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

목차

1. 소개

Hellgate Framework는 Unity로 일반 Application 및 게임 개발 함에 있어 꼭 필요한 부분들을 종합하여 Framework화 한 프로젝트입니다. 강력하고 빠른 속도로 개발을 할 수 있으며 사용도 간단하여 개발의 편의성을 보장됩니다.

Hellgate Framework는 모든 기능에 예제와 Reference 문서를 제공하고 있습니다. 그리고 다양한 예제 코드를 통해 빠른 이해와 사용에 도움을 주고 실제로 예제 코드가 실행되는 것을 눈으로 확인 함으로써 이해와 사용의 도움은 배가 됩니다. 현재 Google Play에서 예제 Application까지 다운로드 가능한 것이 가장 큰 장점입니다.

Hellgate Framework는 실제 프로젝트를 진행하면서 몸으로 경험하고 느끼고 확인된 결과를 현 프로젝트에 반영되었습니다. 그러므로 프로젝트를 진행함에 있어 반복적인 개발 및 변경되는 사양에 빠르고 정확하게 대응 할 수 있습니다. 대부분의 프로젝트는 전투 Scene 개발 이외의 부분은 형식과 방법이 거의 동일하다고 볼 수 있습니다. 이 부분을 해결하기 위한 프로젝트입니다.

Hellgate Framework 프로젝트는 진행중입니다. 현 프로젝트는 완료된 프로젝트가 아닌 계속 개발 중에 있으며 더 좋은 방향, 더 편리함, 더 강력함 등 항상 더욱 더 발전을 위해 노력 중입니다. 그리고 빠른 버그 대응 및 업데이트가 됩니다.

Hellgate Framework가 자랑하고자 하는 기능
Loading Job Controller : AssetBundle load, Http Request/Response, Intent 등의 Job 처리 후 Scene 전환, Loading Bar 표현, Data 전달.
Reflection Convert : Reflection을 이용한 Dictionary, List로 Instance 생성이 간단합니다. 물론 역으로 가능합니다.
Sqlite ORM : 꼭 필요한 부분만 간단하고 강력하게 사용 가능 합니다. OM도 지원합니다.
Scene Manager/Controller : SS-System를 이용한 Scene 호출/삭제/활성/비활성
AssetBundle : Build, Initial Download, Manager.
Http : Requrt/Response, UI 연동.
Encrypt : Unity의 PlayerPrefs(Game Sessions)을 암호화하여 사용 가능합니다.

이외에도 많은 기능이 있습니다.

2. 설치 방법

Hellgate Framework는 NGUI, SS-System 두개의 Asset을 다운로드 필요합니다. NGUI는 UI를 관련하여 사용 되고 있으며, UGUI를 사용 하셔도 좋지만 일부 기능과 예제를 사용 할 수 없습니다. SS-System은 무료이며 Scene Manager에 기본이 되고 있습니다.

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 코드 수정

다운로드 후 SS-System에서 코드 수정이 필요합니다.
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");
}
```

이것으로 프로젝트를 진행 준비가 완료 되었습니다. 이제

