Foundation Cloud Client

3/24/2015 | Nicholas Ventimiglia | http://unity3dfoundation.com

A Cloud host-able Asp.Net MVC solution specialized for Unity3d

- Asp.Net MVC is a world class C# 4.0 open source web server
- Graphical (Html) Developer and Administration portal
- Multi-tenancy ready, host multiple games off a single install
- All server Api's have a Unity3d service client ready to go !
- Account Api with sign in, sign up, account recovery, and guest mode
- Support for branded emails for Account Recover and New Users (Welcome)
- Storage Api with support for dynamic (untyped) objects
- Realtime API for integration in the Realtime cloud messaging service
- Easily extend the solution with custom ApiControllers
- Uses easy to host SQL database using Entity Framework
- Data tables are flat and normalized making a conversion to Redis, MongoDB or RavenDB fast !

Platforms

Desktop, Webplayer, Android, iOS, Windows Store

Setup and Configuration

Once the package is unpacked, you will need to do some light configuration

- On the main menu run **Tools/Foundation/Cloud Settings**
    - **Application Key** is found when you create an app on the server.
    - In the Service URL place the url to your cloud instance.
    - If you do not have a private instance**http://unity3dfoundation** is available for testing.

Dependencies

Lobby depends on a number of other packages. **These packages are open source and included**.

- FullSerializer Json Library
- Foundation.Tasks Async Library

It is recommended that you take a look at the AsyncTask readme before continuing. Async tasks are a pattern for running work in the background and 'waiting' for it. Tasks are used **heavily** in Foundation Cloud.

Example

There is an example scene located under**Foundation/Cloud/Example** Take a look at this scene as it demonstrates every possible operation in it's script file.

Guide

Foundation cloud's code is not meant to be modified. Below, I will describe the major components of the package and how to use them.

CloudSession

Cloud session is a cache for holding the user's identity and persisting between sessions. Persistence is maintained by use of PlayerPrefs. CloudSession is used internally by all services (specifically the account service) and should not be used directly. That said, reading your account Id might be necessary from time to time.

CloudAccount

Static service for communicating with the Accounts / Authentication Service

```
/// <summary>
/// Static instance.
/// </summary>
public static readonly CloudAccount Instance;

/// <summary>
/// SignIn from cache (remember me)
/// </summary>
/// <returns>true if authenticated</returns>
public void SignIn();
```

```
/// <summary>
/// Sign out and clear cache
/// </summary>
public void SignOut();

/// <summary>
/// Requests a new guest account. Use for 'Skip Sign In' option.
/// </summary>
/// <returns></returns>
public Task Guest();

/// <summary>
/// New account created in memory
/// </summary>
/// <returns></returns>
public Task SignUp(string email, string password);

/// <summary>
/// Sign in request
/// </summary>
/// <param name="email"></param>
/// <param name="password"></param>
/// <returns></returns>
public Task SignIn(string email, string password);

/// <summary>
/// Update account details
/// </summary>
/// <param name="email"></param>
/// <param name="password"></param>
/// <returns></returns>
public Task Update(string email, string password);

/// <summary>
/// Tells the server to send out an recovery email. This email will contain a reset token.
/// </summary>
/// <param name="email"></param>
/// <returns></returns>
public Task Recovery(string email);

/// <summary>
/// Updates the account with a new password. Token from Recovery Email.
/// </summary>
/// <param name="token"></param>
/// <param name="password"></param>
/// <returns></returns>
public Task Reset(string token, string password);

/// <summary>
/// Deletes the current account
/// </summary>
/// <param name="email"></param>
/// <param name="password"></param>
/// <returns></returns>
public Task Delete(string email, string password);
```

CloudRealtime

Service for Realtime Messaging Authentication. Foundation Cloud Server supports authenticating your Realtime messenger server side. This allows you to keep your private key off the client if you choose to. For more information please see http://realtime.co

```
/// <summary>
```

```
/// Static Instance.
/// </summary>
public static readonly CloudRealtime Instance;

/// <summary>
/// New account created in memory
/// </summary>
/// <returns>Realtime Authentication Token to use</returns>
public Task<RealtimeToken> SignIn(Dictionary<string, string[] >channels);
```

CloudQuery

The CloudQuery is an object for constructing an odata query in a linq-like fashion. This allows one to filter, sort and truncate a search for a collection of objects easily.

Example Use :

```
var query = new CloudQuery<DemoObject>().WhereGreaterThan("Id", 5).OrderBy("String").Take(5);
```

```
///<summary>
/// Where the property contains
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">String, Number</param>
public CloudQuery<T> WhereContains(string key, object value);

///<summary>
/// Where the property string value starts with
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">String, Number</param>
public CloudQuery<T> WhereStartsWith(string key, object value);

///<summary>
/// Where the property string value ends with
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">String, Number</param>
public CloudQuery<T> WhereEndsWiths(string key, object value);

///<summary>
/// Where the property value equals
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">String, Number</param>
public CloudQuery<T> WhereEquals(string key, object value);

///<summary>
/// Where the property value does not equal
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">String, Number</param>
public CloudQuery<T> WhereNotEquals(string key, object value);

///<summary>
/// Where the (numeric) property value is greater than
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">Number, DateTime</param>
public CloudQuery<T> WhereGreaterThan(string key, object value);
```

```
///<summary>
/// Where the (numeric) property value is less than
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">Number, DateTime</param>
public CloudQuery<T> WhereLessThan(string key, object value);

///<summary>
/// Where the (numeric) property value is greater than or equal to
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">Number, DateTime</param>
public CloudQuery<T> WhereGreaterThanOrEqualTo(string key, object value);

///<summary>
/// Where the (numeric) property value is less than or equal to
/// </summary>
/// <param name="key">property name</param>
/// <param name="value">Number, DateTime</param>
public CloudQuery<T> WhereLessThanOrEqualTo(string key, object value);

///<summary>
/// Sorts with the highest value first
/// </summary>
/// <summary>
/// Orders the selection by the property
/// </summary>
/// <param name="key">property name</param>
public CloudQuery<T> OrderByDescending(string key);

/// <summary>
/// Sorts with the lowest value first
/// </summary>
/// <param name="key">property name</param>
public CloudQuery<T> OrderBy(string key);

/// <summary>
/// Skip, For paging.
/// </summary>
/// <param name="count"></param>
public CloudQuery<T> Skip(int count);

/// <summary>
/// Truncate, For paging.
/// </summary>
/// <param name="count"></param>
public CloudQuery<T> Take(int count);
```

