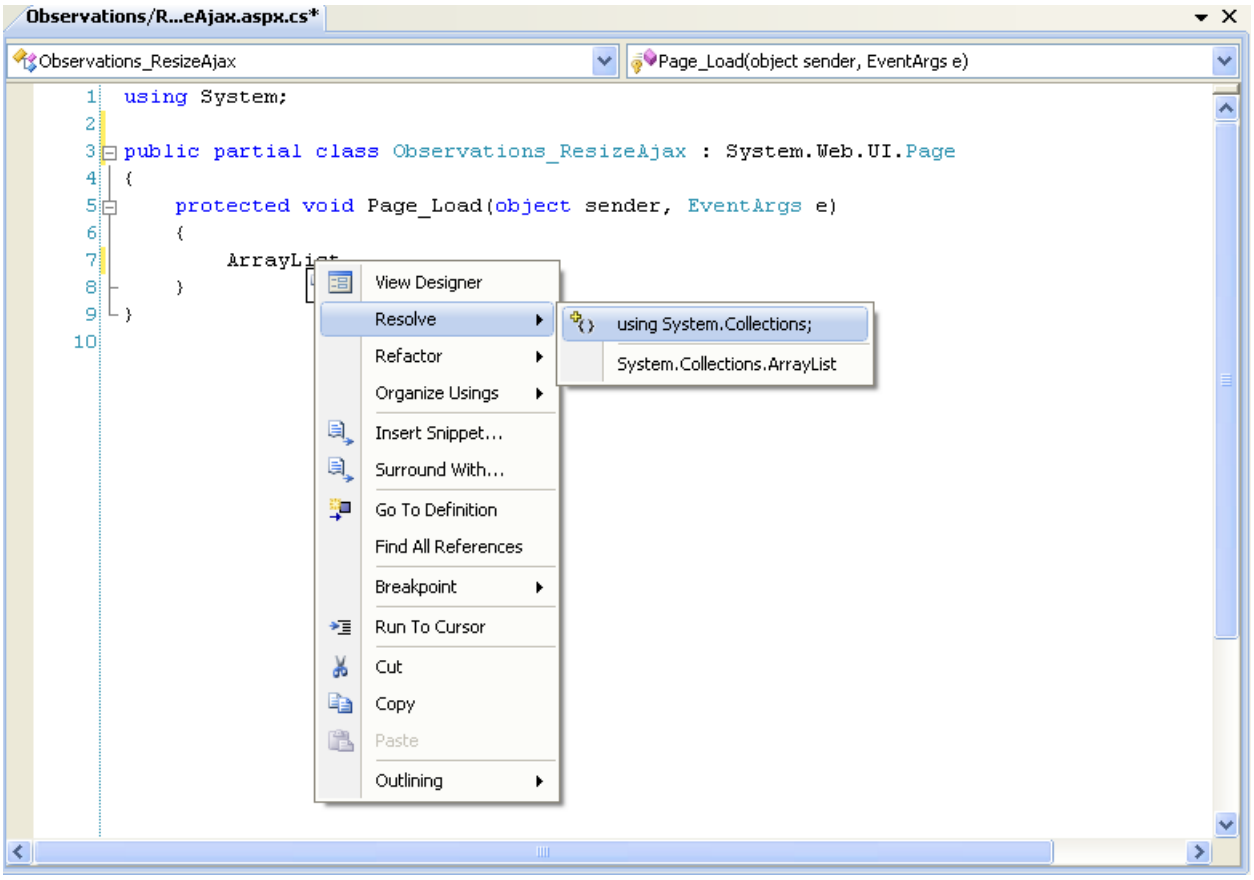

Figure 5: Adding references.

The New C# & VB.Net

Though the basics of C# remain the same, C# 2008 is shipped with great functionalities. In the days prior to the release of Orcas we had to define getters and setters methods inside the public properties. This did not involve any hard logic. All it entailed was hard-work! Orcas makes this task simpler by defining **Automatic Properties**. With Automatic properties, there is no need to define private properties. We leave this task for the compiler to do at compile-time. Let's look at Table 1 for an example:

Table 1: Automatic Properties

Before Orcas	With/In Orcas

`public class Person {`
 	
`private string _firstName;`
`public string FirstName {`
	`public class Person {`
	`public string FirstName`
	`{ get; set; }`
	`}`

```
            get {
                return _firstName;
            }
            set {
                _firstName = value;
            }
        }
}
```

When the C# compiler encounters an empty get/set property implementation, it automatically generates a private field within our class, and implements a public getter and setter property implementation.

Object initialization got a break from the old-fashioned C++ way of initializing objects. Objects can now be initialized in the same way as arrays. Let's understand the new way of achieving it in Table 2.

