

PDC Round-Up

PDC 2008 was a first – there were *four* keynotes. This is because Microsoft made numerous announcements compared with usual PDCs. Some of the big announcements hardly made it into a keynote (Oslo for example). So it's worth taking some time to reflect on the event and look at what was announced and how it may impact our lives as developers.

Windows 7

Windows 7 was the announcement that got the media buzzing, probably because this was the most familiar product to that audience. There are certainly some interesting changes to the UI:

- The new unification of the task bar so they have combined the functionality of the normal stacked task bar with the quick start toolbar
- Configurable system tray
- Jump lists that combine an MRU list with application defined actions
- Ribbons everywhere – the standard applications have had an Office 2007 makeover
- Libraries that are configurable views on to the file system, potentially across machines, targeted at specific kinds of content
- Multitouch is now supported by the OS bringing the Surface Computing UI paradigm to the client OS

These are obviously features that UI developers need to think about when building apps targeted for Windows 7. However, there are OS features that will affect the developer's world as a user of the OS:

- Multi-monitor support is now very simple to configure
- Virtual hard disks (VHDs) are now mountable in the file system and also bootable
- The magnifier is more powerful and accessible (think of [ZoomIt](#) on steroids)
- Remote desktop can now display multi monitor

The core of Windows 7, however, is still Vista which means good things like UAC are still there but the great thing this time is they haven't rewritten the driver model which means any hardware with a Vista driver will continue to work (which was the main pain people went through with Vista when it was first released).

Windows Azure

If Windows 7 got most of the press at PDC, Azure was the announcement that has possibly the most far-reaching consequences. The problem space is one of web scale. When you launch an Internet facing product or site, what infrastructure do you launch on? If you look at the speed that MySpace, Facebook, Twitter and others gained users, the ability to bring sufficient hardware resources to bear as growth accelerates is a huge challenge. In addition, you then need to manage those hardware resources in data centres' which, again, is a non-trivial task. Windows Azure is a platform that runs on Microsoft owned data centres' that you lease resources from – this is the Software **and Services**

vision that Microsoft have been talking about for some time – in that they not only sell software but also provide services to their customers.

The idea is that you write your applications to target Azure and then Microsoft map the resources you requested to your application as they host it. You don't have to worry about hardware failures, connectivity, OS patching, even application rolling upgrades - Microsoft take care of that side of things. But what does it mean to "target Azure"? Azure applications are normal web based applications with a couple of big differences:

- Components run under a role. A web role is one that is Internet accessible, a worker role is a component that is only accessible inside azure.
- You need to supply two configuration files: one defining the application and one specifying requested resources (how many instances of each component are required to run).

You may have noticed an issue though; if you don't "own" a machine but only resources where on earth do you save your data? This is provided via the Azure Storage Service. Azure Storage provides storage for three types of data: blobs, structured tabular data and queues. Blobs are for unstructured data. Queues are designed to be for communication between components in your application (web to worker role for example). Tables are for structured data, although we have to be careful here – we're not talking about relational data or even data with a defined schema, simply items of data with associated properties where each row, in theory, could have completely different properties (although this would be hardly practical).

So far this describes the base infrastructure of Azure. Layered on top of Azure are higher order services: .NET Services, SQL Services, Live Services, SharePoint Services and Dynamics Services. There was little or no detail about SharePoint Services and Dynamics Services so I'll give a bit more detail on the other three.

.Net Services

The .NET Services layer has gone by a couple of names over the last few years: first as BizTalk Services as an incubation project, then as Zurich while being productionised. Now named .NET Services it provides three blocks of functionality

- Access Control: they provide a Security Token Service to allow you to control authentication potentially using federation and authorization using claims
- Service Bus: they provide a service bus "in the cloud" where services and consumers can rendezvous at an Internet accessible location even though both parties are behind firewalls. This enables a number of scenarios such as: bridging on premises systems to web based ones and publish / subscribe – where the sender is disconnected from the final recipients of messages
- Workflow: they provide a constrained set of workflow activities that will allow you to orchestrate services and perform content based routing

SQL Services

Formally known as SQL Server Data Services (SSDS) this service sits on top of Azure providing relational storage. It would, however, be a mistake to think of this as having access to full blown SQL Server as, although the underlying infrastructure is on SQL Server, only a subset of functionality is

exposed. Currently there is just a relational store but the further functionality is on its way in terms of reporting, analytics and synchronization.

