uniqtem

Hellgate Framework Version 1.4.0

- Download Asset Store
- Scripts Reference
- Basic Unity project framework
- Requires Unity 5.0 or higher.
- Requires NGUI: Next-Gen UI kit.
- Requires SS-System 3.0: UI and Scene Manager.
- External MiniJSON.
- Scripts in c#

---

uniqtem

目次

目次
1. 紹介
2. 設置方法
    2.1. Asset Download
    2.2. SS-System コード修正
3. Hello World!!
    3.1. Scenes in build 設定
    3.2. Asset Labels 設定
    3.3. 実行
4. AssetBundles
    4.1. Initial download
    4.2. AssetBundle Manager
5. Http
    5.1. Http Manager
6. Object Pooling
    6.1. Object Pool Manager
7. Reflection
    7.1. Data Convert
8. Register/Encrypt
    8.1. Register
    8.2. Encrypt
9. Scene Manager & Controller
    9.1. Scene Manager
    9.2. Scene Controller
    9.3. Loading Job Controller
10. Sqlite3
    10.1. Sqlite3 ORM(Object-Relational Mapping)
    10.2. Sqlite3 OM(Object Mapping)
11. Runtime Build Version
12. Editor
    12.1. Build AssetBundles
    12.2. Json Converter for Excel
    12.3. Json Converter for Sqlite DB
    12.3. Create Sqlite DB

1. 紹介

Hellgate Frameworkは、Unityで一般Applicationやゲーム開発するにあたり、必要な部分を総合してFramework化したプロジェクトです。強力かつ高速に開発を行うことができ、使用も簡単で、開発の利便性を確保します。

Hellgate Frameworkは、すべての機能にサンプルとReferenceドキュメントを提供しています。そして、様々なサンプルコードを使用して、理解と使用を助け、実際にサンプルコードが実行されることを目で確認することにより、理解と使用の助けは倍になります。現在Google Playからサンプル Applicationまでダウンロードできるのが最大の利点です。

Hellgate Frameworkは、実際のプロジェクトを進行しながら、経験や感覚や確認された結果を現在のプロジェクトに反映しました。したがって、プロジェクトを進行するにあたり、反復的な開発と変更されている仕様に迅速かつ的確に対応することができます。ほとんどのプロジェクトは、戦闘Scene開発以外は、形式と方法がほぼ同一であると見ることができます。この部分を解決するためのプロジェクトです。

Hellgate Frameworkプロジェクトは進行中です。現在のプロジェクトは、完成したプロジェクトではなく、継続開発中であり、より良い方向、より利便性、より力強さなど、常により一層の発展のために努力しています。そして、高速バグ対応とアップデートもしていきます。

Hellgate Frameworkが自慢する機能
Loading Job Controller：AssetBundle load、Http Request/Response、IntentなどのJob処理後Scene切り替え、Loading Bar表現、Data伝達。
Reflection Convert：Dictionary、Listを利用してInstance生成が簡単です。もちろん、逆も可能です。
Sqlite ORM：必ず必要な部分だけを簡単かつ強力に使用可能です。 OMもできます。
Scene Manager/Controller：SS-Systemを用いたScene呼び出し/削除/アクティブ/非アクティブ
AssetBundle：Build、Initial Download、Manager。
Http：Requrt/Response、UI連動。
Encrypt：UnityのPlayerPrefs（Game Sessions）を暗号化して使用可能です。

他にも多くの機能があります。

2. 設置方法

Hellgate FrameworkはNGUI、SS-System二つのAssetが必要です。NGUIはUI関連に使用しています。NGUIを使わなくても大丈夫ですが一部の技能とサンプルを使うことができません。
SS-Systemは無料、Scene Managerのベースです。

2.1. Asset Download

NGUI：https://www.assetstore.Unity.com/en/#!/content/2413
SS-System：https://www.assetstore.Unity.com/en/#!/content/12947

2.2. SS-System コード修正

SS-Systemのコード修正が必要です。
1. SSSystem/Scripts/SSRoot.cs

```
    protected virtual void Update()
    {
        #if UNITY_EDITOR
        Rename();
//          FindCameras();
        FindEventSystems();
        #endif
    }
```

2. SSSystem/Scripts/Compatibility/nGUI/SSAutoAddUICamera.cs

```
    private void Awake()
    {
        gameObject.AddComponent<UICamera> ();
//          UnityEngineInternal.APIUpdaterRuntimeServices.AddComponent(gameObject,
"Assets/SSSystem/Scripts/Compatibility/nGUI/SSAutoAddUICamera.cs (8,3)", "UICamera");
    }
```

以上でプロジェクトの準備が完了しました。

```
using Hellgate;
```

だけでHellgate Frameworkすべての技能を使うのができます。

3. Hello World!!

Hellgate Frameworkで提供されるサンプルコードを起動して機能の把握ができます。すべての技能についてサンプルコードがあります。Google Playでダウンロードもできます。

Google play：https://play.google.com/store/apps/details?id=com.uniqtem.hellgate

3.1. Scenes in build 設定

Unity Editor/File/Build SettingsのAssets/Hellgate/Examples/ScenesのScene全体を設定します。（[図1]参照）

