Create great mobile apps with Azure and Visual Studio

Matteo Pagani
Windows AppConsult Engineer @ Microsoft
mail: matteo.pagani@microsoft.com
twitter: @qmatteoq
Announcements from Connect()

New for Connect(); 2016

Visual Studio 2017 RC
Fully-featured IDE, productivity for any apps

Visual Studio for Mac Preview
A mobile-first, cloud-first IDE. Made for the Mac

Team Foundation Server 2017
The collaboration platform for every developer

Visual Studio Mobile Center Preview
Mission control for mobile apps
Announcements from Connect()

- Microsoft joins the Linux foundation as platinum member
- Google joins the .NET Foundation for open source projects
- Samsung announced that Tizen now supports also .NET Core and C# development
- SQL Server for Linux Preview is now available
How Microsoft can help

Create mobile experiences...
New applications or new end points to existing applications for Android, iOS, and Windows

...connected to the business...
Enterprise-ready platform that integrates modern mobile experiences with existing business assets

...with mobile-era agility
Achieve speed with faster development cycles and continuous measure and learn

Xamarin
Visual Studio

Microsoft Azure

Visual Studio Team Services
HockeyApp
Notification Hubs
Mobile Push Is Everywhere

Breaking news
News/Media

SMS replacement, Deals, Back-office
Banking/Insurance

Prescriptions, Appointments, LOB (maintenance)
Healthcare

Reservation changes, Deals, Back-office
Travel/Hospitality/Airlines

Orders, Product UX, Back-office
Discrete manufacturing/Auto

Offers, Orders, Back-office
Retail
Different types of Notifications

- Windows WNS
  - Toast
  - Tile
  - Badge
  - Raw
- Windows Phone (MPNS)
  - Toast
  - Tile
  - Raw
- Apple
- Google
- Amazon
- Baidu
Push Notification Lifecycle

**Registration at app launch**
- Client app contacts Platform Notification Service, to retrieve current channel (for example: ChannelURIs, device tokens, registrationIds)
- App updates handle in back-end

**Sending Notification**
- App back-end send notification to PNS
- PNS pushes the notification to the app on the device

**Maintenance**
- Delete expired handles when PNS rejects them
Challenges of Push Notifications

Platform dependency

• Different communication protocols to PNS
• Different presentation formats and capabilities

Routing

• Provide a way to send a message to a device/channel
• Target user or group of interest
• Back-end has to maintain an identification repository

Scale

• Need to store current handles for each device
• Broadcast to millions of devices with low latency
Payload Snippets

Windows / Windows Phone
<toast><visual>
  <binding template="ToastText01">
    <text id="1">Hello from a .NET App!</text>
  </binding>
</visual></toast>

iOS
{"aps":{"alert":"Hello from .NET!"}}

Android
{"data":{"message":"Hello from .NET!"}}
From Any Back-End to All Mobile Platforms
# Using Notification Hubs

## One-time set up
- Create a Notification Hub

## Register
- Client app retrieves its current handle (PNS)
- Client app registers on Notification Hub (current handle)

## Send Notification
- App back-end sends a message to Notification Hub
- Notification Hub pushes it to PNS’
Advantages of Using Notification Hubs

- **X-plat: from any back-end to any mobile platform**
- **No need to store device information in the app back-end**
- **Routing and interest groups**
- **Personalization and localization**
- **Broadcast at scale, multicast, unicast**
- **Telemetry**
Sending native notifications

- The mobile app subscribes to the Notification Hub just by sending the channel URL
- The backend sends the notification by describing the full payload

The server

```csharp
private async void OnRegisterForNotificationsClicked(object sender, RoutedEventArgs e)
{
    PushNotificationChannel channel = await PushNotificationChannelManager.CreatePushNotificationChannelForApplicationAsync();
    if (channel != null)
    {
        NotificationHub hub = new NotificationHub("uwpsample", ConnectionString);
        await hub.RegisterNativeAsync(channel.Uri);
    }
}

private async void OnSendNotificationsClicked(object sender, RoutedEventArgs e)
{
    string xml = @"<toast>
        <visual>
            <binding template=""ToastGeneric""> await
            <text>Hello insiders!</text>
            <text>This is a notification from Notification Hub</text>
        </binding>
    </visual>
</toast>";
    NotificationHubClient client = NotificationHubClient.CreateClientFromConnectionString(ConnectionString, "uwpsample");
    await client.SendWindowsNativeNotificationAsync(xml);
}
```
Sending template notifications

- The mobile apps subscribes to the Notification Hub and, other than sending the URL, it specifies the notification payload with some parameters.
- The backend sends the notification specifying just the parameters.