```
using Hellgate;
```

만으로 **Hellgate Framework**의 모든 기능을 사용 할 수 있습니다.

3. Hello World!!

Hellgate Framework에서 제공하는 예제 코드를 돌려보면서 기능 파악을 해 볼 수 있습니다.
모든 기능에 대한 예제 코드가 존재하며 Google Play에서 다운로드도 가능합니다.
Google Play : https://play.google.com/store/apps/details?id=com.uniqtem.hellgate

3.1. Scenes in build 설정

Unity Editor/File/Build Settings에 Assets/Hellgate/Examples/Scenes의 Scene 전체를 설정합니다. ([그림 1] 참고)

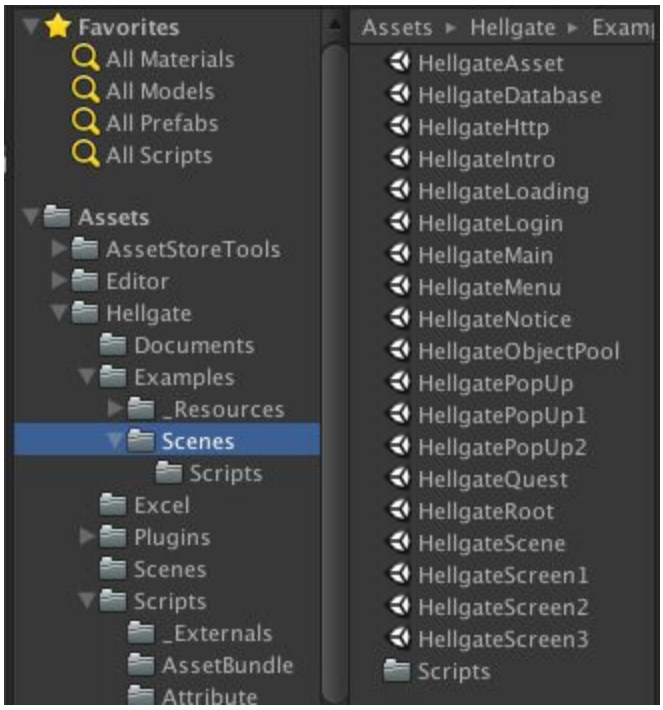

그림 1.

[그림 2]와 같이 표시 되면 Scenes in build 설정은 완료입니다.

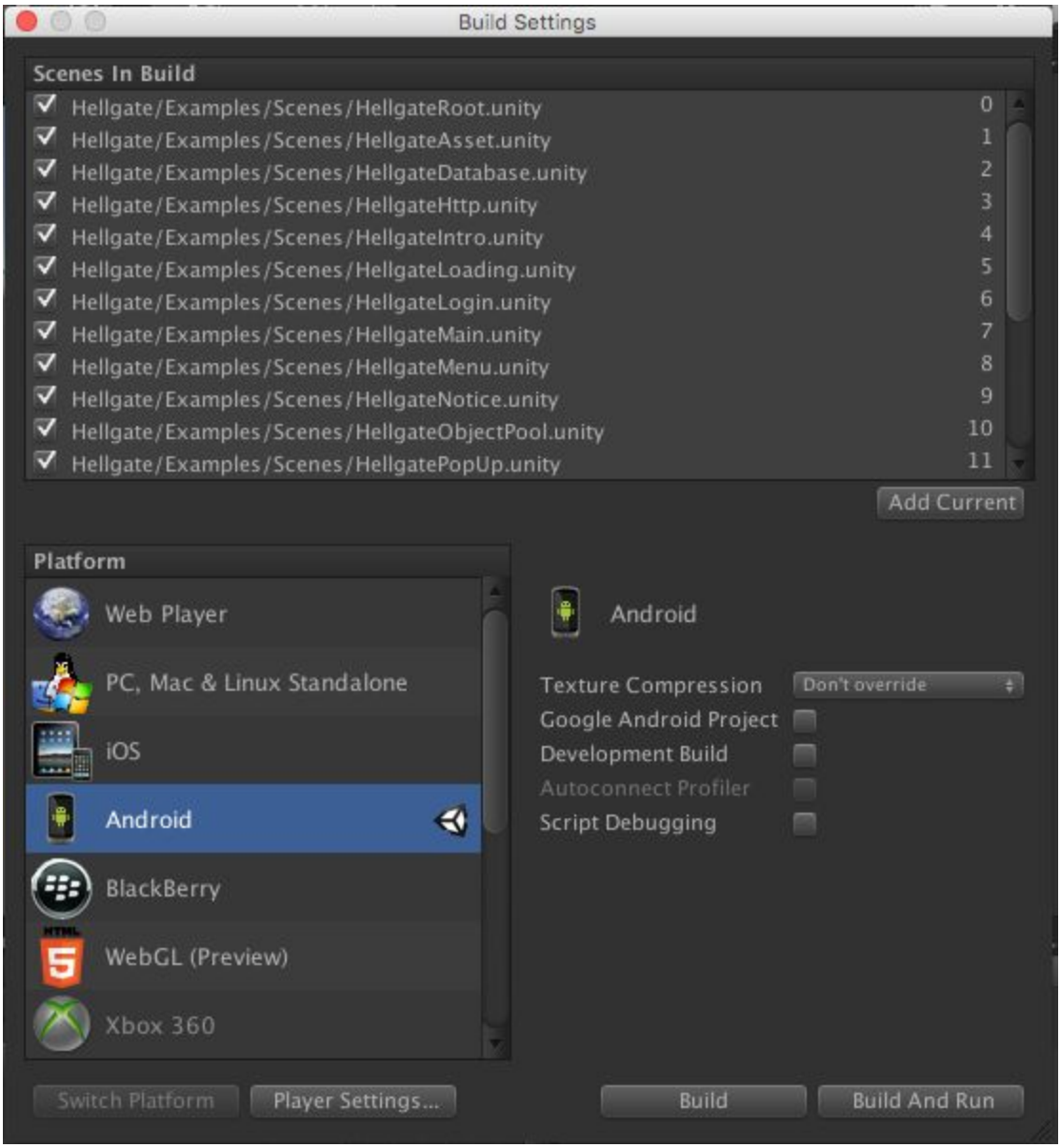

그림 2.

3.2. Asset Labels 설정

Assets/Hellgate/Examples/AssetBundle에 폴더를 Asset Labels를 설정 해 줍니다.
- Asset -> hellgateasset.Unity
- Http -> hellgateasset.Unity
- Main -> hellgatemain.Unity
- Master -> hellgatemaster.Unity
- ObjectPool -> hellgateobjectpool.Unity
- Quest -> hellgatequest.Unity
- Scene -> hellgatescene.Unity