Table 2: Object Initializers

Before Orcas	With/In Orcas
`Person objEmployee = new Person();` `objEmployee.FirstName = "Punit";`	`Person objEmployee` `= new Person { FirstName="Punit"};`
All the public properties earlier required to be initialized using '.' Operator.	Use of '.' Operator to initialize the public properties is now no more required.

ASP.NET

ASP.NET IDE now has a **split view** with itself (see **Figure 6**). Developers and designers have the ability to view both the HTML markups as well as the source code at the same instance. The split window helps designers to visualize layers and graphics while coding.

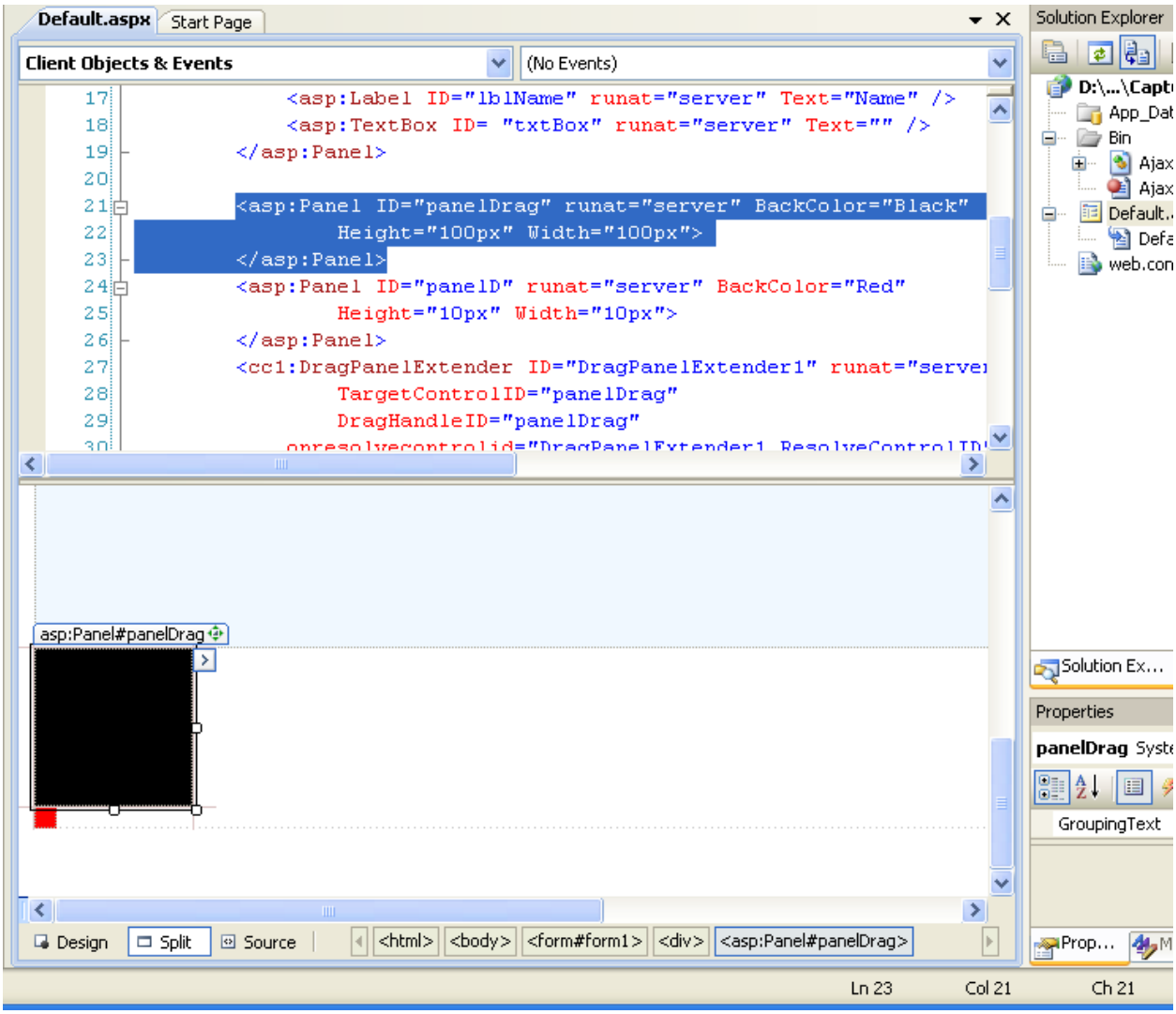

Figure 6: Split View in IDE

Visualizing cascaded **master pages** in VS 2005 was a tedious task. This isn't as tough now! Master pages now can have more than one column control place holder, which means managing different layouts is now feasible. A new master page '**AJAX Master Page**' is introduced in this package. This master page has two content place holders: one for heading and other for the content.

