uFrame API Documentation

0.9

Generated by Doxygen 1.8.5

Thu Jan 16 2014 22:05:45

Contents

Chapter 1

Overview

uFrame Features Overview

`uFrame` is a game development framework for `Unity 3D` developed by professionals with decades of experience. It is based on a flavor of the common `MVVM` (`Model View ViewModel`) framework. A controller has also been added for an extra level of control giving it the name we like to call "MVCVM".

> **Note**: uFrame MVCVM is not a replacement of Unity's component model. uFrame is designed to work directly with it.

Quick To Get Started

Follow the Getting Started Tutorial and get started on you're game in minutes. You'll be amazed at how quickly you can get a started.

Game Series Tutorials walks you through creating a minimal game and demonstrating the power of using uFrame. You'll be instilled with the knowledge to develop you're next title no matter the complexity.

Examples + Checkers Game Included

All code and assets are included to help you learn the ins and outs of creating a game with uFrame.

Unified design pattern for the game and UI.

[u]Frame isn't just designed for UI but the actual game as well. This creates a unified code base that is easy to use, understand and work together.

Scene Management

- **Loading Screen** Comes with the basic installation of uFrame and works directly with the GameManager when switching scenes. Doesn't required any additional setup. Look at the SceneSwitching example in the "Examples" folder.
- Helper methods to load scenes so they can be used similarly to prefabs for additional nesting.
- Dynamically load multiple scenes and [u]Frame will piece them together with a single line of code

MVVM Architecture

Commonly known from Micrsoft WPF, MVVM design is extremly powerful. If you arn't already familiar with MVVM there are loads of resources on the net to learn more.

Added Controllers

A perfect place to connect to you're web services, GameCenter, or even facebook. Controllers also allow View-Models to be decorated creating a flexible and maintainable structure to any element of a game. Decorate View-Models wire them up and make you're game come to life without touching any import visual aspects to your game.

Strong Binding Support

In [u]Frame MVCVM Bindings make it easy to wire just about any event that can occur in you're game. With binding helpers that support chaining views become straight forward and really easy to understand.

Bind in code or with a uFrame Binding Component**

Editor Integration for a rapid workflow

- Generate Editable Templates directly from the Project Folder Menu to quickly add required elements to you're game.
- Generate View Wizard will automagically create a View, ViewModel, Prefab, and throw it into the current scene.

Dependency Injection

A light, and unity friendly version of dependency injection has been implemented to work directly with [u]Frame.

Team Friendly

- A consistent framework for every developer to adhere to.
- With bindings and scene management a Designer can work in one scene and a developer in another. A single line of code can pull both worlds together.

Develop in half the time and half the cost.

When creating a game it can often be difficult to get started. A lot of time can be spent on how things connect together, especially if your using components from different authors. When you are an indie company time is money and being distracted is the last thing any startup needs. Developing your own framework only takes time away from the point (GETTING YOUR GAME OUT).

Mobile Compatible

- Tested and designed to run on any mobile device. Have the confidence of portability and stop worrying about what works and what doesn't on mobile and spend more time developing.

Unity or Unity Pro

- [u]Frame supports both Unity and Unity Pro.

Community Powered

Have ideas or contributions for [u]Frame? Let us know about them. We will work hard to incorporate as many user contributions and other ideas as possible.

Chapter 2

Understanding uFrame

This section breaks down the `uFrame` design patterns and concepts that are critical to know when setting up a game. Click the links to learn more.

Using the GameManager - The GameManager will track and maintain Game instances in the current scene and will persist from scene to scene.

Creating A Game Class - An entry point that should setup and load a scene. Only one can be active at a time.

Creating Controllers -Provides an additional layer for decorating ViewModels.

Creating Views - A visual representation of a ViewModel through bindings.

Creating ViewModels - A logical representation of an entity or View that provides Properties and Commands that can be observed or bound to.

2.1 Using the GameManager

The GameManager is a singleton that will not be destroyed when switching from level to level. The GameManager also manages all of the Games in the current scene, and allows switching active context to other games that exist in other scenes. Only one GameManager should exist per scene as a GameObject with a GameManager component attached to it. Adding the GameManger to a scene can easily be done by clicking the `Create Manager` from the `Unity->uFrame` menu.

2.1.1 Accessing Game Instances

Accessing The Active Game

```
var currentGameInstance = GameManager.ActiveGame;
Debug.Log("The current active game is " + currentGameInstance.GetType().Name);
```

Accessing The All Game Instances

```
foreach (var game in GameManager.Instance.Games) {
    Debug.Log(game.GetType().Name);
}
```

2.1.2 Switching Games

Switching to a another Game in the same level

The active Game can be swapped out by invoking the static method `SwitchGame` on GameManager.

Note: Call SwitchGame with a game that already exists or create a new one by passing null to the second parameter.

```
GameManager.SwitchGame<MyGameType>(
     // Setup the controller here
    (game)=>{ SETUP_GAME_HERE },
    // If this parameter is null then it will create the controller
    null
 );
```

Switching to a another Game in a different level

It is possible to load a Game from another scene simply by knowing the `Level Name` and the `Type` of the Game. In the example below, the GameManager will load another scene and load the active game based off of the `Type` passed into the `SwitchGameAndLevel` method. If it does not find the game in the scene specified it will throw an error. Here is an example:

Note: Seperate different sections of a game into different levels by passing each additional scene as a parameter to the `SwitchGameAndLevel` method.

```
GameManager.SwitchGameAndLevel<MyGameType>(
    (game)=>{ SETUP_GAME_HERE }, // A lambda to setup the controller before it is loaded. Argument can be
       null.
    // The first parameter is required to load the level containing the controller but you
    // can add as many additional scenes as you want.
    "MyMainGameLevel", // The level containing the controller
    "MyMainGameUI" // Load this level as well
);
```

2.2 Creating A Game Class

/// The game entry point. The ViewModel container is now available so register any singleton viewmodels. ///

A game instance is responsible for controlling the current scene. It's responsibilities should primarily be the following.

- Register Mappings for Dependency Injection.
- Registering instances of ViewModels.
- Registering instances of controllers (If being dynamically created. Controllers in the scene will do this automatically)
- Creating any views that don't already exist in the scene.

A custom game can easily be implemented by creating a class and adding it to a game object in the scene via the `Unity Component` menu and keeping it at the root level of the scene. Here is an hypothetical example of a basic Game Class.

```
public class MyGame : Game
{
    public CaptureTheFlagGameController _CTFController;
    public HUDController _HudController;
    public InventoryController _InventoryController;

    public override void Setup()
    {
        base.Setup();

        // Initialize long running models here
        Container.RegisterInstance(CreateController<CaptureTheFlagGameController>());
        Container.RegisterInstance(CreateController<HUDController>());
        Container.RegisterInstance(CreateController<InventoryController>());

        Container.RegisterInstance(_InventoryController.Create());

        Container.RegisterInstance(_HudController.Create());

        Container.RegisterInstance(_CTFController.CreatePlayer());
```

```
        // MyGameViewModel has properties with inject attributes that will
        // automatically be set to references above.
        // The create method creates a MyGameViewModel
        Container.RegisterInstance<MyGameViewModel>(_CTFController.Create());
    }

    // A coroutine to load the scene and directly works with the loading screen via the
       `UpdateProgressDelegate`
    public override IEnumerator Load(UpdateProgressDelegate progress)
    {
        // Load the surroundings from a seperate scene stored in the "__ROOT__"
        // gameobject at the root of the scene.
        yield return LoadAdditive("__ROOT__", "MyAwesomeCaptureTheFlagMap", (root) =>
        {
            // Get our level view from the root object returned
            LevelView = root.GetComponent<MyLevelView>();
        });
    }

    // After this game has been loaded
    public override void OnLoaded()
    {
        base.OnLoaded();
        // Perform any logic after everything has been loaded.
    }
}
```

2.2.1 Calling a Game Event from a Controller

To call an event on a Game use the `GameEvent` method from a controller.

For Example:

```
// .... PlayerController.cs
public void PlayerDied() {
    // The reload method exists in the Game base class.  But, you can call any method
    // that exists on the current active Game.
    GameEvent("Reload");
}
// ....
```

See: Using the GameManager , Controller

2.3 Creating Controllers

Controllers are a new addition to MVVM that fits nicely in the uFrame game development paradigm and makes elements in a game easier to maintain/extend. In order to create a Controller in `uFrame`, derive a class from the `uFrame` `Controller` class. You can always create a controller more quickly by right clicking on an `Element` folder and clicking `New Controller`.

By practice keeping a basic set of rules when using a controller should be considered. (While not required it is recommended) A controller should never have a reference to a view. A controller is responsible for creating a view model and wiring up appropriate subscriptions. A controller doesn't have to contain all `ViewModel` logic and should be used only when needed.

Example** For a more concrete example of using a controller examine the following:

- The Game class registers `MyViewModel` via it's `Setup` method by invoking a create method on the `MyController` controller.

```
// MyGame.cs ---
// Set in Unity's Inspector
public MyElementController _MyElementController;
public override void Setup()
{
    base.Setup();
    Container.RegisterInstance<MyViewModel>(_MyElementController.Create());
}
```

- The create method on the controller subscribes to an `ICommand` called `Hit` and invokes a Game event called "GameOver".

```
// MyController.cs ---
public MyViewModel Create()
{
    var myViewModel = new MyViewModel();
    this.SubscribeToCommand( myViewModel.Hit, ()=>{
        GameEvent("GameOver");
    });
    return myViewModel;
}
```

- The view `MyView` uses `MyViewModel` and has a collision binding that binds to an `ICommand` property called `Hit` on the ViewModel.

```
public class MyElementView : View<PlayerModel>
{
    // This method will bind the model to the view.
    public override void Bind()
    {
        this.BindCollision(()=>Model.Hit,CollisionEventType.OnCollisionEnter)
            .When(p=>p.tag == "Enemy")
            .Subscribe(()=>Debug.Log("Enemy hit player")
            ;
    }
}
```

- The view has "GameContainer" selected on the 'View Model From' option in the Views inspector. This will pull the ViewModel from the GameContainer registered in the Game.
- A collision happens on the view and triggers the ICommand
- The controller now receives this event (as shown above) and can update the other `Elements` in the game as necessary (In this example it simply invokes "GameOver" on `MyGame`)

2.4 Creating Views

A View is a visual representation of a ViewModel. For example: A UI dialog, Player, Weapon, etc...

In `uFrame` a View is a class derived from 'View', attached as a component to Prefabs or GameObjects in a scene.

- A prefab of the View should be stored under a "Resources" folder for default path resolution in `uFrame`.

You can easily generate a new View with a corresponding ViewModel using the Create View dialog. Access the dialog by right clicking on an `Element` folder and clicking on `New View` Fill out the options in the dialog.

2.4.1 View Life cycle

1. **Awake()**
   - CreateModel() `Instantiate the ViewModel with no initialization and only for referencing`
2. **OnEnable()**
   - Tell any parent view about this view adding to it's `ChildView` collection.
3. **Start()** or `Model` Property is set and already bound.
   - SetupBindings() `Root Views First ascending down the hierarchy`
     - Unbind() `If already bound`
     - **InitializeModel()** `Load any inspector properties into the ViewModel here.`

  - **Bind()** will add to the Bindings Collection
  - Invoke Bind() on each binding
  - SetupBindings() on any childview of this view

4. **LateUpdate()**
   - **ReverseBind()** on all two way bindings

2.4.2 Views in the scene

A View, when used as a component on a `Prefab`, can always be dragged into a scene and placed under a Controller without having to dynamically instantiate the View. When dragging a View into the scene, will use the ViewModel type specified in the TypeParameter of view. If a view is in the scene you can choose a subscription type

2.4.3 RegisterTypes

- **None** = The view model will be created automatically for each instance
- **Subscribe** = Subscribe to the view model type that is in the current `GameContainer` initialized in the current `Game`

2.4.4 Insantiating Views

Instantiating a View is as simple as calling one of the InstantiateView overload methods inside of a Game or View.

For Example:

```
InstantiateView(new WeaponModel() { _Ammo = 20 });
```

When instantiating a View, it will automatically search for the View prefab based off of the type name without 'Model'.

For instance `WeaponModel` would find a prefab named `Weapon` inside of a resources folder.

A View Prefab can also be instantiated by passing in the prefab name or prefab game object manually to the `InstantieView` method.

```
InstantiateView("M16", new WeaponModel() { _Ammo = 20 });
InstantiateView(_M16PrefabObject, new WeaponModel() { _Ammo = 20 });
```

> Note: In the example above "M16" could be passed as a relative path like "Weapons\M16".

To extend the View searching functionality of `uFrame`, see: ExtendinguFrame

2.4.5 Bindings

View Bindings are what connect a `ViewModel` and a `View` together. In uFrame a binding can be added via a binding component or directly in code by overriding the **Bind()** method of a `View`

Each binding extension method (like below) supports chaining with various methods that further customize that type of binding.

Example

```
public override void Bind()
{
    // Add bindings here or add binding components in unity.
    this.BindProperty(() => Model._State, state => gameObject.SetActive(state != MyElementState.Dead));

    this.BindProperty(() => Model._Color, color => _GraphicsRenderer.material.SetColor("_Color", color));
```

```
    this.BindCollision(CollisionEventType.OnTriggerEnter, () => Model.
      EnteredZone)
        .When(go => go.tag == "RedZone")
        .Subscribe(() => Debug.Log("RedZone Entered"))
        ;

    this.BindCollision(CollisionEventType.OnTriggerExit, () => Model.
      ExitedZone)
       .When(go => go.tag == "RedZone")
        ;
}
```

2.4.6 Property Bindings

Property Bindings bind a ViewModel Property to any given target(s).

Example of a property binding.

```
// A One-Way binding
this.BindProperty(() => Model.IsAlive, value => gameObject.SetActive(value));
// A Two-way binding
this.BindProperty(() => Model.IsAlive, value => gameObject.SetActive(value), ()=>gameObject.active);
```

2.4.7 View Collection Bindings

Collection Bindings bind a ModelCollection<T> Property on a `ViewModel` to any given View Collection.

```
// Simple binding (see comments below for default implementation info)
this.BindToViewCollection(() => Model._Cubes, _Cubes);
// Complex binding
this.BindToViewCollection(
    () => Model._Cubes, _Cubes) // Cubes can be null

    // Default adds to _Cubes if not set
    .SetAddHandler((obj) => Debug.Log("View was added"))
    // Default removes from _Cubes if not set
    .SetRemoveHandler((obj) => Debug.Log("View was removed"))
    // Default uses THIS
    .SetParent(_customParentGameObject)
    // Default if not set
    .SetCreateHandler((binding,model)=>InstantiateView(binding.ViewName,model))
    // Binds the collection immediately if true. Defaults to true if not set
    .Immediate(true)
    // The View that will be used when instantiating the ViewModel
    .SetView("BasicPlate")
    ;
```

2.4.8 Collision/Trigger Bindings

Collision Bindings bind a ViewModel Command to a collision/trigger event.

```
this.BindCollision(() => Model.LightsOnCommand, CollisionEventType.OnTriggerEnter)
    // When() used to only invoke the command when the expression is true
    .When((c) => c.tag == "LightsTrigger")
    // Subscribe() can be used on any Command Binding
    .Subscribe(()=>Debug.Log("Lights are on"))
    ;
this.BindCollision(() => Model.LightsOffCommand, CollisionEventType.OnTriggerExit)
    .When((c) => c.tag == "LightsTrigger")
    .Subscribe(()=>Debug.Log("Lights are off"))
    ;
```

Chain Methods:

- When(Action<`GameObject`>) : used to filter the collision.
- Subscribe : Subscribe to this binding and will be invoked only when the binding is invoked.

2.4.9 Event Bindings

Event bindings make it easy to have binding functionality when a custom event occurs. For Example

```
private IEnumerator Wait() {
    yield return new WaitForSeconds(2.0f);
    Event("MY_CUSTOM_EVENT");
}
public override void Bind() {
    this.BindEvent(()=>Model.LightsOncommand,"MY_CUSTOM_EVENT");
}
```

Note: These can be used also when implementing some sort of input like a swipe, touch..etc

2.4.10 Key Event Bindings

Key Event Bindings bind a key to a `ViewModel` command.

```
this.BindKey( () => Model.LightsOnCommand, Keys.L )
    .On(KeyBindingEventType.KeyUp)
    .Subscribe(()=>Debug.Log("L key was pressed"))
    ;
```

2.4.11 Mouse Event Bindings

Mouse Event Bindings bind a mouse interaction to a `ViewModel` command.

```
this.BindMouseEvent( () => Model.LightsOnCommand, MouseEventType.OnMouseDown )
    .Subscribe(()=>Debug.Log("Mouse was clicked"))
    ;
```

2.5 Creating ViewModels

In `uFrame` a ViewModel is a class that holds state for a View. In order to create a `ViewModel` in `uFrame`, derive a class from ViewModel.

2.5.1 Model Properties

`P<T>` is a special class that wraps a type into a property that can be bound, and signals events when changed.

To easily create ViewModel Properties see Installing ViewModel Code Snippets

Example of a ViewModel

```
public class MyElementViewModel : ViewModel
{
    public readonly ModelCollection<int> _CollectionProperty = new ModelCollection<int>();
    public IEnumerable<int> Collection
    {
        get { return _CollectionProperty.Value; }
        set { _CollectionProperty.Value = value.ToList(); }
    }

    public readonly P<Color> _ColorProperty = new P<Color>(Color.black);
    public Color Color
    {
        get { return _Color.Value; }
        set { _Color.Value = value; }
    }

    public ICommand BindableCommand
    {
        get;
        set;
    }
```

```
    public MyElementViewModel() {
        BindableCommand = new Command(()=>Color=Color.black);
    }
}
```

2.5.2 ViewModel Commands

All View Model Commands

2.5.3 Binding Performance on IOS

On iOS, dynamically compiling expression trees is not supported. The default implementation for P<T> is using reflection to grab public P<T> fields from the derived ViewModel (It's only invoked once).

> Note: On PC & Other platforms that allow JIT, uFrame will compile an expression tree to access the field directly. Performance is not an issue with these platforms.

Chapter 3

Quick Start Tutorial

3.1 Creating a project.

Step 1. Create A New Project or load an existing project. Step 2. Install [u]Frame via the Asset Store or unitypackage.

Step 3. In an empty scene Click "[u]Frame" on the Unity menu and then click "Create Manager". This will add the uFrame GameManager to the scene.

Now that the scene is setup to use [u]Frame a **Game** is needed to pull all the "∗∗Elements∗∗" together (more on this later). The **Game** will govern this scene at a high level. To create a **Game** do the following.

> Note: You can always create a **Game** manually or you can generate one more quickly by doing the following.

Step 4. From the project window create a folder called "Elements" on the root level of your assets folder.

Step 5. Right-click on the newly created "Elements Folder" and click New Game". For this tutorial just call it "MyGame"

Step 6. Now right-click on the same "Elements" folder you created in step 4 and click New Element Structure". Call this "MyElement".

Adding a new element structure creates a folder structure for an element. Doing this will automagically generate a structure like the following.

Step 7. Now click on "_GameManager in the scene". In the inspector click on the plus next to Games and you will see "My" in the list. Click add to add it to the scene. The game should always be on the root level of a scene.

> Note: If "My" doesn't show up in the Games list make sure there are no compile errors.

Step 8. Now In the same inspector specify the default loading level that comes with uFrame ("Loading").

Step 9. As a precaution double check that a few scripts are in the correct execution order.

Step 7. Now Load up "MyGame" in you're favorite editor and you.

> We highly recommend Visual Studio 2012 with UnityVS plugin at unitvs.com. If you can afford JetBrains resharper it will move things along faster.

Once loaded take a second to look at the file generated and the comments provided to familiarize yourself with the overrides. If you want to know more about methods that can be overridden please consult the documentation.

The main method to consult in MyGame is the `Load` method. The GameManager will start this coroutine when the scene loads. You can invoke the `UpdateProgressDelegate` that is passed to this method as the first parameter (passing a value 0.0f to 1.0f and a message) in order to let the loading screen know how much progress has been completed.

Note: [u]Frame comes with a default loading screen ("Loading.unity") to make it easier to get up and running. The loading scene works directly with uFrame when Switching to a different Game in another scene.

3.2 Installing ViewModel Code Snippets

Using the snippets provided with uFrame, it can really increase productivity when creating ViewModels. In Visual Studio open the code snippets manager. Click add and navigate to the uFrame directory in the project and select the "Snippets" folder. For a view model property type "vmp" and press tab. For a view model collection type "vmc" and press tab.

Chapter 4

Serialization

uFrame uses a extended version called SimpleJSON that is available on the UnityWiki. Click the link below to learn more about this simple library.

http://wiki.unity3d.com/index.php/SimpleJSON

4.1 ViewModel Serialization

Any ViewModel in uFrame can serialized and deserialized directly. In fact the ToString() method of ViewModel invokes the deserialize method of ViewModel providing a json representation.