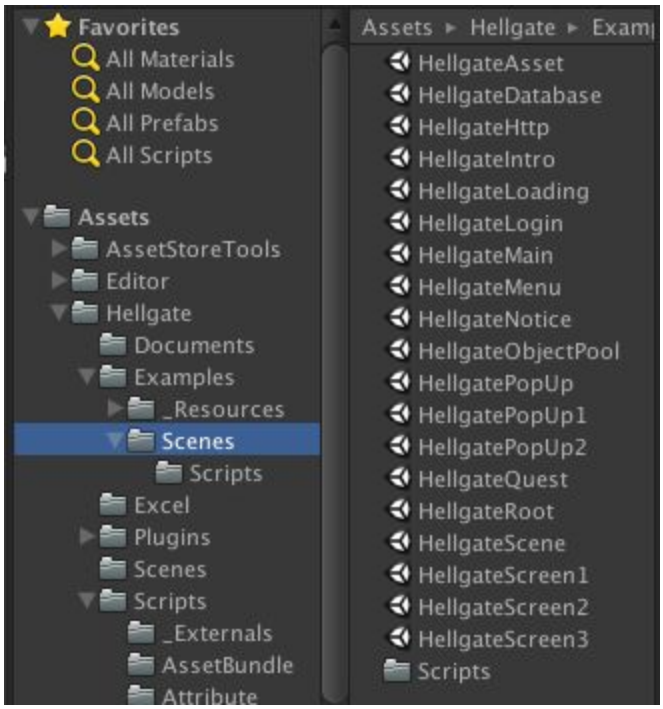

図 1.

[図2]のように表示されたらScenes in build設定は完了です。

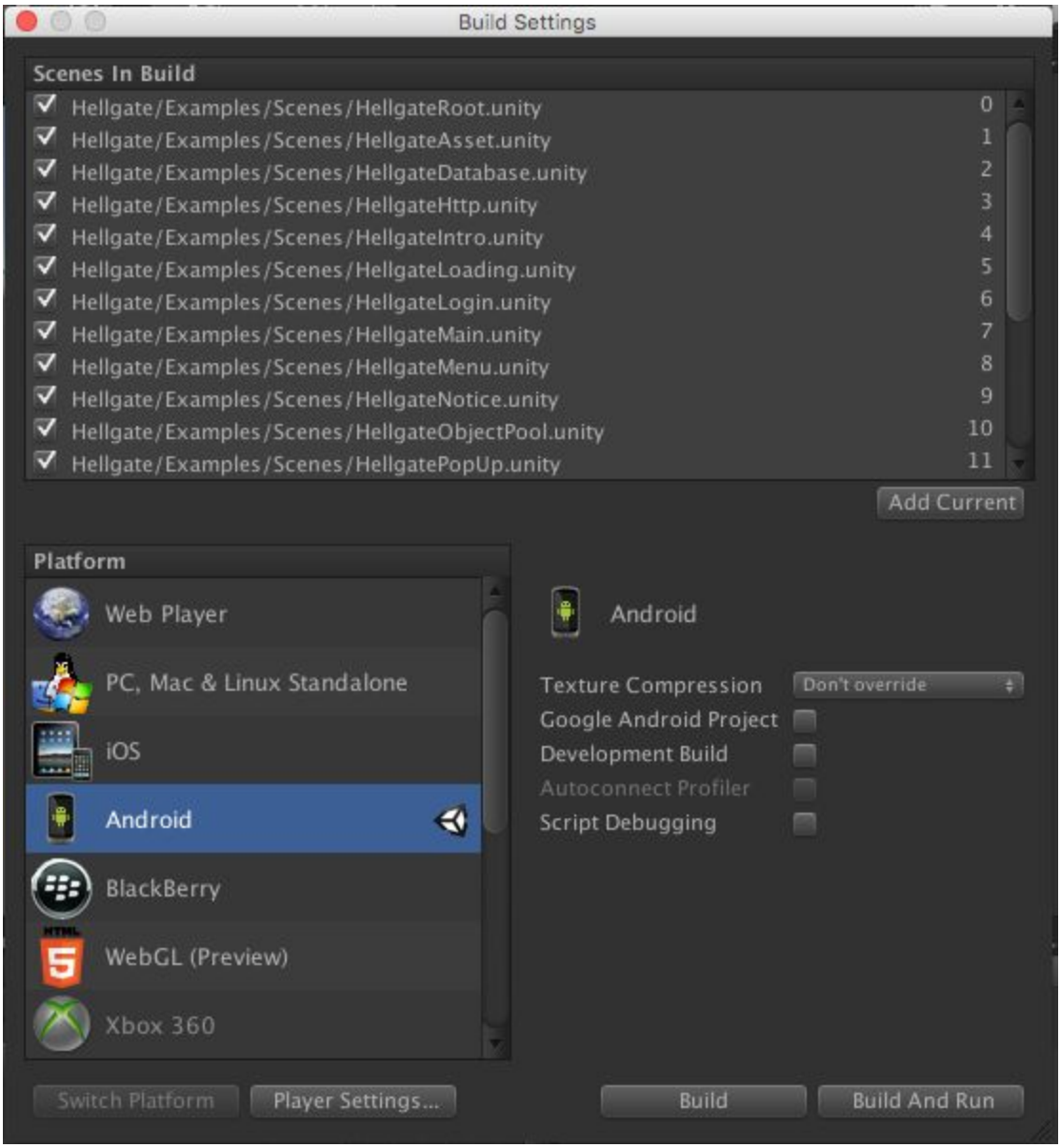

図 2.