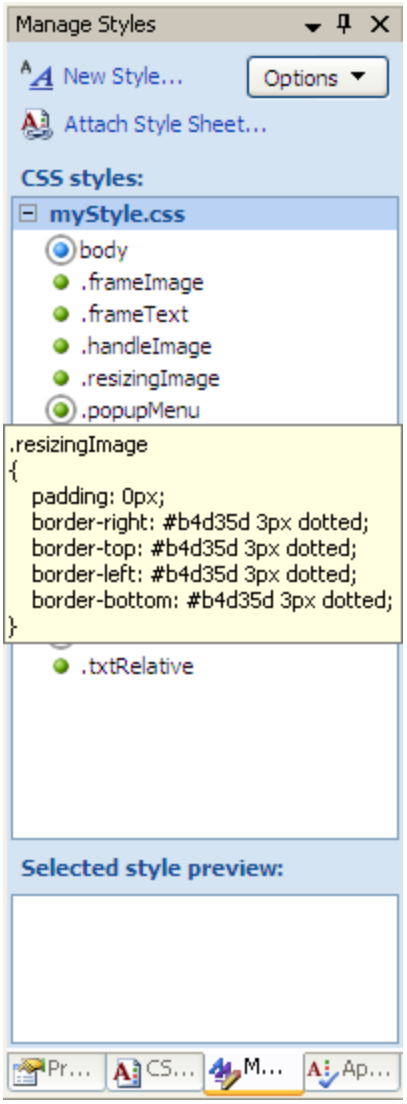

Figure 7: Managing CSS

Things became much nearer to Dreamweaver and Sharepoint Designer 2007. Just like Dreamweaver, Visual Studio Orcas has extremely rich CSS property Window. We can not just create CSS using the 'Build Layout', but can also preview it. 'Manage Styles' is the new window that allows us to easily create, manage, and refractor CSS rules within style-sheets. CSS became as easy as applying styles in Microsoft Word. When we select a HTML element, we can view the CSS properties and rules that apply to it in the 'CSS Properties' window. (see **Figure 7**)

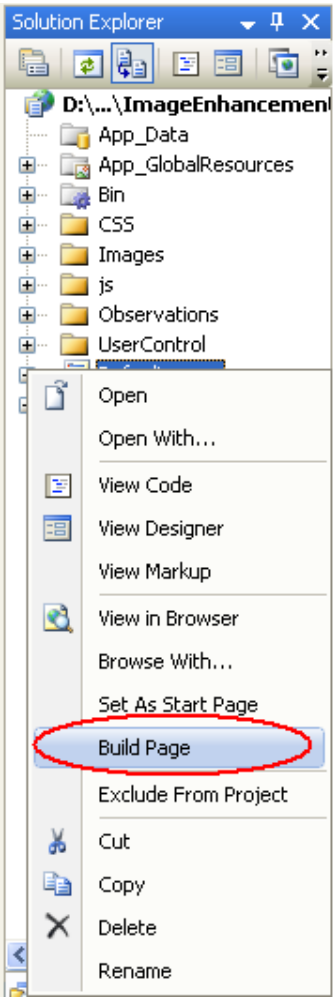

Figure 8: Building Single Page

Instead of building entire site, we can now build specific pages. This makes team development easier (see **Figure 8**)