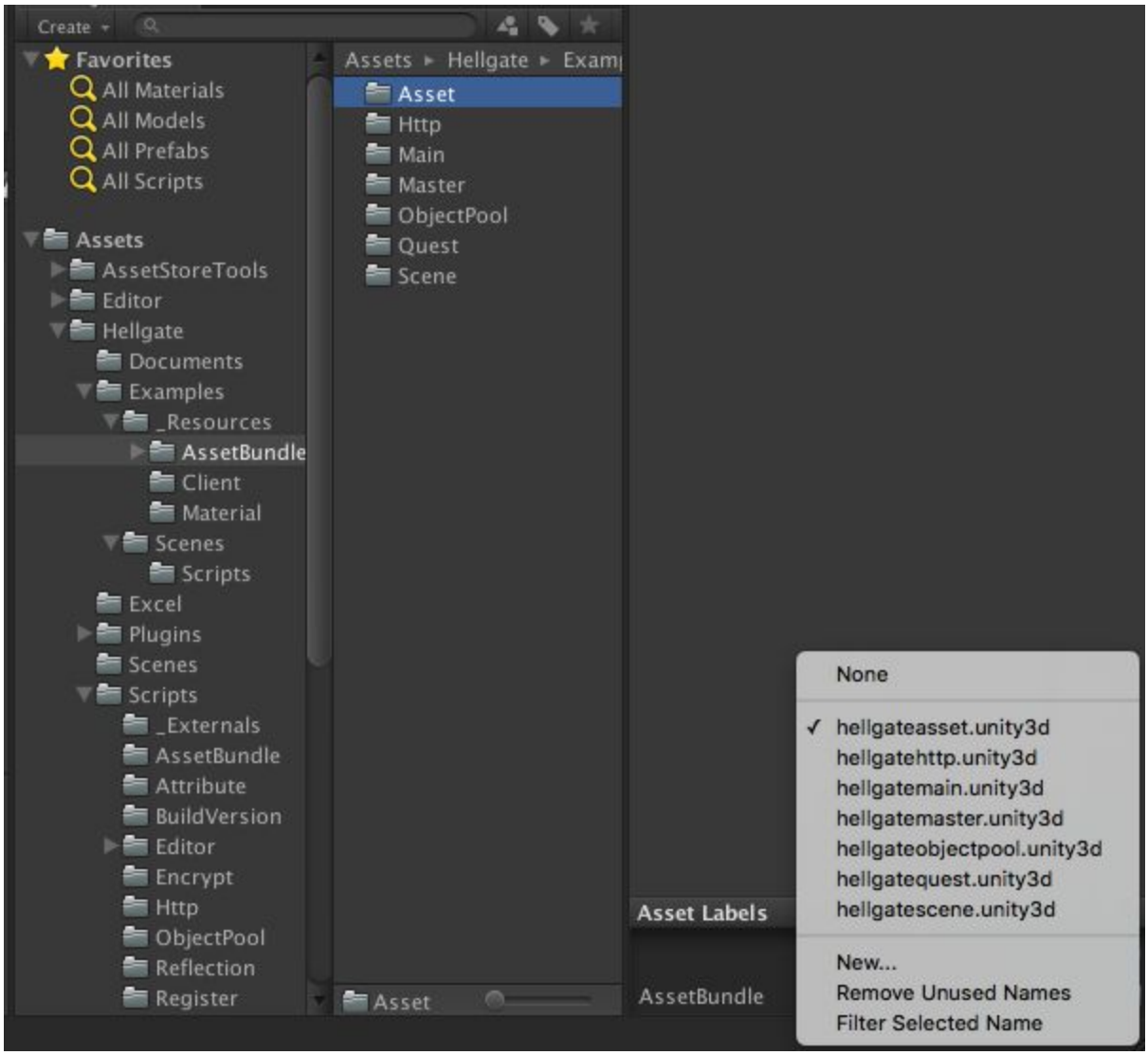

그림 3.

3.3. 실행

Assets/Hellgate/Examples/Scenes/HellgateRoot.unity에서 실행하면 완료. [그림 4]가 실행 화면 입니다. 간혹 Dropbox 서버 문제로 Timeover가 걸릴 수도 있습니다.

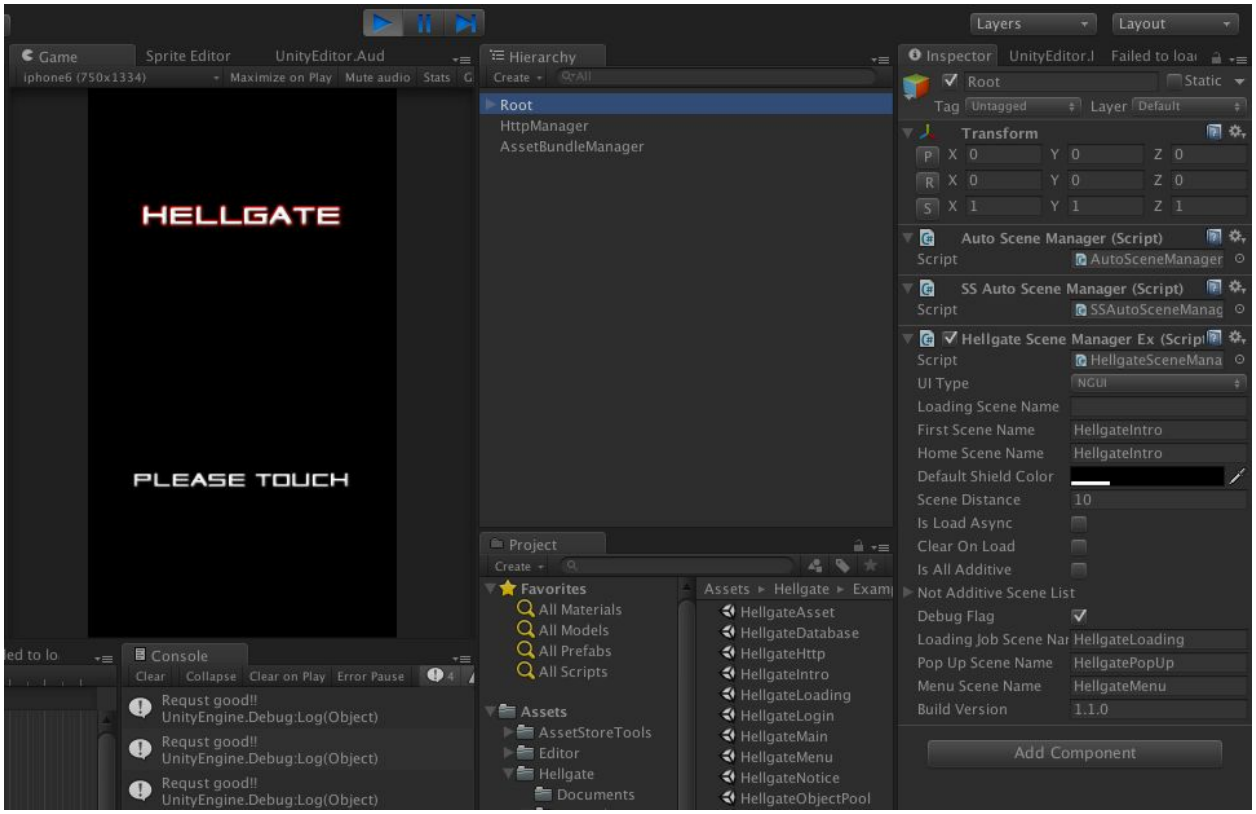

그림 4.

4. AssetBundles

Hellgate Framework에서는 Resource 다운로드, 사용, 관리 등 Unity의 AssetBundles을 사용하고 있습니다. Unity로 개발을 한다면 꼭 한번 이해하고 넘어가야 할 부분 일 것 같습니다. 그리고 Unity 5.0이상의 AssetBundles 사양을 따르고 있으며 그 이하 버젼은 지원하지 않습니다.

4.1. Initial download

Application이 실행 후에 Resource list를 다운로드 할 수 있는 기능입니다. 많은 게임에서 최초에 리소스를 다운로드, 업데이트 등을 한 뒤 게임을 진행 할 수 있도록 하는데 바로 그 기능입니다.

Resource list를 관리 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" : 리소스 전체 삭제 후 전체 다운로드
"minor" : "assetbundle":[] 의 상위 버젼으로 변경시 하위 array 버젼 check 후 다운로드/업데이트

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 Manager를 통해서 간단하게 Resource Load가 가능합니다. Editor에서 실행시에는 AssetBundle에서 load하지 않고 해당 경로에서 직접 Load하기 때문에 다시 다운로드 받지 않고도 갱신 됩니다. 그러므로 Editor 테스트가 편합니다.