To Serialize a ViewModel simply just invoke the ToString method of a ViewModel. [u]Frame directly support json serialization.

```
var jsonString = new MyViewModel() { _MyCustomValue = "HELLO WORLD" }.ToString();
```

Deserializing works in a similar manner, just invoke the deserialize method of ViewModel with a string of json.

```
var copy = new MyViewModel();
copy.Deserialize(JSON.parse(jsonString));
```

`ViewModels` have built in support for serializing all `P` fields. It is possible to be more specific with serializing and deserializing by overriding the `Serialize` and `Deserialize` methods of `ViewModel`. For Example:

```
public class MyViewModel : ViewModel {
    public string _MyCustomValue;
    public override JSONNode Serialize()
    {
        // Keep all of the serialization that ViewModel Provides
        var node = base.Serialize();
        // Add additional info to the serialization
        node.Add("_MyCustomValue", this._MyCustomValue);
        return node;
    }

    public virtual void Deserialize(JSONNode node)
    {
        // Keep all of the de-serialization that ViewModel Provides
        base.Deserialize(node);
        this._MyCustomValue = node["_MyCustomValue"];
    }
}
```

Chapter 5

Namespace Index

5.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

A view class which attaches as a component directly to a game object. The responsibility of this view is to bind a data model 'TModel' to the game object 103

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

Chapter 9

Namespace Documentation

9.1 Package SimpleJSON

Classes

- class **JSON**
- class JSONArray
- class JSONClass
- class JSONData
- class JSONLazyCreator
- class JSONNode

Enumerations

- enum JSONBinaryTag {
  JSONBinaryTag.Array = 1, JSONBinaryTag.Class = 2, JSONBinaryTag.Value = 3, JSONBinaryTag.IntValue = 4,
  JSONBinaryTag.DoubleValue = 5, JSONBinaryTag.BoolValue = 6, JSONBinaryTag.FloatValue = 7 }

9.1.1 Enumeration Type Documentation

9.1.1.1 enum SimpleJSON.JSONBinaryTag

Enumerator

Array

Class

Value

IntValue

DoubleValue

BoolValue

FloatValue

Chapter 10

Class Documentation

10.1 BindableProperty Class Reference

A bindable property that can be easily wired for binding.

Public Member Functions

- BindableProperty (object bindableObject, MemberInfo bindableMember)

Properties

- MemberInfo BindableMember `[get, set]`
- object BindableObject `[get, set]`
- Func< object > GetDelegate `[get]`
- object Value `[get, set]`

10.1.1 Detailed Description

A bindable property that can be easily wired for binding.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 BindableProperty.BindableProperty (object *bindableObject,* MemberInfo *bindableMember*)

10.1.3 Property Documentation

10.1.3.1 MemberInfo BindableProperty.BindableMember `[get],[set]`

10.1.3.2 object BindableProperty.BindableObject `[get],[set]`

10.1.3.3 Func<object> BindableProperty.GetDelegate `[get]`

10.1.3.4 object BindableProperty.Value `[get],[set]`

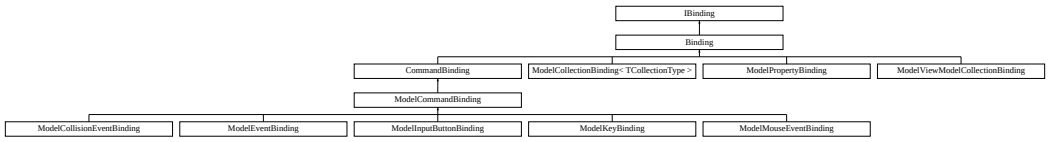
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/BindableProperty.cs

10.2 Binding Class Reference

The base class for all bindings.

Inheritance diagram for Binding:



Public Member Functions

- virtual void Bind ()

  Set-up the binding. This should almost always be implemented in a deriving class.
- virtual void Unbind ()

  Unbind this binding

Protected Member Functions

- Binding ()
- Binding (ViewBase sourceView, string modelMemberName)

  Constructor

Properties

- bool CanTwoWayBind `[get]`

  Does this instance type implement ITwoWayBinding?
- Func< object > GetTargetValueDelegate `[get, set]`

  A delegate for Getting the target value and is required for a two-way binding.
- bool IsBound `[get, set]`
- bool IsComponent `[get, set]`

  Was this loaded from a component in the Unity Inspector?
- string ModelMemberName `[get, set]`

  The source ViewModel member name that is being bound to.
- ModelPropertyBase ModelProperty `[get, set]`

  The Model Property that is being bound to. Will call the ModelPropertySelector if null.
- Func< ModelPropertyBase > ModelPropertySelector `[get, set]`

  A selector that will select the model property. This should be set manually if reflection shouldn't be used.
- Action< object > SetTargetValueDelegate `[get, set]`

  A delegate to set the value of the target member(s).
- IViewModelObserver Source `[get, set]`

  The owner view that this Binding belongs to
- object SourceValue `[get]`

  The value of the ViewModel Member
- ViewBase SourceView `[get]`
- bool TwoWay `[get, set]`

  Is this a two-way binding.

10.2.1 Detailed Description

The base class for all bindings.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 Binding.Binding ( ) [protected]

10.2.2.2 Binding.Binding (ViewBase *sourceView*, string *modelMemberName*) [protected]

Constructor

Parameters

sourceView	The View that will own this binding.
modelMember-Name	The member of the ViewModel.

10.2.3 Member Function Documentation

10.2.3.1 virtual void Binding.Bind ( ) [virtual]

Set-up the binding. This should almost always be implemented in a deriving class.

Implements IBinding.

Reimplemented in ModelViewModelCollectionBinding, CommandBinding, ModelCollectionBinding< TCollection-Type >, ModelEventBinding, ModelCommandBinding, and ModelPropertyBinding.

10.2.3.2 virtual void Binding.Unbind ( ) [virtual]

Unbind this binding

Implements IBinding.

Reimplemented in ModelViewModelCollectionBinding, CommandBinding, ModelCollectionBinding< TCollection-Type >, ModelPropertyBinding, ModelEventBinding, and ModelCommandBinding.

10.2.4 Property Documentation

10.2.4.1 bool Binding.CanTwoWayBind [get]

Does this instance type implement ITwoWayBinding?

10.2.4.2 Func<object> Binding.GetTargetValueDelegate [get], [set]

A delegate for Getting the target value and is required for a two-way binding.

10.2.4.3 bool Binding.IsBound [get], [set]

10.2.4.4 bool Binding.IsComponent [get], [set]

Was this loaded from a component in the Unity Inspector?

10.2.4.5 string Binding.ModelMemberName [get], [set]

The source ViewModel member name that is being bound to.

10.2.4.6 ModelPropertyBase Binding.ModelProperty [get], [set]

The Model Property that is being bound to. Will call the ModelPropertySelector if null.

10.2.4.7 Func<ModelPropertyBase> Binding.ModelPropertySelector [get], [set]

A selector that will select the model property. This should be set manually if reflection shouldn't be used.

10.2.4.8 Action<object> Binding.SetTargetValueDelegate [get], [set]

A delegate to set the value of the target member(s).

10.2.4.9 IViewModelObserver Binding.Source [get], [set]

The owner view that this Binding belongs to

10.2.4.10 object Binding.SourceValue [get]

The value of the ViewModel Member

10.2.4.11 ViewBase Binding.SourceView [get]

10.2.4.12 bool Binding.TwoWay [get], [set]

Is this a two-way binding.

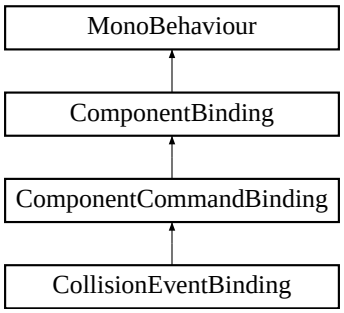
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/Binding.cs

10.3 CollisionEventBinding Class Reference

A component for binding to a collision.

Inheritance diagram for CollisionEventBinding:



Public Attributes

- CollisionEventType _CollisionEvent

Protected Member Functions

- override IBinding GetBinding ()

  The binding provider. Create the binding that the component will add to the source view here.
- virtual void OnCollisionEnter (Collision collision)
- virtual void OnCollisionExit (Collision collision)
- virtual void OnCollisionStay (Collision collision)
- virtual void OnTriggerEnter (Collider other)
- virtual void OnTriggerExit (Collider other)
- virtual void OnTriggerStay (Collider other)

Additional Inherited Members

10.3.1 Detailed Description

A component for binding to a collision.

10.3.2 Member Function Documentation

10.3.2.1 override IBinding CollisionEventBinding.GetBinding ( ) [protected], [virtual]

The binding provider. Create the binding that the component will add to the source view here.

Returns

The binding that will be added to the source view.

Implements ComponentBinding.

10.3.2.2 virtual void CollisionEventBinding.OnCollisionEnter (Collision *collision*) [protected], [virtual]

10.3.2.3 virtual void CollisionEventBinding.OnCollisionExit (Collision *collision*) [protected], [virtual]

10.3.2.4 virtual void CollisionEventBinding.OnCollisionStay (Collision *collision*) [protected], [virtual]

10.3.2.5 virtual void CollisionEventBinding.OnTriggerEnter (Collider *other*) [protected], [virtual]

10.3.2.6 virtual void CollisionEventBinding.OnTriggerExit (Collider *other*) [protected], [virtual]

10.3.2.7 virtual void CollisionEventBinding.OnTriggerStay (Collider *other*) [protected], [virtual]

10.3.3 Member Data Documentation

10.3.3.1 CollisionEventType CollisionEventBinding._CollisionEvent

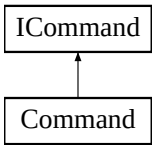
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/CollisionEventBinding.cs

10.4 Command Class Reference

A ViewModel command that can be executed. IEnumerator is always used so that any command can be a coroutine.

Inheritance diagram for Command:



Public Member Functions

- Command (Action @delegate)
- IEnumerator Execute ()

Protected Member Functions

- virtual void OnOnCommandComplete ()
- virtual void OnOnCommandExecuting ()

Properties

- Action Delegate `[get, set]`

Events

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

10.4.1 Detailed Description

A ViewModel command that can be executed. IEnumerator is always used so that any command can be a coroutine.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 Command.Command (Action @ *delegate*)

10.4.3 Member Function Documentation

10.4.3.1 IEnumerator Command.Execute ( )

Implements ICommand.

10.4.3.2 virtual void Command.OnOnCommandComplete ( ) `[protected]`, `[virtual]`

10.4.3.3 virtual void Command.OnOnCommandExecuting ( ) `[protected]`, `[virtual]`

10.4.4 Property Documentation

10.4.4.1 Action Command.Delegate [get],[set],[protected]

10.4.5 Event Documentation

10.4.5.1 CommandEvent Command.OnCommandExecuted

10.4.5.2 CommandEvent Command.OnCommandExecuting

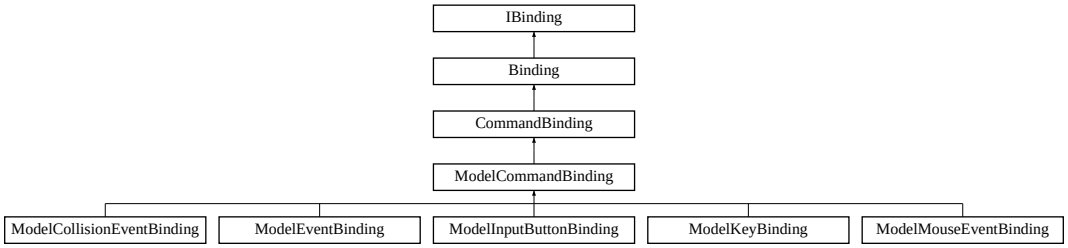
The documentation for this class was generated from the following file:

- Scripts/Base/Commands/Command.cs

10.5 CommandBinding Class Reference

Base class for a command binding. Use this class if a different type of command binding is needed.

Inheritance diagram for CommandBinding:



Public Member Functions

- override void Bind ()

  Set-up the binding. This should almost always be implemented in a deriving class.
- bool CanExecute ()
- void ExecuteCommand ()
- CommandBinding Subscribe (Action execute, bool before=false)
- CommandBinding Throttle (float seconds)
- override void Unbind ()

  Unbind this binding
- CommandBinding When (Func< bool > condition)

Properties

- object Argument [get, set]
- Func< ICommand > CommandDelegate [get, set]
- bool ExecuteBefore [get, set]
- ICommand Command [get, set]
- List< Predicate< object > > Conditions [get, set]

Additional Inherited Members

10.5.1 Detailed Description

Base class for a command binding. Use this class if a different type of command binding is needed.

10.5.2 Member Function Documentation

10.5.2.1 override void CommandBinding.Bind ( ) [virtual]

Set-up the binding. This should almost always be implemented in a deriving class.

Reimplemented from Binding.

Reimplemented in ModelEventBinding, and ModelCommandBinding.

10.5.2.2 bool CommandBinding.CanExecute ( )

10.5.2.3 void CommandBinding.ExecuteCommand ( )

10.5.2.4 CommandBinding CommandBinding.Subscribe (Action *execute*, bool *before* = false)

10.5.2.5 CommandBinding CommandBinding.Throttle (float *seconds*)

10.5.2.6 override void CommandBinding.Unbind ( ) [virtual]

Unbind this binding

Reimplemented from Binding.

Reimplemented in ModelEventBinding, and ModelCommandBinding.

10.5.2.7 CommandBinding CommandBinding.When (Func< bool > *condition*)

10.5.3 Property Documentation

10.5.3.1 object CommandBinding.Argument [get], [set]

10.5.3.2 ICommand CommandBinding.Command [get], [set], [protected]

10.5.3.3 Func<ICommand> CommandBinding.CommandDelegate [get], [set]

10.5.3.4 List<Predicate<object> > CommandBinding.Conditions [get], [set], [protected]

10.5.3.5 bool CommandBinding.ExecuteBefore [get], [set]

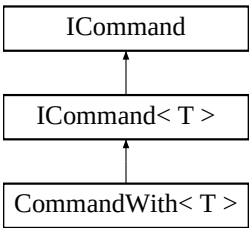
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/CommandBinding.cs

10.6 CommandWith< T > Class Template Reference

A command with an argument of type T. Not usually bound to directly but used to forward a command to a parent viewmodel

Inheritance diagram for CommandWith< T >:



Public Member Functions

- CommandWith (Action< T > @delegate)
- CommandWith (T parameter, Action< T > @delegate)
- virtual IEnumerator Execute ()

Protected Member Functions

- virtual void OnOnCommandComplete ()
- virtual void OnOnCommandExecuting ()

Properties

- T Parameter `[get, set]`
- Action< T > Delegate `[get, set]`

Events

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

10.6.1 Detailed Description

A command with an argument of type T. Not usually bound to directly but used to forward a command to a parent viewmodel

Template Parameters

T	The argument parameter.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 CommandWith< T >.CommandWith (Action< T > @ *delegate*)

10.6.2.2 CommandWith< T >.CommandWith (T *parameter,* Action< T > @ *delegate*)

10.6.3 Member Function Documentation

10.6.3.1 virtual IEnumerator CommandWith< T >.Execute ( ) `[virtual]`

Implements ICommand.

10.6.3.2 virtual void CommandWith< T >.OnOnCommandComplete ( ) [protected], [virtual]

10.6.3.3 virtual void CommandWith< T >.OnOnCommandExecuting ( ) [protected], [virtual]

10.6.4 Property Documentation

10.6.4.1 Action<T> CommandWith< T >.Delegate [get], [set], [protected]

10.6.4.2 T CommandWith< T >.Parameter [get], [set]

10.6.5 Event Documentation

10.6.5.1 CommandEvent CommandWith< T >.OnCommandExecuted

10.6.5.2 CommandEvent CommandWith< T >.OnCommandExecuting

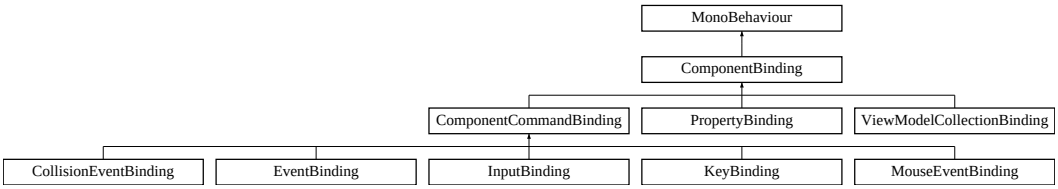
The documentation for this class was generated from the following file:

- Scripts/Base/Commands/CommandWith.cs

10.7 ComponentBinding Class Reference

A Unity3d Component that will provide a binding to a specified View

Inheritance diagram for ComponentBinding:



Public Member Functions

- virtual IEnumerable < KeyValuePair< string, ModelPropertyBase > > FilterBindableProperties (Dictionary< string, ModelPropertyBase > modelProperties)

  Override this method to filter the list of properties that are displayed in the Binding Inspector

Public Attributes

- string _ModelMemberName
- ViewModelObserver _SourceView

Protected Member Functions

- virtual void Awake ()
- abstract IBinding GetBinding ()

  The binding provider. Create the binding that the component will add to the source view here.

Properties

- IBinding Binding [get, set]
  The binding that has been created for this component.

10.7.1 Detailed Description

A Unity3d Component that will provide a binding to a specified View

10.7.2 Member Function Documentation

10.7.2.1 virtual void ComponentBinding.Awake ( ) [protected],[virtual]

Reimplemented in EventBinding.

10.7.2.2 virtual IEnumerable<KeyValuePair<string, ModelPropertyBase> > ComponentBinding.FilterBindableProperties (Dictionary< string, ModelPropertyBase > *modelProperties*) [virtual]

Override this method to filter the list of properties that are displayed in the Binding Inspector

Parameters

modelProperties	

Returns

10.7.2.3 abstract IBinding ComponentBinding.GetBinding ( ) [protected],[pure virtual]

The binding provider. Create the binding that the component will add to the source view here.

Returns

The binding that will be added to the source view.

Implemented in MouseEventBinding, InputBinding, PropertyBinding, KeyBinding, CollisionEventBinding, ViewModelCollectionBinding, and EventBinding.

10.7.3 Member Data Documentation

10.7.3.1 string ComponentBinding._ModelMemberName

10.7.3.2 ViewModelObserver ComponentBinding._SourceView

10.7.4 Property Documentation

10.7.4.1 IBinding ComponentBinding.Binding [get],[set]

The binding that has been created for this component.

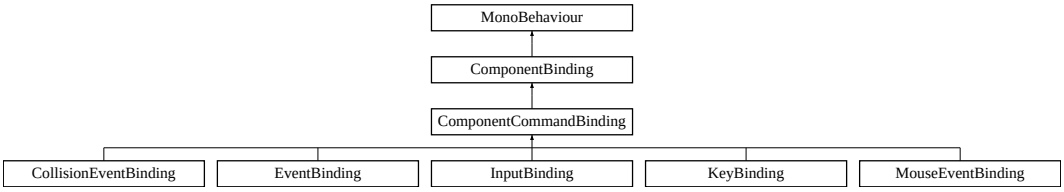
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ComponentBinding.cs

10.8 ComponentCommandBinding Class Reference

A component that will create a command binding and requires a component for the command to work.

Inheritance diagram for ComponentCommandBinding:



Public Attributes

- Component _TargetComponent

Properties

- ModelCommandBinding CommandBinding `[get]`

  Simply a wrapper of "Binding" property cast to ModelCommandBinding
- Component Component `[get, set]`

Additional Inherited Members

10.8.1 Detailed Description

A component that will create a command binding and requires a component for the command to work.

10.8.2 Member Data Documentation

10.8.2.1 Component ComponentCommandBinding._TargetComponent

10.8.3 Property Documentation

10.8.3.1 ModelCommandBinding ComponentCommandBinding.CommandBinding `[get]`

Simply a wrapper of "Binding" property cast to ModelCommandBinding

10.8.3.2 Component ComponentCommandBinding.Component `[get],[set]`

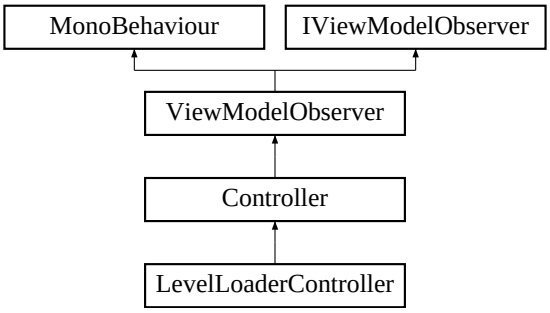
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ComponentCommandBinding.cs

10.9 Controller Class Reference

A controller is a integral part of uFrame and is used for an extra layer connecting services and "Elements" of a game together. A controller also provides the creation of a ViewModel and bind to command to provide additional functionality.

Inheritance diagram for Controller:



Public Member Functions

- virtual void GameEvent (string message, params object[] additionalParamters)

  Send an event to our game
- abstract void Setup (IGameContainer container)
- override void AddBinding (IBinding binding)

Protected Member Functions

- virtual void Awake ()
- void SubscribeToCommand (ICommand command, Action action)
- virtual void OnDestroy ()
- virtual void OnDisable ()
- virtual void OnEnable ()
- virtual void Start ()

Properties

- IGameContainer Container `[get]`
- string ControllerName `[get]`

  The friendly name of the controller. If this' type name ends with controller it will be removed.

10.9.1 Detailed Description

A controller is a integral part of uFrame and is used for an extra layer connecting services and "Elements" of a game together. A controller also provides the creation of a ViewModel and bind to command to provide additional functionality.

10.9.2 Member Function Documentation

10.9.2.1 override void Controller.AddBinding (IBinding *binding*) `[virtual]`

Reimplemented from ViewModelObserver.

10.9.2.2 virtual void Controller.Awake ( ) `[protected]`, `[virtual]`

10.9.2.3 virtual void Controller.GameEvent (string *message*, params object[] *additionalParamters*) `[virtual]`

Send an event to our game

Parameters

message	
additional-Paramters	

10.9.2.4 virtual void Controller.OnDestroy ( ) [protected], [virtual]

10.9.2.5 virtual void Controller.OnDisable ( ) [protected], [virtual]

10.9.2.6 virtual void Controller.OnEnable ( ) [protected], [virtual]

10.9.2.7 abstract void Controller.Setup (IGameContainer *container*) [pure virtual]

Implemented in LevelLoaderController.

10.9.2.8 virtual void Controller.Start ( ) [protected], [virtual]

Reimplemented in LevelLoaderController.

10.9.2.9 void Controller.SubscribeToCommand (ICommand *command*, Action *action*) [protected]

10.9.3 Property Documentation

10.9.3.1 IGameContainer Controller.Container [get]

10.9.3.2 string Controller.ControllerName [get]

The friendly name of the controller. If this' type name ends with controller it will be removed.