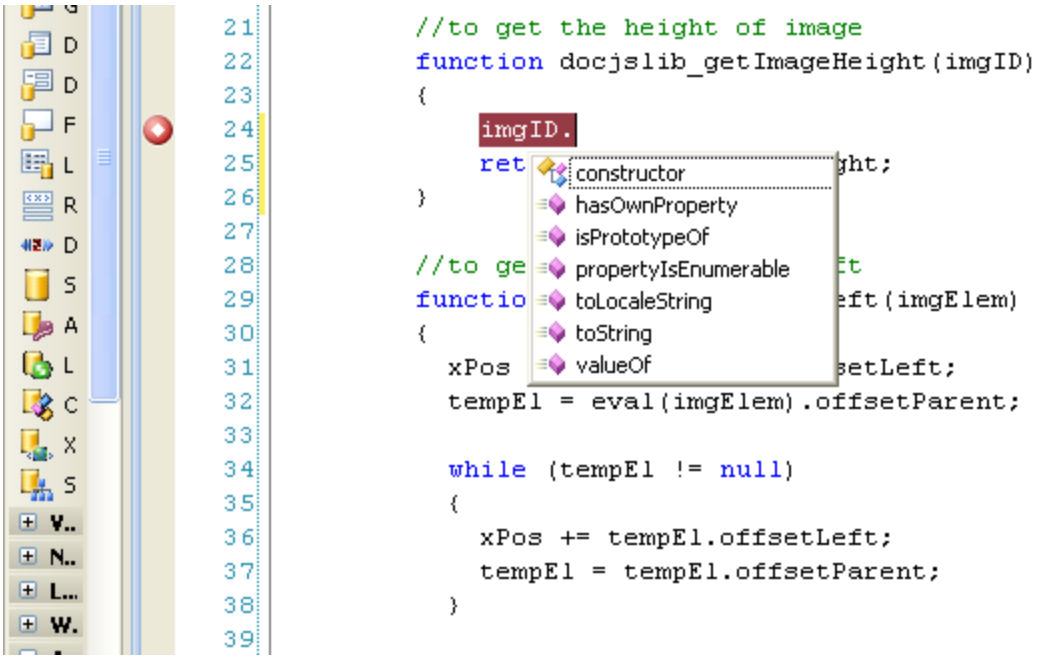

Figure 9: Intellisense in JavaScript

Javascript in Orcas has an inbuilt intellisense (see **Figure 9**), giving richer **debugging** support to the developers! The JavaScript intellisense supports rich type inferencing. In other words, it automatically detects the type of object and makes relevant suggestions rather than showing general suggestions (as was the case in VS 2005). XML **commenting** is also possible with Javascript functions. This increases the scope of documentation.

Client-side debugging helps make implementation of Ajax.Net easier.

Ajax.Net

Let's now have a look on Ajax – one of the technologies that fascinate me the most. Unlike VS2005, we need not explicitly install Ajax Extensions. Every site in Orcas is an Ajax Enabled website. Developers can take in advantage of the rich-UI, client side and server side integration and develop Web applications that run independent of modern browsers, even when integrated with Web Services or ASP.Net Services.

We find the same Script Manager, Script Manager Proxy, Update Panel, Update Progress and Timer in the toolbox. A wonderful Ajax Control Toolkit, compatible with Orcas, can be downloaded from CodePlex.

Support of common formats (such as XAML) gives designers direct control over layout and data binding with controls. Designers can now leverage a look and feel (L&F) similar to Office 2007 and Sharepoint. And when integrated with Ajax, it has a different appeal in itself!

Data Access through ADO.NET

In VS2005, dataset and the code that connects it are in the same file. Designing a multi-tier application using VS 2005 is a difficult task.

Visual Studio Orcas allows us to develop n-tier application with n-service layers, an access layer and a business logic layer. With such a model, it is easy to share validation logic between entities. Tables and the dataset now reside in different layers.

Orcas introduces LINQ (Language Integrated Query). Before LINQ was introduced, displaying data from a table involved two steps:

1. Executing a SQL Query or a Stored Procedure
2. Filtering a dataset using ADO.NET

Both activities are independent of each other as they involve two different applications and languages. This works absolutely well until the data is pooled from a database engine. However, when data is stored in XML files, HTML files or in emails, this method does not deliver results. Developers need knowledge of SQL and XPath. LINQ removes such dependencies. LINQ merges these worlds by using one language – the same language in which we code: C# or VB.NET. Hence, LINQ provides us an efficient way to handle such data processing in a language with which we are most familiar.

LINQ can be considered similar to a SQL query except for its syntax. A LINQ query starts with a 'from' clause (instead of select), followed by field name, then by keyword 'in' and Data Sources (listed in intellisense)

```
string[] names =
 {"Punit", "Pranjali", "Gautam", "Parasaran" };

IEnumerable<string> result =
    from name in names select name;
```

The main difference here is that C# always puts the "select" part as the last part of the command, however VB.NET puts it first.

```
Dim names As String() =
  {"Punit", "Pranjali", "Gautam", "Parasaran" }

Dim result As IEnumerable(Of String) = _
 Select name From name in names
```

Other features & advantages

Some of Orcas' other new features are:

1. The **Microsoft Synchronization Services** for ADO.NET provide an application programming interface (API) to synchronize data between data services and a local store.
2. Microsoft **SQL Server 2005 Compact Edition**: Offers low maintenance, compact, embedded database for single-user client applications for Tablet PCs, Pocket PCs, smart phones and desktops.
3. Windows **Workflow** Foundation: Design workflows as easily as designing flow charts.
4. Windows **Presentation** Framework: Rich UI interfacing now becomes easier with XOML and XAML.
5. Data Access does not require any knowledge of specialized languages like SQL and XPath. LINQ can instead replace them.
6. Building applications on different platforms becomes easier.
7. Windows Forms continue to provide compelling features to develop large applications.
8. **Look and Feel** of the IDE sees a remarkable difference.
9. Team suite continues to provide a testing and documentation environment.

There's much more to be explored and to be written on how Microsoft Visual Studio 2008 makes life easier for developers, architects and managers. I hope this white paper has given developers useful insights into what Visual Studio Orcas offers them.