# uFrame API Docs

1.06

Generated by Doxygen 1.8.6

Mon Jun 2 2014 12:52:15

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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

# Chapter 2

# Class Index

## 2.1 Class List

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

# Chapter 3

# Class Documentation

## 3.1 BindableProperty Class Reference

A wrapper for any class property so it can easily be bound to.

### 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]`

  *Get the value of the property*

### 3.1.1 Detailed Description

A wrapper for any class property so it can easily be bound to.

### 3.1.2 Property Documentation

#### 3.1.2.1 object BindableProperty.Value `[get],[set]`

Get the value of the property

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

- Assets/uFrameComplete/uFrame/Base/Bindings/BindableProperty.cs

## 3.2 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 (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).*
- ViewBase Source `[get, set]`

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

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

  *Is this a two-way binding.*

## 3.2.1 Detailed Description

The base class for all bindings.

## 3.2.2 Constructor & Destructor Documentation

### 3.2.2.1 Binding.Binding ( ViewBase *sourceView*, string *modelMemberName* ) `[protected]`

Constructor

**Parameters**

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

### 3.2.3 Member Function Documentation

#### 3.2.3.1 virtual void Binding.Bind ( ) [virtual]

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

Implements IBinding.

Reimplemented in ModelViewModelCollectionBinding, ModelViewPropertyBinding, CommandBinding, ModelCollectionBinding< TCollectionType >, ModelEventBinding, ModelPropertyBinding, and ModelCommandBinding.

#### 3.2.3.2 virtual void Binding.Unbind ( ) [virtual]

Unbind this binding

Implements IBinding.

Reimplemented in ModelViewModelCollectionBinding, CommandBinding, ModelViewPropertyBinding, ModelCollectionBinding< TCollectionType >, ModelPropertyBinding, ModelEventBinding, and ModelCommandBinding.

### 3.2.4 Property Documentation

#### 3.2.4.1 bool Binding.CanTwoWayBind [get]

Does this instance type implement ITwoWayBinding?

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

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

#### 3.2.4.3 bool Binding.IsComponent [get], [set]

Was this loaded from a component in the Unity Inspector?

#### 3.2.4.4 string Binding.ModelMemberName [get], [set]

The source ViewModel member name that is being bound to.

#### 3.2.4.5 ModelPropertyBase Binding.ModelProperty [get], [set]

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

#### 3.2.4.6 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.

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

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

#### 3.2.4.8 ViewBase Binding.Source [get],[set]

The owner view that this Binding belongs to

#### 3.2.4.9 object Binding.SourceValue [get]

The value of the ViewModel Member

#### 3.2.4.10 bool Binding.TwoWay [get],[set]

Is this a two-way binding.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/Binding.cs

## 3.3 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

### 3.3.1 Detailed Description

A component for binding to a collision.

### 3.3.2 Member Function Documentation

#### 3.3.2.1 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.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/CollisionEventBinding.cs

## 3.4 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

- object **Sender** `[get, set]`
- object **Parameter** `[get, set]`
- Action **Delegate** `[get, set]`

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

### 3.4.1 Detailed Description

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

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

- Assets/uFrameComplete/uFrame/Base/Commands/Command.cs

## 3.5 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** ()
- virtual object **GetArgument** ()
- CommandBinding **SetParameter** (object value)
- CommandBinding **SetParameterSelector** (Func< object > commandArgSelector)
- CommandBinding **Subscribe** (Action execute, bool before=false)
- CommandBinding **Throttle** (float seconds)
- override void Unbind ()

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

### Protected Attributes

- readonly List< Action > **_UnbindActions** = new List<Action>()

### Properties

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

### Additional Inherited Members

#### 3.5.1 Detailed Description

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

#### 3.5.2 Member Function Documentation

##### 3.5.2.1 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.

##### 3.5.2.2 override void CommandBinding.Unbind ( ) `[virtual]`

Unbind this binding

Reimplemented from Binding.

Reimplemented in ModelEventBinding, and ModelCommandBinding.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/CommandBinding.cs

## 3.6 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

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

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

### 3.6.1 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. |

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

- Assets/uFrameComplete/uFrame/Base/Commands/CommandWith.cs

## 3.7 CommandWithSender< TSender > Class Template Reference

Inheritance diagram for CommandWithSender< TSender >:

### Public Member Functions

- **CommandWithSender** (Action< TSender > @delegate)
- **CommandWithSender** (TSender sender, Action< TSender > @delegate)
- virtual IEnumerator **Execute** ()

### Protected Member Functions

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

### Properties

- object **Sender** `[get, set]`
- object **Parameter** `[get, set]`
- Action< TSender > **Delegate** `[get, set]`

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

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

- Assets/uFrameComplete/uFrame/Base/Commands/CommandWith.cs

## 3.8 CommandWithSenderAndArgument< TSender, TArgument > Class Template Reference

Inheritance diagram for CommandWithSenderAndArgument< TSender, TArgument >:

### Public Member Functions

- **CommandWithSenderAndArgument** (Action< TSender, TArgument > @delegate)
- **CommandWithSenderAndArgument** (TSender sender, Action< TSender, TArgument > @delegate)
- virtual IEnumerator **Execute** ()

### Protected Member Functions

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

### Properties

- object **Sender** `[get, set]`
- object **Parameter** `[get, set]`
- Action< TSender, TArgument > **Delegate** `[get, set]`

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

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

- Assets/uFrameComplete/uFrame/Base/Commands/CommandWith.cs

## 3.9 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**
- ViewBase **_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.*

### 3.9.1 Detailed Description

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

### 3.9.2 Member Function Documentation

#### 3.9.2.1 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**

#### 3.9.2.2 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, UFPropertyBinding, KeyBinding, CollisionEventBinding, ViewModelCollectionBinding, and EventBinding.