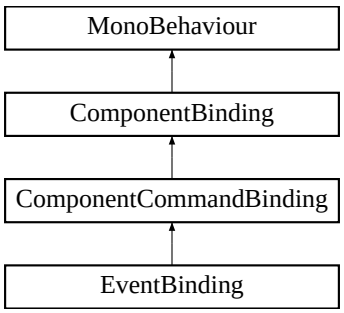
The documentation for this class was generated from the following file:

- Scripts/Base/Controllers/Controller.cs

10.10 EventBinding Class Reference

The event binding component that will add an event binding to a source view.

Inheritance diagram for EventBinding:



Public Attributes

- string _EventName

Protected Member Functions

- override IBinding GetBinding ()

  The binding provider. Create the binding that the component will add to the source view here.

- override void Awake ()

Additional Inherited Members

10.10.1 Detailed Description

The event binding component that will add an event binding to a source view.

10.10.2 Member Function Documentation

10.10.2.1 override void EventBinding.Awake ( ) [protected],[virtual]

Reimplemented from ComponentBinding.

10.10.2.2 override IBinding EventBinding.GetBinding ( ) [protected],[virtual]

The binding provider. Create the binding that the component will add to the source view here.

Returns

The binding that will be added to the source view.

Implements ComponentBinding.

10.10.3 Member Data Documentation

10.10.3.1 string EventBinding._EventName

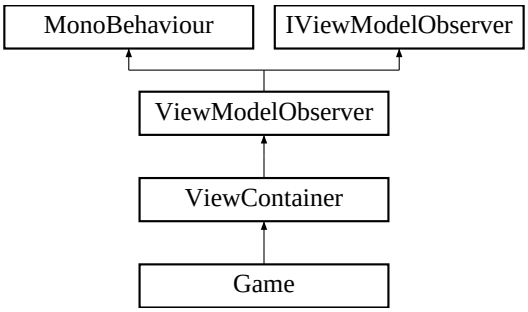
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/EventBinding.cs

10.11 Game Class Reference

The main entry point for a game that is managed and accessible via GameManager. Only one will available at a time. This class when derived form should setup the container and load anything needed to properly run a game. This could include ViewModel Registering in the Container, Instantiating Views, Instantiating or Initializing Controllers.

Inheritance diagram for Game:



Public Member Functions

- T CreateController< T > ()

  Creates a controller (A GameObject with the controller of type T attached to it).
- Controller CreateController (Type t)

  Creates a controller (A GameObject with the controller of type T attached to it).
- void InitializeControllers ()
- virtual IEnumerator Load (UpdateProgressDelegate progress)

  This method should do any set up necessary to load the controller and is invoked when you call GameStateManager.-SwitchGame(). This should call StartCoroutine(Controller.Load) on any regular controller in the scene.
- virtual void OnLoaded ()

  This method is called when this controller has started loading
- virtual void OnLoading ()

  This method is called when the load function has completed
- void RegisterController (Controller controller, bool removeExisting=false)

  Registers a controller based off of its type
- virtual void Reload ()

  This method simply starts the load method as a coroutine and should be overriden to add any reload logic that is necessary
- virtual void Setup ()
- virtual void Unload ()
- void UnregisterController (Controller controller)

  Unregisters the controller.
- void UnregisterController (string controllerName)

  Unregisters the controller.

Protected Member Functions

- Game ()
- virtual void Awake ()
- virtual void OnDestroy ()

Properties

- Controller this[string controllerName] `[get]`

  Gets the Game with the specified controllerName.
- IGameContainer Container `[get, set]`
- Dictionary< string, Controller > Controllers `[get]`

  Gets the controllers that have been registered with this game.

10.11.1 Detailed Description

The main entry point for a game that is managed and accessible via GameManager. Only one will available at a time. This class when derived form should setup the container and load anything needed to properly run a game. This could include ViewModel Registering in the Container, Instantiating Views, Instantiating or Initializing Controllers.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 Game.Game ( ) [protected]

10.11.3 Member Function Documentation

10.11.3.1 virtual void Game.Awake ( ) [protected], [virtual]

10.11.3.2 Controller Game.CreateController (Type *t*)

Creates a controller (A GameObject with the controller of type T attached to it).

Returns

The controller.

10.11.3.3 T Game.CreateController< T > ( )

Creates a controller (A GameObject with the controller of type T attached to it).

Returns

The controller.

Template Parameters

T	The controller type.

Type Constraints

T **:** ***Controller***

10.11.3.4 void Game.InitializeControllers ( )

10.11.3.5 virtual IEnumerator Game.Load (UpdateProgressDelegate *progress*) [virtual]

This method should do any set up necessary to load the controller and is invoked when you call GameStateManager.SwitchGame(). This should call StartCoroutine(Controller.Load) on any regular controller in the scene.

Returns

10.11.3.6 virtual void Game.OnDestroy ( ) [protected], [virtual]

10.11.3.7 virtual void Game.OnLoaded ( ) [virtual]

This method is called when this controller has started loading

10.11.3.8 virtual void Game.OnLoading ( ) [virtual]

This method is called when the load function has completed

10.11.3.9 void Game.RegisterController (Controller *controller*, bool *removeExisting* = false)

Registers a controller based off of its type

Parameters

controller	The controller to be registered with this game.
removeExisting	Should we remove the existing controller if one of the exact same type exists

10.11.3.10 virtual void Game.Reload ( ) [virtual]

This method simply starts the load method as a coroutine and should be overriden to add any reload logic that is necessary

10.11.3.11 virtual void Game.Setup ( ) [virtual]

10.11.3.12 virtual void Game.Unload ( ) [virtual]

10.11.3.13 void Game.UnregisterController (Controller *controller*)

Unregisters the controller.

Parameters

controller	Controller.

10.11.3.14 void Game.UnregisterController (string *controllerName*)

Unregisters the controller.

Parameters

controllerName	Controller name.

10.11.4 Property Documentation

10.11.4.1 IGameContainer Game.Container [get],[set]

10.11.4.2 Dictionary<string, Controller> Game.Controllers [get]

Gets the controllers that have been registered with this game.

The controllers.

10.11.4.3 Controller Game.this[string controllerName] [get]

Gets the Game with the specified controllerName.

Parameters

controllerName	Controller name.

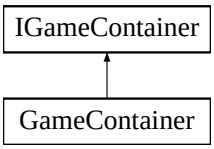
The documentation for this class was generated from the following file:

- Scripts/Base/Controllers/Game.cs

10.12 GameContainer Class Reference

A ViewModel Container and a factory for Controllers and commands.

Inheritance diagram for GameContainer:



Public Member Functions

- void Clear ()
- void Inject (object obj)

  Injects registered types/mappings into an object
- void Register< TSource, TTarget > ()

  Register a type mapping
- TBase RegisterInstance< TBase > (TBase instance=null)

  Register an instance of a type.
- object RegisterInstance (Type type, object instance=null)

  Register an instance of a type.
- T Resolve< T > ()

  If an instance of T exist then it will return that instance otherwise it will create a new one based off mappings.
- object Resolve (Type instanceType)

  If an instance of instanceType exist then it will return that instance otherwise it will create a new one based off mappings.

Properties

- Dictionary< Type, Type > Mappings `[get, set]`
- Dictionary< Type, object > Instances `[get, set]`

10.12.1 Detailed Description

A ViewModel Container and a factory for Controllers and commands.

10.12.2 Member Function Documentation

10.12.2.1 void GameContainer.Clear ( )

Implements IGameContainer.

10.12.2.2 void GameContainer.Inject (object *obj*)

Injects registered types/mappings into an object

Parameters

obj	

Implements IGameContainer.

10.12.2.3 void GameContainer.Register< TSource, TTarget > ( )

Register a type mapping

Template Parameters

TSource	The base type.
TTarget	The concrete type

Implements IGameContainer.

10.12.2.4 object GameContainer.RegisterInstance (Type *type*, object *instance* = `null`)

Register an instance of a type.

Parameters

type	The type of the instance
instance	

Returns

10.12.2.5 TBase GameContainer.RegisterInstance< TBase > (TBase *instance* = `null`)

Register an instance of a type.

Template Parameters

TBase	

Parameters

instance	

Returns

Type Constraints

TBase* : *class

TBase* : *new()

10.12.2.6 object GameContainer.Resolve (Type *instanceType*)

If an instance of instanceType exist then it will return that instance otherwise it will create a new one based off mappings.

Parameters

instanceType	The type of instance to resolve

Returns

The/An instance of 'instanceType'

Implements IGameContainer.

10.12.2.7 T GameContainer.Resolve< T > ( )

If an instance of T exist then it will return that instance otherwise it will create a new one based off mappings.

Template Parameters

T	The type of instance to resolve

Returns

The/An instance of 'instanceType'

Implements IGameContainer.

Type Constraints

T* : *class

T* : *new()

10.12.3 Property Documentation

10.12.3.1 Dictionary<Type, object> GameContainer.Instances [get],[set],[protected]

10.12.3.2 Dictionary<Type, Type> GameContainer.Mappings [get],[set]

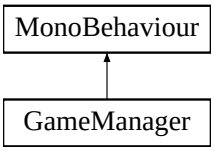
The documentation for this class was generated from the following file:

- Scripts/Base/Controllers/GameContainer.cs

10.13 GameManager Class Reference

A singleton that manages our current game and all the games in the scene. This component will persist through every level

Inheritance diagram for GameManager:



Public Member Functions

- virtual void AddGame (Game game)

  Adds a controler to the list of registered controllers. You shouldn't have to use this method directly. It is used by a game to register itself.
- void ApplyRenderSettings ()
- void Awake ()
- string GetPath (string elementPath, string path)
- void LoadRenderSettings ()
- void OnDestroy ()
- virtual void RemoveGame (Game game)

  Removes the game from this manager. This will only happen if a Game is destroyed
- virtual IEnumerator Start ()

Static Public Member Functions

- static Coroutine SwitchGame< T > (T controller=null, Action< T > setup=null, UpdateProgressDelegate progress=null)

  This switches the game from one to the other invoking a sequence of actions SwitchGame
- static void SwitchGameAndLevel< T > (SwitchLevelSettings< T > settings)

  Loads the other levels asynchronously and then switches the game assuming that it will exist in the scene after loading is finished.
- static void SwitchGameAndLevel< T > (Action< T > setup, params string[] levels)

  Loads the other levels asynchronously and then switches the game assuming that it will exist in the scene after loading is finished.

Public Attributes

- Color _AmbientLight = new Color(0.2f, 0.2f, 0.2f, 1.0f)
- string _ControllerScriptsPath = "@ElementPath/"
- float _FlareStrength = 1.0f
- bool _Fog
- Color _FogColor = new Color(0.5f, 0.5f, 0.5f, 1.0f)
- float _FogDensity = 0.01f
- FogMode _FogMode = FogMode.ExponentialSquared
- float _HaloStrength = 0.5f
- float _LinearFogEnd = 300.0f
- float _LinearFogStart = 0.0f
- string _LoadingLevel

  A level that displays a progress bar and message
- Material _SkyboxMaterial
- Game _Start

  Set this to the game that will load when the game starts
- string _StartupScene
- string _ViewModelScriptsPath = "@ElementPath/"
- string _ViewPrefabsPath = "@ElementPath/Resources/"
- string _ViewsScriptsPath = "@ElementPath/"

Static Protected Member Functions

- static void DefaultUpdateProgress (string message, float progress)

Properties

- static Game ActiveGame [get, set]

  The current running game
- static IGameContainer Container [get]
- static GameManager Instance [get, set]

  The current instance of GameManager
- static LevelLoadViewModel LoadingViewModel [get, set]

  The view model that is used for loading a scene. Bind to this to be notified of progress changes
- static ISwitchLevelSettings SwitchLevelSettings [get, set]
- Type ContainerType [get, set]
- List< Game > Games [get, set]

  A list of all the game in the scene. Each game registers itself with this manager and is added to this list.

10.13.1 Detailed Description

A singleton that manages our current game and all the games in the scene. This component will persist through every level

10.13.2 Member Function Documentation

10.13.2.1 virtual void GameManager.AddGame (Game *game*) [virtual]

Adds a controler to the list of registered controllers. You shouldn't have to use this method directly. It is used by a game to register itself.

Parameters

game	The game being added.

10.13.2.2 void GameManager.ApplyRenderSettings ( )

10.13.2.3 void GameManager.Awake ( )

10.13.2.4 static void GameManager.DefaultUpdateProgress (string *message*, float *progress*) [static], [protected]

10.13.2.5 string GameManager.GetPath (string *elementPath*, string *path*)

10.13.2.6 void GameManager.LoadRenderSettings ( )

10.13.2.7 void GameManager.OnDestroy ( )

10.13.2.8 virtual void GameManager.RemoveGame (Game *game*) [virtual]

Removes the game from this manager. This will only happen if a Game is destroyed

Parameters

game	

10.13.2.9 virtual IEnumerator GameManager.Start ( ) [virtual]

10.13.2.10 static Coroutine GameManager.SwitchGame< T > (T *controller* = null, Action< T > *setup* = null, UpdateProgressDelegate *progress* = null) [static]

This switches the game from one to the other invoking a sequence of actions SwitchGame

```
- Invoke the current controllers Unload() method.
- Set the CurrentController Property to the new game
- New Controller Load() method is invoked via StartCoroutine
- New Controller OnLoading() method is invoked
- After the Load() Coroutine method is complete it will invoke the ActiveGame Game's OnLoaded() method
```

Template Parameters

T	The game type

Parameters

progress	
setup	
controller	

Returns

Type Constraints

T **:** ***Game***

10.13.2.11 static void GameManager.SwitchGameAndLevel< T > (SwitchLevelSettings< T > *settings*) [static]

Loads the other levels asynchronously and then switches the game assuming that it will exist in the scene after loading is finished.

Template Parameters

T	The type of game

Returns

Type Constraints

T **:** ***Game***

10.13.2.12 static void GameManager.SwitchGameAndLevel< T > (Action< T > *setup*, params string[] *levels*) [static]

Loads the other levels asynchronously and then switches the game assuming that it will exist in the scene after loading is finished.

Template Parameters

T	The type of the Game to switch to

Parameters

setup	Setup the Game?
levels	Load these levels additively?

Returns

Type Constraints

T **:** ***Game***

10.13.3 Member Data Documentation

10.13.3.1 Color GameManager._AmbientLight = new Color(0.2f, 0.2f, 0.2f, 1.0f)

10.13.3.2 string GameManager._ControllerScriptsPath = "@ElementPath/"

10.13.3.3 float GameManager._FlareStrength = 1.0f

10.13.3.4 bool GameManager._Fog

10.13.3.5 Color GameManager._FogColor = new Color(0.5f, 0.5f, 0.5f, 1.0f)

10.13.3.6 float GameManager._FogDensity = 0.01f

10.13.3.7 FogMode GameManager._FogMode = FogMode.ExponentialSquared

10.13.3.8 float GameManager._HaloStrength = 0.5f

10.13.3.9 float GameManager._LinearFogEnd = 300.0f

10.13.3.10 float GameManager._LinearFogStart = 0.0f

10.13.3.11 string GameManager._LoadingLevel

A level that displays a progress bar and message

10.13.3.12 Material GameManager._SkyboxMaterial

10.13.3.13 Game GameManager._Start

Set this to the game that will load when the game starts

10.13.3.14 string GameManager._StartupScene

10.13.3.15 string GameManager._ViewModelScriptsPath = "@ElementPath/"

10.13.3.16 string GameManager._ViewPrefabsPath = "@ElementPath/Resources/"

10.13.3.17 string GameManager._ViewsScriptsPath = "@ElementPath/"

10.13.4 Property Documentation

10.13.4.1 Game GameManager.ActiveGame [static], [get], [set]

The current running game

10.13.4.2 IGameContainer GameManager.Container [static], [get]

10.13.4.3 Type GameManager.ContainerType [get], [set]

10.13.4.4 List<Game> GameManager.Games [get], [set]

A list of all the game in the scene. Each game registers itself with this manager and is added to this list.

10.13.4.5 GameManager GameManager.Instance [static], [get], [set]

The current instance of GameManager

10.13.4.6 LevelLoadViewModel GameManager.LoadingViewModel [static], [get], [set]

The view model that is used for loading a scene. Bind to this to be notified of progress changes

The loading view model.

10.13.4.7 ISwitchLevelSettings GameManager.SwitchLevelSettings [static], [get], [set]

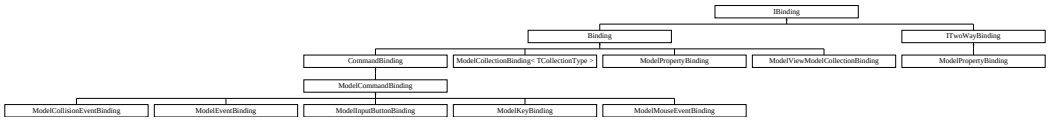
The documentation for this class was generated from the following file:

- Scripts/Base/Controllers/GameManager.cs

10.14 IBinding Interface Reference

Interface for all bindings

Inheritance diagram for IBinding:



Public Member Functions

- void Bind ()
- void Unbind ()

Properties

- bool CanTwoWayBind [get]
- bool IsComponent [get, set]
- string ModelMemberName [get, set]
- IViewModelObserver Source [get, set]
- bool TwoWay [get, set]

10.14.1 Detailed Description

Interface for all bindings

10.14.2 Member Function Documentation

10.14.2.1 void IBinding.Bind ( )

Implemented in Binding, ModelViewModelCollectionBinding, CommandBinding, ModelCollectionBinding< T-CollectionType >, ModelEventBinding, ModelCommandBinding, and ModelPropertyBinding.

10.14.2.2 void IBinding.Unbind ( )

Implemented in ModelViewModelCollectionBinding, Binding, CommandBinding, ModelCollectionBinding< T-CollectionType >, ModelPropertyBinding, ModelEventBinding, and ModelCommandBinding.

10.14.3 Property Documentation

10.14.3.1 bool IBinding.CanTwoWayBind [get]

10.14.3.2 bool IBinding.IsComponent [get], [set]

10.14.3.3 string IBinding.ModelMemberName [get], [set]

10.14.3.4 IViewModelObserver IBinding.Source [get], [set]

10.14.3.5 bool IBinding.TwoWay [get], [set]

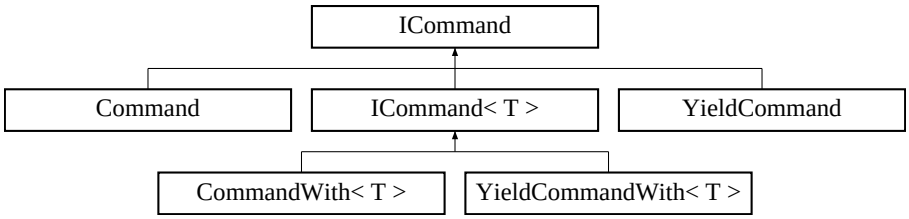
The documentation for this interface was generated from the following file:

- Scripts/Base/Bindings/IBinding.cs

10.15 ICommand Interface Reference

The base command interface for implementing a command in a ViewModel

Inheritance diagram for ICommand:



Public Member Functions

- IEnumerator Execute ()

Events

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

10.15.1 Detailed Description

The base command interface for implementing a command in a ViewModel

10.15.2 Member Function Documentation

10.15.2.1 IEnumerator ICommand.Execute ( )

Implemented in YieldCommand, YieldCommandWith< T >, CommandWith< T >, and Command.

10.15.3 Event Documentation

10.15.3.1 CommandEvent ICommand.OnCommandExecuted

10.15.3.2 CommandEvent ICommand.OnCommandExecuting

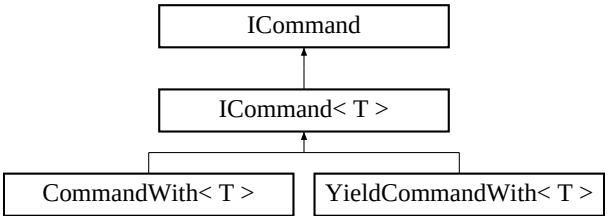
The documentation for this interface was generated from the following file:

- Scripts/Base/Commands/ICommand.cs

10.16 ICommand< T > Interface Template Reference

A base command interface for implementing a command with a parameter in a ViewModel

Inheritance diagram for ICommand< T >:



Properties

- T Parameter `[get, set]`

Additional Inherited Members

10.16.1 Detailed Description

A base command interface for implementing a command with a parameter in a ViewModel

Template Parameters

T	

10.16.2 Property Documentation

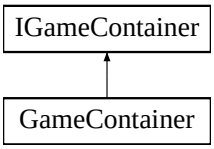
10.16.2.1 T ICommand< T >.Parameter [get], [set]

The documentation for this interface was generated from the following file:

- Scripts/Base/Commands/ICommand.cs

10.17 IGameContainer Interface Reference

Inheritance diagram for IGameContainer:



Public Member Functions

- void Clear ()
- void Inject (object obj)

  Injects registered types/mappings into an object
- void Register< TSource, TTarget > ()

  Register a type mapping
- TBase RegisterInstance< TBase > (TBase @default=null)

  Register an instance of a type.
- object RegisterInstance (Type type, object @default=null)

  Register an instance of a type.
- T Resolve< T > ()

  If an instance of T exist then it will return that instance otherwise it will create a new one based off mappings.
- object Resolve (Type instanceType)

  If an instance of instanceType exist then it will return that instance otherwise it will create a new one based off mappings.

10.17.1 Member Function Documentation

10.17.1.1 void IGameContainer.Clear ( )

Implemented in GameContainer.

10.17.1.2 void IGameContainer.Inject (object *obj*)

Injects registered types/mappings into an object

Parameters

obj	

Implemented in GameContainer.

10.17.1.3 void IGameContainer.Register< TSource, TTarget > ( )

Register a type mapping

Template Parameters

TSource	The base type.
TTarget	The concrete type

Implemented in GameContainer.

10.17.1.4 object IGameContainer.RegisterInstance (Type *type*, object @ *default* = `null`)

Register an instance of a type.

