# uFrame API Docs

Generated by Doxygen 1.8.6

Tue Aug 5 2014 04:13:27

ii CONTENTS

# **Contents**

		Index	1
1.1	Class F	lierarchy	1
Clas	a Inday		_
		: <b>-</b>	<b>5</b> 5
2.1	Class L	151	5
Clas	s Docur	nentation	9
3.1	Bindabl	eProperty Class Reference	9
	3.1.1	Detailed Description	9
	3.1.2	Property Documentation	9
3.2	Binding	Class Reference	9
	3.2.1	Detailed Description	10
	3.2.2	Constructor & Destructor Documentation	10
	3.2.3	Member Function Documentation	11
	3.2.4	Property Documentation	11
3.3	Collisio	nEventBinding Class Reference	12
	3.3.1	Detailed Description	12
	3.3.2	Member Function Documentation	12
3.4	Comma	and Class Reference	13
	3.4.1	Detailed Description	13
3.5	Comma	andBinding Class Reference	14
	3.5.1	Detailed Description	14
	3.5.2	Member Function Documentation	15
3.6	Comma	andWith< T > Class Template Reference	15
	3.6.1	Detailed Description	16
3.7	Comma	andWithSender< TSender > Class Template Reference	16
3.8	Comma	and With Sender And Argument $<$ TS ender, TArgument $>$ Class Template Reference	17
3.9	Compo	nentBinding Class Reference	17
	3.9.1	Detailed Description	18
	3.9.2	Member Function Documentation	18
	3.9.3	Property Documentation	19
3.10	Compo	nentCommandBinding Class Reference	19
	3.10.1	Detailed Description	19
	3.10.2	Property Documentation	19
3.11	Control	ler Class Reference	19
	3.11.1	Detailed Description	20
	3.11.2	Constructor & Destructor Documentation	21
	3.11.3	Member Function Documentation	22
	3.11.4	Property Documentation	23
	Class 2.1 Class 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10	Class Index 2.1 Class L  Class Docur 3.1 Bindabl	Class   Index

3.12	DefaultTypeResolver Class Reference	23
3.13	DiagramInfoAttribute Class Reference	23
3.14	ElementService Class Reference	24
	3.14.1 Detailed Description	24
3.15	EventBinding Class Reference	24
	3.15.1 Detailed Description	25
	3.15.2 Member Function Documentation	25
3.16	FileSerializerStorage Class Reference	25
3.17	GameContainer Class Reference	26
	3.17.1 Detailed Description	26
	3.17.2 Member Function Documentation	27
3.18	GameManager Class Reference	29
	3.18.1 Detailed Description	30
	3.18.2 Member Function Documentation	31
	3.18.3 Member Data Documentation	32
	3.18.4 Property Documentation	33
3.19	IBinding Interface Reference	33
	3.19.1 Detailed Description	34
3.20	IBindingProvider Interface Reference	34
3.21	ICommand Interface Reference	34
	3.21.1 Detailed Description	35
3.22	ICommand< T > Interface Template Reference	35
3.23	ICommandWith< T > Interface Template Reference	35
	3.23.1 Detailed Description	35
3.24	IGameContainer Interface Reference	35
	3.24.1 Member Function Documentation	36
3.25	UsonSerializable Interface Reference	40
3.26	IModelCollection Interface Reference	40
3.27	InjectAttribute Class Reference	41
	3.27.1 Detailed Description	41
3.28	InputBinding Class Reference	41
	3.28.1 Member Function Documentation	42
3.29	ISerializer Interface Reference	42
3.30	ISerializerStorage Interface Reference	42
3.31	ISerializerStream Interface Reference	43
3.32	ITwoWayBinding Interface Reference	44
	3.32.1 Member Function Documentation	44
3.33	ITypeResolver Interface Reference	44
3.34	IUFSerializable Interface Reference	45
3.35	IView Interface Reference	45