### 3.9.3 Property Documentation

#### 3.9.3.1 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:

- Assets/uFrameComplete/uFrame/Base/Bindings/ComponentBinding.cs

## 3.10 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

### 3.10.1 Detailed Description

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

### 3.10.2 Property Documentation

#### 3.10.2.1 ModelCommandBinding ComponentCommandBinding.CommandBinding [get]

Simply a wrapper of "Binding" property cast to ModelCommandBinding

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ComponentCommandBinding.cs

## 3.11 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.

Inheritance diagram for Controller:

### Public Member Functions

- void **AddBinding** (IBinding binding)
- void **RemoveBinding** (IBinding binding)
- void **Unbind** ()
- virtual void GameEvent (string message, params object[] additionalParamters)

  *Send an event to our game*
- virtual void **Setup** (IGameContainer container)
- ModelPropertyBinding **SubscribeToProperty**< **TViewModel, TBindingType** > (TViewModel source, P< TBindingType > sourceProperty, Action< TViewModel, TBindingType > changedAction)
- ModelPropertyBinding **SubscribeToProperty**< **TBindingType** > (P< TBindingType > sourceProperty, Action< TBindingType > targetSetter)
- Coroutine **StartCoroutine** (IEnumerator routine)
- void **StopCoroutine** (string name)
- void **StopAllCoroutines** ()
- void **ExecuteCommand** (ICommand command, object argument)
- virtual void **ExecuteCommand** (ICommand command)
- void **ExecuteCommand**< **TArgument** > (ICommandWith< TArgument > command, TArgument argument)
- abstract void **WireCommands** (ViewModel viewModel)
- virtual ViewModel **CreateEmpty** ()
- abstract void **Initialize** (ViewModel viewModel)
- virtual ViewModel **Create** ()
- virtual ViewModel **Create** (Action< ViewModel > preInitializer)
- TViewModel **Ensure**< **TViewModel** > ()
- TViewModel **EnsureByName**< **TViewModel** > (string instanceName)
- virtual ViewModel **GetByName** (string resolveName, bool initialize=true, Action< ViewModel > preInitializer=null)
- virtual ViewModel **GetByType** (Type viewModelType, bool initialize=true, Action< ViewModel > preInitializer=null)

### Protected Member Functions

- void **SubscribeToCommand** (ICommand command, Action action)
- virtual ViewModel **ResolveByName** (string resolveName)

### Properties

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

  *The friendly name of the controller. If this' type name ends with controller it will be removed.*
- List< IBinding > **Bindings** [get, set]

### 3.11.1 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.

### 3.11.2 Member Function Documentation

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

Send an event to our game

**Parameters**

| | |
|---|---|
| *message* | |
| *additional-Paramters* | |

### 3.11.3 Property Documentation

#### 3.11.3.1 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:

- Assets/uFrameComplete/uFrame/Base/Controllers/Controller.cs

## 3.12 DiagramInfoAttribute Class Reference

Inheritance diagram for DiagramInfoAttribute:

### Public Member Functions

- **DiagramInfoAttribute** (string diagramName)

### Properties

- string **DiagramName** [get, set]

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

- Assets/uFrameComplete/uFrame/Base/DiagramInfoAttribute.cs

## 3.13 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

#### 3.13.1 Detailed Description

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

#### 3.13.2 Member Function Documentation

##### 3.13.2.1 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.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/EventBinding.cs

## 3.14 GameContainer Class Reference

A ViewModel Container and a factory for Controllers and commands.

Inheritance diagram for GameContainer:

### Public Member Functions

- IEnumerable< TType > ResolveAll< TType > ()

  *Resolves all instances of TType or subclasses of TType. Either named or not.*
- void Clear ()

*Clears all type-mappings and instances.*

- void Inject (object obj)

  *Injects registered types/mappings into an object*
- void Register< TSource, TTarget > (string name=null)

  *Register a type mapping*
- TBase RegisterInstance< TBase > (TBase instance=null, bool injectNow=true)

  *Register an instance of a type.*
- void RegisterInstance (string name, object instance, bool injectNow=true)

  *Register a named instance*
- object RegisterInstance (Type type, object instance=null, bool injectNow=true)

  *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.*
- T Resolve< T > (string name)

  *Resolve by the name*
- object Resolve (Type instanceType, bool requireInstance=false)

  *If an instance of instanceType exist then it will return that instance otherwise it will create a new one based off mappings.*
- void InjectAll ()

  *Injects everything that is registered at once*
- void **RegisterAdapter**< **TFor, TBase, TConcrete** > ()
- TBase **ResolveAdapter**< **TBase** > (Type tfor)
- TBase **ResolveAdapter**< **TFor, TBase** > ()

### Properties

- TypeMappingCollection **Mappings** `[get, set]`
- Dictionary< Type, object > **Instances** `[get, set]`
- Dictionary< string, object > **NamedInstances** `[get, set]`
- Dictionary< Type, Dictionary < Type, Type > > **AdapterMappings** `[get, set]`

## 3.14.1 Detailed Description

A ViewModel Container and a factory for Controllers and commands.

## 3.14.2 Member Function Documentation

### 3.14.2.1 void GameContainer.Clear ( )

Clears all type-mappings and instances.

Implements IGameContainer.

### 3.14.2.2 void GameContainer.Inject ( object *obj* )

Injects registered types/mappings into an object

**Parameters**

| | |
|---|---|
| *obj* | |

Implements IGameContainer.

#### 3.14.2.3 void GameContainer.InjectAll ( )

Injects everything that is registered at once

Implements IGameContainer.

#### 3.14.2.4 void GameContainer.Register< TSource, TTarget > ( string *name* = `null` )

Register a type mapping

**Template Parameters**

| | |
|---|---|
| *TSource* | The base type. |
| *TTarget* | The concrete type |

Implements IGameContainer.

#### 3.14.2.5 void GameContainer.RegisterInstance ( string *name*, object *instance*, bool *injectNow* = `true` )

Register a named instance

**Parameters**

| | |
|---|---|
| *name* | The name for the instance to be resolved. |
| *instance* | The instance that will be resolved be the name |
| *injectNow* | Perform the injection immediately |

Implements IGameContainer.

#### 3.14.2.6 object GameContainer.RegisterInstance ( Type *type*, object *instance* = `null`, bool *injectNow* = `true` )

Register an instance of a type.