Parameters

type	
default	

Returns

10.17.1.5 TBase IGameContainer.RegisterInstance< TBase > (TBase @ *default* = `null`)

Register an instance of a type.

Template Parameters

TBase	

Parameters

default	

Returns

Type Constraints

TBase* : *class

TBase* : *new()

10.17.1.6 object IGameContainer.Resolve (Type *instanceType*)

If an instance of instanceType exist then it will return that instance otherwise it will create a new one based off mappings.

Parameters

instanceType	The type of instance to resolve

Returns

The/An instance of 'instanceType'

Implemented in GameContainer.

10.17.1.7 T IGameContainer.Resolve< T > ( )

If an instance of T exist then it will return that instance otherwise it will create a new one based off mappings.

Template Parameters

T	The type of instance to resolve

Returns

The/An instance of 'instanceType'

Implemented in GameContainer.

Type Constraints

T* : *class

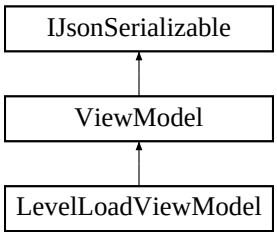
T* : *new()

The documentation for this interface was generated from the following file:

- Scripts/Base/Controllers/IGameContainer.cs

10.18 IJsonSerializable Interface Reference

Inheritance diagram for IJsonSerializable:



Public Member Functions

- void Deserialize (JSONNode node)
- JSONNode Serialize ()

10.18.1 Member Function Documentation

10.18.1.1 void IJsonSerializable.Deserialize (JSONNode *node*)

Implemented in ViewModel.

10.18.1.2 JSONNode IJsonSerializable.Serialize ( )

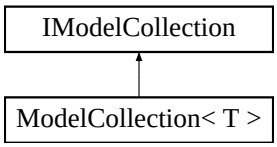
Implemented in ViewModel.

The documentation for this interface was generated from the following file:

- Scripts/Base/IJsonSerializable.cs

10.19 IModelCollection Interface Reference

Inheritance diagram for IModelCollection:



Properties

- IEnumerable< object > Value [get]

Events

- ModelCollectionChanged Changed

10.19.1 Property Documentation

10.19.1.1 IEnumerable<object> IModelCollection.Value [get]

10.19.2 Event Documentation

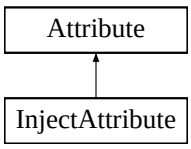
10.19.2.1 ModelCollectionChanged IModelCollection.Changed

The documentation for this interface was generated from the following file:

- Scripts/Base/ViewModels/ModelCollection.cs

10.20 InjectAttribute Class Reference

Inheritance diagram for InjectAttribute:

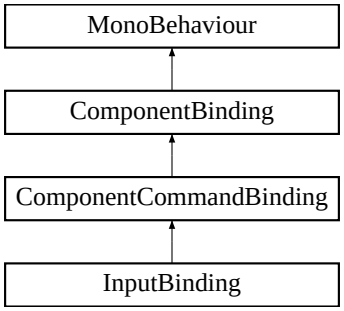


The documentation for this class was generated from the following file:

- Scripts/Base/Controllers/InjectAttribute.cs

10.21 InputBinding Class Reference

Inheritance diagram for InputBinding:



Public Member Functions

- void Update ()

Public Attributes

- string _ButtonName
- InputButtonEventType _EventType

Protected Member Functions

- override IBinding GetBinding ()

  The binding provider. Create the binding that the component will add to the source view here.

Additional Inherited Members

10.21.1 Member Function Documentation

10.21.1.1 override IBinding InputBinding.GetBinding ( ) [protected],[virtual]

The binding provider. Create the binding that the component will add to the source view here.

Returns

The binding that will be added to the source view.

Implements ComponentBinding.

10.21.1.2 void InputBinding.Update ( )

10.21.2 Member Data Documentation

10.21.2.1 string InputBinding._ButtonName

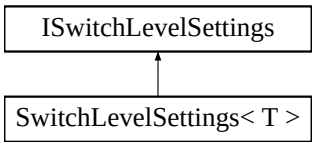
10.21.2.2 InputButtonEventType InputBinding._EventType

The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/MouseEventBinding.cs

10.22 ISwitchLevelSettings Interface Reference

Inheritance diagram for ISwitchLevelSettings:



Public Member Functions

- void InvokeControllerSetup (Game game)

Properties

- string[] Levels [get, set]
- Action< LevelLoadProgress > ProgressUpdated [get, set]
- Type StartControllerType [get]

10.22.1 Member Function Documentation

10.22.1.1 void ISwitchLevelSettings.InvokeControllerSetup (Game *game*)

10.22.2 Property Documentation

10.22.2.1 string [] ISwitchLevelSettings.Levels [get],[set]

10.22.2.2 Action<LevelLoadProgress> ISwitchLevelSettings.ProgressUpdated [get],[set]

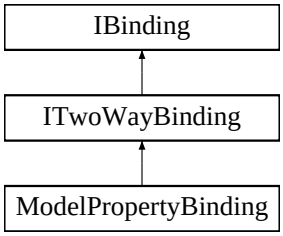
10.22.2.3 Type ISwitchLevelSettings.StartControllerType [get]

The documentation for this interface was generated from the following file:

- Scripts/Common/ISwitchLevelSettings.cs

10.23 ITwoWayBinding Interface Reference

Inheritance diagram for ITwoWayBinding:



Public Member Functions

- void BindReverse ()

  Will be called every update frame

Additional Inherited Members

10.23.1 Member Function Documentation

10.23.1.1 void ITwoWayBinding.BindReverse ( )

Will be called every update frame

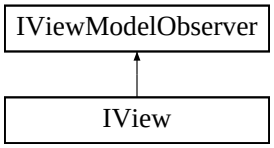
Implemented in ModelPropertyBinding.

The documentation for this interface was generated from the following file:

- Scripts/Base/Bindings/ITwoWayBinding.cs

10.24 IView Interface Reference

Inheritance diagram for IView:



Properties

- ViewModel ViewModelObject `[get]`

  Gets the view model object.
- Type ViewModelType `[get]`

  Gets the type of the view model.
- string ViewName `[get, set]`

  The name of the prefab that created this view

Additional Inherited Members

10.24.1 Property Documentation

10.24.1.1 ViewModel IView.ViewModelObject `[get]`

Gets the view model object.

The view model object.

10.24.1.2 Type IView.ViewModelType `[get]`

Gets the type of the view model.

The type of the model.

10.24.1.3 string IView.ViewName [get], [set]

The name of the prefab that created this view

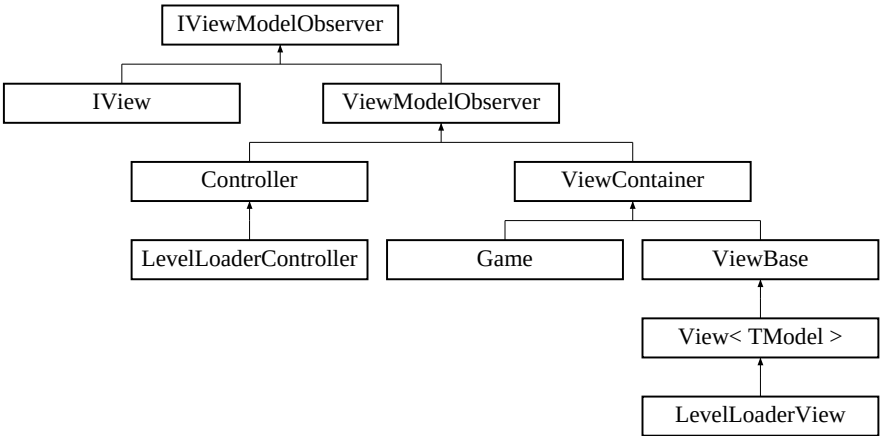
The documentation for this interface was generated from the following file:

- Scripts/Base/Views/IView.cs

10.25 IViewModelObserver Interface Reference

Potential future use.

Inheritance diagram for IViewModelObserver:



Public Member Functions

- void ExecuteCommand (ICommand command)
- void Unbind ()
- void AddBinding (IBinding binding)
- void RemoveBinding (IBinding binding)

Properties

- GameObject gameObject [get]
- Rigidbody rigidbody [get]
- Transform transform [get]
- bool enabled [get, set]
- List< IBinding > Bindings [get, set]

10.25.1 Detailed Description

Potential future use.

10.25.2 Member Function Documentation

10.25.2.1 void IViewModelObserver.AddBinding (IBinding *binding*)

Implemented in ViewModelObserver, ViewBase, and Controller.

10.25.2.2 void IViewModelObserver.ExecuteCommand (ICommand *command*)

Implemented in ViewModelObserver.

10.25.2.3 void IViewModelObserver.RemoveBinding (IBinding *binding*)

Implemented in ViewModelObserver.

10.25.2.4 void IViewModelObserver.Unbind ( )

Implemented in ViewModelObserver, and ViewBase.

10.25.3 Property Documentation

10.25.3.1 List<IBinding> IViewModelObserver.Bindings [get],[set]

10.25.3.2 bool IViewModelObserver.enabled [get],[set]

10.25.3.3 GameObject IViewModelObserver.gameObject [get]

10.25.3.4 Rigidbody IViewModelObserver.rigidbody [get]

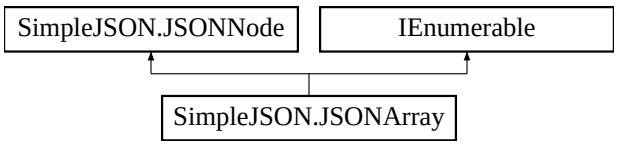
10.25.3.5 Transform IViewModelObserver.transform [get]

The documentation for this interface was generated from the following file:

- Scripts/Base/Bindings/IViewModelObserver.cs

10.26 SimpleJSON.JSONArray Class Reference

Inheritance diagram for SimpleJSON.JSONArray:



Public Member Functions

- override void Add (string aKey, JSONNode aItem)
- IEnumerator GetEnumerator ()
- override JSONNode Remove (int aIndex)
- override JSONNode Remove (JSONNode aNode)
- override void Serialize (System.IO.BinaryWriter aWriter)
- override string ToString ()
- override string ToString (string aPrefix)

Properties

- override IEnumerable< JSONNode > Childs [get]
- override int Count [get]
- override JSONNode this[int aIndex] [get, set]
- override JSONNode this[string aKey] [get, set]

Additional Inherited Members

10.26.1 Member Function Documentation

10.26.1.1 override void SimpleJSON.JSONArray.Add (string *aKey*, JSONNode *aItem*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.26.1.2 IEnumerator SimpleJSON.JSONArray.GetEnumerator ( )

10.26.1.3 override JSONNode SimpleJSON.JSONArray.Remove (int *aIndex*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.26.1.4 override JSONNode SimpleJSON.JSONArray.Remove (JSONNode *aNode*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.26.1.5 override void SimpleJSON.JSONArray.Serialize (System.IO.BinaryWriter *aWriter*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.26.1.6 override string SimpleJSON.JSONArray.ToString ( )

10.26.1.7 override string SimpleJSON.JSONArray.ToString (string *aPrefix*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.26.2 Property Documentation

10.26.2.1 override IEnumerable<JSONNode> SimpleJSON.JSONArray.Childs [get]

10.26.2.2 override int SimpleJSON.JSONArray.Count [get]

10.26.2.3 override JSONNode SimpleJSON.JSONArray.this[int aIndex] [get], [set]

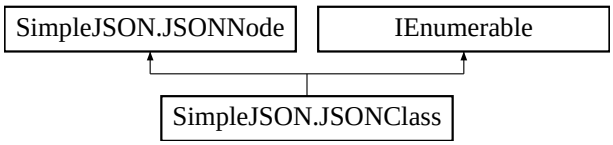
10.26.2.4 override JSONNode SimpleJSON.JSONArray.this[string aKey] [get], [set]

The documentation for this class was generated from the following file:

- Scripts/Base/SimpleJSON.cs

10.27 SimpleJSON.JSONClass Class Reference

Inheritance diagram for SimpleJSON.JSONClass:



Public Member Functions

- override void Add (string aKey, JSONNode aItem)
- IEnumerator GetEnumerator ()
- override JSONNode Remove (string aKey)
- override JSONNode Remove (int aIndex)
- override JSONNode Remove (JSONNode aNode)
- override void Serialize (System.IO.BinaryWriter aWriter)
- override string ToString ()
- override string ToString (string aPrefix)

Properties

- override IEnumerable< JSONNode > Childs `[get]`
- override int Count `[get]`
- override JSONNode this[string aKey] `[get, set]`
- override JSONNode this[int aIndex] `[get, set]`

Additional Inherited Members

10.27.1 Member Function Documentation

10.27.1.1 override void SimpleJSON.JSONClass.Add (string *aKey*, JSONNode *aItem*) `[virtual]`

Reimplemented from SimpleJSON.JSONNode.

10.27.1.2 IEnumerator SimpleJSON.JSONClass.GetEnumerator ( )

10.27.1.3 override JSONNode SimpleJSON.JSONClass.Remove (string *aKey*) `[virtual]`

Reimplemented from SimpleJSON.JSONNode.

10.27.1.4 override JSONNode SimpleJSON.JSONClass.Remove (int *aIndex*) `[virtual]`

Reimplemented from SimpleJSON.JSONNode.

10.27.1.5 override JSONNode SimpleJSON.JSONClass.Remove (JSONNode *aNode*) `[virtual]`

Reimplemented from SimpleJSON.JSONNode.

10.27.1.6 override void SimpleJSON.JSONClass.Serialize (System.IO.BinaryWriter *aWriter*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.27.1.7 override string SimpleJSON.JSONClass.ToString ( )

10.27.1.8 override string SimpleJSON.JSONClass.ToString (string *aPrefix*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.27.2 Property Documentation

10.27.2.1 override IEnumerable<JSONNode> SimpleJSON.JSONClass.Childs [get]

10.27.2.2 override int SimpleJSON.JSONClass.Count [get]

10.27.2.3 override JSONNode SimpleJSON.JSONClass.this[int aIndex] [get], [set]

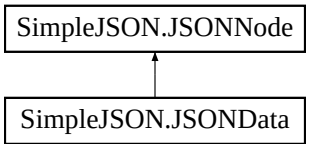
10.27.2.4 override JSONNode SimpleJSON.JSONClass.this[string aKey] [get], [set]

The documentation for this class was generated from the following file:

- Scripts/Base/SimpleJSON.cs

10.28 SimpleJSON.JSONData Class Reference

Inheritance diagram for SimpleJSON.JSONData:



Public Member Functions

- JSONData (string aData)
- JSONData (float aData)
- JSONData (double aData)
- JSONData (bool aData)
- JSONData (int aData)
- override void Serialize (System.IO.BinaryWriter aWriter)
- override string ToString ()
- override string ToString (string aPrefix)

Properties

- override string Value [get, set]

Additional Inherited Members

10.28.1 Constructor & Destructor Documentation

10.28.1.1 SimpleJSON.JSONData.JSONData (string *aData*)

10.28.1.2 SimpleJSON.JSONData.JSONData (float *aData*)

10.28.1.3 SimpleJSON.JSONData.JSONData (double *aData*)

10.28.1.4 SimpleJSON.JSONData.JSONData (bool *aData*)

10.28.1.5 SimpleJSON.JSONData.JSONData (int *aData*)

10.28.2 Member Function Documentation

10.28.2.1 override void SimpleJSON.JSONData.Serialize (System.IO.BinaryWriter *aWriter*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.28.2.2 override string SimpleJSON.JSONData.ToString ( )

10.28.2.3 override string SimpleJSON.JSONData.ToString (string *aPrefix*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.28.3 Property Documentation

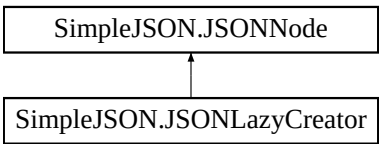
10.28.3.1 override string SimpleJSON.JSONData.Value [get],[set]

The documentation for this class was generated from the following file:

- Scripts/Base/SimpleJSON.cs

10.29 SimpleJSON.JSONLazyCreator Class Reference

Inheritance diagram for SimpleJSON.JSONLazyCreator:



Public Member Functions

- JSONLazyCreator (JSONNode aNode)
- JSONLazyCreator (JSONNode aNode, string aKey)
- override void Add (JSONNode aItem)
- override void Add (string aKey, JSONNode aItem)
- override bool Equals (object obj)
- override int GetHashCode ()

- override string ToString ()
- override string ToString (string aPrefix)

Static Public Member Functions

- static bool operator!= (JSONLazyCreator a, object b)
- static bool operator== (JSONLazyCreator a, object b)

Properties

- override JSONArray AsArray [get]
- override bool AsBool [get, set]
- override double AsDouble [get, set]
- override float AsFloat [get, set]
- override int AsInt [get, set]
- override JSONClass AsObject [get]
- override JSONNode this[int aIndex] [get, set]
- override JSONNode this[string aKey] [get, set]

10.29.1 Constructor & Destructor Documentation

10.29.1.1 SimpleJSON.JSONLazyCreator.JSONLazyCreator (JSONNode *aNode*)

10.29.1.2 SimpleJSON.JSONLazyCreator.JSONLazyCreator (JSONNode *aNode*, string *aKey*)

10.29.2 Member Function Documentation

10.29.2.1 override void SimpleJSON.JSONLazyCreator.Add (JSONNode *aItem*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.29.2.2 override void SimpleJSON.JSONLazyCreator.Add (string *aKey*, JSONNode *aItem*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.29.2.3 override bool SimpleJSON.JSONLazyCreator.Equals (object *obj*)

10.29.2.4 override int SimpleJSON.JSONLazyCreator.GetHashCode ( )

10.29.2.5 static bool SimpleJSON.JSONLazyCreator.operator!= (JSONLazyCreator *a*, object *b*) [static]

10.29.2.6 static bool SimpleJSON.JSONLazyCreator.operator== (JSONLazyCreator *a*, object *b*) [static]

10.29.2.7 override string SimpleJSON.JSONLazyCreator.ToString ( )

10.29.2.8 override string SimpleJSON.JSONLazyCreator.ToString (string *aPrefix*) [virtual]

Reimplemented from SimpleJSON.JSONNode.

10.29.3 Property Documentation

10.29.3.1 override JSONArray SimpleJSON.JSONLazyCreator.AsArray [get]

10.29.3.2 override bool SimpleJSON.JSONLazyCreator.AsBool [get], [set]

10.29.3.3 override double SimpleJSON.JSONLazyCreator.AsDouble [get], [set]

10.29.3.4 override float SimpleJSON.JSONLazyCreator.AsFloat [get], [set]

10.29.3.5 override int SimpleJSON.JSONLazyCreator.AsInt [get], [set]

10.29.3.6 override JSONClass SimpleJSON.JSONLazyCreator.AsObject [get]

10.29.3.7 override JSONNode SimpleJSON.JSONLazyCreator.this[int aIndex] [get], [set]

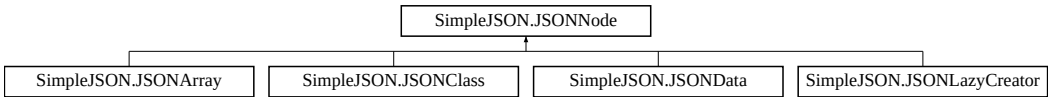
10.29.3.8 override JSONNode SimpleJSON.JSONLazyCreator.this[string aKey] [get], [set]

The documentation for this class was generated from the following file:

- Scripts/Base/SimpleJSON.cs

10.30 SimpleJSON.JSONNode Class Reference

Inheritance diagram for SimpleJSON.JSONNode:



Public Member Functions

- virtual void Add (string aKey, JSONNode aItem)
- virtual void Add (JSONNode aItem)
- virtual JSONNode Remove (string aKey)
- virtual JSONNode Remove (int aIndex)
- virtual JSONNode Remove (JSONNode aNode)
- override string ToString ()
- virtual string ToString (string aPrefix)
- override bool Equals (object obj)
- override int GetHashCode ()
- string SaveToBase64 ()
- string SaveToCompressedBase64 ()
- void SaveToCompressedFile (string aFileName)
- void SaveToCompressedStream (System.IO.Stream aData)
- void SaveToFile (string aFileName)
- void SaveToStream (System.IO.Stream aData)
- virtual void Serialize (System.IO.BinaryWriter aWriter)

Static Public Member Functions

- static implicit operator JSONNode (string s)
- static implicit operator string (JSONNode d)
- static bool operator!= (JSONNode a, object b)
- static bool operator== (JSONNode a, object b)
- static JSONNode Deserialize (System.IO.BinaryReader aReader)
- static JSONNode LoadFromBase64 (string aBase64)
- static JSONNode LoadFromCompressedBase64 (string aBase64)
- static JSONNode LoadFromCompressedFile (string aFileName)
- static JSONNode LoadFromCompressedStream (System.IO.Stream aData)
- static JSONNode LoadFromFile (string aFileName)
- static JSONNode LoadFromStream (System.IO.Stream aData)
- static JSONNode Parse (string aJSON)

Properties

- virtual IEnumerable< JSONNode > Childs `[get]`
- virtual int Count `[get]`
- IEnumerable< JSONNode > DeepChilds `[get]`
- virtual string Value `[get, set]`
- virtual JSONNode this[int aIndex] `[get, set]`
- virtual JSONNode this[string aKey] `[get, set]`
- virtual JSONArray AsArray `[get]`
- virtual bool AsBool `[get, set]`
- virtual double AsDouble `[get, set]`
- virtual float AsFloat `[get, set]`
- virtual int AsInt `[get, set]`
- virtual JSONClass AsObject `[get]`
- virtual Quaternion AsQuaternion `[get, set]`
- virtual Vector2 AsVector2 `[get, set]`
- virtual Vector3 AsVector3 `[get, set]`
- virtual Vector4 AsVector4 `[get, set]`

10.30.1 Member Function Documentation

10.30.1.1 virtual void SimpleJSON.JSONNode.Add (string *aKey,* JSONNode *aItem*) `[virtual]`

Reimplemented in SimpleJSON.JSONLazyCreator, SimpleJSON.JSONClass, and SimpleJSON.JSONArray.