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 Framework에서는 통신 유틸리티 모듈을 제공합니다. UI 연동까지 간단하게 가능합니다.

5.1. Http Manager

Http Manager를 통해 request/response 처리가 가능합니다.

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 Framework에서는 Object Pool을 제공합니다. 객체를 필요에 따라서 생성하고 파괴하는 방식이 아닌 적절한 개수를 미리 생성하여 이를 Active true/false 하는 방식입니다.

6.1. Object Pool Manager

Object Pool Manager를 통해서 Object를 생성 및 재활용 합니다.

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 Framework에서는 Reflection을 이용해 List, Dictionary를 Class Instance 생성하는 것이 코드 한 줄로 가능합니다. 물론 역변환도 가능합니다.

7.1. Data Convert

Data Convert는 특히 Json 데이터를 활용 하는데 유용합니다.
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 Framework에서는 Unity의 PlayerPrefs(Game Session)기능을 암호화 하여 사용 가능합니다. 암호화는 3DESC, MD5, SHA1 알고리즘을 사용하고 있습니다.

8.1. Register

PlayerPrefs class 대신 Register class를 사용합니다. 사용 방법은 동일합니다.

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 Request에서 url이나 post 등 암호화를 해주면 보안에 도움이 됩니다.

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 Framework에서는 SS-System을 이용하여 수 많은 Scene들을 편리하게 관리 할 수 있습니다. Screen, PopUp, Menu, Loading Job 등 다양한 형태로 Scene을 관리, 표현 할 수 있으며 간단한 사용법을 자랑 합니다. 기본 사양은 SS-System을 사용합니다.

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

9.1. Scene Manager

Scene Manager static instance를 통해서 Scene 호출/삭제/활성/비활성 등을 할 수 있습니다.

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 Controller는 Scene Manager를 통해 load 된 scene에 해당 됩니다.

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 Controller는 Scene 이동시 AssetBundles, Http, Intent 등을 Job 형태로 자동 처리 후 Scene 이동 및 Event를 보내는 Controller입니다.

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 Framework에서는 Sqlite3 OM과 ORM을 지원합니다. Query도 사용 가능합니다.