3.2. Asset Labels 設定

Assets/Hellgate/Examples/AssetBundleにフォルダをAsset Labelsを設定します。
- Asset -> hellgateasset.Unity
- Http -> hellgateasset.Unity
- Main -> hellgatemain.Unity
- Master -> hellgatemaster.Unity
- ObjectPool -> hellgateobjectpool.Unity
- Quest -> hellgatequest.Unity
- Scene -> hellgatescene.Unity

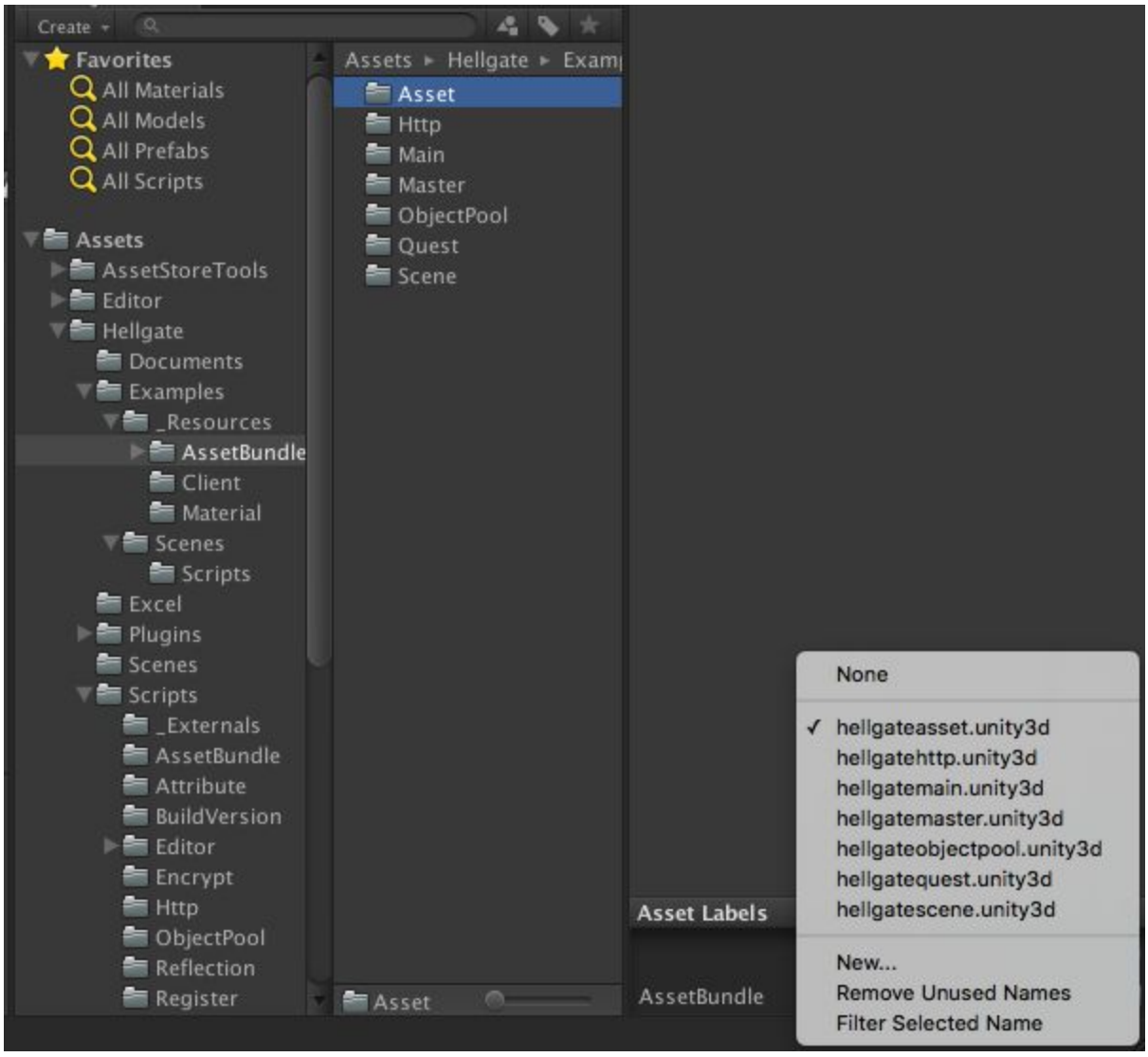

図 3.

3.3. 実行

Assets/Hellgate/Examples/Scenes/Hellgate Root.unityで実行すると、完了。 [図4]が実行画面です。たまにDropboxサーバーの問題でTimeoverになる場合があります。