**Parameters**

| | |
|---|---|
| *type* | The type of the instance |
| *instance* | |

**Returns**

#### 3.14.2.7 TBase GameContainer.RegisterInstance< TBase > ( TBase *instance* = `null`, bool *injectNow* = `true` )

Register an instance of a type.

**Template Parameters**

| | |
|---|---|
| *TBase* | |

**Parameters**

| | |
|---|---|
| *instance* | |
| *injectNow* | |

**Returns**

**Type Constraints**

***TBase* : *class***

#### 3.14.2.8 object GameContainer.Resolve ( Type *instanceType*, bool *requireInstance* = `false` )

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 |
| *requireInstance* | If true will return null if an instance isn't registered. |

**Returns**

The/An instance of 'instanceType'

Implements IGameContainer.

#### 3.14.2.9 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***

#### 3.14.2.10 T GameContainer.Resolve< T > ( string *name* )

Resolve by the name

**Template Parameters**

| | |
|---|---|
| *T* | |

**Parameters**

| | |
|---|---|
| *name* | |

**Returns**

Implements IGameContainer.

**Type Constraints**

***T* : *class***

#### 3.14.2.11 IEnumerable<TType> GameContainer.ResolveAll< TType > ( )

Resolves all instances of TType or subclasses of TType. Either named or not.

**Template Parameters**

| | |
|---|---|
| *TType* | The Type to resolve |

**Returns**

List of objects.

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

- Assets/uFrameComplete/uFrame/Base/Controllers/GameContainer.cs

## 3.15 GameManager Class Reference

A singleton that manages our current Scene Manager and all the games types in the scene. This component will persist through every scene

Inheritance diagram for GameManager:

### Public Member Functions

- virtual void AddGame (SceneManager sceneManager)

  *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 **OnEnable** ()
- void **Awake** ()
- void **Start** ()
- virtual void **Startup** ()
- string **GetPath** (string elementPath, string path)
- void **LoadRenderSettings** ()
- void **OnDestroy** ()
- virtual void RemoveGame (SceneManager sceneManager)

  *Removes the Scene Manager from this manager. This will only happen if a Game is destroyed*

### Static Public Member Functions

- static void **ProgressUpdated** (string message, float progress)
- static Coroutine **SwitchGame**< **T** > (Action< T > setup, UpdateProgressDelegate progress=null)
- static Coroutine SwitchGame< TGame > (TGame controller, Action< TGame > 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.*
- static IEnumerator **Load** ()

### Public Attributes

- Color **_AmbientLight** = new Color(0.2f, 0.2f, 0.2f, 1.0f)
- 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**
- SceneManager _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/"
- bool **_DontUseAsyncLoading** = false

### Static Protected Member Functions

- static void **DefaultUpdateProgress** (string message, float progress)

### Properties

- static SceneManager ActiveSceneManager `[get, set]`

  *The current running game*
- static IGameContainer **Container** `[get]`
- static LevelLoadViewModel **Progress** `[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< SceneManager > 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.*
- static bool **IsPro** `[get]`

### 3.15.1 Detailed Description

A singleton that manages our current Scene Manager and all the games types in the scene. This component will persist through every scene

### 3.15.2 Member Function Documentation

#### 3.15.2.1 virtual void GameManager.AddGame ( SceneManager *sceneManager* ) `[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**

| | |
|---|---|
| *sceneManager* | The game being added. |

#### 3.15.2.2 virtual void GameManager.RemoveGame ( SceneManager *sceneManager* ) `[virtual]`

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

**Parameters**

| | |
|---|---|
| *sceneManager* | |

#### 3.15.2.3 static Coroutine GameManager.SwitchGame< TGame > ( TGame *controller*, Action< TGame > *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**

| | |
|---|---|
| *TGame* | The Scene Manager |

**Parameters**

| | |
|---|---|
| *progress* | |
| *setup* | |

| | |
|---|---|
| *controller* | |

**Returns**

**Type Constraints**

***TGame*** **:** ***SceneManager***

#### 3.15.2.4 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*** **:** ***SceneManager***

#### 3.15.2.5 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*** **:** ***SceneManager***

### 3.15.3 Member Data Documentation

#### 3.15.3.1 string GameManager._LoadingLevel

A level that displays a progress bar and message

#### 3.15.3.2 SceneManager GameManager._Start

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

### 3.15.4 Property Documentation

#### 3.15.4.1 SceneManager GameManager.ActiveSceneManager [static], [get], [set]

The current running game

#### 3.15.4.2 List<SceneManager> 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.

#### 3.15.4.3 GameManager GameManager.Instance [static], [get], [set]

The current instance of GameManager

#### 3.15.4.4 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.

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

- Assets/uFrameComplete/uFrame/Base/Controllers/GameManager.cs

## 3.16 GameType Class Reference

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

- Assets/uFrameComplete/uFrame/Base/Controllers/SceneManager.cs

## 3.17 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]
- ViewBase **Source** [get, set]
- bool **TwoWay** [get, set]

### 3.17.1 Detailed Description

Interface for all bindings

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

- Assets/uFrameComplete/uFrame/Base/Bindings/IBinding.cs

## 3.18 IBindingProvider Interface Reference

Inheritance diagram for IBindingProvider:

### Public Member Functions

- void **Bind** (ViewBase view)
- void **Unbind** (ViewBase viewBase)

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

- Assets/uFrameComplete/uFrame/Base/Bindings/IBindingProvider.cs

## 3.19 ICommand Interface Reference

The base command interface for implementing a command in a ViewModel

Inheritance diagram for ICommand:

### Public Member Functions

- IEnumerator **Execute** ()

### Properties

- object **Sender** `[get, set]`
- object **Parameter** `[get, set]`

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

### 3.19.1 Detailed Description

The base command interface for implementing a command in a ViewModel

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

- Assets/uFrameComplete/uFrame/Base/Commands/ICommand.cs

## 3.20 ICommand< T > Interface Template Reference

Inheritance diagram for ICommand< T >:

### Additional Inherited Members

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

- Assets/uFrameComplete/uFrame/Base/Commands/ICommand.cs

## 3.21 ICommandWith< T > Interface Template Reference

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

Inheritance diagram for ICommandWith< T >:

### Additional Inherited Members

#### 3.21.1 Detailed Description

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

**Template Parameters**

| | |
|---|---|
| *T* | |

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

- Assets/uFrameComplete/uFrame/Base/Commands/ICommand.cs

## 3.22 IGameContainer Interface Reference

Inheritance diagram for IGameContainer:

### Public Member Functions

- void Clear ()

  *Clears all type mappings and instances.*
- void Inject (object obj)

  *Injects registered types/mappings into an object*
- void Register< TSource, TTarget > (string name=null)

  *Register a type mapping*
- TBase RegisterInstance< TBase > (TBase @default=null, bool injectNow=true)

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

*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, bool requireInstance=false)

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

- void InjectAll ()

  *Injects everything that is registered at once*

- void RegisterInstance (string name, object instance, bool injectNow=true)

  *Register a named instance*

- T Resolve< T > (string name)

  *Resolve by the name*

- TType **ResolveAdapter< TFor, TType >** ()

### 3.22.1 Member Function Documentation

#### 3.22.1.1 void IGameContainer.Clear ( )

Clears all type mappings and instances.

Implemented in GameContainer.

#### 3.22.1.2 void IGameContainer.Inject ( object *obj* )

Injects registered types/mappings into an object

**Parameters**

| | |
|---|---|
| *obj* | |

Implemented in GameContainer.

#### 3.22.1.3 void IGameContainer.InjectAll ( )

Injects everything that is registered at once

Implemented in GameContainer.

#### 3.22.1.4 void IGameContainer.Register< TSource, TTarget > ( string *name* = `null` )

Register a type mapping

**Template Parameters**

| | |
|---|---|
| *TSource* | The base type. |
| *TTarget* | The concrete type |

Implemented in GameContainer.

#### 3.22.1.5 object IGameContainer.RegisterInstance ( Type *type*, object @ *default* = `null`, bool *injectNow* = `true` )

Register an instance of a type.

**Parameters**

| | |
|---|---|
| *type* | |
| *default* | |

**Returns**

#### 3.22.1.6 void IGameContainer.RegisterInstance ( string *name*, object *instance*, bool *injectNow* = `true` )

Register a named instance

**Parameters**

| | |
|---|---|
| *name* | The name for the instance to be resolved. |
| *instance* | The instance that will be resolved be the name |
| *injectNow* | Perform the injection immediately |

Implemented in GameContainer.

#### 3.22.1.7 TBase IGameContainer.RegisterInstance< TBase > ( TBase @ *default* = `null`, bool *injectNow* = `true` )

Register an instance of a type.

**Template Parameters**

| | |
|---|---|
| *TBase* | |

**Parameters**

| | |
|---|---|
| *default* | |

**Returns**

**Type Constraints**

***TBase* : *class***

#### 3.22.1.8 object IGameContainer.Resolve ( Type *instanceType*, bool *requireInstance* = `false` )

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 |
| *requireInstance* | Will cause an exception if an instance hasn't been registered |

**Returns**

The/An instance of 'instanceType'

Implemented in GameContainer.

#### 3.22.1.9 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***

#### 3.22.1.10 T IGameContainer.Resolve< T > ( string *name* )

Resolve by the name

**Template Parameters**

| | |
|---|---|
| *T* | |

**Parameters**

| | |
|---|---|
| *name* | |

**Returns**

Implemented in GameContainer.

**Type Constraints**

***T* : *class***

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

- Assets/uFrameComplete/uFrame/Base/Controllers/IGameContainer.cs

## 3.23 IJsonSerializable Interface Reference

Inheritance diagram for IJsonSerializable:

### Public Member Functions

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

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

- Assets/uFrameComplete/uFrame/Base/IJsonSerializable.cs

## 3.24 IModelCollection Interface Reference

Inheritance diagram for IModelCollection:

### Public Member Functions

- void **AddObject** (object item)
- void **RemoveObject** (object item)
- void **Clear** ()

### Properties

- IEnumerable< object > **Value** `[get]`
- Type **ItemType** `[get]`

### Events

- ModelCollectionChanged **Changed**

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

- Assets/uFrameComplete/uFrame/Base/ViewModels/ModelCollection.cs

## 3.25 InjectAttribute Class Reference

Used by the injection container to determine if a property or field should be injected.

Inheritance diagram for InjectAttribute:

### Public Member Functions

- **InjectAttribute** (string name)

### Properties

- string **Name** `[get, set]`

#### 3.25.1 Detailed Description

Used by the injection container to determine if a property or field should be injected.

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

- Assets/uFrameComplete/uFrame/Base/Controllers/InjectAttribute.cs

## 3.26 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

### 3.26.1 Member Function Documentation

#### 3.26.1.1 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.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/MouseEventBinding.cs

## 3.27 ITwoWayBinding Interface Reference

Inheritance diagram for ITwoWayBinding:

### Public Member Functions

- void BindReverse ()

  *Will be called every update frame*

### Additional Inherited Members

#### 3.27.1 Member Function Documentation

##### 3.27.1.1 void ITwoWayBinding.BindReverse ( )

Will be called every update frame

Implemented in ModelPropertyBinding.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ITwoWayBinding.cs

## 3.28 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

#### 3.28.1 Property Documentation

##### 3.28.1.1 ViewModel IView.ViewModelObject `[get]`

Gets the view model object.

The view model object.

##### 3.28.1.2 Type IView.ViewModelType `[get]`

Gets the type of the view model.

The type of the model.

##### 3.28.1.3 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:

- Assets/uFrameComplete/uFrame/Base/Views/IView.cs

## 3.29 IViewModelObserver Interface Reference

Potential future use.