**The server**

```csharp
private async void OnRegisterForNotificationsClicked(object sender, RoutedEventArgs e)
{
    PushNotificationChannel channel = await PushNotificationChannelManager.CreatePushNotificationChannelForApplicationAsync();
    if (channel != null)
    {
        NotificationHub hub = new NotificationHub("uwpsample");
        var xml = "<toast><visual><binding template="ToastGeneric"><text>$(title)</text><text>$(message)</text></binding></visual></toast>";
        await hub.RegisterTemplateAsync(channel.Uri, xml, "SampleTemplate");
    }
}

private async void OnSendNotificationsClicked(object sender, RoutedEventArgs e)
{
    NotificationHubClient client = NotificationHubClient.CreateClientFromConnectionString(ConnectionString, "uwpsample");
    Dictionary<string, string> properties = new Dictionary<string, string>
    {
        {"title", "Template notification"},
        {"message", "This is the message"}
    };
    await client.SendTemplateNotificationAsync(properties);
}
```
Demo
Notification Hub
Azure App Service
One integrated offering

- Web Apps
- Mobile Apps
- Logic Apps
- API Apps
### What Do Azure Apps Provide?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for .NET, Node.js, PHP, Java and Python platforms</td>
<td>Supports a wide range of programming languages and frameworks.</td>
</tr>
<tr>
<td>Built-in AutoScale and Load Balancing</td>
<td>Enables automatic scaling and load balancing for applications.</td>
</tr>
<tr>
<td>High availability with Auto-Patching features</td>
<td>Ensures high availability with auto-patching features for applications.</td>
</tr>
<tr>
<td>Continuous Deployment with Git, TFS, GitHub</td>
<td>Supports continuous deployment with popular version control systems.</td>
</tr>
<tr>
<td>SQL Databases, MySQL, DocumentDB, Search, MongoDB</td>
<td>Provides scalable database options for various applications.</td>
</tr>
<tr>
<td>WordPress, Umbraco, Joomla, Drupal</td>
<td>Supports popular content management systems for web development.</td>
</tr>
</tbody>
</table>
Supported Publishing Methods

- FTP://
- Visual Studio Online
- WEBDEPLOY
- git
- DROPBOX
- More...
What is Continuous Deployment?

- Variance between tested code and live production is minimal
- When code is checked in, it is automatically deployed
- Promotes
  - Business agility
  - Technical agility
  - Operational agility
Compute Scaling

Scale Up
- also known as Vertical Scaling
  - Increase resources capacity within existing node

Scale Out
- also known as Horizontal Scaling
  - Increase resources capacity by adding nodes
Web App Scaling by Instance / CPU (Basic, Standard or Premium Plan)

<table>
<thead>
<tr>
<th>Scale</th>
<th>FIXITPLAN-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoscale</td>
<td>On</td>
</tr>
<tr>
<td>Instances</td>
<td>1</td>
</tr>
</tbody>
</table>

**Instances**

- **Scale by**: schedule and performance rules
- **Description**: Create your own set of rules. Create a schedule that adjusts your instance counts based on time and performance metrics.
  - Default, scale 1 - 2
  - CPU Percentage > 80 (increase count by 1)
- **Settings**
  - CPU Percentage < 60 (decrease count by 1)
  - Add Rule
  - Add Profile
Web App Auto-Scaling – Adding Metrics

New metrics also allow scale up and scale down rules
API apps
REST APIs Matter

REST

- Stands for ‘Representational State Transfer’
- Architectural style
- Rely on HTTP
- Almost every device connected already use HTTP
REST API in Terms of Language

Nouns
- URIs – things

Verbs
- Actions to take on nouns
  - GET, PUT, DELETE, POST, PATCH, HEAD, OPTIONS
  - Idempotent vs. Non-Idempotent

Adjectives
- Mime types – further describe nouns
  - Content-type: application/vnd+company.category+json
JSON Collection

https://api.contoso.com/v1.0/addresses
{
  "value": [
    {"street":"1st Avenue","city":"Seattle"},
    {"street":"124th Ave NE","city":"Redmond"},
    {"street":"35th Ave SE","city":"Cleveland"},
    {"street":"148th Ave NE","city":"Denver"},
  ],
  "@nextLink":"<opaque-url>"
}
Why Azure API Apps?