iv CONTENTS

	3.35.1 Property Documentation	46
3.36	IViewModelObserver Interface Reference	46
	3.36.1 Detailed Description	46
3.37	JSONArray Class Reference	46
3.38	JSONClass Class Reference	47
3.39	JSONData Class Reference	48
3.40	JSONLazyCreator Class Reference	49
3.41	JSONNode Class Reference	49
3.42	JsonStream Class Reference	51
3.43	KeyBinding Class Reference	52
	3.43.1 Detailed Description	52
	3.43.2 Member Function Documentation	52
3.44	LevelLoaderSceneManager Class Reference	53
3.45	$\label{eq:modelCollection} \mbox{ModelCollection} < \mbox{T} > \mbox{Class Template Reference}  .  .  .  .  .  .  .  .  .  $	53
	3.45.1 Detailed Description	54
3.46	${\sf ModelCollectionBinding} {< \   {\sf TCollectionType} > {\sf Class\ Template\ Reference}  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  $	54
	3.46.1 Member Function Documentation	55
3.47	ModelCollectionChangeEvent Class Reference	55
3.48	$\label{thm:modelCollectionChangeEventWith} \textbf{ModelCollectionChangeEventWith} < \textbf{T} > \textbf{Class Template Reference} \qquad \dots \qquad \dots \qquad \dots \qquad \dots$	56
3.49	ModelCollisionEventBinding Class Reference	56
	3.49.1 Detailed Description	57
	3.49.2 Member Function Documentation	57
	3.49.3 Property Documentation	58
3.50	ModelCommandBinding Class Reference	58
	3.50.1 Detailed Description	58
	3.50.2 Member Function Documentation	59
3.51	ModelEventBinding Class Reference	59
	3.51.1 Detailed Description	60
	3.51.2 Member Function Documentation	60
3.52	ModelInputButtonBinding Class Reference	60
3.53	ModelKeyBinding Class Reference	60
	3.53.1 Detailed Description	61
	3.53.2 Member Function Documentation	61
3.54	ModelMouseEventBinding Class Reference	62
3.55	ModelPropertyBase Class Reference	62
	3.55.1 Detailed Description	63
	3.55.2 Member Function Documentation	63
	3.55.3 Property Documentation	64
	3.55.4 Event Documentation	64
3.56	ModelPropertyBinding Class Reference	64

	3.56.1 Detailed Description	65
	3.56.2 Member Function Documentation	65
3.57	ModelViewModelCollectionBinding Class Reference	65
	3.57.1 Detailed Description	66
	3.57.2 Member Function Documentation	66
3.58	ModelViewPropertyBinding Class Reference	66
	3.58.1 Member Function Documentation	67
3.59	MouseEventBinding Class Reference	67
	3.59.1 Member Function Documentation	68
3.60	P< T > Class Template Reference	68
	3.60.1 Detailed Description	69
	3.60.2 Member Function Documentation	69
	3.60.3 Property Documentation	69
3.61	RegisteredInstance Class Reference	70
3.62	SceneContext Class Reference	70
	3.62.1 Detailed Description	70
	3.62.2 Member Function Documentation	70
	3.62.3 Property Documentation	71
3.63	SceneManager Class Reference	71
	3.63.1 Detailed Description	72
	3.63.2 Member Function Documentation	72
	3.63.3 Property Documentation	74
3.64	StateLoaderResolver Class Reference	75
3.65	StringSerializerStorage Class Reference	75
3.66	TypeInstanceCollection Class Reference	76
3.67	TypeMapping Class Reference	76
3.68	TypeMappingCollection Class Reference	76
3.69	TypeRelation Class Reference	77
3.70	TypeRelationCollection Class Reference	77
3.71	UFGroup Class Reference	77
3.72	UFPropertyBinding Class Reference	78
	3.72.1 Detailed Description	78
	3.72.2 Member Function Documentation	78
3.73	UFRequireInstanceMethod Class Reference	79
3.74	UFToggleGroup Class Reference	79
3.75	View< TModel > Class Template Reference	80
	3.75.1 Detailed Description	81
	3.75.2 Member Function Documentation	81
	3.75.3 Property Documentation	82
3.76	ViewBase Class Reference	82

1 Hierarchical Index 1

	3.76.1 Detailed Description	84
	3.76.2 Member Function Documentation	84
	3.76.3 Member Data Documentation	87
	3.76.4 Property Documentation	87
	3.76.5 Event Documentation	88
3.77	ViewComponent Class Reference	88
3.78	ViewContainer Class Reference	88
	3.78.1 Detailed Description	89
	3.78.2 Member Function Documentation	89
3.79	ViewEventTrigger Class Reference	91
3.80	ViewModel Class Reference	91
	3.80.1 Detailed Description	92
	3.80.2 Member Function Documentation	92
	3.80.3 Property Documentation	93
3.81	ViewModelCollectionBinding Class Reference	93
	3.81.1 Member Function Documentation	94
	ViewModelCommandInfo Class Reference	94
	ViewModelPropertyInfo Class Reference	94
3.84	ViewResolver Class Reference	95
	3.84.1 Detailed Description	95
	3.84.2 Member Function Documentation	95
3.85	YieldCommand Class Reference	95
3.86	$\label{eq:commandWith} \textbf{YieldCommandWith} < \textbf{T} > \textbf{Class Template Reference} \qquad \dots \qquad \dots \qquad \dots \qquad \dots$	96
	3.86.1 Detailed Description	
3.87	$\label{eq:commandWithSender} \textbf{YieldCommandWithSender} < \textbf{T} > \textbf{Class Template Reference}  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  $	97
	3.87.1 Detailed Description	98
3.88	$\label{thm:commandWithSenderAndArgument} \textbf{YieldCommandWithSenderAndArgument} < \textbf{TSender}, \textbf{TArgument} > \textbf{Class Template Reference}  .  .$	98
	3.88.1 Detailed Description	99
Index		100
dox		.00
4 LII:-	ararahigal Indov	
1 Hie	erarchical Index	
1.1 Cla	ass Hierarchy	
This inhe	eritance list is sorted roughly, but not completely, alphabetically:	
Attrib	oute	
D	biagramInfoAttribute	23
	njectAttribute	41
		41
U	FGroup Control of the	77