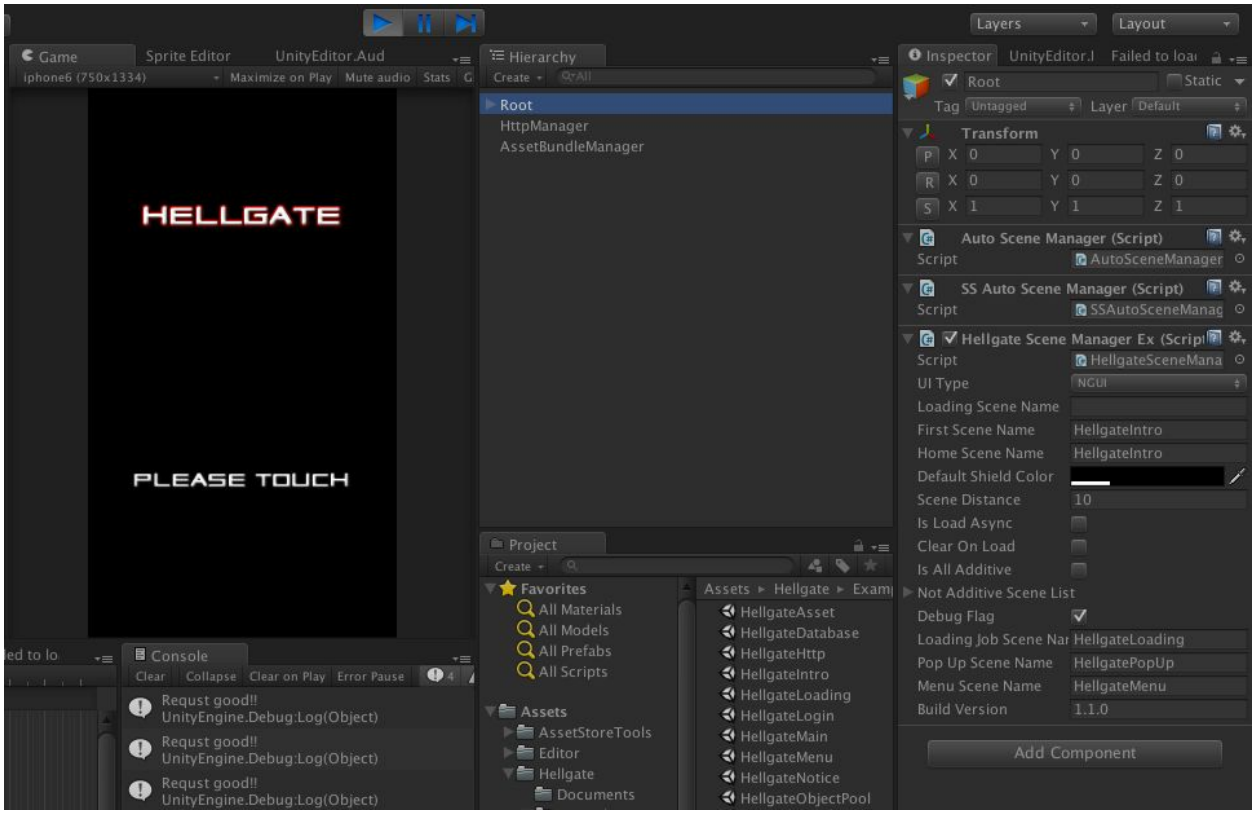

図 4.

4. AssetBundles

Hellgate FrameworkではResourceダウンロード、使用、管理などUnityのAssetBundlesを使用しています。 Unityで開発をするなら一度理解して乗り超えなければならない部分であると思います。そしてUnity5.0以上AssetBundles仕様に従っており、それ以下のバージョンはサポートしていません。

4.1. Initial download

Applicationが実行した後にResource listをダウンロードができます。色々なゲームで、最初にResourceをダウンロード、更新などをした後、ゲームを進行することができるようにするため、まさにその機能です。

Resource listを管理Json from
https://dl.dropboxusercontent.com/u/95277951/hellgate/8118ea697fed5008b03c4a67c4eeb5bb33ae2775/pc/resource.json

```
{
        "resource": {
           "major": 1,
           "minor": 1,
           "version": 1,
           "name": "resource"
        },
        "assetbundle": [
           {
              "name": "hellgatemaster",
              "version": 1
           },
           {
              "name": "hellgatemain",
              "version": 1
           }, ...
        ]
}
```

"major"：リソース全体を削除した後、完全なダウンロード
"minor"： "assetbundle"：[]の上位バージョンに変更する場合、サブarrayバージョンcheck後ダウンロード/アップデート

Ex)

```
// downloader instance
// response json from
AssetBundleInitialDownloader aDownloader = new AssetBundleInitialDownloader ("URL");
aDownloader.aEvent = CallbackDownloader;
aDownloader.Download ();

/// <summary>
/// Callbacks the downloader.
/// </summary>
/// <param name="status">Status.</param>
private void CallbackDownloader (AssetBundleInitalStatus status)
{
    if (status == AssetBundleInitalStatus.Start) {
        // download start
    } else if (status == AssetBundleInitalStatus.Over) {
        // not download list
    } else if (status == AssetBundleInitalStatus.HttpTimeover) {
        // request time over
    } else if (status == AssetBundleInitalStatus.HttpOver) {
        // request over
    } else if (status == AssetBundleInitalStatus.HttpError) {
        // request error
    } else if (status == AssetBundleInitalStatus.DownloadOver) {
        // download over
    }
}

/// <summary>
/// Update this instance.
/// Initial download progress bar and status
/// </summary>
void Update ()
{
    // AssetBundle status.
    if (aStatus == AssetBundleInitalStatus.Start && aDownloader != null) {
        slider.value = aDownloader.Progress;
        pLabel.text = aDownloader.SProgress;
        cLabel.text = aDownloader.CurretIndex + " / " + aDownloader.DownloadCount;
    }
}
```

4.2. AssetBundle Manager

AssetBundle Managerを利用して簡単にResource Loadが可能です。Editorで実行時には、AssetBundleでloadせずに、そのパスから直接Loadするので再度ダウンロードせずに更新されます。したがって、Editorテストが簡単です。