Inheritance diagram for IViewModelObserver:

### Public Member Functions

- void **AddBinding** (IBinding binding)
- void **RemoveBinding** (IBinding binding)
- void **Unbind** ()

### Properties

- List< IBinding > **Bindings** `[get, set]`

### 3.29.1 Detailed Description

Potential future use.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/IViewModelObserver.cs

## 3.30 JSONArray Class Reference

Inheritance diagram for 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

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

- Assets/uFrameComplete/uFrame/Base/SimpleJSON.cs

## 3.31 JSONClass Class Reference

Inheritance diagram for 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

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

- Assets/uFrameComplete/uFrame/Base/SimpleJSON.cs

## 3.32 JSONData Class Reference

Inheritance diagram for 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

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

- Assets/uFrameComplete/uFrame/Base/SimpleJSON.cs

## 3.33 JSONLazyCreator Class Reference

Inheritance diagram for 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]`

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

- Assets/uFrameComplete/uFrame/Base/SimpleJSON.cs

## 3.34 JSONNode Class Reference

Inheritance diagram for 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 **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]`

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

- Assets/uFrameComplete/uFrame/Base/SimpleJSON.cs

## 3.35 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

### 3.35.1 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

### 3.35.2 Member Function Documentation

#### 3.35.2.1 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.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/KeyBinding.cs

## 3.36 LevelLoaderSceneManager Class Reference

Inheritance diagram for LevelLoaderSceneManager:

### Protected Member Functions

- void **Awake** ()

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

- Assets/uFrameComplete/uFrame/Base/Controllers/LevelLoaderSceneContainer.cs

## 3.37 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)
- void **AddObject** (object item)
- void **RemoveObject** (object item)
- **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]`
- Type **ItemType** `[get]`

### Events

- ModelCollectionChanged **Changed**
- ModelCollectionChangedWith **ChangedWith**

### Additional Inherited Members

#### 3.37.1 Detailed Description

An observable collection to use in viewmodels.

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

- Assets/uFrameComplete/uFrame/Base/ViewModels/ModelCollection.cs

## 3.38 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

#### 3.38.1 Member Function Documentation

##### 3.38.1.1 override void ModelCollectionBinding< TCollectionType >.Bind ( ) `[virtual]`

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

Reimplemented from Binding.

##### 3.38.1.2 override void ModelCollectionBinding< TCollectionType >.Unbind ( ) `[virtual]`

Unbind this binding

Reimplemented from Binding.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelViewModelCollectionBinding.cs

## 3.39 ModelCollectionChangeEvent Class Reference

Inheritance diagram for ModelCollectionChangeEvent:

### Properties

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

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

- Assets/uFrameComplete/uFrame/Base/ViewModels/ModelCollection.cs

## 3.40 ModelCollectionChangeEventWith< T > Class Template Reference

Inheritance diagram for ModelCollectionChangeEventWith< T >:

### Properties

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

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

- Assets/uFrameComplete/uFrame/Base/ViewModels/ModelCollection.cs

## 3.41 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

- override object GetArgument ()

  *Overriden to supply the CommandArgumentSelector result value if its not equal to null*
- ModelCollisionEventBinding SetParameterSelector (Func< GameObject, object > commandArgSelector)

  *Set the parameter that will be passed to the command.*
- CommandBinding Subscribe (Action< GameObject > action, bool before=false)

  *Subscribe to this collision binding with a reference to the collider.*
- 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

#### 3.41.1 Detailed Description

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

#### 3.41.2 Member Function Documentation

##### 3.41.2.1 override object ModelCollisionEventBinding.GetArgument ( ) `[virtual]`

Overriden to supply the CommandArgumentSelector result value if its not equal to null

**Returns**

The object that will be passed as the argument to the command.

Reimplemented from CommandBinding.

##### 3.41.2.2 ModelCollisionEventBinding ModelCollisionEventBinding.SetParameterSelector ( Func< GameObject, object > *commandArgSelector* )

Set the parameter that will be passed to the command.

**Parameters**

| | |
|---|---|
| *commandArgSelector* | A selector that will select the object to pass to the command with the collider as the first argument |

**Returns**

##### 3.41.2.3 CommandBinding ModelCollisionEventBinding.Subscribe ( Action< GameObject > *action*, bool *before* = `false` )

Subscribe to this collision binding with a reference to the collider.

**Parameters**

| | |
|---|---|
| *action* | The action to perform with the collider as the parameter. |
| *before* | Execute the action before the action executes. Defaults to false. |

**Returns**

This so it can be further chained.

##### 3.41.2.4 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.

### 3.41.3 Property Documentation

#### 3.41.3.1 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:

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelCollisionEventBinding.cs

## 3.42 ModelCommandBinding Class Reference

A base class for binding to a ViewModel command.

Inheritance diagram for ModelCommandBinding:

### Public Member Functions

- 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

### 3.42.1 Detailed Description

A base class for binding to a ViewModel command.

### 3.42.2 Member Function Documentation

#### 3.42.2.1 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.

#### 3.42.2.2 override void ModelCommandBinding.Unbind ( ) [virtual]

Unbind this binding

Reimplemented from CommandBinding.

Reimplemented in ModelEventBinding.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelCommandBinding.cs

## 3.43 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

### 3.43.1 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.

### 3.43.2 Member Function Documentation

#### 3.43.2.1 override void ModelEventBinding.Bind ( ) [virtual]

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

Reimplemented from ModelCommandBinding.

#### 3.43.2.2 override void ModelEventBinding.Unbind ( ) [virtual]

Unbind this binding

Reimplemented from ModelCommandBinding.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelEventBinding.cs

## 3.44 ModelInputButtonBinding Class Reference

Inheritance diagram for ModelInputButtonBinding:

### Properties

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

### Additional Inherited Members

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelMouseEventBinding.cs

## 3.45 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

### 3.45.1 Detailed Description

Binds a key to a ViewModel command.

### 3.45.2 Member Function Documentation

#### 3.45.2.1 ModelKeyBinding ModelKeyBinding.RequireAlt ( )

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

**Returns**

This to respect chaining.

#### 3.45.2.2 ModelKeyBinding ModelKeyBinding.RequireControl ( )

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

**Returns**

This to respect chaining.