Benefits of App Services
- Automatic operating system patching
- Enterprise grade security
- High availability
- Support for many platforms and languages
- Auto scaling and load balancing
- WebJobs for background processing
- Easy deployment, including continuous delivery
- Access on-premises data

Additional Benefits
- Bring your API as-is
- Simple access control
- Connectivity to SaaS platforms
- Swagger metadata
- Logic App integration
- Visual Studio tooling and support
- Public and private marketplaces
- Automatic dependency deployment
- Automatic updates
Swagger

Goal
- Define a standard, language-agnostic interface to REST APIs
- Discover and understand the service’s capabilities

Does not required
- Access to source code
- Documentation
- Network traffic inspection

Benefits
- Human readable
- Machine readable
Swagger Hello World

```json
{
    "swagger": "2.0",
    "info": {
        "title": "MyClient",
        "version": "1.0.0"
    },
    "host": "swaggersample.azurewebsites.net",
    "paths": {
        "/api/HelloWorld": {
            "get": {
                "operationId": "GetGreeting",
                "produces": [
                    "application/json"
                ],
                "responses": {
                    "200": {
                        "description": "GETs a greeting.",
                        "schema": {
                            "type": "string"
                        }
                    }
                }
            }
        }
    }
}
```
Swagger UI

Definition

• Collection of:
  • HTML
  • JavaScript
  • and CSS assets

Goal

• Dynamically generate beautiful documentation and sandbox
• From a Swagger-compliant API

Note

• Part of the Swagger project
Swagger UI (continued)

Swagger Petstore
This is a sample server Petstore server. You can find out more about Swagger at [http://swagger.io](http://swagger.io) or on IRC. freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

Find out more about Swagger
[http://swagger.io](http://swagger.io)
[Contact the Developer](#)

Apache 2.0

<table>
<thead>
<tr>
<th>pet : Everything about your Pets</th>
<th>store : Access to Petstore orders</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>POST /user</code> Create user</td>
<td><code>Show/Hide</code> List Operations Expand Operations</td>
</tr>
<tr>
<td><code>POST /user/createdWithArray</code> Creates list of users with given input array</td>
<td></td>
</tr>
<tr>
<td><code>POST /user/createdWithList</code> Creates list of users with given input array</td>
<td></td>
</tr>
<tr>
<td><code>GET /user/login</code> Logs user into the system</td>
<td></td>
</tr>
<tr>
<td><code>GET /user/logout</code> Logs out current logged in user session</td>
<td></td>
</tr>
<tr>
<td><code>DELETE /user/{username}</code> Delete user</td>
<td></td>
</tr>
<tr>
<td><code>GET /user/{username}</code> Get user by user name</td>
<td></td>
</tr>
<tr>
<td><code>PUT /user/{username}</code> Updated user</td>
<td></td>
</tr>
</tbody>
</table>

[ API URL: Av2, API VERSION: 1.0.0 ]
Swashbuckle

Goal
• Documenting ASP.NET Web API projects with Swagger

Combine
• ApiExplorer
• Swagger/Swagger-UI

Install
• Available as NuGet Package
AutoRest

Definition

• .NET based code generation tool
• Turns language agnostic REST API specifications into client libraries in multiple languages.
• Transforms swagger spec into implementation

Supported language

• .NET
• Java
• JavaScript
• Ruby
• Python

Benefits

• Expand reach of API
• Reduce time spent to authoring client libraries
Azure App Service Gateway

• Built-in authentication services that implement OAuth 2.0 and OpenID Connect and work with multiple identity providers.

Supported identity providers

• Azure Active Directory
• Microsoft Account
• Google
• Twitter
• Facebook
Azure Mobile Apps
Engage employees, partners and customers on any device at any time

Enterprise Grade Apps
Fully Managed Platform
High Productivity Development
Mobile SDKs
Gateway to backend capabilities
Data / Notifications / Auth

Native SDKs

Cross-Platform SDKs

Native
Android
iOS
Windows

Cross-plat
Cordova/PhoneGap
Xamarin
Sencha
Titanium
HTML 5/JS

.NET and node back-ends available
Why offline sync?