10.30.1.2 virtual void SimpleJSON.JSONNode.Add (JSONNode *aItem*) `[virtual]`

Reimplemented in SimpleJSON.JSONLazyCreator.

10.30.1.3 static JSONNode SimpleJSON.JSONNode.Deserialize (System.IO.BinaryReader *aReader*) `[static]`

10.30.1.4 override bool SimpleJSON.JSONNode.Equals (object *obj*)

10.30.1.5 override int SimpleJSON.JSONNode.GetHashCode ( )

10.30.1.6 static JSONNode SimpleJSON.JSONNode.LoadFromBase64 (string *aBase64*) `[static]`

10.30.1.7 static JSONNode SimpleJSON.JSONNode.LoadFromCompressedBase64 (string *aBase64*) [static]

10.30.1.8 static JSONNode SimpleJSON.JSONNode.LoadFromCompressedFile (string *aFileName*) [static]

10.30.1.9 static JSONNode SimpleJSON.JSONNode.LoadFromCompressedStream (System.IO.Stream *aData*) [static]

10.30.1.10 static JSONNode SimpleJSON.JSONNode.LoadFromFile (string *aFileName*) [static]

10.30.1.11 static JSONNode SimpleJSON.JSONNode.LoadFromStream (System.IO.Stream *aData*) [static]

10.30.1.12 static implicit SimpleJSON.JSONNode.operator JSONNode (string *s*) [static]

10.30.1.13 static implicit SimpleJSON.JSONNode.operator string (JSONNode *d*) [static]

10.30.1.14 static bool SimpleJSON.JSONNode.operator!= (JSONNode *a*, object *b*) [static]

10.30.1.15 static bool SimpleJSON.JSONNode.operator== (JSONNode *a*, object *b*) [static]

10.30.1.16 static JSONNode SimpleJSON.JSONNode.Parse (string *aJSON*) [static]

10.30.1.17 virtual JSONNode SimpleJSON.JSONNode.Remove (string *aKey*) [virtual]

Reimplemented in SimpleJSON.JSONClass.

10.30.1.18 virtual JSONNode SimpleJSON.JSONNode.Remove (int *aIndex*) [virtual]

Reimplemented in SimpleJSON.JSONClass, and SimpleJSON.JSONArray.

10.30.1.19 virtual JSONNode SimpleJSON.JSONNode.Remove (JSONNode *aNode*) [virtual]

Reimplemented in SimpleJSON.JSONClass, and SimpleJSON.JSONArray.

10.30.1.20 string SimpleJSON.JSONNode.SaveToBase64 ( )

10.30.1.21 string SimpleJSON.JSONNode.SaveToCompressedBase64 ( )

10.30.1.22 void SimpleJSON.JSONNode.SaveToCompressedFile (string *aFileName*)

10.30.1.23 void SimpleJSON.JSONNode.SaveToCompressedStream (System.IO.Stream *aData*)

10.30.1.24 void SimpleJSON.JSONNode.SaveToFile (string *aFileName*)

10.30.1.25 void SimpleJSON.JSONNode.SaveToStream (System.IO.Stream *aData*)

10.30.1.26 virtual void SimpleJSON.JSONNode.Serialize (System.IO.BinaryWriter *aWriter*) [virtual]

Reimplemented in SimpleJSON.JSONData, SimpleJSON.JSONClass, and SimpleJSON.JSONArray.

10.30.1.27 override string SimpleJSON.JSONNode.ToString ( )

10.30.1.28 virtual string SimpleJSON.JSONNode.ToString (string *aPrefix*) [virtual]

Reimplemented in SimpleJSON.JSONLazyCreator, SimpleJSON.JSONData, SimpleJSON.JSONClass, and SimpleJSON.JSONArray.

10.30.2 Property Documentation

10.30.2.1 virtual JSONArray SimpleJSON.JSONNode.AsArray [get]

10.30.2.2 virtual bool SimpleJSON.JSONNode.AsBool [get], [set]

10.30.2.3 virtual double SimpleJSON.JSONNode.AsDouble [get], [set]

10.30.2.4 virtual float SimpleJSON.JSONNode.AsFloat [get], [set]

10.30.2.5 virtual int SimpleJSON.JSONNode.AsInt [get], [set]

10.30.2.6 virtual JSONClass SimpleJSON.JSONNode.AsObject [get]

10.30.2.7 virtual Quaternion SimpleJSON.JSONNode.AsQuaternion [get], [set]

10.30.2.8 virtual Vector2 SimpleJSON.JSONNode.AsVector2 [get], [set]

10.30.2.9 virtual Vector3 SimpleJSON.JSONNode.AsVector3 [get], [set]

10.30.2.10 virtual Vector4 SimpleJSON.JSONNode.AsVector4 [get], [set]

10.30.2.11 virtual IEnumerable<JSONNode> SimpleJSON.JSONNode.Childs [get]

10.30.2.12 virtual int SimpleJSON.JSONNode.Count [get]

10.30.2.13 IEnumerable<JSONNode> SimpleJSON.JSONNode.DeepChilds [get]

10.30.2.14 virtual JSONNode SimpleJSON.JSONNode.this[int aIndex] [get], [set]

10.30.2.15 virtual JSONNode SimpleJSON.JSONNode.this[string aKey] [get], [set]

10.30.2.16 virtual string SimpleJSON.JSONNode.Value [get], [set]

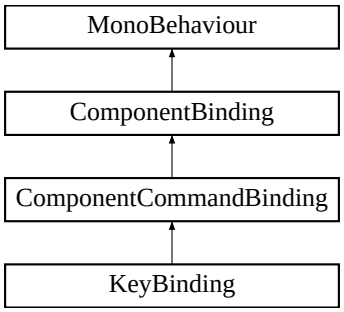
The documentation for this class was generated from the following file:

- Scripts/Base/SimpleJSON.cs

10.31 KeyBinding Class Reference

A component that will process a key binding as well as provide a key binding instance to the source view. Note. Even when adding this binding via code the component will still be added because a component is needed to process a keypress

Inheritance diagram for KeyBinding:



Public Attributes

- bool _Alt
- bool _Control
- KeyCode _Key
- KeyBindingEventType _KeyEventType = KeyBindingEventType.KeyDown
- bool _Shift

Protected Member Functions

- override IBinding GetBinding ()

  The binding provider. Create the binding that the component will add to the source view here.
- virtual bool IsKey (ModelKeyBinding keyBinding)
- void Update ()

Additional Inherited Members

10.31.1 Detailed Description

A component that will process a key binding as well as provide a key binding instance to the source view. Note. Even when adding this binding via code the component will still be added because a component is needed to process a keypress

10.31.2 Member Function Documentation

10.31.2.1 override IBinding KeyBinding.GetBinding ( ) [protected],[virtual]

The binding provider. Create the binding that the component will add to the source view here.

Returns

The binding that will be added to the source view.

Implements ComponentBinding.

10.31.2.2 virtual bool KeyBinding.IsKey (ModelKeyBinding *keyBinding*) [protected],[virtual]

10.31.2.3 void KeyBinding.Update ( ) [protected]

10.31.3 Member Data Documentation

10.31.3.1 bool KeyBinding._Alt

10.31.3.2 bool KeyBinding._Control

10.31.3.3 KeyCode KeyBinding._Key

10.31.3.4 KeyBindingEventType KeyBinding._KeyEventType = KeyBindingEventType.KeyDown

10.31.3.5 bool KeyBinding._Shift

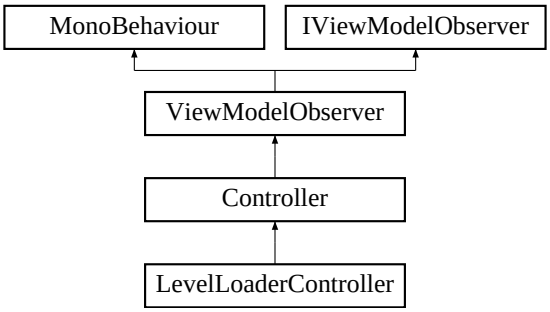
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/KeyBinding.cs

10.32 LevelLoaderController Class Reference

A [u]Frame built-in controller to manage loading a level via GameManager Add this in a level-loading scene along with LevelLoadViewModel and a LevelLoaderView.

Inheritance diagram for LevelLoaderController:



Public Member Functions

- void ProgressUpdated (string message, float progress)
- override void Setup (IGameContainer container)

Protected Member Functions

- IEnumerator Load ()
- override void Start ()

Properties

- static ISwitchLevelSettings Settings `[get]`

  The settings at which the level will be loaded
- LevelLoadViewModel Progress `[get]`

10.32.1 Detailed Description

A [u]Frame built-in controller to manage loading a level via GameManager Add this in a level-loading scene along with LevelLoadViewModel and a LevelLoaderView.

10.32.2 Member Function Documentation

10.32.2.1 IEnumerator LevelLoaderController.Load ( ) [protected]

10.32.2.2 void LevelLoaderController.ProgressUpdated (string *message*, float *progress*)

10.32.2.3 override void LevelLoaderController.Setup (IGameContainer *container*) [virtual]

Implements Controller.

10.32.2.4 override void LevelLoaderController.Start ( ) [protected], [virtual]

Reimplemented from Controller.

10.32.3 Property Documentation

10.32.3.1 LevelLoadViewModel LevelLoaderController.Progress [get], [protected]

10.32.3.2 ISwitchLevelSettings LevelLoaderController.Settings [static], [get]

The settings at which the level will be loaded

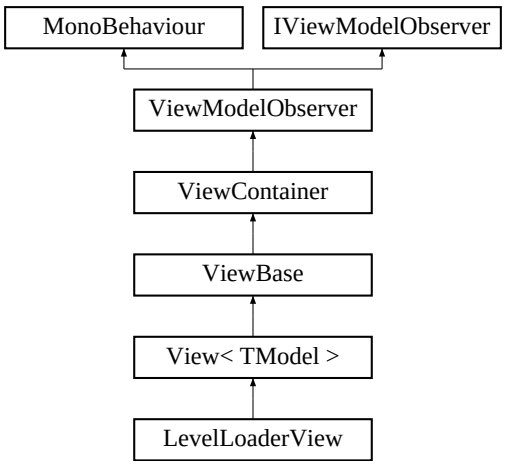
The settings.

The documentation for this class was generated from the following file:

- Scripts/Common/LevelLoaderController.cs

10.33 LevelLoaderView Class Reference

Inheritance diagram for LevelLoaderView:



Public Member Functions

- override void Bind ()
- override ViewModel CreateModel ()

Protected Member Functions

- override void InitializeModel (LevelLoadViewModel viewViewModel)

Additional Inherited Members

10.33.1 Member Function Documentation

10.33.1.1 override void LevelLoaderView.Bind ( ) [virtual]

Implements ViewBase.

10.33.1.2 override ViewModel LevelLoaderView.CreateModel ( ) [virtual]

Reimplemented from ViewBase.

10.33.1.3 override void LevelLoaderView.InitializeModel (LevelLoadViewModel *viewViewModel*) [protected]

The documentation for this class was generated from the following file:

- Scripts/Common/LevelLoaderView.cs

10.34 LevelLoadProgress Struct Reference

A struct for passing a message and a progress indicator

Public Member Functions

- LevelLoadProgress (string message, float progress)

  Level load progress

Properties

- string Message [get, set]

  Simply a message saying what is happening
- float Progress [get, set]

  Progress should be a normalized value ranging from 0f - 1.0f

10.34.1 Detailed Description

A struct for passing a message and a progress indicator

10.34.2 Constructor & Destructor Documentation

10.34.2.1 LevelLoadProgress.LevelLoadProgress (string *message*, float *progress*)

Level load progress

Parameters

message	What is happening?
progress	How complete are you. Range 0f - 1.0f

10.34.3 Property Documentation

10.34.3.1 string LevelLoadProgress.Message [get],[set]

Simply a message saying what is happening

10.34.3.2 float LevelLoadProgress.Progress [get],[set]

Progress should be a normalized value ranging from 0f - 1.0f

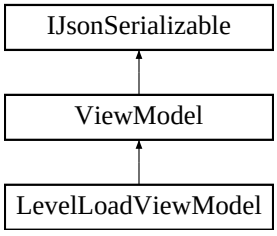
The documentation for this struct was generated from the following file:

- Scripts/Common/LevelLoadProgress.cs

10.35 LevelLoadViewModel Class Reference

The view model that is used when a level/scene is loading.

Inheritance diagram for LevelLoadViewModel:



Public Attributes

- readonly P< float > _Progress = new P<float>(0f)
- readonly P< string > _Status = new P<string>("Loading...")

Properties

- float Progress [get, set]
- string Status [get, set]

Additional Inherited Members

10.35.1 Detailed Description

The view model that is used when a level/scene is loading.

10.35.2 Member Data Documentation

10.35.2.1 readonly P<float> LevelLoadViewModel._Progress = new P<float>(0f)

10.35.2.2 readonly P<string> LevelLoadViewModel._Status = new P<string>("Loading...")

10.35.3 Property Documentation

10.35.3.1 float LevelLoadViewModel.Progress [get],[set]

10.35.3.2 string LevelLoadViewModel.Status [get],[set]

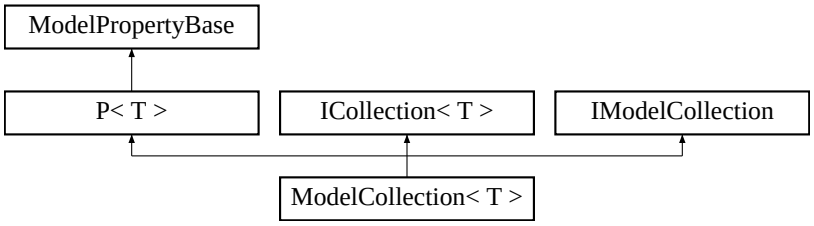
The documentation for this class was generated from the following file:

- Scripts/Common/LevelLoadViewModel.cs

10.36 ModelCollection< T > Class Template Reference

An observable collection to use in viewmodels.

Inheritance diagram for ModelCollection< T >:



Public Member Functions

- delegate void ModelCollectionChangedWith (ModelCollectionChangeEventWith< T > changeArgs)
- ModelCollection ()
- ModelCollection (IEnumerable< T > enumerable)
- virtual void Add (T item)
- override bool CanSetValue (List< T > value)
- virtual void Clear ()
- virtual bool Contains (T item)
- void CopyTo (T[] array, int arrayIndex)
- override void Deserialize (JSONNode node)
- IEnumerator< T > GetEnumerator ()
- virtual bool Remove (T item)
- override JSONNode Serialize ()
- override string ToString ()

Protected Member Functions

- virtual void OnChangedWith (ModelCollectionChangeEventWith< T > changeargs)

Properties

- int Count [get]
- bool IsReadOnly [get]
- Action< T > OnAdd [get, set]
- Action< T > OnRemove [get, set]
- override Type ValueType [get]

Events

- ModelCollectionChanged Changed
- ModelCollectionChangedWith ChangedWith

Additional Inherited Members

10.36.1 Detailed Description

An observable collection to use in viewmodels.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 ModelCollection< T >.ModelCollection ( )

10.36.2.2 ModelCollection< T >.ModelCollection (IEnumerable< T > *enumerable*)

10.36.3 Member Function Documentation

10.36.3.1 virtual void ModelCollection< T >.Add (T *item*) [virtual]

10.36.3.2 override bool ModelCollection< T >.CanSetValue (List< T > *value*)

10.36.3.3 virtual void ModelCollection< T >.Clear ( ) [virtual]

10.36.3.4 virtual bool ModelCollection< T >.Contains (T *item*) [virtual]

10.36.3.5 void ModelCollection< T >.CopyTo (T[] *array*, int *arrayIndex*)

10.36.3.6 override void ModelCollection< T >.Deserialize (JSONNode *node*) [virtual]

Implements ModelPropertyBase.

10.36.3.7 IEnumerator<T> ModelCollection< T >.GetEnumerator ( )

10.36.3.8 delegate void ModelCollection< T >.ModelCollectionChangedWith (ModelCollectionChangeEventWith< T > *changeArgs*)

10.36.3.9 virtual void ModelCollection< T >.OnChangedWith (ModelCollectionChangeEventWith< T > *changeargs*) [protected], [virtual]

10.36.3.10 virtual bool ModelCollection< T >.Remove (T *item*) [virtual]

10.36.3.11 override JSONNode ModelCollection< T >.Serialize ( ) [virtual]

Implements ModelPropertyBase.

10.36.3.12 override string ModelCollection< T >.ToString ( )

10.36.4 Property Documentation

10.36.4.1 int ModelCollection< T >.Count [get]

10.36.4.2 bool ModelCollection< T >.IsReadOnly [get]

10.36.4.3 Action<T> ModelCollection< T >.OnAdd [get], [set]

10.36.4.4 Action<T> ModelCollection< T >.OnRemove [get], [set]

10.36.4.5 override Type ModelCollection< T >.ValueType [get]

10.36.5 Event Documentation

10.36.5.1 ModelCollectionChanged ModelCollection< T >.Changed

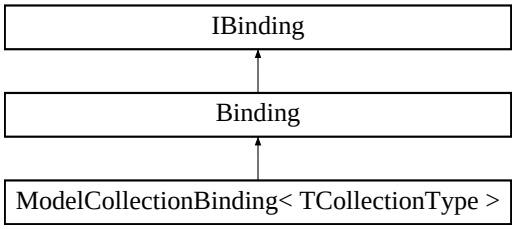
10.36.5.2 ModelCollectionChangedWith ModelCollection< T >.ChangedWith

The documentation for this class was generated from the following file:

- Scripts/Base/ViewModels/ModelCollection.cs

10.37 ModelCollectionBinding< TCollectionType > Class Template Reference

Inheritance diagram for ModelCollectionBinding< TCollectionType >:



Public Member Functions

- override void Bind ()

  Set-up the binding. This should almost always be implemented in a deriving class.
- void Immediate ()
- ModelCollectionBinding < TCollectionType > SetAddHandler (Action< TCollectionType > onAddHandler)
- ModelCollectionBinding < TCollectionType > SetRemoveHandler (Action< TCollectionType > onRemoveHandler)
- override void Unbind ()

  Unbind this binding

Properties

- ModelCollection< TCollectionType > Collection [get]
- bool IsImmediate [get, set]

- Action< TCollectionType > OnAdd [get, set]
- Action< TCollectionType > OnRemove [get, set]

Additional Inherited Members

10.37.1 Member Function Documentation

10.37.1.1 override void ModelCollectionBinding< TCollectionType >.Bind ( ) [virtual]

Set-up the binding. This should almost always be implemented in a deriving class.

Reimplemented from Binding.

10.37.1.2 void ModelCollectionBinding< TCollectionType >.Immediate ( )

10.37.1.3 ModelCollectionBinding<TCollectionType> ModelCollectionBinding< TCollectionType >.SetAddHandler (Action< TCollectionType > *onAddHandler*)

10.37.1.4 ModelCollectionBinding<TCollectionType> ModelCollectionBinding< TCollectionType >.SetRemoveHandler (Action< TCollectionType > *onRemoveHandler*)

10.37.1.5 override void ModelCollectionBinding< TCollectionType >.Unbind ( ) [virtual]

Unbind this binding

Reimplemented from Binding.

10.37.2 Property Documentation

10.37.2.1 ModelCollection<TCollectionType> ModelCollectionBinding< TCollectionType >.Collection [get]

10.37.2.2 bool ModelCollectionBinding< TCollectionType >.IsImmediate [get],[set]

10.37.2.3 Action<TCollectionType> ModelCollectionBinding< TCollectionType >.OnAdd [get],[set]

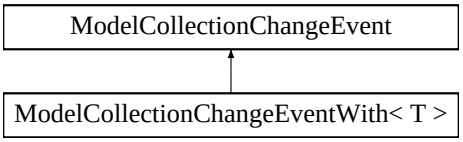
10.37.2.4 Action<TCollectionType> ModelCollectionBinding< TCollectionType >.OnRemove [get],[set]

The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelViewModelCollectionBinding.cs

10.38 ModelCollectionChangeEvent Class Reference

Inheritance diagram for ModelCollectionChangeEvent:



Properties

- ModelCollectionAction Action [get, set]
- object[] NewItems [get, set]
- object[] OldItems [get, set]

10.38.1 Property Documentation

10.38.1.1 ModelCollectionAction ModelCollectionChangeEvent.Action [get],[set]

10.38.1.2 object [] ModelCollectionChangeEvent.NewItems [get],[set]

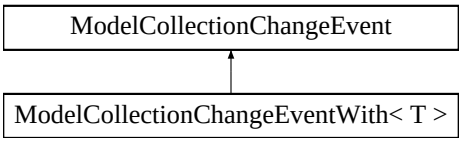
10.38.1.3 object [] ModelCollectionChangeEvent.OldItems [get],[set]

The documentation for this class was generated from the following file:

- Scripts/Base/ViewModels/ModelCollection.cs

10.39 ModelCollectionChangeEventWith< T > Class Template Reference

Inheritance diagram for ModelCollectionChangeEventWith< T >:



Properties

- T[] NewItemsOfT [get, set]
- T[] OldItemsOfT [get, set]

10.39.1 Property Documentation

10.39.1.1 T [] ModelCollectionChangeEventWith< T >.NewItemsOfT [get],[set]

10.39.1.2 T [] ModelCollectionChangeEventWith< T >.OldItemsOfT [get],[set]

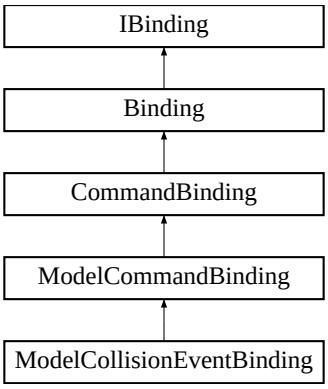
The documentation for this class was generated from the following file:

- Scripts/Base/ViewModels/ModelCollection.cs

10.40 ModelCollisionEventBinding Class Reference

A collision binding that will trigger a command when executed. Use chaining when possible to provide additional options for this binding.