Ex)

```
// sprite load(Single)
AssetBundleData data = new AssetBundleData ("AssetBundle Name");
data.objName = "Name";
data.type = typeof(Sprite);

AssetBundleManager.Instance.LoadAssetBundle (data, delegate(object obj) {
    Sprite temp = obj as Sprite;
    sprite.sprite2D = temp;
});

// prefabs load(Multi)
string[] loads = new string[] {
    "CubeGreen", "CubeYellow"
};

List<AssetBundleData> datas = new List<AssetBundleData> ();
for (int i = 0; i < loads.Length; i++) {
    AssetBundleData data = new AssetBundleData ("AssetBundle Name", loads [i]);
    datas.Add (data);
}

AssetBundleManager.Instance.LoadAssetBundle (datas, delegate(object obj) {
    List<object> objs = obj as List<object>;
    cubes = Util.GetListObjects<GameObject> (objs);

    Instantiate (cubes [ran.Next (0, loads.Length)]);
});
```

5. Http

Hellgate Frameworkでは、通信ユーティリティモジュールを提供しています。 UI連動まで簡単に可能です。

5.1. Http Manager

Http Managerを利用してrequest/response処理が可能です。

Ex)

```
        // Set base url(Static).
        HttpData.BASE_URL = "base url";

        // Set base url(Static).
        HttpData http = new HttpData ("uri");
//          http.popUp = true; // loading popup ui on/off
//          http.retry = 3; // retry count
//          http.timeout = 10; // time out second
        http.finishedDelegate = delegate (WWW www) {
            if (www == null) { // time over
            } else if (www.error != null) { // error
            } else {
                // done
            }
        };
        HttpManager.Instance.GET (http);
```

6. Object Pooling

Hellgate Frameworkでは、Object Poolを提供しています。Objectを必要に応じて生成し、破壊する方式ではなく、適切な数を事前に作成して、これをActive true/falseする方式です。

6.1. Object Pool Manager

Object Pool Managerを利用してObjectを作成し、リサイクルします。

Ex)

```
public GameObject prefab;

void Awake ()
{
    // if you want to create in advance
    ObjectPoolManager.Init (prefab);
}

 // object set active(true)
GameObject temp = ObjectPoolManager.Spawn (prefab);
// object delay 2 second set active(false)
ObjectPoolManager.DelayDespawn (temp, 2f);
// object set active(false)
ObjectPoolManager.Despawn (temp);
```

7. Reflection

Hellgate Frameworkでは、Reflectionを利用してList、DictionaryをClass instance生成することが1行のコードでできます。もちろん逆変換も可能です。

7.1. Data Convert

Data Convertは、特にJsonデータを活用するのに役立ちます。
Json：
https://dl.dropboxusercontent.com/u/95277951/hellgate/8118ea697fed5008b03c4a67c4eeb5bb33ae2775/reflection.json

Ex)

```
HttpData http = new HttpData ("reflection", "json");
      http.finishedDelegate = delegate (WWW www) {
         if (www == null) { // time over
         } else if (www.error != null) { // error
         } else {
            // class reference
            // dictoinary -> HellgateReflectionDataEx instance
            HellgateReflectionDataEx data = Reflection.Convert<HellgateReflectionDataEx>
((IDictionary)MiniJSON.Json.Deserialize (www.text));

            // HellgateReflectionDataEx instance -> dictoinary
            IDictionary iDic = Reflection.Convert<HellgateReflectionDataEx> (data);
         }
      };
      HttpManager.Instance.GET (http);
```

8. Register/Encrypt

Hellgate Frameworkでは、UnityのPlayerPrefs（Game Session）機能を暗号化して使用可能です。暗号化は、3DESC、MD5、SHA1アルゴリズムを使用しています。

8.1. Register

PlayerPrefs classの代わりにRegister classを使用します。使用方法は同じです。

Ex)

```
Register.SetInt ("KeyInt", 10);
Register.SetFloat ("KeyFloat", 20.20f);
Register.SetString ("KeyString", "Hellgate Framework");

int num = Register.GetInt ("KeyInt", 0);
float f = Register.GetFloat ("keyFloat", 0);
string str = Register.GetString ("KeyString", "defaultValue");
```

8.2. Encrypt

Http Requestでurlやpostなどの暗号化をしてくれれば、セキュリティに役立ちます。

Ex)

```
// create manifest url
string encrypt = Encrypt.SHA1Key (BuildVersionBindings.GetBuildVersion () + "Hellgate");
//      Debug.Log (encrypt);
List<string> param = new List<string> ();
param.Add (BASE_URL);
param.Add (encrypt);
param.Add (Util.GetDevice ());
param.Add ("manifest");

string url = Http.CreateURL (param, "json");
HttpData hD = new HttpData (url);
hD.popUp = false;
hD.finishedDelegate = CallbackManifest;
HttpManager.Instance.GET (hD);
```

9. Scene Manager & Controller

Hellgate Frameworkでは、SS-Systemを利用して、多くのSceneを簡単に管理することができます。 Screen、PopUp、Menu、Loading Jobなど、様々な形でSceneを管理、表現することができ、簡単な使い方を誇りを持っています。基本仕様は、SS-Systemを使用します。

SS-System Advance : http://anh-pham.appspot.com/sssystem/en/ssadvance.pdf

9.1. Scene Manager

Scene Manager static instanceを利用してScene呼び出し/削除/アクティブ/非アクティブなどができます。

Ex)

```
// load scene
SceneManager.Instance.Screen ("scene name");

// load popup
SceneManager.Instance.PopUp ("popup name");

// close popup (stack)
SceneManager.Instance.Close ();

// load menu
SceneManager.Instance.LoadMenu ("menu name");

 // load message popup
SceneManager.Instance.PopUp ("Yes and No.", PopUpType.YesAndNo);
SceneManager.Instance.PopUp ("Okay.", PopUpType.Ok);

// load main menu
SceneManager.Instance.LoadMainMenu ();

// show main menu
SceneManager.Instance.ShowMainMenu ();

// hide main menu
SceneManager.Instance.HideMainMenu ();
```