10.1. Sqlite3 ORM(Object-Relational Mapping)

Class 기반에 간단한 Attribute으로 설정하여 사용합니다.
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 기반으로 사용 합니다.

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 Framework에서는 Runtime에서 Build Version을 가져 올 수 있습니다. Version정보를 별도 Config로 관리 할 필요 없습니다.

Ex)

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

12. Editor

Hellgate Framework에서는 여러 가지의 Editor가 있습니다. 경로는 Unity Editor/Window/Hellgate 입니다.

12.1. Build AssetBundles

AssetBundle을 실제로 만들어 내는 Editor로써 경로는 Unity Editor/Window/Hellgate/Build AssetBundles입니다. Asset Labels에 설정 후 [그림 5]와 같이 Output folder를 지정하면, AssetBundle이 만들어 지게 됩니다. Editor의 Build Settings Platform에 따라 Target이 설정되며 PC/Android/iOS만 지원합니다.

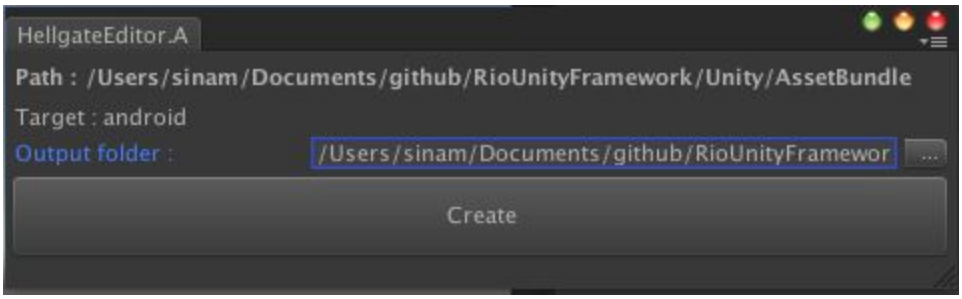

그림 5.

12.2. Json Converter for Excel