Inheritance diagram for ModelCollisionEventBinding:



Public Member Functions

- ModelCollisionEventBinding When (Predicate< GameObject > predicate)

  A filter to determine when a collision should invoke the command this is bound to.

Properties

- CollisionEventType CollisionEvent `[get, set]`

  The collision/trigger event that will invoke the command this is bound to.

Additional Inherited Members

10.40.1 Detailed Description

A collision binding that will trigger a command when executed. Use chaining when possible to provide additional options for this binding.

10.40.2 Member Function Documentation

10.40.2.1 ModelCollisionEventBinding ModelCollisionEventBinding.When (Predicate< GameObject > *predicate*)

A filter to determine when a collision should invoke the command this is bound to.

Parameters

predicate	Return true if the command should be invoked. Use the GameObject parameter to filter colliders.

Returns

This so it can be further chained.

10.40.3 Property Documentation

10.40.3.1 CollisionEventType ModelCollisionEventBinding.CollisionEvent `[get],[set]`

The collision/trigger event that will invoke the command this is bound to.

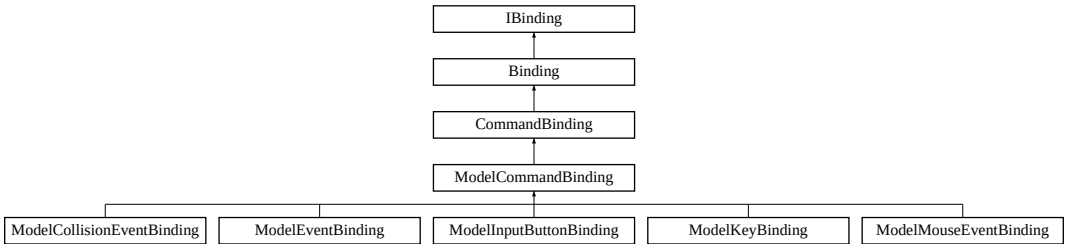
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelCollisionEventBinding.cs

10.41 ModelCommandBinding Class Reference

A base class for binding to a ViewModel command.

Inheritance diagram for ModelCommandBinding:



Public Member Functions

- ModelCommandBinding ()
- override void Bind ()

  Set-up the binding. This should almost always be implemented in a deriving class.
- override void Unbind ()

  Unbind this binding

Properties

- ComponentCommandBinding Component `[get, set]`

Additional Inherited Members

10.41.1 Detailed Description

A base class for binding to a ViewModel command.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 ModelCommandBinding.ModelCommandBinding ( )

10.41.3 Member Function Documentation

10.41.3.1 override void ModelCommandBinding.Bind ( ) `[virtual]`

Set-up the binding. This should almost always be implemented in a deriving class.

Reimplemented from CommandBinding.

Reimplemented in ModelEventBinding.

10.41.3.2 override void ModelCommandBinding.Unbind ( ) `[virtual]`

Unbind this binding

Reimplemented from CommandBinding.

Reimplemented in ModelEventBinding.

10.41.4 Property Documentation

10.41.4.1 ComponentCommandBinding ModelCommandBinding.Component [get],[set]

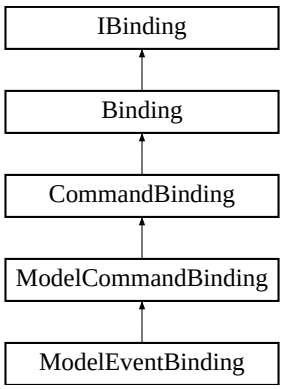
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelCommandBinding.cs

10.42 ModelEventBinding Class Reference

An event binding. Basically a wrapper for a .NET event so events can be triggered by a string. They can easily be bound and is mainly for conveniance.

Inheritance diagram for ModelEventBinding:



Public Member Functions

- ModelEventBinding (string eventName)
- override void Bind ()

  Set-up the binding. This should almost always be implemented in a deriving class.
- override void Unbind ()

  Unbind this binding

Properties

- virtual string EventName [get, set]

Additional Inherited Members

10.42.1 Detailed Description

An event binding. Basically a wrapper for a .NET event so events can be triggered by a string. They can easily be bound and is mainly for conveniance.

10.42.2 Constructor & Destructor Documentation

10.42.2.1 ModelEventBinding.ModelEventBinding (string *eventName*)

10.42.3 Member Function Documentation

10.42.3.1 override void ModelEventBinding.Bind ( ) [virtual]

Set-up the binding. This should almost always be implemented in a deriving class.

Reimplemented from ModelCommandBinding.

10.42.3.2 override void ModelEventBinding.Unbind ( ) [virtual]

Unbind this binding

Reimplemented from ModelCommandBinding.

10.42.4 Property Documentation

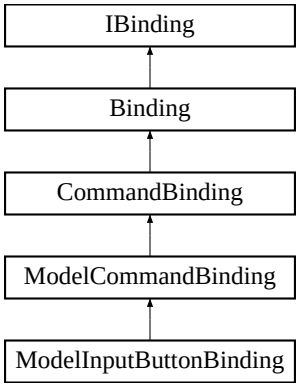
10.42.4.1 virtual string ModelEventBinding.EventName [get],[set]

The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelEventBinding.cs

10.43 ModelInputButtonBinding Class Reference

Inheritance diagram for ModelInputButtonBinding:



Properties

- string ButtonName [get, set]
- InputButtonEventType EventType [get, set]

Additional Inherited Members

10.43.1 Property Documentation

10.43.1.1 string ModelInputButtonBinding.ButtonName [get],[set]

10.43.1.2 InputButtonEventType ModelInputButtonBinding.EventType [get],[set]

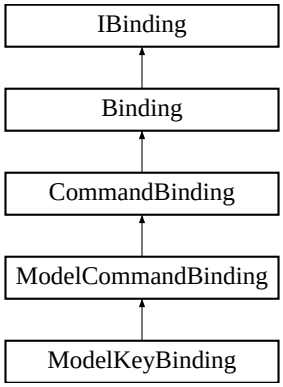
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelMouseEventBinding.cs

10.44 ModelKeyBinding Class Reference

Binds a key to a ViewModel command.

Inheritance diagram for ModelKeyBinding:



Public Member Functions

- ModelKeyBinding (KeyCode key)
- ModelKeyBinding On (KeyBindingEventType eventType)
- ModelKeyBinding RequireAlt ()

  When invoked Alt must be pressed along with 'Key' for the command to be invoked
- ModelKeyBinding RequireControl ()

  When invoked Control must be pressed along with 'Key' for the command to be invoked
- ModelKeyBinding RequireShift ()

  When invoked Shift must be pressed along with 'Key' for the command to be invoked

Properties

- bool Alt `[get, set]`
- bool Control `[get, set]`
- KeyCode Key `[get, set]`
- KeyBindingEventType KeyEventType `[get, set]`
- bool Shift `[get, set]`

Additional Inherited Members

10.44.1 Detailed Description

Binds a key to a ViewModel command.

10.44.2 Constructor & Destructor Documentation

10.44.2.1 ModelKeyBinding.ModelKeyBinding (KeyCode *key*)

10.44.3 Member Function Documentation

10.44.3.1 ModelKeyBinding ModelKeyBinding.On (KeyBindingEventType *eventType*)

10.44.3.2 ModelKeyBinding ModelKeyBinding.RequireAlt ( )

When invoked Alt must be pressed along with 'Key' for the command to be invoked

Returns

This to respect chaining.

10.44.3.3 ModelKeyBinding ModelKeyBinding.RequireControl ( )

When invoked Control must be pressed along with 'Key' for the command to be invoked

Returns

This to respect chaining.

10.44.3.4 ModelKeyBinding ModelKeyBinding.RequireShift ( )

When invoked Shift must be pressed along with 'Key' for the command to be invoked

Returns

This to respect chaining.

10.44.4 Property Documentation

10.44.4.1 bool ModelKeyBinding.Alt [get],[set]

10.44.4.2 bool ModelKeyBinding.Control [get],[set]

10.44.4.3 KeyCode ModelKeyBinding.Key [get],[set]

10.44.4.4 KeyBindingEventType ModelKeyBinding.KeyEventType [get],[set]

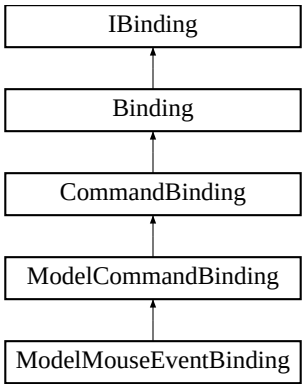
10.44.4.5 bool ModelKeyBinding.Shift [get],[set]

The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelKeyBinding.cs

10.45 ModelMouseEventBinding Class Reference

Inheritance diagram for ModelMouseEventBinding:



Properties

- MouseEventType EventType `[get, set]`

Additional Inherited Members

10.45.1 Property Documentation

10.45.1.1 MouseEventType ModelMouseEventBinding.EventType `[get],[set]`

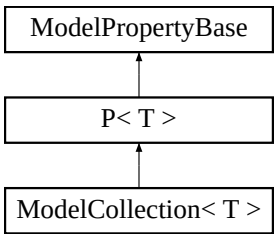
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelMouseEventBinding.cs

10.46 ModelPropertyBase Class Reference

A base class for model properties.

Inheritance diagram for ModelPropertyBase:



Public Member Functions

- delegate void PropertyChangedHandler (object value)
- abstract void Deserialize (JSONNode node)
- void QuietlySetValue (object value)

  Sets the value without invoking any OnPropertyChanged events. This is useful for two-way bindings
- abstract JSONNode Serialize ()

Static Public Member Functions

- static object DeserializeObject (Type valueType, JSONNode node)
- static JSONNode SerializeObject (Type valueType, object value)

Protected Attributes

- object _value

Properties

- virtual object ObjectValue [get, set]

  The value of this model property
- virtual Type ValueType [get]

  The value type of this property

Events

- PropertyChangedHandler PropertyChanged

  When the value has changed

10.46.1 Detailed Description

A base class for model properties.

10.46.2 Member Function Documentation

10.46.2.1 abstract void ModelPropertyBase.Deserialize (JSONNode *node*) [pure virtual]

Implemented in ModelCollection< T >, and P< T >.

10.46.2.2 static object ModelPropertyBase.DeserializeObject (Type *valueType*, JSONNode *node*) [static]

10.46.2.3 delegate void ModelPropertyBase.PropertyChangedHandler (object *value*)

10.46.2.4 void ModelPropertyBase.QuietlySetValue (object *value*)

Sets the value without invoking any OnPropertyChanged events. This is useful for two-way bindings

Parameters

value	

10.46.2.5 abstract JSONNode ModelPropertyBase.Serialize ( ) [pure virtual]

Implemented in ModelCollection< T >, and P< T >.

10.46.2.6 static JSONNode ModelPropertyBase.SerializeObject (Type *valueType*, object *value*) [static]

10.46.3 Member Data Documentation

10.46.3.1 object ModelPropertyBase._value [protected]

10.46.4 Property Documentation

10.46.4.1 virtual object ModelPropertyBase.ObjectValue [get],[set]

The value of this model property

10.46.4.2 virtual Type ModelPropertyBase.ValueType [get]

The value type of this property

10.46.5 Event Documentation

10.46.5.1 PropertyChangedHandler ModelPropertyBase.PropertyChanged

When the value has changed

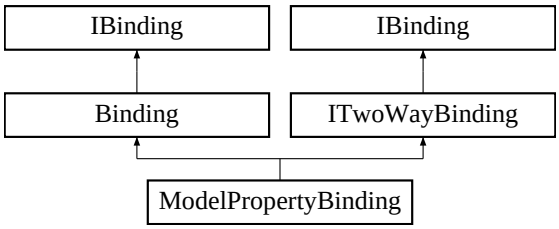
The documentation for this class was generated from the following file:

- Scripts/Base/ViewModels/ModelPropertyBase.cs

10.47 ModelPropertyBinding Class Reference

A class that contains a binding from a ViewModel to a Target

Inheritance diagram for ModelPropertyBinding:



Public Member Functions

- override void Bind ()

  Set-up the binding. This should almost always be implemented in a deriving class.
- void BindReverse ()

  If the value has changed apply the value to the property without reinvoking the SetTargetDelegate. It's important to not reinvoke the SetTargetDelegate because it will create a stack overflow. But only the SetTargetDelegate should be ignored because there may be other bindings to this property and when it changes they should definately know about it.
- override void Unbind ()

  Unbind remove the property changed event handler and the sets the model property to null so it can be refreshed if a new model is set

Additional Inherited Members

10.47.1 Detailed Description

A class that contains a binding from a ViewModel to a Target

10.47.2 Member Function Documentation

10.47.2.1 override void ModelPropertyBinding.Bind ( ) [virtual]

Set-up the binding. This should almost always be implemented in a deriving class.

Reimplemented from Binding.

10.47.2.2 void ModelPropertyBinding.BindReverse ( )

If the value has changed apply the value to the property without reinvoking the SetTargetDelegate. It's important to not reinvoke the SetTargetDelegate because it will create a stack overflow. But only the SetTargetDelegate should be ignored because there may be other bindings to this property and when it changes they should definately know about it.

Implements ITwoWayBinding.

10.47.2.3 override void ModelPropertyBinding.Unbind ( ) [virtual]

Unbind remove the property changed event handler and the sets the model property to null so it can be refreshed if a new model is set

Reimplemented from Binding.

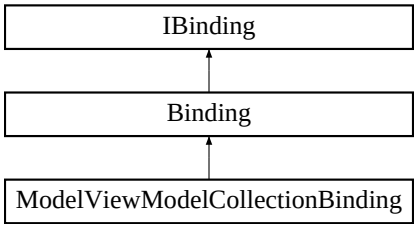
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelPropertyBinding.cs

10.48 ModelViewModelCollectionBinding Class Reference

Class for a view collection binding. Binds a ViewModel collection to a set of corresponding Views

Inheritance diagram for ModelViewModelCollectionBinding:



Public Member Functions

- override void Bind ()

  Set-up the binding. This should almost always be implemented in a deriving class.
- ModelViewModelCollectionBinding Immediate (bool immediate=true)
- ModelViewModelCollectionBinding SetAddHandler (Action< ViewBase > onAdd)

- ModelViewModelCollectionBinding SetCreateHandler (Func< ModelViewModelCollectionBinding, ViewModel, ViewBase > onCreateView)
- ModelViewModelCollectionBinding SetParent (Transform parent)
- ModelViewModelCollectionBinding SetRemoveHandler (Action< ViewBase > onRemove)
- ModelViewModelCollectionBinding SetView (string viewName)
- override void Unbind ()

  Unbind this binding

Properties

- IModelCollection Collection `[get]`
- bool IsImmediate `[get, set]`
- Action< ViewBase > OnAddView `[get, set]`
- Func < ModelViewModelCollectionBinding, ViewModel, ViewBase > OnCreateView `[get, set]`
- Action< ViewBase > OnRemoveView `[get, set]`
- Transform Parent `[get, set]`
- string ViewName `[get, set]`

Additional Inherited Members

10.48.1 Detailed Description

Class for a view collection binding. Binds a ViewModel collection to a set of corresponding Views

10.48.2 Member Function Documentation

10.48.2.1 override void ModelViewModelCollectionBinding.Bind ( ) `[virtual]`

Set-up the binding. This should almost always be implemented in a deriving class.

Reimplemented from Binding.

10.48.2.2 ModelViewModelCollectionBinding ModelViewModelCollectionBinding.Immediate (bool *immediate* = `true`)

10.48.2.3 ModelViewModelCollectionBinding ModelViewModelCollectionBinding.SetAddHandler (Action< ViewBase > *onAdd*)

10.48.2.4 ModelViewModelCollectionBinding ModelViewModelCollectionBinding.SetCreateHandler (Func< ModelViewModelCollectionBinding, ViewModel, ViewBase > *onCreateView*)

10.48.2.5 ModelViewModelCollectionBinding ModelViewModelCollectionBinding.SetParent (Transform *parent*)

10.48.2.6 ModelViewModelCollectionBinding ModelViewModelCollectionBinding.SetRemoveHandler (Action< ViewBase > *onRemove*)

10.48.2.7 ModelViewModelCollectionBinding ModelViewModelCollectionBinding.SetView (string *viewName*)

10.48.2.8 override void ModelViewModelCollectionBinding.Unbind ( ) `[virtual]`

Unbind this binding

Reimplemented from Binding.

10.48.3 Property Documentation

10.48.3.1 IModelCollection ModelViewModelCollectionBinding.Collection [get]

10.48.3.2 bool ModelViewModelCollectionBinding.IsImmediate [get], [set]

10.48.3.3 Action<ViewBase> ModelViewModelCollectionBinding.OnAddView [get], [set]

10.48.3.4 Func<ModelViewModelCollectionBinding, ViewModel, ViewBase> ModelViewModelCollectionBinding.-OnCreateView [get], [set]

10.48.3.5 Action<ViewBase> ModelViewModelCollectionBinding.OnRemoveView [get], [set]

10.48.3.6 Transform ModelViewModelCollectionBinding.Parent [get], [set]

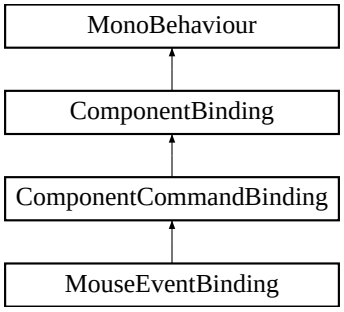
10.48.3.7 string ModelViewModelCollectionBinding.ViewName [get], [set]

The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ModelViewModelCollectionBinding.cs

10.49 MouseEventBinding Class Reference

Inheritance diagram for MouseEventBinding:



Public Attributes

- MouseEventType _EventType

Protected Member Functions

- override IBinding GetBinding ()

  The binding provider. Create the binding that the component will add to the source view here.
- virtual void OnBecameInvisible ()
- virtual void OnBecameVisible ()
- virtual void OnMouseDown ()
- virtual void OnMouseDrag ()
- virtual void OnMouseEnter ()
- virtual void OnMouseExit ()
- virtual void OnMouseOver ()
- virtual void OnMouseUp ()
- virtual void OnMouseUpAsButton ()

Additional Inherited Members

10.49.1 Member Function Documentation

10.49.1.1 override IBinding MouseEventBinding.GetBinding ( ) [protected], [virtual]

The binding provider. Create the binding that the component will add to the source view here.

Returns

The binding that will be added to the source view.

Implements ComponentBinding.

10.49.1.2 virtual void MouseEventBinding.OnBecameInvisible ( ) [protected], [virtual]

10.49.1.3 virtual void MouseEventBinding.OnBecameVisible ( ) [protected], [virtual]

10.49.1.4 virtual void MouseEventBinding.OnMouseDown ( ) [protected], [virtual]

10.49.1.5 virtual void MouseEventBinding.OnMouseDrag ( ) [protected], [virtual]

10.49.1.6 virtual void MouseEventBinding.OnMouseEnter ( ) [protected], [virtual]

10.49.1.7 virtual void MouseEventBinding.OnMouseExit ( ) [protected], [virtual]

10.49.1.8 virtual void MouseEventBinding.OnMouseOver ( ) [protected], [virtual]

10.49.1.9 virtual void MouseEventBinding.OnMouseUp ( ) [protected], [virtual]

10.49.1.10 virtual void MouseEventBinding.OnMouseUpAsButton ( ) [protected], [virtual]

10.49.2 Member Data Documentation

10.49.2.1 MouseEventType MouseEventBinding._EventType

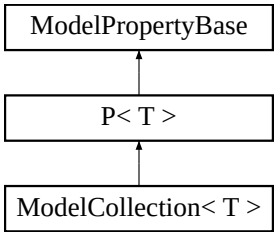
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/MouseEventBinding.cs

10.50 P< T > Class Template Reference

A typed ViewModel Property Class

Inheritance diagram for P< T >:



Public Member Functions

- P ()
- P (T value)
- void Bind (Action< T > target, bool immediate=true)

  Bind the specified target to this property.
- virtual bool CanSetValue (T value)
- override void Deserialize (JSONNode node)

  Deserialize the specified node into `Value`.
- override bool Equals (object obj)
- override int GetHashCode ()
- override JSONNode Serialize ()

  Serializes this object

Properties

- T Value `[get, set]`

  Gets or sets the value.
- override Type ValueType `[get]`

  Gets the type of the value.

Additional Inherited Members

10.50.1 Detailed Description

A typed ViewModel Property Class

Template Parameters

T	

10.50.2 Constructor & Destructor Documentation

10.50.2.1 P< T >.P ( )

10.50.2.2 P< T >.P (T *value*)

10.50.3 Member Function Documentation

10.50.3.1 void P< T >.Bind (Action< T > *target*, bool *immediate* = `true`)

Bind the specified target to this property.

Parameters

target	Target.

Template Parameters

TBindingType	The 1st type parameter.

10.50.3.2 virtual bool P< T >.CanSetValue (T *value*) [virtual]

10.50.3.3 override void P< T >.Deserialize (JSONNode *node*) [virtual]

Deserialize the specified node into `Value`.

Parameters

node	Node.

Implements ModelPropertyBase.

10.50.3.4 override bool P< T >.Equals (object *obj*)

10.50.3.5 override int P< T >.GetHashCode ( )

10.50.3.6 override JSONNode P< T >.Serialize ( ) [virtual]

Serializes this object

Implements ModelPropertyBase.

10.50.4 Property Documentation

10.50.4.1 T P< T >.Value [get], [set]

Gets or sets the value.

The value.

10.50.4.2 override Type P< T >.ValueType [get]

Gets the type of the value.

The type of the value.