#### 3.45.2.3 ModelKeyBinding ModelKeyBinding.RequireShift ( )

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

**Returns**

This to respect chaining.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelKeyBinding.cs

## 3.46 ModelMouseEventBinding Class Reference

Inheritance diagram for ModelMouseEventBinding:

### Properties

- MouseEventType **EventType** `[get, set]`

### Additional Inherited Members

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelMouseEventBinding.cs

## 3.47 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*

## 3.47.1 Detailed Description

A base class for model properties.

## 3.47.2 Member Function Documentation

### 3.47.2.1 void ModelPropertyBase.QuietlySetValue ( object *value* )

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

**Parameters**

| | |
|---|---|
| *value* | |

## 3.47.3 Property Documentation

### 3.47.3.1 virtual object ModelPropertyBase.ObjectValue `[get],[set]`

The value of this model property

### 3.47.3.2 virtual Type ModelPropertyBase.ValueType `[get]`

The value type of this property

### 3.47.4 Event Documentation

#### 3.47.4.1 PropertyChangedHandler ModelPropertyBase.PropertyChanged

When the value has changed

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

- Assets/uFrameComplete/uFrame/Base/ViewModels/ModelPropertyBase.cs

## 3.48 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*

### Properties

- bool **IsImmediate** `[get, set]`

### Additional Inherited Members

### 3.48.1 Detailed Description

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

### 3.48.2 Member Function Documentation

#### 3.48.2.1 override void ModelPropertyBinding.Bind ( ) `[virtual]`

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

Reimplemented from Binding.

#### 3.48.2.2 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.

#### 3.48.2.3 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:

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelPropertyBinding.cs

## 3.49 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

- ModelViewModelCollectionBinding **Immediate** (bool immediate=true)
- ModelViewModelCollectionBinding **SetAddHandler** (Action< ViewBase > onAdd)
- ModelViewModelCollectionBinding **SetCreateHandler** (Func< ViewModel, ViewBase > onCreateView)
- ModelViewModelCollectionBinding **SetParent** (Transform parent)
- ModelViewModelCollectionBinding **SetRemoveHandler** (Action< ViewBase > onRemove)
- ModelViewModelCollectionBinding **SetView** (string viewName)
- override void Unbind ()

  *Unbind this binding*
- override void Bind ()

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

### Protected Member Functions

- void **AddLookup** (GameObject obj, ViewModel viewModel)
- void **RemoveLookup** (ViewModel model)

### Properties

- IModelCollection **Collection** [get]
- bool **IsImmediate** [get, set]
- Action< ViewBase > **OnAddView** [get, set]
- Func< ViewModel, ViewBase > **OnCreateView** [get, set]
- Action< ViewBase > **OnRemoveView** [get, set]

- Transform **Parent** `[get, set]`
- string **ViewName** `[get, set]`
- Dictionary< int, GameObject > **GameObjectLookup** `[get, set]`
- Dictionary< ViewModel, int > **ObjectIdLookup** `[get, set]`

### 3.49.1 Detailed Description

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

### 3.49.2 Member Function Documentation

#### 3.49.2.1 override void ModelViewModelCollectionBinding.Bind ( ) `[virtual]`

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

Reimplemented from Binding.

#### 3.49.2.2 override void ModelViewModelCollectionBinding.Unbind ( ) `[virtual]`

Unbind this binding

Reimplemented from Binding.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelViewModelCollectionBinding.cs

## 3.50 ModelViewPropertyBinding Class Reference

Inheritance diagram for ModelViewPropertyBinding:

### Public Member Functions

- override void Bind ()

  *Set-up the binding. This should almost always be implemented in a deriving class.*
- ModelViewPropertyBinding **SetView** (string viewName)
- ModelViewPropertyBinding **SetParent** (Transform parent)
- override void Unbind ()

  *Unbind this binding*

### Properties

- Transform **Parent** `[get, set]`
- string **ViewName** `[get, set]`
- Func
  < ModelViewModelCollectionBinding,
  ViewModel, ViewBase > **OnCreateView** `[get, set]`

### Additional Inherited Members

### 3.50.1 Member Function Documentation

#### 3.50.1.1 override void ModelViewPropertyBinding.Bind ( ) [virtual]

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

Reimplemented from Binding.

#### 3.50.1.2 override void ModelViewPropertyBinding.Unbind ( ) [virtual]

Unbind this binding

Reimplemented from Binding.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ModelPropertyBinding.cs

## 3.51 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

### 3.51.1 Member Function Documentation

#### 3.51.1.1 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.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/MouseEventBinding.cs

## 3.52 P< T > Class Template Reference

A typed ViewModel Property Class

Inheritance diagram for P< T >:

### Public Member Functions

- **P** (T value)
- 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

### 3.52.1 Detailed Description

A typed ViewModel Property Class

**Template Parameters**

| | |
|---|---|
| *T* | |

### 3.52.2 Member Function Documentation

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

Deserialize the specified node into `Value`.

**Parameters**

| | |
|---|---|
| *node* | Node. |

Implements ModelPropertyBase.

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

Serializes this object

Implements ModelPropertyBase.

### 3.52.3 Property Documentation

#### 3.52.3.1 T P< T >.Value [get], [set]

Gets or sets the value.

The value.

#### 3.52.3.2 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:

- Assets/uFrameComplete/uFrame/Base/ViewModels/P.cs

## 3.53 SceneManager 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 SceneManager:

### Public Member Functions

- 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*
- 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 **RegisterRootView** (ViewBase viewBase)

### Protected Member Functions

- virtual void **Awake** ()
- virtual void **OnDestroy** ()

### Properties

- IGameContainer **Container** `[get, set]`
- static ISwitchLevelSettings Settings `[get]`

  *The settings at which the level will be loaded*
- List< ViewBase > **RootViews** `[get, set]`