9.2. Scene Controller

Scene ControllerはScene Managerを使用してloadされたsceneです。

Ex)

```
public class HellgateSceneController : SceneController
{
   public override void OnSet (object data)
   {
      base.OnSet (data);
      // init
   }

   public override void OnShow ()
   {
      base.OnShow ();
      // show
   }

   /// <summary>
   /// Android back key
   /// </summary>
   public override void OnKeyBack ()
   {
      // message popup
      base.Quit ("Exit ?");
   }

   ....
}
```

9.3. Loading Job Controller

Loading Job ControllerはScene移動の時AssetBundles、Http、IntentなどをJob形で自動処理した後Scene移動とEventを送るControllerです。

Ex)

```
	// http request list
	List<HttpData> https = new List<HttpData> ();
	https.Add (new HttpData ("URL1"));
	https.Add (new HttpData ("URL2"));

	// load asset bundle list
	List<AssetBundleData> assetBundles = new List<AssetBundleData> ();
	assetBundles.Add (new AssetBundleData ("AssetBundle Name", "Name1", typeof (Sprite)));
	assetBundles.Add (new AssetBundleData ("AssetBundle Name", "Name2", typeof
(GameObject)));

	LoadingJobData data = new LoadingJobData ("Next Scene");
	data.https = https;
	data.assetBundles = assetBundles;

	// send intent
	data.PutExtra ("title", "Scene");
	data.PutExtra ("int", 10);
	data.PutExtra ("float", 20.20);

//	data.assetBundleasync = true; // load async assetbundle
//	data.popUp = true; // loading progress
//	data.nextScenePopUp = false; // the next scene is not the popup

	SceneManager.Instance.LoadingJob (data);
```

10. Sqlite3

Hellgate FrameworkではSqlite3 OMとORMができます。 Queryも使用可能です。

10.1. Sqlite3 ORM(Object-Relational Mapping)

Classベースの簡単なAttributeに設定して使用します。
Wiki：https://en.wikipedia.org/wiki/Object-relational_mapping

Ex)

```
/// <summary>
/// table class.
/// </summary>
[Table]
public class HellgateTableName
{
    [Ignore]
    private int idx = 0;
    private int column1 = 0;
    private string column2 = "";

    public int Idx {
        get {
            return idx;
        }
    }

    public int Column1 {
        get {
            return column1;
        } set {
            column1 = value;
        }
    }

    public string Column2 {
        get {
            return column2;
        } set {
            column2 = value;
        }
```

```
}
    Query query = new Query ("DB name.db");

    // insert
    HellgateTableName data = new HellgateTableName ();
    data.Column1 = 10;
    data.Column2 = "test";
    query.INSERT<HellgateTableName> (data);

    // insert batch
    List<HellgateTableName> list = new List<HellgateTableName> ();

    HellgateTableName data1 = new HellgateTableName ();
    data1.Column1 = 10;
    data1.Column2 = "test1";

    HellgateTableName data2 = new HellgateTableName ();
    data2.Column1 = 20;
    data2.Column2 = "test2";

    list.Add (data1);
    list.Add (data2);
    query.INSERT_BATCH<HellgateTableName> (list);

    // Update
    HellgateTableName data = new HellgateTableName ();
    data.Column1 = 10;
    data.Column2 = "test";
//      query.UPDATE<HellgateTableName> (data, "add query");
    query.UPDATE<HellgateTableName> (data, "key", 1);

    // all select
    HellgateTableName[] list = query.SELECT<HellgateTableName> ();

    // select
//      HellgateTableName[] list = query.SELECT<HellgateTableName> ("add query");
    HellgateTableName[] list = query.SELECT<HellgateTableName> ("key", 1);

    // all delete
    query.DELETE<HellgateTableName> ();

    // delete
//      query.DELETE<HellgateTableName> ("add query");
    query.DELETE<HellgateTableName> ("key", 1);
    ....
```

10.2. Sqlite3 OM(Object Mapping)

Dictionary、Listに基づいて使用します。

Ex)

```
Query query = new Query ("DB name.db");

// insert
Dictionary<string, object> data = new Dictionary<string, object> ();
data.Add ("column1", 1);
data.Add ("column2", "test");
query.INSERT ("table name", data);

// update
Dictionary<string, object> data = new Dictionary<string, object> ();
data.Add ("column1", 1);
data.Add ("column2", "test");
//    query.UPDATE ("table name", data, "add query");
query.UPDATE ("table name", data, "key", 1);

// all select
DataTable data = query.SELECT ("table name");

// select
//    DataTable data = query.SELECT ("table name", "add query");
DataTable data = query.SELECT ("table name", "key", 1);

// delete
//    query.DELETE ("table name", "add query");
query.DELETE ("table name", "key", 1);

// all delete
query.DELETE ("table name");
....
```

11. Runtime Build Version

Hellgate Frameworkでは、RuntimeでBuild Versionを持って来ることができます。 Version情報を別途Configに管理する必要はありません。

Ex)

```
string version = BuildVersionBindings.GetBuildVersion ();
```

12. Editor

Hellgate Frameworkでは、種々のEditorがあります。パスはUnity Editor/Window/Hellgateです。

12.1. Build AssetBundles

AssetBundleを実際に作り出すEditorとしてパスはUnity Editor/Window/Hellgate/Build AssetBundlesです。 Asset Labelsに設定した後、[図5]に示すようにOutput folderを指定すると、AssetBundleが作られます。 EditorのBuild Settings PlatformのTargetで設定、PC/Android/iOSのみサポートします。