Excel 파일을 이용하여 Json을 만들어 내는 Editor입니다. Master data를 만들때 유용하며 경로는 Unity Editor/Window/Hellgate/Json Converter for Excel입니다. 실행 화면은 [그림 6]과 같으며 Excel file 경로와 Output folder 경로만 설정 하면 Json이 만들어 집니다. 그리고 [그림 7]은 Excel Sheet 양식입니다. Sheet에 Ignore 설정도 가능하니 활용 가능합니다.

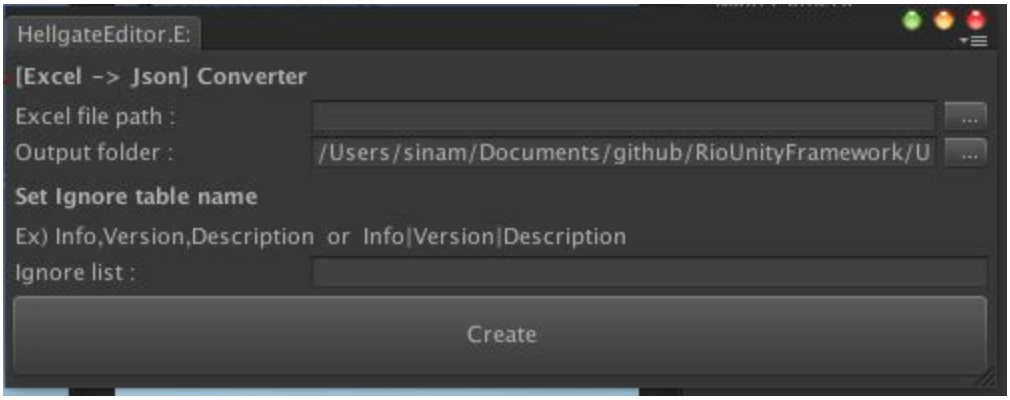

그림 6.

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

그림 7.

12.3. Json Converter for Sqlite DB

Sqlite DB 파일을 이용하여 Json을 만들어 내는 Editor입니다. Master data를 만들때 유용하며 경로는 Unity Editor/Window/Hellgate/Json Converter for Sqlite DB입니다. 실행 화면은 [그림 8]과 같으며 Sqlite DB file 경로와 Output folder 경로만 설정 하면 Json이 만들어 집니다. 그리고 [그림 9]은 Table 작성 양식입니다. Table에 Ignore 설정도 가능하니 활용 가능합니다.

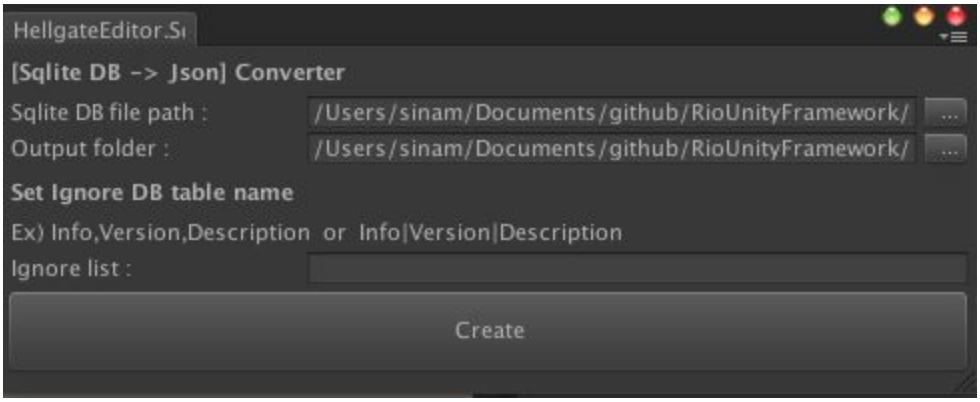

그림 8.

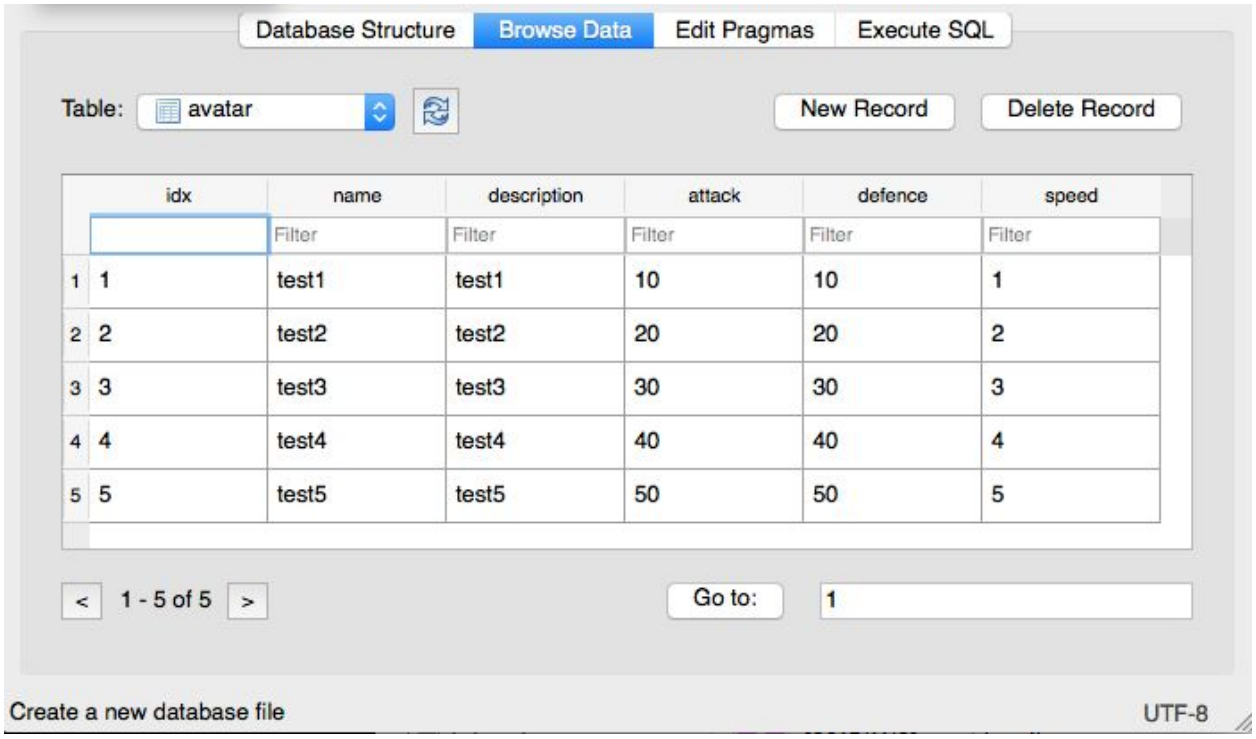

그림 9.

12.3. Create Sqlite DB

Table Auto Generated Attribute가 설정된 Class 읽어 Sqlite DB를
Assets/StreamingAssets에 만들어 주는 Editor입니다.(참고 [그림 10], [그림 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;
}
```