## 3.53.1 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.

## 3.53.2 Member Function Documentation

### 3.53.2.1 virtual IEnumerator SceneManager.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**

### 3.53.2.2 virtual void SceneManager.OnLoaded ( ) `[virtual]`

This method is called when this controller has started loading

### 3.53.2.3 virtual void SceneManager.OnLoading ( ) `[virtual]`

This method is called when the load function has completed

### 3.53.2.4 virtual void SceneManager.Reload ( ) `[virtual]`

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

## 3.53.3 Property Documentation

### 3.53.3.1 ISwitchLevelSettings SceneManager.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:

- Assets/uFrameComplete/uFrame/Base/Controllers/SceneManager.cs

## 3.54 TypeMapping Class Reference

### Properties

- Type **From** `[get, set]`
- Type **To** `[get, set]`
- string **Name** `[get, set]`

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

- Assets/uFrameComplete/uFrame/Base/Controllers/GameContainer.cs

## 3.55 TypeMappingCollection Class Reference

Inheritance diagram for TypeMappingCollection:

### Properties

- Type **this[Type from]** `[get, set]`
- Type **this[Type from, string name]** `[get, set]`

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

- Assets/uFrameComplete/uFrame/Base/Controllers/GameContainer.cs

## 3.56 UFGroup Class Reference

Inheritance diagram for UFGroup:

### Public Member Functions

- **UFGroup** (string name)

### Properties

- string **Name** `[get, set]`

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

- Assets/uFrameComplete/uFrame/Base/Views/ViewBase.cs

## 3.57 UFPropertyBinding 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 UFPropertyBinding:

### 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

#### 3.57.1 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.

Note: NGUI added a propertybinding class so this one is renamed to UFPropertyBinding.

#### 3.57.2 Member Function Documentation

##### 3.57.2.1 override IBinding UFPropertyBinding.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.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/UFPropertyBinding.cs

## 3.58 UFRequireInstanceMethod Class Reference

Inheritance diagram for UFRequireInstanceMethod:

### Public Member Functions

- **UFRequireInstanceMethod** (string methodName)

### Properties

- string **MethodName** `[get, set]`

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

- Assets/uFrameComplete/uFrame/Base/Views/ViewBase.cs

## 3.59 UFToggleGroup Class Reference

Inheritance diagram for UFToggleGroup:

### Public Member Functions

- **UFToggleGroup** (string name)

### Properties

- string **Name** `[get, set]`

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

- Assets/uFrameComplete/uFrame/Base/Views/ViewBase.cs

## 3.60 View< TModel > Class Template Reference

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

**Template Parameters**

| | |
|---|---|
| *TModel* | The ViewModel Type |

Inheritance diagram for View< TModel >:

### Public Member Functions

- sealed override void InitializeViewModel (ViewModel model)

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

### Protected Member Functions

- virtual void InitializeViewModel (TModel viewModel)

  *The method InitializeViewModel should be overriden to initialize anything from the Inspector Editor.*

### 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

#### 3.60.1 Detailed Description

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

**Template Parameters**

| | |
|---|---|
| *TModel* | The ViewModel Type |

**Type Constraints**

***TModel* : *ViewModel***

***TModel* : *new()***

#### 3.60.2 Member Function Documentation

##### 3.60.2.1 sealed override void View< TModel >.InitializeViewModel ( ViewModel *model* ) [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.

##### 3.60.2.2 virtual void View< TModel >.InitializeViewModel ( TModel *viewModel* ) [protected], [virtual]

The method InitializeViewModel should be overriden to initialize anything from the Inspector Editor.

**Parameters**

| | |
|---|---|
| *viewModel* | |

#### 3.60.3 Property Documentation

##### 3.60.3.1 TModel View< TModel >.Model [get], [set]

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

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

- Assets/uFrameComplete/uFrame/Base/Views/View.cs

## 3.61 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.*
- override void **AddBinding** (IBinding binding)
- virtual void **Awake** ()
- abstract void **Bind** ()
- abstract ViewModel **CreateModel** ()
- virtual void **OnDestroy** ()
- virtual void **OnDisable** ()
- virtual void **OnEnable** ()
- 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 **AfterBind** ()
- virtual void **Start** ()
- override void Unbind ()

  *Unbind the current bindings.*
- virtual void Event (string eventname)

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

  *This method should be overriden to Initialize the ViewModel with any options specified in a unity component inspector.*
- void **ExecuteCommand** (ICommand command, object argument)
- virtual void **ExecuteCommand** (ICommand command)
- void **ExecuteCommand**< **TArgument** > (ICommandWith< TArgument > command, ViewModel sender, TArgument argument)
- void **ExecuteCommand**< **TArgument** > (ICommandWith< TArgument > command, TArgument argument)

## Public Attributes

- bool _LogEvents

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

## Protected Member Functions

- ViewModel **RequestViewModel** (Controller controller)
- ViewModel **ResolveViewModel** (Controller controller=null)
- virtual void **LateUpdate** ()

## Properties

- List< IBindingProvider > **BindingProviders** `[get, set]`
- IEnumerable< ViewModel > **ChildViewModels** `[get]`
- List< ViewBase > **ChildViews** `[get, set]`
- bool **Instantiated** `[get, set]`
- ViewBase **ParentView** `[get, set]`
- 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*
- virtual bool **IsMultiInstance** `[get]`
- bool **OverrideViewModel** `[get, set]`
- bool **ForceResolveViewModel** `[get, set]`

### Events

- ViewEvent EventTriggered

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

### 3.61.1 Detailed Description

The base class for a View that binds to a ViewModel

### 3.61.2 Member Function Documentation

#### 3.61.2.1 virtual void ViewBase.Event ( string *eventname* ) `[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 |

#### 3.61.2.2 abstract void ViewBase.InitializeViewModel ( ViewModel *model* ) `[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 >.

#### 3.61.2.3 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.

#### 3.61.2.4 override void ViewBase.Unbind ( ) `[virtual]`

Unbind the current bindings.

Reimplemented from ViewModelObserver.