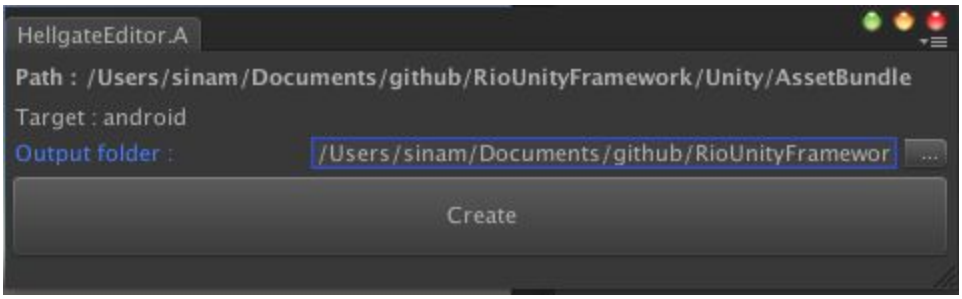

図 5.

12.2. Json Converter for Excel

Excelファイルを利用して、Jsonを作り出すEditorです。 Master dataを作成するときに便利です。パスはUnity Editor/Window/Hellgate/Json Converter for Excelです。実行画面は、[図6]と同じで、Excel fileパスとOutput folderパスを設定するだけでJsonが作成されます。そして[図7]は、Excel Sheetフォームです。 SheetにIgnoreの設定も可能なので活用できます。

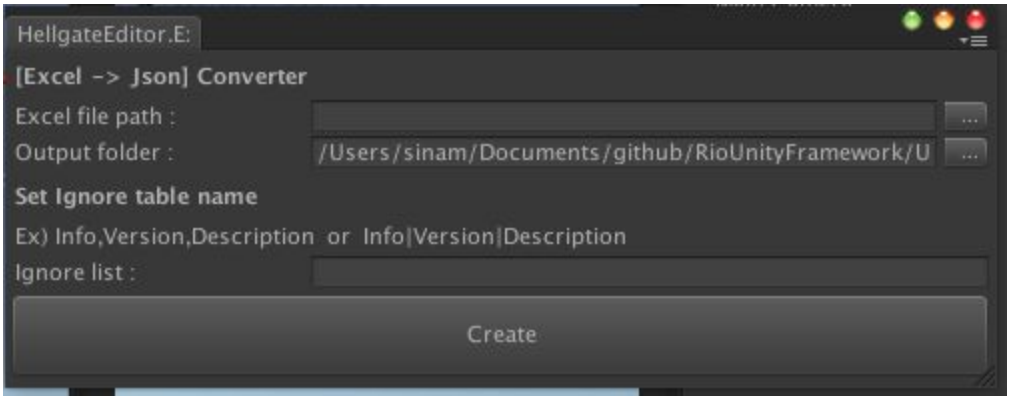

図 6.

idx	name	description	attack	defence	speed
1	aaa	aaaaa	10	10	1
2	bbb	bbbbb	20	20	2
3	ccc	ccccc	30	30	3
4	ddd	ddddd	40	40	4
5	eee	eeeee	50	50	5

図 7.

12.3. Json Converter for Sqlite DB

Excelファイルを利用して、Jsonを作り出すEditorです。 Master dataを作成するときに便利です。パスはUnity Editor/Window/Hellgate/Json Converter for Excelです。実行画面は、[図6]と同じで、Excel fileパスとOutput folderパスを設定するだけでJsonが作成されます。そして[図7]は、Excel Sheetフォームです。 SheetにIgnoreの設定も可能なので活用できます。

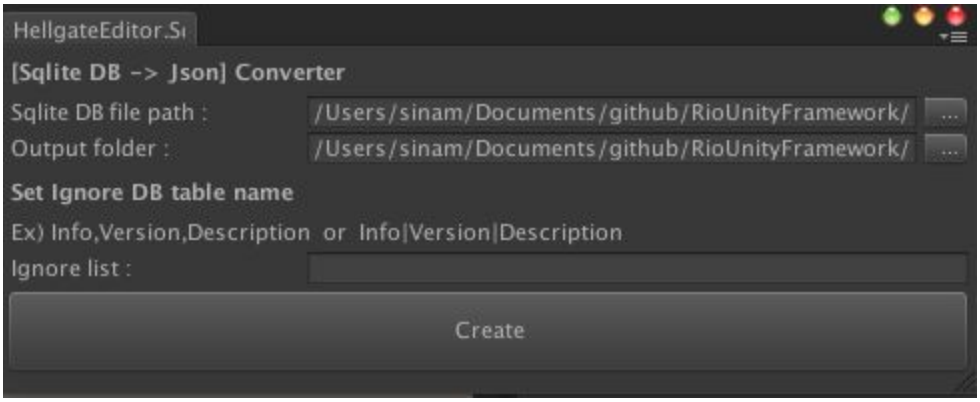

図 8.