Live Services

Live Services is the infrastructure upon which Live Mesh is based. It allows you to manage devices, contacts and data with a uniform API. It allows the storage and synchronization of data and applications across devices with a base set of services called the Live Operating Environment. There are implementations of LOE for devices and also in the cloud allowing applications to use a standard API whether they are running on the locally or in the cloud.

The first player in this space was Amazon with the Amazon Web Service platform. With Azure, Microsoft has now entered the playing field with a very rich set of services that go beyond those which Amazon is currently providing. For many companies Azure will allow a low risk and low overhead entry into providing web scale services to their customers and so the possibilities for the platform are potentially big.

Oslo

Oslo is Microsoft's new modelling platform that aims to allow developers to create applications declaratively, using models that are then consumed by model aware runtimes that produce the intended functionality. It takes an approach that was used successfully by both BizTalk and Windows Workflow and generalizes it for any application platform willing to participate. Oslo consists of three pieces:

The Repository

This is a relational database that holds the models in the form of data and it is this data that is processed by runtimes to generate the application. The repository has structure to deal with access control and versioning

Quadrant

When people think of modelling them generally think in terms of graphical tools creating diagrams that show their intention using a notation that is tied to the domain they are modelling – class diagrams in OO design for example. Quadrant is a new graphical tool that provides the ability to work with any kind of data giving a flexible and highly customizable viewing capability. But more than that, it is extensible with new diagram types to allow the building and rendering of visual Domain Specific Languages (DSLs) that allow applications and services to be built using a very natural and accessible “visual syntax”.

M

Although people generally think of graphical tools when they think of modelling, developers tend to spend most of their time writing text based code. M is a new text based language for creating models including the ability to design text based DSLs. This gives developers both a familiar development experience and the power of DSLs to dramatically reduce the amount of code they have to write to build their functionality

WF 4.0

When it was introduced in .NET 3.0, Windows Workflow introduced a powerful new way to write applications that provided an infrastructure for long running processing and the increased transparency on what that processing was doing. However, there were issues with WF 3.0 that were impossible to resolve with the way it was written – particularly in terms of the serialization speed and footprint during persistence. As a result WF 4.0 is a ground up rewrite of workflow that keeps the same concepts of the processing model but changes the API significantly. The big changes include:

- Performance: they claim execution speed increases of one to two orders of magnitude
- Persistence: the way data flows around the workflow has been completely remodelled so they only need to persist the state of the currently executing activities and their ancestors in the activity tree
- Asynchrony: writing asynchronous activities has been hugely simplified. In addition a lightweight a sync model has been introduced to allow the activity to perform things like a sync IO without danger of being unloaded
- XAML: XAML is now the default workflow authoring model allowing a very flexible deployment model out of the box
- Flow control: flowchart based workflow is now available
- Composability: all of the different execution models – sequential, state machine, flowchart and rules based – are all composable within each other
- Workflow Services: the workflow service implementation has now been made fully declarative and also supports data based correlation as well as context channel based correlation
- Designer re-hosting: although possible in 3.0 and 3.5 this was a huge undertaking. The re-hosting model is now hugely simplified needing just a small number of lines of code to wire the WF designer into custom applications

The 3.5 runtime and designer will still be shipped in Visual Studio 10 and there is interop functionality to allow 3.5 based activities to be hosted in 4.0 workflows

Dublin

One of the most frustrating things about workflow has been the necessity to write the hosting code for the workflow – which is a non-trivial piece of code to get right and the lack of any out-of-the-box management functionality. Dublin is an extension to the Windows Server Application Server Role designed for hosting workflow and workflow based services. Deploying a workflow into Dublin is as simple as dropping a workflow XAML file on to a Dublin “virtual directory”. The management tools integrate directly into the IIS manager MMC and provide visibility of running workflows and their tracking data. Deploying into Dublin is also supported from Quadrant.

And More ...

PDC also saw the announcements of: new features of both C# 4.0 and VB 10; more concurrency support in the libraries and Visual Studio; WCF 4.0; WPF 4.0; Silverlight 3.0; the Surface computing SDK and much more.

This PDC saw the most announcements of new technology that I can remember at one of these conferences. These new waves of technology for me, make it an exciting time to be a developer (although one that is going to involve ramping up on many new APIs and platforms).