- Make apps **resilient** against intermittent network connectivity
- Allow end-users to create and modify data even when there is **no network access**
- Sync data **across multiple devices**
- Improve app responsiveness by **caching** server data locally on the device
- Detect and **handle conflicts** when the same record is modified by more than one client

Why offline sync?
Authentication

Same as API and Web Apps

Access levels
- Anonymous (or roll your own)
- Authenticated

Authenticated
- Azure AD (User account or service principal)
- Microsoft account
- Social: Facebook, Google, Twitter

CORS
Demo

Xamarin and App Services
How Microsoft can help

Create mobile experiences...
New applications or new end points to existing applications for Android, iOS, and Windows

...connected to the business...
Enterprise-ready platform that integrates modern mobile experiences with existing business assets

...with mobile-era agility
Achieve speed with faster development cycles and continuous measure and learn

Xamarin
Visual Studio

Microsoft Azure

Visual Studio Team Services
HockeyApp
Continuous Integration & Delivery (CI, CD)

- Automate and orchestrate your build, test, and release processes
- Smooth deployment and beta testing of mobile apps
- Close the loop between development and operations

Visual Studio Team Services and TFS
HockeyApp
Xamarin Test Cloud
Integration and delivery
Agile planning

Tools for the entire team

Monitor all your team’s work—including operational issues—using built-in Kanban boards.
Track work in progress to ensure you’ve got a streamlined channel from idea to delivery.
Dashboards and charts

Build dashboards that provide visibility to your team and stakeholders. Stop worrying where to find things and instead use Visual Studio Team Services or Team Foundation Server to get everyone on the same page.
Mobile tasks

Build and deployment

Simplify the creation of powerful build deployment pipelines with the deployment tasks

Dozens of tasks exist for build and deployment options across a wide variety of Microsoft, non-Microsoft, and OSS technologies

Build steps are easily extensible and an ecosystem of third-party steps are available from the Visual Studio Team Services store
Quality is hard
Android diversification
iOS diversification

Form factors

- iPhones
- iPod touch
- iPads

iOS 9 features

- 3D Touch
  - iPhone 6s series only
- Apple pencil
  - iPad Pro only
- Multi Tasking
  - Only select iPads
Application complexity

- Thousands of iOS and Android APIs
- Public web APIs
- Backend integrations
- Third-party libraries
- Memory, CPU, and network constraints
Real devices

2,000+ devices
Automate your app testing and run them on over 2,000 different real devices. Test everything users do, as well as any performance problems with step-by-step memory and performance tracking.

Script C# or Ruby

Xamarin Test Recorder Preview
Detailed reports

Track results
The rich, detailed reports available through Xamarin Test Cloud gives users the ability to analyze reports that include results, screenshots, and performance metrics.

Trend analysis
Compare results to previous runs to find regression bugs or performance bottlenecks. Have insight into the quality of your application.
Demo
Xamarin Test Cloud
Continuous monitoring & learning

- Get insights about availability and crashes of apps and services
- Analyze results in beta testing and platforms/devices coverage
- Learn from production usage and connect to direct marketing actions

HockeyApp
Azure Mobile Engagement
Visual Studio Team Services and HockeyApp
Distribution

In-house App Store for testers
Upload beta versions of your application to the HockeyApp store to allow testers to install and test these versions on actual devices

Deployment made easy
HockeyApp's desktop app automatically tracks all necessary information about your latest build in order to make uploading beta versions very easy
Crash reports

No additional code
By integrating HockeyApp’s open source SDK for Android, iOS, Mac, and Windows your apps can send crash reports directly to HockeyApp, no additional code needed

Powerful crash analysis
HockeyApp processes and symbolicates all crash reports. This gives meaningful stack traces with friendly class names, methods, and accurate line numbers
Feedback

A conversation with your users
All feedback is handled as discussions. You can manage these discussions in the web interface or through email. Search discussions, mark them as completed or create work items based on them. Give your users a voice outside of the app store

All development phases
You can use the feedback feature for all builds of your app or just beta versions. Either way, HockeyApp makes it easy for users to tell you what's what. Let them post ideas right from within your app
Beta test coverage

Real usage matters
Advanced metrics to see which devices were used, how long the app was used for, and which language was tested. No additional setup

Detailed charts
HockeyApp features both raw data from analytics as well as live, interactive charts for the most important metrics
Demo

HockeyApp
How Microsoft can help

Create mobile experiences...
New applications or new end points to existing applications for Android, iOS, and Windows

...connected to the business...
Enterprise-ready platform that integrates modern mobile experiences with existing business assets

...with mobile-era agility
Achieve speed with faster development cycles and continuous measure and learn

Xamarin
Visual Studio

Microsoft Azure

Visual Studio Team Services
HockeyApp
Questions?

Samples: https://github.com/qmatteoq/AppServices
Slides: https://doc.co/ZpQrYs/nqqAqB

Matteo Pagani
Windows AppConsult Engineer @ Microsoft
mail: matteo.pagani@microsoft.com
twitter: @qmatteoq