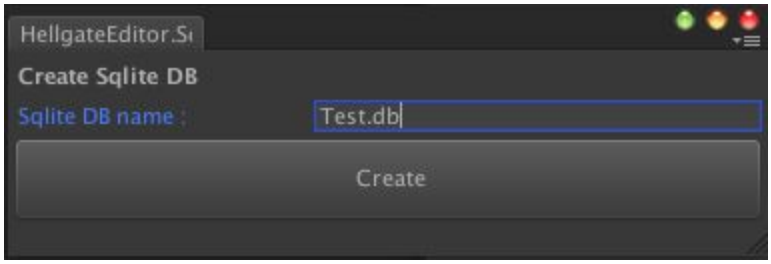

그림 10.

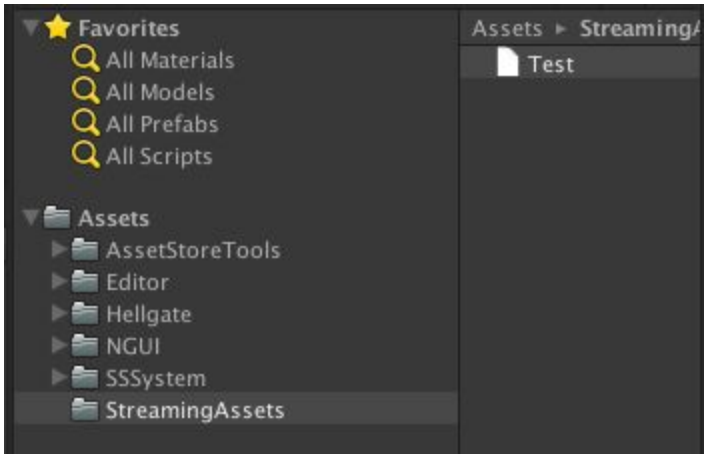
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
AssetStoreTools
Editor
Hellgate
NGUI
SSSystem
StreamingAssets
Assets ▸ Streaming
Test

그림 11.