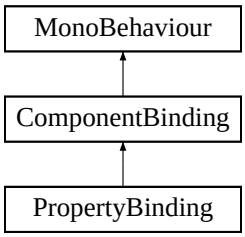
The documentation for this class was generated from the following file:

- Scripts/Base/ViewModels/P.cs

10.51 PropertyBinding Class Reference

A component for a property binding. A component property binding will use reflection to pull the member information so if performance is an issue I would recommend a code only binding.

Inheritance diagram for PropertyBinding:



Public Attributes

- Component _TargetComponent
- List< string > _TargetProperties = new List<string>()
- bool _TwoWay = false

Protected Member Functions

- override IBinding GetBinding ()

  The binding provider. Create the binding that the component will add to the source view here.

Protected Attributes

- MemberInfo _targetPropertyInfo
- object _targetPropertyObject

Properties

- BindableProperty TargetProperty [get]

Additional Inherited Members

10.51.1 Detailed Description

A component for a property binding. A component property binding will use reflection to pull the member information so if performance is an issue I would recommend a code only binding.

10.51.2 Member Function Documentation

10.51.2.1 override IBinding PropertyBinding.GetBinding ( ) [protected], [virtual]

The binding provider. Create the binding that the component will add to the source view here.

Returns

The binding that will be added to the source view.

Implements ComponentBinding.

10.51.3 Member Data Documentation

10.51.3.1 Component PropertyBinding._TargetComponent

10.51.3.2 List<string> PropertyBinding._TargetProperties = new List<string>()

10.51.3.3 MemberInfo PropertyBinding._targetPropertyInfo [protected]

10.51.3.4 object PropertyBinding._targetPropertyObject [protected]

10.51.3.5 bool PropertyBinding._TwoWay = false

10.51.4 Property Documentation

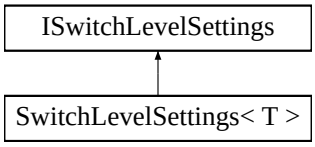
10.51.4.1 BindableProperty PropertyBinding.TargetProperty [get]

The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/PropertyBinding.cs

10.52 SwitchLevelSettings< T > Class Template Reference

Inheritance diagram for SwitchLevelSettings< T >:



Public Member Functions

- SwitchLevelSettings ()
- SwitchLevelSettings (Action< T > setup)

Properties

- string[] Levels [get, set]
- Action< LevelLoadProgress > ProgressUpdated [get, set]
- Action< T > Setup [get, set]
- Type StartControllerType [get]

10.52.1 Constructor & Destructor Documentation

10.52.1.1 SwitchLevelSettings< T >.SwitchLevelSettings ( )

10.52.1.2 SwitchLevelSettings< T >.SwitchLevelSettings (Action< T > *setup*)

10.52.2 Property Documentation

10.52.2.1 string [] SwitchLevelSettings< T >.Levels [get],[set]

10.52.2.2 Action<LevelLoadProgress> SwitchLevelSettings< T >.ProgressUpdated [get],[set]

10.52.2.3 Action<T> SwitchLevelSettings< T >.Setup [get],[set]

10.52.2.4 Type SwitchLevelSettings< T >.StartControllerType [get]

The documentation for this class was generated from the following file:

- Scripts/Common/SwitchLevelSettings.cs

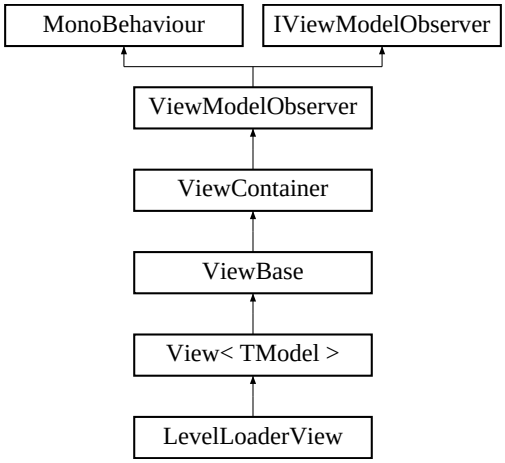
10.53 View< TModel > Class Template Reference

IOS: On ios this class can be used but if u have a class definition like "public class MyView\T\ : View\T\" it will not work because of mono ios compiler limitations.

Workaround: create a subclass from ViewBase and implement its methods

A view class which attaches as a component directly to a game object. The responsibility of this view is to bind a data model 'TModel' to the game object

Inheritance diagram for View< TModel >:



Protected Member Functions

- sealed override void InitializeModel (ViewModel model)

  This method should be overriden to Initialize the ViewModel with any options specified in a unity component inspector.
- virtual void InitializeModel (TModel viewViewModel)

Properties

- TModel Model `[get, set]`

  Gets or sets the ViewModel. Note: The setter will reinvoke the bind method. To set quietly use ViewModelObject
- override Type ViewModelType `[get]`

Additional Inherited Members

10.53.1 Detailed Description

IOS: On ios this class can be used but if u have a class definition like "public class MyView\T\ : View\T\" it will not work because of mono ios compiler limitations.

Workaround: create a subclass from ViewBase and implement its methods

A view class which attaches as a component directly to a game object. The responsibility of this view is to bind a data model 'TModel' to the game object

Template Parameters

TModel	The ViewModel Type

Type Constraints

TModel* : *ViewModel

TModel* : *new()

10.53.2 Member Function Documentation

10.53.2.1 sealed override void View< TModel >.InitializeModel (ViewModel *model*) [protected],[virtual]

This method should be overriden to Initialize the ViewModel with any options specified in a unity component inspector.

Parameters

model	The model to initialize.

Implements ViewBase.

10.53.2.2 virtual void View< TModel >.InitializeModel (TModel *viewViewModel*) [protected],[virtual]

10.53.3 Property Documentation

10.53.3.1 TModel View< TModel >.Model [get],[set]

Gets or sets the ViewModel. Note: The setter will reinvoke the bind method. To set quietly use ViewModelObject

10.53.3.2 override Type View< TModel >.ViewModelType [get]

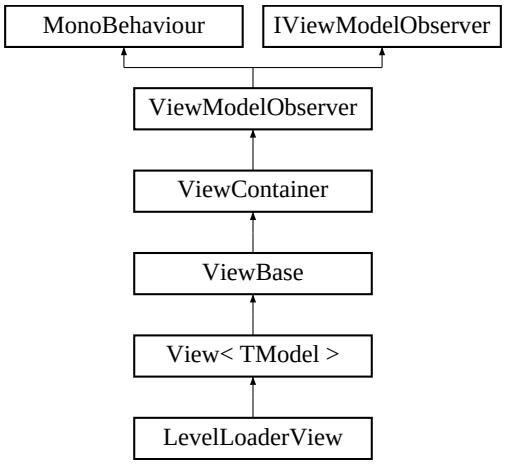
The documentation for this class was generated from the following file:

- Scripts/Base/Views/View.cs

10.54 ViewBase Class Reference

The base class for a View that binds to a ViewModel

Inheritance diagram for ViewBase:



Public Member Functions

- delegate void ViewEvent (string eventName)

  The View Event delegate that takes a string for the event name.
- virtual void Awake ()
- abstract void Bind ()
- virtual ViewModel CreateModel ()
- virtual void OnDestroy ()
- virtual void OnDisable ()
- virtual void OnEnable ()
- override void Unbind ()
- void SetupBindings ()

This method will setup all bindings on this view. Bindings don't actually occur on a view until this method is called. In the bind method it will simply add to the collection of bindings. You should never have to call this method manually.

- virtual void Start ()
- override void AddBinding (IBinding binding)

Public Attributes

- bool _LogEvents

  Should we log an event for each View event that occurs.
- ViewModelRegistryType _ViewModelFrom = ViewModelRegistryType.CreateNew

  Where should the viewmodel come from or how should it be instantiated.

Protected Member Functions

- virtual void Event (string eventname)

  Invoke a .NET event on this view. This is a convinience method for Event Bindings.
- abstract void InitializeModel (ViewModel model)

  This method should be overriden to Initialize the ViewModel with any options specified in a unity component inspector.
- virtual void LateUpdate ()

Properties

- IEnumerable< ViewModel > ChildViewModels `[get]`
- List< ViewBase > ChildViews `[get, set]`
- bool Instantiated `[get, set]`
- ViewBase ParentView `[get]`
- ViewModel ParentViewModel `[get]`
- virtual ViewModel ViewModelObject `[get, set]`
- abstract Type ViewModelType `[get]`
- string ViewName `[get, set]`

  The name of the prefab that created this view

Events

- ViewEvent EventTriggered

  An event that is invoked whe calling Event("MyEvent")

10.54.1 Detailed Description

The base class for a View that binds to a ViewModel

10.54.2 Member Function Documentation

10.54.2.1 override void ViewBase.AddBinding (IBinding *binding*) `[virtual]`

Reimplemented from ViewModelObserver.

10.54.2.2 virtual void ViewBase.Awake ( ) [virtual]

10.54.2.3 abstract void ViewBase.Bind ( ) [pure virtual]

Implemented in LevelLoaderView.

10.54.2.4 virtual ViewModel ViewBase.CreateModel ( ) [virtual]

Reimplemented in LevelLoaderView.

10.54.2.5 virtual void ViewBase.Event (string *eventname*) [protected],[virtual]

Invoke a .NET event on this view. This is a convinience method for Event Bindings.

Parameters

eventname	The name of the event that occured

10.54.2.6 abstract void ViewBase.InitializeModel (ViewModel *model*) [protected],[pure virtual]

This method should be overriden to Initialize the ViewModel with any options specified in a unity component inspector.

Parameters

model	The model to initialize.

Implemented in View< TModel >.

10.54.2.7 virtual void ViewBase.LateUpdate ( ) [protected],[virtual]

10.54.2.8 virtual void ViewBase.OnDestroy ( ) [virtual]

10.54.2.9 virtual void ViewBase.OnDisable ( ) [virtual]

10.54.2.10 virtual void ViewBase.OnEnable ( ) [virtual]

10.54.2.11 void ViewBase.SetupBindings ( )

This method will setup all bindings on this view. Bindings don't actually occur on a view until this method is called. In the bind method it will simply add to the collection of bindings. You should never have to call this method manually.

10.54.2.12 virtual void ViewBase.Start ( ) [virtual]

10.54.2.13 override void ViewBase.Unbind ( ) [virtual]

Reimplemented from ViewModelObserver.

10.54.2.14 delegate void ViewBase.ViewEvent (string *eventName*)

The View Event delegate that takes a string for the event name.

Parameters

eventName	The event that has occured.

10.54.3 Member Data Documentation

10.54.3.1 bool ViewBase._LogEvents

Should we log an event for each View event that occurs.

10.54.3.2 ViewModelRegistryType ViewBase._ViewModelFrom = ViewModelRegistryType.CreateNew

Where should the viewmodel come from or how should it be instantiated.

10.54.4 Property Documentation

10.54.4.1 IEnumerable<ViewModel> ViewBase.ChildViewModels [get]

10.54.4.2 List<ViewBase> ViewBase.ChildViews [get],[set]

10.54.4.3 bool ViewBase.Instantiated [get],[set]

10.54.4.4 ViewBase ViewBase.ParentView [get]

10.54.4.5 ViewModel ViewBase.ParentViewModel [get]

10.54.4.6 virtual ViewModel ViewBase.ViewModelObject [get],[set]

10.54.4.7 abstract Type ViewBase.ViewModelType [get]

10.54.4.8 string ViewBase.ViewName [get],[set]

The name of the prefab that created this view

10.54.5 Event Documentation

10.54.5.1 ViewEvent ViewBase.EventTriggered

An event that is invoked whe calling Event("MyEvent")

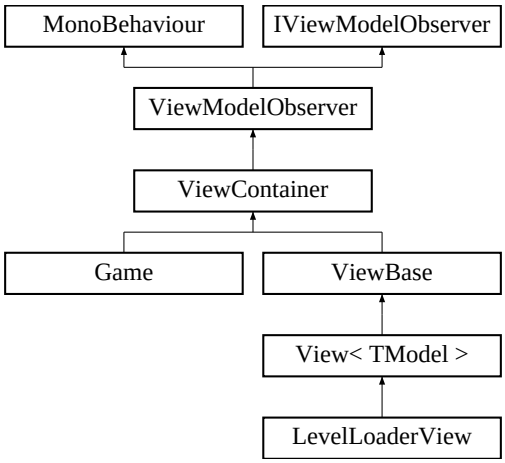
The documentation for this class was generated from the following file:

- Scripts/Base/Views/ViewBase.cs

10.55 ViewContainer Class Reference

A base class for all view containers. Simply just utility methods for views and events.

Inheritance diagram for ViewContainer:



Public Member Functions

- virtual TView CreateView< TView > ()
- virtual TView CreateView< TView > (ViewModel model)
- virtual TView CreateView< TView > (ViewModel model, Vector3 position)
- virtual TView CreateView< TView > (ViewModel model, Vector3 position, Quaternion rotation)
- ViewBase InstantiateView (ViewModel model)
- ViewBase InstantiateView (ViewModel model, Vector3 position)
- ViewBase InstantiateView (ViewModel model, Vector3 position, Quaternion rotation)
- ViewBase InstantiateView (GameObject prefab, ViewModel model)
- ViewBase InstantiateView (GameObject prefab, ViewModel model, Vector3 position)
- ViewBase InstantiateView (string viewName)
- ViewBase InstantiateView (string viewName, ViewModel model)

  Instantiates a view.
- ViewBase InstantiateView (string viewName, ViewModel model, Vector3 position)

  Instantiates a view.
- ViewBase InstantiateView (string viewName, ViewModel model, Vector3 position, Quaternion rotation)

  Instantiates a view.
- ViewBase InstantiateView (GameObject prefab, ViewModel model, Vector3 position, Quaternion rotation)

  Instantiates a view.
- Coroutine LoadAdditive (string rootObjectName, string levelName, Action< GameObject > complete=null)
- Coroutine Task (Func< IEnumerator > coroutine)

Additional Inherited Members

10.55.1 Detailed Description

A base class for all view containers. Simply just utility methods for views and events.

10.55.2 Member Function Documentation

10.55.2.1 virtual TView ViewContainer.CreateView< TView > ( ) [virtual]

Type Constraints

TView* : *ViewBase

10.55.2.2 virtual TView ViewContainer.CreateView< TView > (ViewModel *model*) [virtual]

Type Constraints

TView **:** ***ViewBase***

10.55.2.3 virtual TView ViewContainer.CreateView< TView > (ViewModel *model,* Vector3 *position*) [virtual]

Type Constraints

TView **:** ***ViewBase***

10.55.2.4 virtual TView ViewContainer.CreateView< TView > (ViewModel *model,* Vector3 *position,* Quaternion *rotation*) [virtual]

Type Constraints

TView **:** ***ViewBase***

10.55.2.5 ViewBase ViewContainer.InstantiateView (ViewModel *model*)

10.55.2.6 ViewBase ViewContainer.InstantiateView (ViewModel *model,* Vector3 *position*)

10.55.2.7 ViewBase ViewContainer.InstantiateView (ViewModel *model,* Vector3 *position,* Quaternion *rotation*)

10.55.2.8 ViewBase ViewContainer.InstantiateView (GameObject *prefab,* ViewModel *model*)

10.55.2.9 ViewBase ViewContainer.InstantiateView (GameObject *prefab,* ViewModel *model,* Vector3 *position*)

10.55.2.10 ViewBase ViewContainer.InstantiateView (string *viewName*)

10.55.2.11 ViewBase ViewContainer.InstantiateView (string *viewName,* ViewModel *model*)

Instantiates a view.

Parameters

viewName	The name of the prefab/view to instantiate
model	The model that will be passed to the view.

Returns

The instantiated view

10.55.2.12 ViewBase ViewContainer.InstantiateView (string *viewName,* ViewModel *model,* Vector3 *position*)

Instantiates a view.

Parameters

viewName	The name of the prefab/view to instantiate

model	The model that will be passed to the view.
position	The position to instantiate the view.

Returns

The instantiated view

10.55.2.13 ViewBase ViewContainer.InstantiateView (string *viewName*, ViewModel *model*, Vector3 *position*, Quaternion *rotation*)

Instantiates a view.

Parameters

viewName	The name of the prefab/view to instantiate
model	The model that will be passed to the view.
position	The position to instantiate the view.
rotation	The rotation to instantiate the view with.

Returns

The instantiated view

10.55.2.14 ViewBase ViewContainer.InstantiateView (GameObject *prefab*, ViewModel *model*, Vector3 *position*, Quaternion *rotation*)

Instantiates a view.

Parameters

prefab	The prefab/view to instantiate
model	The model that will be passed to the view.
position	The position to instantiate the view.
rotation	The rotation to instantiate the view with.

Returns

The instantiated view

10.55.2.15 Coroutine ViewContainer.LoadAdditive (string *rootObjectName*, string *levelName*, Action< GameObject > *complete* = `null`)

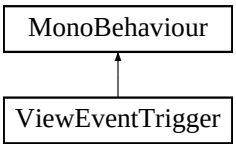
10.55.2.16 Coroutine ViewContainer.Task (Func< IEnumerator > *coroutine*)

The documentation for this class was generated from the following file:

- Scripts/Base/ViewContainer.cs

10.56 ViewEventTrigger Class Reference

Inheritance diagram for ViewEventTrigger:



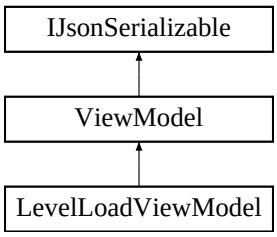
The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ViewEventTrigger.cs

10.57 ViewModel Class Reference

A data structure that contains information/data needed for a 'View'

Inheritance diagram for ViewModel:



Public Member Functions

- virtual void Deserialize (JSONNode node)
- ICommand ForwardThisTo< T > (ICommand< T > command)
- ICommand ForwardThisTo< T > (Func< ICommand< T >> commandSelector)
- virtual IEnumerable
  < ModelPropertyBase > GetProperties ()

  Override this method to skip using reflection. This can drastically improve performance especially IOS
- virtual JSONNode Serialize ()
- override string ToString ()

Static Public Member Functions

- static Dictionary< string,
  PropertyInfo > GetReflectedCommands (Type modelType)
- static Dictionary< string,
  FieldInfo > GetReflectedModelProperties (Type modelType)

Protected Member Functions

- ICommand Command (Action command)
- ICommand Command (Func< IEnumerator > command)

Properties

- Dictionary< string, ICommand > Commands [get]
- ModelPropertyBase this[string bindingPropertyName] [get]

  Access a model property via string. This is optimized using a compiled delegate to access derived classes properties so use as needed

10.57.1 Detailed Description

A data structure that contains information/data needed for a 'View'

10.57.2 Member Function Documentation

10.57.2.1 ICommand ViewModel.Command (Action *command*) [protected]

10.57.2.2 ICommand ViewModel.Command (Func< IEnumerator > *command*) [protected]

10.57.2.3 virtual void ViewModel.Deserialize (JSONNode *node*) [virtual]

Implements IJsonSerializable.

10.57.2.4 ICommand ViewModel.ForwardThisTo< T > (ICommand< T > *command*)

Type Constraints

T* : *ViewModel

10.57.2.5 ICommand ViewModel.ForwardThisTo< T > (Func< ICommand< T >> *commandSelector*)

Type Constraints

T* : *ViewModel

10.57.2.6 virtual IEnumerable<ModelPropertyBase> ViewModel.GetProperties ( ) [virtual]

Override this method to skip using reflection. This can drastically improve performance especially IOS

Returns

10.57.2.7 static Dictionary<string, PropertyInfo> ViewModel.GetReflectedCommands (Type *modelType*) [static]

Parameters

modelType	

Returns

10.57.2.8 static Dictionary<string, FieldInfo> ViewModel.GetReflectedModelProperties (Type *modelType*) [static]

10.57.2.9 virtual JSONNode ViewModel.Serialize ( ) [virtual]

Implements IJsonSerializable.

10.57.2.10 override string ViewModel.ToString ( )

10.57.3 Property Documentation

10.57.3.1 Dictionary<string, ICommand> ViewModel.Commands [get]

10.57.3.2 ModelPropertyBase ViewModel.this[string bindingPropertyName] [get]

Access a model property via string. This is optimized using a compiled delegate to access derived classes properties so use as needed

Parameters

bindingPropertyName	The name of the property/field to access

Returns

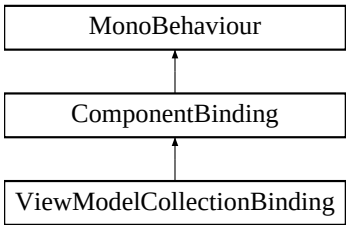
ModelPropertyBase The Model Property class. Use value to get the value of the property

The documentation for this class was generated from the following file:

- Scripts/Base/Views/ViewModel.cs

10.58 ViewModelCollectionBinding Class Reference

Inheritance diagram for ViewModelCollectionBinding:



Public Attributes

- bool _Immediate
- Transform _Parent
- Component _TargetComponent
- string _ViewName

Protected Member Functions

- override IBinding GetBinding ()

  The binding provider. Create the binding that the component will add to the source view here.

Additional Inherited Members

10.58.1 Member Function Documentation

10.58.1.1 override IBinding ViewModelCollectionBinding.GetBinding ( ) [protected],[virtual]

The binding provider. Create the binding that the component will add to the source view here.

Returns

The binding that will be added to the source view.

Implements ComponentBinding.