CloudStorage

CloudStorage is a service for creating, reading updating and deleting objects. You may use this service for all sorts of objects you might create such as saved games, high scores, player profiles, or all three. Before use you will need to decorate your data objects with two of three storage annotations.

[StorageTable]

Required StorageTable is placed on the class of an object that will be used by CloudStorage. It includes the name of the entity's table.

[StorageIdentity]

Required StorageIdentity flags the unique identity field of the object. It is strongly recommended that this filed be a string property. When creating an identity for a new instance of an object please use a GUID.

[StorageScore]

Optional Flags a field as the default sorting value. When a object query is requested the default 'orderby' will be defined by the StorageScore value. This annotation was included for easy transitioning to a high speed Redis database serverside.

StorageObject Example

```
[StorageTable("DemoObject")]
class DemoObject
{
[StorageIdentity]
public string Id { get; set; }
[StorageScore]
public int Score { get; set; }
}
```

CloudStorage Interface

```
/// <summary>
/// Static Instance
/// </summary>
public static readonly CloudStorage Instance;

/// <summary
/// Reads a collection of objects which match a cloud query.
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="query"></param>
/// <returns></returns>
public HttpTask<T[]> Get<T>(CloudQuery<T> query);

/// <summary>
/// Reads a single of object from the server.
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="id"></param>
/// <returns></returns>
public HttpTask<T> Get<T>(string id) ;

/// <summary>
/// Reads a set of objects from the server.
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="ids"></param>
/// <returns></returns>
public HttpTask<T[]> GetSet<T>(string[] ids);
```

```
/// <summary>
/// Saves a new object serverside.
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="entity"></param>
/// <returns></returns>
public HttpTask Create<T>(T entity;

/// <summary>
/// Saves a new object serverside. Includes write protection (AVL).
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="entity"></param>
/// <param name="acl">protection group</param>
/// <param name="param">User name</param>
/// <returns></returns>
public HttpTask Create<T>(T entity, StorageACL acl, string param) ;

/// <summary>
/// Saves an existing object serverside
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="entity"></param>
/// <returns></returns>
public HttpTask Update<T>(T entity);

/// <summary>
/// Saves an existing object set serverside
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="entities"></param>
/// <returns></returns>
public HttpTask UpdateSet<T>(T[] entities);

/// <summary>
/// Updates a single property on an existing object
/// </summary>
/// <param name="id">ObjectId</param>
/// <param name="propertyName"></param>
/// <param name="propertyValue"></param>
/// <returns></returns>
public HttpTask UpdateProperty(string id, string propertyName, string propertyValue) ;

/// <summary>
/// Increments / Decrements a single property.
/// </summary>
/// <param name="id">ObjectId</param>
/// <param name="propertyName"></param>
/// <param name="delta">change</param>
/// <returns></returns>
public HttpTask UpdateDelta(string id, string propertyName, float delta = 1) ;

/// <summary>
/// Increments / Decrements a single property.
/// </summary>
/// <param name="id">ObjectId</param>
/// <param name="propertyName"></param>
/// <param name="delta">change</param>
/// <returns></returns>
public HttpTask UpdateDelta(string id,  string propertyName, int delta = 1) ;

/// <summary>
/// Deletes the entity serverside
/// </summary>
/// <typeparam name="T"></typeparam>
/// <param name="entity"></param>
/// <returns></returns>
public HttpTask Delete<T>(T entity) w
```

CloudService

The CloudService wrapps communication with the server and includes common error handling and serialization. It is used internally by the CloudAccount, CloudStorage and CloudRealtime services.

The CloudService is an advanced class for the creating of custom api's. Specifically, if you write a new ApiController serverside (e.g. A strongly typed storage controller) you may use the CloudService to communicate with this new service.

Example Custom Service

```
/// <summary>
/// Service for communicating with the serverside ApiCatController
/// </summary>
public class CatService
{
  public static readonly CatService Instance = new CatService();

  public CloudConfig Config
  {
  get { return CloudConfig.Instance; }
  }

  public CloudSession Session
  {
  get { return CloudSession.Instance; }
  }

  // Include name of controller in the constructor
  public readonly CloudService Service = new CloudService("Cat");

  /// <summary>
  /// Calls the LickSelf ActionMethod on the controller.
  /// </summary>
  public Task<RealtimeToken> LickSelf()
  {
      return Service.Post("LickSelf");
  }
}
```