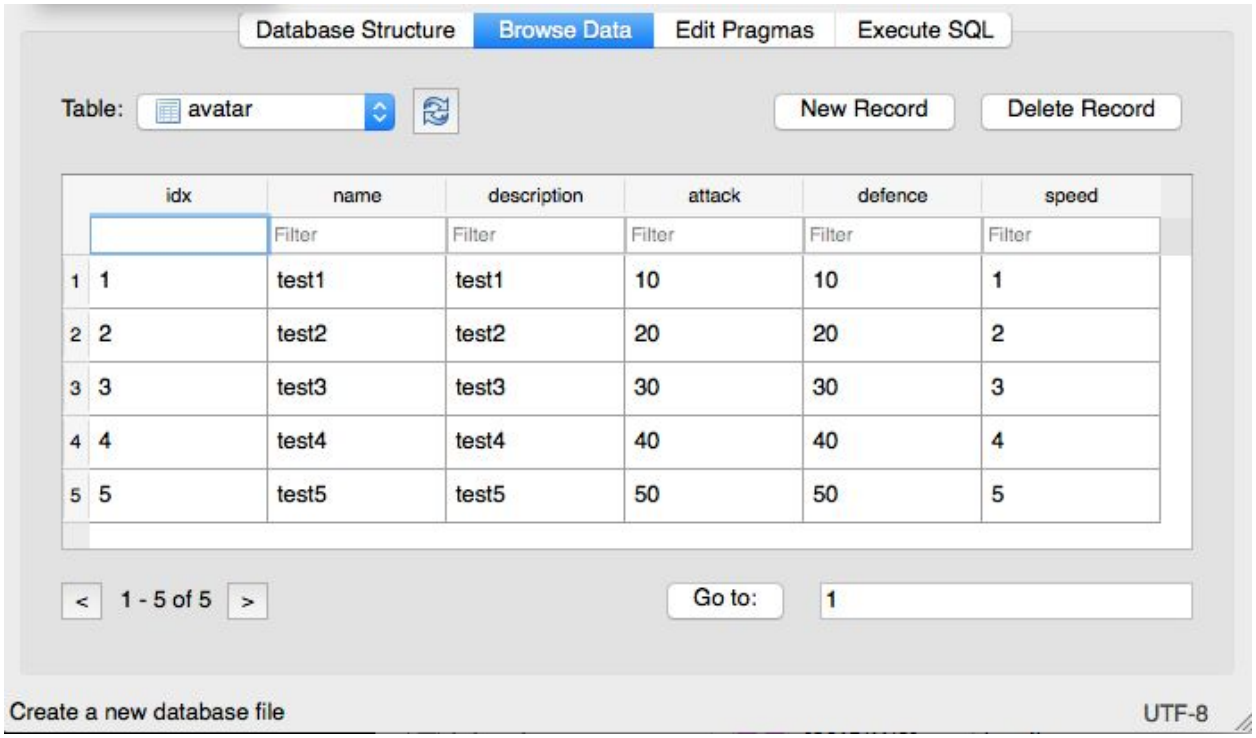

図 9.

12.3. Create Sqlite DB

Table Auto Generated Attributeが設定されたClass読んでSqlite DBを
Assets/StreamingAssetsに作ってくれるEditorです。（参考[図10]、[図11]）

Ex)

```
[Table ("table_sample1", true)]
public class HellgateSampleTableEx
{
    [Column (SqliteDataConstraints.AI)]
    private int idx;
    [Column (new SqliteDataConstraints[] {
        SqliteDataConstraints.NOTNULL, SqliteDataConstraints.UNIQUE
    })]
    private string column1;
    protected float column2;
    // public will not be added to this column.
    public string temp;
}

[Table (true)]
public class Table_sample2
{
    [Column (SqliteDataConstraints.AI)]
    private int idx;
    private string column1;
    private float column2;
    private bool column3;
}
```

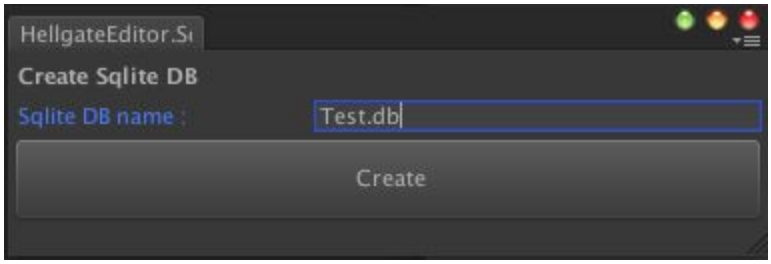

図 10.

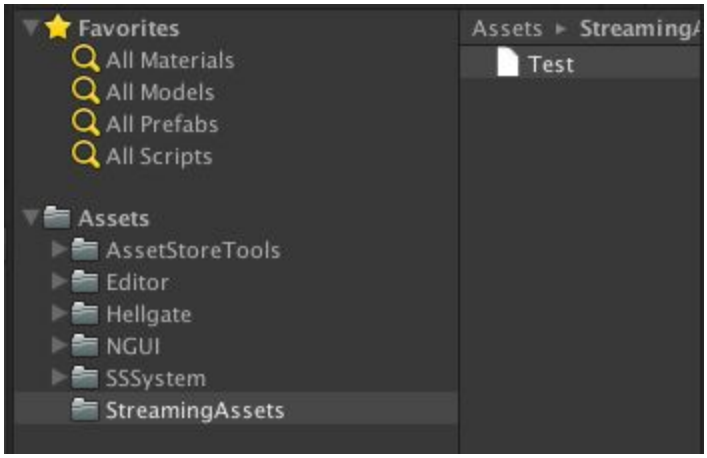
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
AssetStoreTools
Editor
Hellgate
NGUI
SSSystem
StreamingAssets
Assets ▸ Streaming
Test

図 11.