10.58.2 Member Data Documentation

10.58.2.1 bool ViewModelCollectionBinding._Immediate

10.58.2.2 Transform ViewModelCollectionBinding._Parent

10.58.2.3 Component ViewModelCollectionBinding._TargetComponent

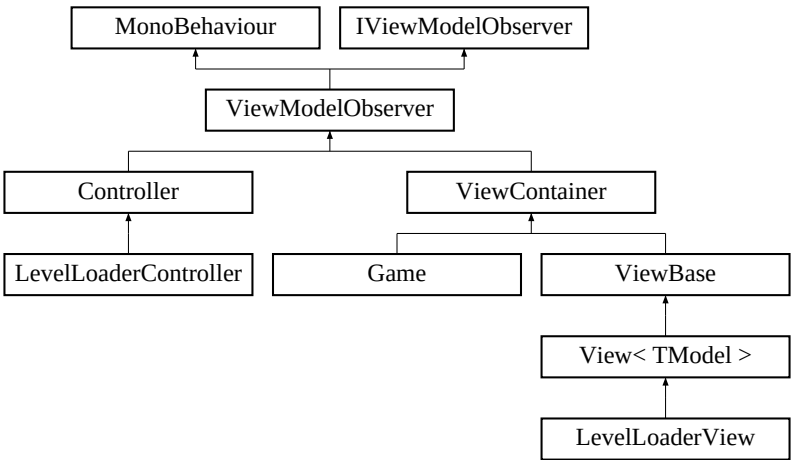
10.58.2.4 string ViewModelCollectionBinding._ViewName

The documentation for this class was generated from the following file:

- Scripts/Base/Bindings/ViewModelCollectionBinding.cs

10.59 ViewModelObserver Class Reference

Inheritance diagram for ViewModelObserver:



Public Member Functions

- virtual void AddBinding (IBinding binding)
- void ExecuteCommand (ICommand command)
- virtual void RemoveBinding (IBinding binding)
- virtual void Unbind ()

Properties

- List< IBinding > Bindings [get, set]

  The bindings that are attached to this ViewModel

10.59.1 Member Function Documentation

10.59.1.1 virtual void ViewModelObserver.AddBinding (IBinding *binding*) [virtual]

Implements IViewModelObserver.

Reimplemented in ViewBase, and Controller.

10.59.1.2 void ViewModelObserver.ExecuteCommand (ICommand *command*)

Implements IViewModelObserver.

10.59.1.3 virtual void ViewModelObserver.RemoveBinding (IBinding *binding*) [virtual]

Implements IViewModelObserver.

10.59.1.4 virtual void ViewModelObserver.Unbind ( ) [virtual]

Implements IViewModelObserver.

Reimplemented in ViewBase.

10.59.2 Property Documentation

10.59.2.1 List<IBinding> ViewModelObserver.Bindings [get],[set]

The bindings that are attached to this ViewModel

The documentation for this class was generated from the following file:

- Scripts/Base/Views/ViewBase.cs

10.60 ViewResolver Class Reference

The View Managers responsibility is to provide prefabes based off of a view model This implementation finds a prefab based off of the ViewModel's type name removing "View" from it.

Public Member Functions

- virtual GameObject FindView (ViewModel model)

  Provides a prefab
- virtual GameObject FindView (string viewName)

  Provides a prefab based off a viewname

10.60.1 Detailed Description

The View Managers responsibility is to provide prefabes based off of a view model This implementation finds a prefab based off of the ViewModel's type name removing "View" from it.

10.60.2 Member Function Documentation

10.60.2.1 virtual GameObject ViewResolver.FindView (ViewModel *model*) [virtual]

Provides a prefab

Parameters

model	The model for the view prefab we are looking for

Returns

10.60.2.2 virtual GameObject ViewResolver.FindView (string *viewName*) [virtual]

Provides a prefab based off a viewname

Parameters

viewName	The name of the view prefab we are looking for

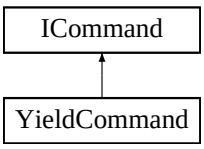
Returns

The documentation for this class was generated from the following file:

- Scripts/Base/Views/ViewResolver.cs

10.61 YieldCommand Class Reference

Inheritance diagram for YieldCommand:



Public Member Functions

- YieldCommand (Func< IEnumerator > enumeratorDelegate)
- IEnumerator Execute ()

Protected Member Functions

- virtual void OnOnCommandComplete ()
- virtual void OnOnCommandExecuting ()

Properties

- Func< IEnumerator > EnumeratorDelegate [get, set]

Events

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

10.61.1 Constructor & Destructor Documentation

10.61.1.1 YieldCommand.YieldCommand (Func< IEnumerator > *enumeratorDelegate*)

10.61.2 Member Function Documentation

10.61.2.1 IEnumerator YieldCommand.Execute ( )

Implements ICommand.

10.61.2.2 virtual void YieldCommand.OnOnCommandComplete ( ) [protected],[virtual]

10.61.2.3 virtual void YieldCommand.OnOnCommandExecuting ( ) [protected],[virtual]

10.61.3 Property Documentation

10.61.3.1 Func<IEnumerator> YieldCommand.EnumeratorDelegate [get],[set],[protected]

10.61.4 Event Documentation

10.61.4.1 CommandEvent YieldCommand.OnCommandExecuted

10.61.4.2 CommandEvent YieldCommand.OnCommandExecuting

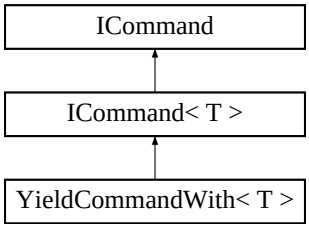
The documentation for this class was generated from the following file:

- Scripts/Base/Commands/Command.cs

10.62 YieldCommandWith< T > Class Template Reference

A coroutine command with a parameter.

Inheritance diagram for YieldCommandWith< T >:



Public Member Functions

- YieldCommandWith (Func< T, IEnumerator > enumeratorDelegate)
- YieldCommandWith (T parameter, Func< T, IEnumerator > enumeratorDelegate)
- IEnumerator Execute ()

Protected Member Functions

- virtual void OnOnCommandComplete ()
- virtual void OnOnCommandExecuting ()

Properties

- T Parameter `[get, set]`
- Func< T, IEnumerator > EnumeratorDelegate `[get, set]`

Events

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

10.62.1 Detailed Description

A coroutine command with a parameter.

Template Parameters

T	

10.62.2 Constructor & Destructor Documentation

10.62.2.1 YieldCommandWith< T >.YieldCommandWith (Func< T, IEnumerator > *enumeratorDelegate*)

10.62.2.2 YieldCommandWith< T >.YieldCommandWith (T *parameter,* Func< T, IEnumerator > *enumeratorDelegate*)

10.62.3 Member Function Documentation

10.62.3.1 IEnumerator YieldCommandWith< T >.Execute ( )

Implements ICommand.

10.62.3.2 virtual void YieldCommandWith< T >.OnOnCommandComplete ( ) `[protected]`, `[virtual]`

10.62.3.3 virtual void YieldCommandWith< T >.OnOnCommandExecuting ( ) `[protected]`, `[virtual]`

10.62.4 Property Documentation

10.62.4.1 Func<T, IEnumerator> YieldCommandWith< T >.EnumeratorDelegate `[get]`, `[set]`, `[protected]`

10.62.4.2 T YieldCommandWith< T >.Parameter `[get]`, `[set]`

10.62.5 Event Documentation

10.62.5.1 CommandEvent YieldCommandWith< T >.OnCommandExecuted

10.62.5.2 CommandEvent YieldCommandWith< T >.OnCommandExecuting

The documentation for this class was generated from the following file:

- Scripts/Base/Commands/YieldCommandWith.cs

Chapter 11

File Documentation

11.1 Scripts/Base/Bindings/BindableProperty.cs File Reference

Classes

- class BindableProperty

  A bindable property that can be easily wired for binding.

11.2 Scripts/Base/Bindings/Binding.cs File Reference

Classes

- class Binding

  The base class for all bindings.

11.3 Scripts/Base/Bindings/CollectionBindings.cs File Reference

Classes

- class **CollectionBindings**

11.4 Scripts/Base/Bindings/CollisionBindings.cs File Reference

Classes

- class **CollisionBindings**

11.5 Scripts/Base/Bindings/CollisionEventBinding.cs File Reference

Classes

- class CollisionEventBinding

  A component for binding to a collision.

11.6 Scripts/Base/Bindings/CollisionEventType.cs File Reference

Enumerations

- enum CollisionEventType { CollisionEventType.OnCollisionEnter, CollisionEventType.OnCollisionExit, CollisionEventType.OnCollisionStay, CollisionEventType.OnTriggerEnter, CollisionEventType.OnTriggerExit, CollisionEventType.OnTriggerStay }

11.6.1 Enumeration Type Documentation

11.6.1.1 enum CollisionEventType

Enumerator

OnCollisionEnter

OnCollisionExit

OnCollisionStay

OnTriggerEnter

OnTriggerExit

OnTriggerStay

11.7 Scripts/Base/Bindings/CommandBinding.cs File Reference

Classes

- class CommandBinding

  Base class for a command binding. Use this class if a different type of command binding is needed.

11.8 Scripts/Base/Bindings/ComponentBinding.cs File Reference

Classes

- class ComponentBinding

  A Unity3d Component that will provide a binding to a specified View

11.9 Scripts/Base/Bindings/ComponentCommandBinding.cs File Reference

Classes

- class ComponentCommandBinding

  A component that will create a command binding and requires a component for the command to work.

11.10 Scripts/Base/Bindings/EventBinding.cs File Reference

Classes

- class EventBinding

  The event binding component that will add an event binding to a source view.

11.11 Scripts/Base/Bindings/IBinding.cs File Reference

Classes

- interface IBinding

  Interface for all bindings

11.12 Scripts/Base/Bindings/ITwoWayBinding.cs File Reference

Classes

- interface ITwoWayBinding

11.13 Scripts/Base/Bindings/IViewModelObserver.cs File Reference

Classes

- interface IViewModelObserver

  Potential future use.

11.14 Scripts/Base/Bindings/KeyBinding.cs File Reference

Classes

- class KeyBinding

  A component that will process a key binding as well as provide a key binding instance to the source view. Note. Even when adding this binding via code the component will still be added because a component is needed to process a keypress

Enumerations

- enum KeyBindingEventType { KeyBindingEventType.Key, KeyBindingEventType.KeyDown, KeyBindingEventType.KeyUp }

11.14.1 Enumeration Type Documentation

11.14.1.1 enum KeyBindingEventType

Enumerator

Key

KeyDown

KeyUp

11.15 Scripts/Base/Bindings/ModelCollisionEventBinding.cs File Reference

Classes

- class ModelCollisionEventBinding

  A collision binding that will trigger a command when executed. Use chaining when possible to provide additional options for this binding.

11.16 Scripts/Base/Bindings/ModelCommandBinding.cs File Reference

Classes

- class ModelCommandBinding

  A base class for binding to a ViewModel command.

11.17 Scripts/Base/Bindings/ModelEventBinding.cs File Reference

Classes

- class ModelEventBinding

  An event binding. Basically a wrapper for a .NET event so events can be triggered by a string. They can easily be bound and is mainly for conveniance.

11.18 Scripts/Base/Bindings/ModelKeyBinding.cs File Reference

Classes

- class ModelKeyBinding

  Binds a key to a ViewModel command.

11.19 Scripts/Base/Bindings/ModelMouseEventBinding.cs File Reference

Classes

- class ModelInputButtonBinding
- class ModelMouseEventBinding

Enumerations

- enum InputButtonEventType { InputButtonEventType.Button, InputButtonEventType.ButtonDown, InputButtonEventType.ButtonUp }

11.19.1 Enumeration Type Documentation

11.19.1.1 enum InputButtonEventType

Enumerator

Button

ButtonDown

ButtonUp

11.20 Scripts/Base/Bindings/ModelPropertyBinding.cs File Reference

Classes

- class ModelPropertyBinding

  A class that contains a binding from a ViewModel to a Target

11.21 Scripts/Base/Bindings/ModelViewModelCollectionBinding.cs File Reference

Classes

- class ModelCollectionBinding< TCollectionType >
- class ModelViewModelCollectionBinding

  Class for a view collection binding. Binds a ViewModel collection to a set of corresponding Views

Typedefs

- using Object = UnityEngine.Object

11.21.1 Typedef Documentation

11.21.1.1 using Object = UnityEngine.Object

11.22 Scripts/Base/Bindings/MouseEventBinding.cs File Reference

Classes

- class InputBinding
- class MouseEventBinding

11.23 Scripts/Base/Bindings/MouseEventType.cs File Reference

Enumerations

- enum MouseEventType { MouseEventType.OnBecameInvisible, MouseEventType.OnBecameVisible, MouseEventType.OnMouseDown, MouseEventType.OnMouseDrag, MouseEventType.OnMouseEnter, MouseEventType.OnMouseExit, MouseEventType.OnMouseOver, MouseEventType.OnMouseUp, MouseEventType.OnMouseUpAsButton }

A Unity mouse event. The comments are from the unity documentation.

11.23.1 Enumeration Type Documentation

11.23.1.1 enum MouseEventType

A Unity mouse event. The comments are from the unity documentation.

Enumerator

OnBecameInvisible

OnBecameVisible

OnMouseDown

OnMouseDrag

OnMouseEnter

OnMouseExit

OnMouseOver

OnMouseUp

OnMouseUpAsButton

11.24 Scripts/Base/Bindings/PropertyBinding.cs File Reference

Classes

- class PropertyBinding

  A component for a property binding. A component property binding will use reflection to pull the member information so if performance is an issue I would recommend a code only binding.

11.25 Scripts/Base/Bindings/PropertyBindings.cs File Reference

Classes

- class **PropertyBindings**

11.26 Scripts/Base/Bindings/ViewBindings.cs File Reference

Classes

- class **ViewBindings**

  Binding extension method that make it easy to bind ViewModels to Views

Typedefs

- using Object = UnityEngine.Object

11.26.1 Typedef Documentation

11.26.1.1 using Object = UnityEngine.Object

11.27 Scripts/Base/Bindings/ViewEventTrigger.cs File Reference

Classes

- class ViewEventTrigger

11.28 Scripts/Base/Bindings/ViewModelCollectionBinding.cs File Reference

Classes

- class ViewModelCollectionBinding

11.29 Scripts/Base/Commands/Command.cs File Reference

Classes

- class Command

  A ViewModel command that can be executed. IEnumerator is always used so that any command can be a coroutine.
- class YieldCommand

11.30 Scripts/Base/Commands/CommandWith.cs File Reference

Classes

- class CommandWith< T >

  A command with an argument of type T. Not usually bound to directly but used to forward a command to a parent viewmodel

11.31 Scripts/Base/Commands/ControllerActionCommand.cs File Reference

11.32 Scripts/Base/Commands/DelegateCommand.cs File Reference

11.33 Scripts/Base/Commands/GameEventCommand.cs File Reference

11.34 Scripts/Base/Commands/ICommand.cs File Reference

Classes

- interface ICommand

  The base command interface for implementing a command in a ViewModel
- interface ICommand< T >

  A base command interface for implementing a command with a parameter in a ViewModel

Functions

- delegate void CommandEvent ()

11.34.1 Function Documentation

11.34.1.1 delegate void CommandEvent ( )

11.35 Scripts/Base/Commands/YieldCommandWith.cs File Reference

Classes

- class YieldCommandWith< T >

  A coroutine command with a parameter.

11.36 Scripts/Base/Controllers/Controller.cs File Reference

Classes

- class Controller

  A controller is a integral part of uFrame and is used for an extra layer connecting services and "Elements" of a game together. A controller also provides the creation of a ViewModel and bind to command to provide additional functionality.

11.37 Scripts/Base/Controllers/Game.cs File Reference

Classes

- class Game

  The main entry point for a game that is managed and accessible via GameManager. Only one will available at a time. This class when derived form should setup the container and load anything needed to properly run a game. This could include ViewModel Registering in the Container, Instantiating Views, Instantiating or Initializing Controllers.

Typedefs

- using Object = UnityEngine.Object

11.37.1 Typedef Documentation

11.37.1.1 using Object = UnityEngine.Object

11.38 Scripts/Base/Controllers/GameContainer.cs File Reference

Classes

- class GameContainer

  A ViewModel Container and a factory for Controllers and commands.

11.39 Scripts/Base/Controllers/GameManager.cs File Reference

Classes

- class GameManager

  A singleton that manages our current game and all the games in the scene. This component will persist through every level

11.40 Scripts/Documentation/GameManager.cs File Reference

11.41 Scripts/Base/Controllers/IGameContainer.cs File Reference

Classes

- interface IGameContainer

11.42 Scripts/Base/Controllers/InjectAttribute.cs File Reference

Classes

- class InjectAttribute

11.43 Scripts/Base/IJsonSerializable.cs File Reference

Classes

- interface IJsonSerializable

11.44 Scripts/Base/SimpleJSON.cs File Reference

Classes

- class **SimpleJSON.JSON**
- class SimpleJSON.JSONArray
- class SimpleJSON.JSONClass
- class SimpleJSON.JSONData
- class SimpleJSON.JSONLazyCreator
- class SimpleJSON.JSONNode

Namespaces

- package SimpleJSON

Enumerations

- enum SimpleJSON.JSONBinaryTag {
  SimpleJSON.JSONBinaryTag.Array = 1, SimpleJSON.JSONBinaryTag.Class = 2, SimpleJSON.JSONBinaryTag.Value = 3, SimpleJSON.JSONBinaryTag.IntValue = 4,
  SimpleJSON.JSONBinaryTag.DoubleValue = 5, SimpleJSON.JSONBinaryTag.BoolValue = 6, SimpleJSON.JSONBinaryTag.FloatValue = 7 }

11.45 Scripts/Base/UFrame.cs File Reference

Classes

- class **UFrame**

  The uFrame static factory class for overriding/customizing core uFrame functionality if needed

11.46 Scripts/Base/ViewContainer.cs File Reference

Classes

- class ViewContainer

  A base class for all view containers. Simply just utility methods for views and events.

11.47 Scripts/Base/ViewModels/ModelCollection.cs File Reference

Classes

- interface IModelCollection
- class ModelCollection< T >

  An observable collection to use in viewmodels.
- class ModelCollectionChangeEvent
- class ModelCollectionChangeEventWith< T >

Enumerations

- enum ModelCollectionAction {
  ModelCollectionAction.Add, ModelCollectionAction.Remove, ModelCollectionAction.Move, ModelCollectionAction.Replace,
  ModelCollectionAction.Reset }

Functions

- delegate void ModelCollectionChanged (ModelCollectionChangeEvent changeArgs)

11.47.1 Enumeration Type Documentation

11.47.1.1 enum ModelCollectionAction

Enumerator

Add

Remove

Move

Replace

Reset

11.47.2 Function Documentation

11.47.2.1 delegate void ModelCollectionChanged (ModelCollectionChangeEvent *changeArgs*)

11.48 Scripts/Base/ViewModels/ModelPropertyBase.cs File Reference

Classes

- class ModelPropertyBase

  A base class for model properties.

11.49 Scripts/Base/ViewModels/P.cs File Reference

Classes

- class P< T >

  A typed ViewModel Property Class

11.50 Scripts/Base/Views/IView.cs File Reference

Classes

- interface IView

11.51 Scripts/Base/Views/View.cs File Reference

Classes

- class View< TModel >

  IOS: On ios this class can be used but if u have a class definition like "public class MyView\T\ : View\T\" it will not work because of mono ios compiler limitations.

  Workaround: create a subclass from ViewBase and implement its methods

  A view class which attaches as a component directly to a game object. The responsibility of this view is to bind a data model 'TModel' to the game object

Enumerations

- enum ViewModelRegistryType { ViewModelRegistryType.CreateNew, ViewModelRegistryType.GameContainer }

11.51.1 Enumeration Type Documentation

11.51.1.1 enum ViewModelRegistryType

Enumerator

CreateNew

GameContainer

11.52 Scripts/Base/Views/ViewBase.cs File Reference

Classes

- class ViewBase

  The base class for a View that binds to a ViewModel
- class ViewModelObserver

11.53 Scripts/Base/Views/ViewExtensions.cs File Reference

Classes

- class **ViewExtensions**

Typedefs

- using Object = UnityEngine.Object

11.53.1 Typedef Documentation

11.53.1.1 using Object = UnityEngine.Object

11.54 Scripts/Base/Views/ViewModel.cs File Reference

Classes

- class ViewModel

  A data structure that contains information/data needed for a 'View'

11.55 Scripts/Base/Views/ViewResolver.cs File Reference

Classes

- class ViewResolver

  The View Managers responsibility is to provide prefabes based off of a view model This implementation finds a prefab based off of the ViewModel's type name removing "View" from it.

11.56 Scripts/Common/ISwitchLevelSettings.cs File Reference

Classes

- interface ISwitchLevelSettings

11.57 Scripts/Common/LevelLoaderController.cs File Reference

Classes

- class LevelLoaderController

  A [u]Frame built-in controller to manage loading a level via GameManager Add this in a level-loading scene along with LevelLoadViewModel and a LevelLoaderView.

Typedefs

- using Object = UnityEngine.Object

Functions

- delegate void UpdateProgressDelegate (string message, float progress)

11.57.1 Typedef Documentation

11.57.1.1 using Object = UnityEngine.Object

11.57.2 Function Documentation

11.57.2.1 delegate void UpdateProgressDelegate (string *message*, float *progress*)

11.58 Scripts/Common/LevelLoaderView.cs File Reference

Classes

- class LevelLoaderView

11.59 Scripts/Common/LevelLoadProgress.cs File Reference

Classes

- struct LevelLoadProgress

  A struct for passing a message and a progress indicator

11.60 Scripts/Common/LevelLoadViewModel.cs File Reference

Classes

- class LevelLoadViewModel

  The view model that is used when a level/scene is loading.

11.61 Scripts/Common/MvcExtensions.cs File Reference

Classes

- class **MvcExtensions**

11.62 Scripts/Common/SwitchLevelSettings.cs File Reference

Classes

- class SwitchLevelSettings< T >

11.63 Scripts/Documentation/Controllers.cs File Reference

11.64 Scripts/Documentation/Games.cs File Reference

11.65 Scripts/Documentation/GettingStarted.cs File Reference

11.66 Scripts/Documentation/Models.cs File Reference

11.67 Scripts/Documentation/Overview.cs File Reference

11.68 Scripts/Documentation/QuickStart.cs File Reference

11.69 Scripts/Documentation/Serialization.cs File Reference

11.70 Scripts/Documentation/Views.cs File Reference

Index