#### 3.61.2.5 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. |

### 3.61.3 Member Data Documentation

#### 3.61.3.1 bool ViewBase._LogEvents

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

### 3.61.4 Property Documentation

#### 3.61.4.1 string ViewBase.ViewName `[get],[set]`

The name of the prefab that created this view

### 3.61.5 Event Documentation

#### 3.61.5.1 ViewEvent ViewBase.EventTriggered

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

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

- Assets/uFrameComplete/uFrame/Base/Views/ViewBase.cs

## 3.62 ViewComponent Class Reference

Inheritance diagram for ViewComponent:

### Public Member Functions

- virtual void **Awake** ()
- virtual void **Bind** (ViewBase view)
- virtual void **Unbind** (ViewBase viewBase)

### Properties

- ViewBase **View** `[get, set]`

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

- Assets/uFrameComplete/uFrame/Base/Views/ViewComponent.cs

## 3.63 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, Vector3 position)

  *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

### 3.63.1 Detailed Description

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

### 3.63.2 Member Function Documentation

#### 3.63.2.1 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

#### 3.63.2.2 ViewBase ViewContainer.InstantiateView ( string *viewName,* Vector3 *position* )

Instantiates a view.

**Parameters**

| | |
|---|---|
| *viewName* | The name of the prefab/view to instantiate |
| *position* | The position to instantiate the view. |

**Returns**

The instantiated view

#### 3.63.2.3 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

#### 3.63.2.4 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

#### 3.63.2.5 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

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

- Assets/uFrameComplete/uFrame/Base/ViewContainer.cs

## 3.64 ViewEventTrigger Class Reference

Inheritance diagram for ViewEventTrigger:

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ViewEventTrigger.cs

## 3.65 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** > (ICommandWith< T > command)
- ICommand **ForwardThisTo**< **T** > (Func< ICommandWith< 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,
  ModelPropertyBase > **Properties** `[get]`
- 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*

### 3.65.1 Detailed Description

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

### 3.65.2 Member Function Documentation

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

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

**Returns**

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

**Parameters**

| | |
|---|---|
| *modelType* | |

**Returns**

### 3.65.3 Property Documentation

#### 3.65.3.1 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:

- Assets/uFrameComplete/uFrame/Base/Views/ViewModel.cs

## 3.66 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

#### 3.66.1 Member Function Documentation

##### 3.66.1.1 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.

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

- Assets/uFrameComplete/uFrame/Base/Bindings/ViewModelCollectionBinding.cs

## 3.67 ViewModelObserver Class Reference

Inheritance diagram for ViewModelObserver:

### Public Member Functions

- virtual void **AddBinding** (IBinding binding)
- virtual void **RemoveBinding** (IBinding binding)
- virtual void **Unbind** ()

### Properties

- List< IBinding > Bindings [get, set]

  *The bindings that are attached to this ViewModel*

#### 3.67.1 Property Documentation

##### 3.67.1.1 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:

- Assets/uFrameComplete/uFrame/Base/Views/ViewModelObserver.cs

## 3.68 ViewModelOverrideAttribute Class Reference

Inheritance diagram for ViewModelOverrideAttribute:

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

- Assets/uFrameComplete/uFrame/Base/Views/ViewBase.cs

## 3.69 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*

### 3.69.1 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.

### 3.69.2 Member Function Documentation

#### 3.69.2.1 virtual GameObject ViewResolver.FindView ( ViewModel *model* ) [virtual]

Provides a prefab

**Parameters**

| | |
|---|---|
| *model* | The model for the view prefab we are looking for |

**Returns**

#### 3.69.2.2 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:

- Assets/uFrameComplete/uFrame/Base/Views/ViewResolver.cs

## 3.70 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

- object **Sender** `[get, set]`
- object **Parameter** `[get, set]`
- Func< IEnumerator > **EnumeratorDelegate** `[get, set]`

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

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

- Assets/uFrameComplete/uFrame/Base/Commands/Command.cs

## 3.71 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 sender, Func< T, IEnumerator > enumeratorDelegate)
- IEnumerator **Execute** ()

### Protected Member Functions

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

### Properties

- object **Sender** `[get, set]`
- object **Parameter** `[get, set]`
- Func< T, IEnumerator > **EnumeratorDelegate** `[get, set]`

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

### 3.71.1 Detailed Description

A coroutine command with a parameter.

**Template Parameters**

| | |
|---|---|
| *T* | |

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

- Assets/uFrameComplete/uFrame/Base/Commands/YieldCommandWith.cs

## 3.72 YieldCommandWithSender< T > Class Template Reference

A coroutine command with a parameter.

Inheritance diagram for YieldCommandWithSender< T >:

### Public Member Functions

- **YieldCommandWithSender** (Func< T, IEnumerator > enumeratorDelegate)
- **YieldCommandWithSender** (T sender, Func< T, IEnumerator > enumeratorDelegate)
- IEnumerator **Execute** ()

### Protected Member Functions

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

### Properties

- object **Sender** `[get, set]`
- object **Parameter** `[get, set]`
- Func< T, IEnumerator > **EnumeratorDelegate** `[get, set]`

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

#### 3.72.1 Detailed Description

A coroutine command with a parameter.

**Template Parameters**

| | |
|---|---|
| *T* | |

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

- Assets/uFrameComplete/uFrame/Base/Commands/YieldCommandWith.cs

## 3.73 YieldCommandWithSenderAndArgument< TSender, TArgument > Class Template Reference

A coroutine command with a parameter.

Inheritance diagram for YieldCommandWithSenderAndArgument< TSender, TArgument >:

### Public Member Functions

- **YieldCommandWithSenderAndArgument** (Func< TSender, TArgument, IEnumerator > enumeratorDelegate)
- **YieldCommandWithSenderAndArgument** (TSender sender, Func< TSender, TArgument, IEnumerator > enumeratorDelegate)
- IEnumerator **Execute** ()

### Protected Member Functions

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

### Properties

- object **Sender** `[get, set]`
- object **Parameter** `[get, set]`
- Func< TSender, TArgument, IEnumerator > **EnumeratorDelegate** `[get, set]`

### Events

- CommandEvent **OnCommandExecuted**
- CommandEvent **OnCommandExecuting**

#### 3.73.1 Detailed Description

A coroutine command with a parameter.

**Template Parameters**

| | |
|---|---|
| *TSender* | |
| *TArgument* | |

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

- Assets/uFrameComplete/uFrame/Base/Commands/YieldCommandWith.cs

# Index