UFRequireInstanceMethod	79
UFToggleGroup	79
BindableProperty	9
Controller	19
ElementService	24
IBinding	33
Binding	9
CommandBinding	14
ModelCommandBinding	58
ModelCollisionEventBinding	56
ModelEventBinding	59
ModelInputButtonBinding	60
ModelKeyBinding	60
ModelMouseEventBinding	62
${\bf ModelCollectionBinding}{<{\bf TCollectionType}>}$	54
ModelPropertyBinding	64
ModelViewModelCollectionBinding	65
ModelViewPropertyBinding	66
ITwoWayBinding	44
ModelPropertyBinding	64
IBindingProvider	34
ViewComponent ICollection< T >	88
ModelCollection< T >	53
ICommand	34
Command	13
ICommand < T >	35
ICommandWith< T >	35
CommandWith < T >	15
CommandWithSender < TSender >	16
CommandWithSenderAndArgument < TSender, TArgument >	17
YieldCommandWith< T >	96

${\sf YieldCommandWithSender} < {\sf T} >$	97
${\bf Yield Command With Sender And Argument < T Sender, T Argument >}$	98
YieldCommand IEnumerable	95
JSONArray	46
JSONClass	47
IGameContainer	35
GameContainer	26
IJsonSerializable	40
$\label{eq:list} \begin{tabular}{l} \textbf{ViewModel} \\ \textbf{IList} < \textbf{T} > \\ \end{tabular}$	91
ModelCollection< T >	53
IModelCollection	40
ModelCollection< T > INotifyPropertyChanged	53
ModelCollection < T >	53
ViewModel	91
ISerializer	42
ISerializerStorage	42
FileSerializerStorage	25
StringSerializerStorage	75
ISerializerStream	43
JsonStream	51
ITypeResolver	44
DefaultTypeResolver	23
StateLoaderResolver	75
IUFSerializable	45
ViewModel	91
IViewModelObserver	46
IView	45
ViewBase	82
View < TModel >	80
ViewModel	91

JSONNode	49
JSONArray	46
JSONClass	47
JSONData	48
JSONLazyCreator List< RegisteredInstance >	49
TypeInstanceCollection List< TypeMapping >	76
TypeMappingCollection List< TypeRelation >	76
TypeRelationCollection	77
ModelCollectionChangeEvent	55
${\bf ModelCollectionChangeEventWith}{<{\bf T}>}$	56
ModelPropertyBase	62
P <t></t>	68
ModelCollection < T > MonoBehaviour	53
ComponentBinding	17
ComponentCommandBinding	19
CollisionEventBinding	12
EventBinding	24
InputBinding	41
KeyBinding	52
MouseEventBinding	67
UFPropertyBinding	78
ViewModelCollectionBinding	93
GameManager	29
LevelLoaderSceneManager	53
ViewComponent	88
ViewContainer	88
SceneManager	71
ViewBase	82
ViewEventTrigger	91
RegisteredInstance	70

2 Class Index 5

	SceneContext	70
	TypeMapping	76
	TypeRelation	77
	ViewModelCommandInfo	94
	ViewModelPropertyInfo	94
	ViewResolver	95
2	Class Index	
2.1	Class List	
He	re are the classes, structs, unions and interfaces with brief descriptions:	
	BindableProperty A wrapper for any class property so it can easily be bound to.	9
	Binding The base class for all bindings.	9
	CollisionEventBinding A component for binding to a collision.	12
	Command  A ViewModel command that can be executed. IEnumerator is always used so that any command can be a coroutine.	13
	CommandBinding  Base class for a command binding. Use this class if a different type of command binding is needed.	14
	CommandWith< T > A command with an argument of type T. Not usually bound to directly but used to forward a command to a parent viewmodel	15
	CommandWithSender < TSender >	16
	CommandWithSenderAndArgument < TSender, TArgument >	17
	ComponentBinding A Unity3d Component that will provide a binding to a specified View	17
	ComponentCommandBinding A component that will create a command binding and requires a component for the command to work.	19
	Controller A controller is a group of commands usually to provide an abstract level	19
	DefaultTypeResolver	23
	DiagramInfoAttribute	23
	ElementService Future name of controller.	24

EventBinding  The event binding component that will add an event binding to a source view.	<b>2</b> 4
FileSerializerStorage	25
GameContainer A ViewModel Container and a factory for Controllers and commands.	26
GameManager  A singleton that manages our current Scene Manager and all the games types in the scene.  This component will persist through every scene	29
IBinding Interface for all bindings	33
IBindingProvider	34
ICommand The base command interface for implementing a command in a ViewModel	34
$\textbf{ICommand} \! < \textbf{T} >$	35
ICommandWith< T > A base command interface for implementing a command with a parameter in a ViewModel	35
IGameContainer	35
IJsonSerializable	40
IModelCollection	40
InjectAttribute Used by the injection container to determine if a property or field should be injected.	41
InputBinding	41
ISerializer	42
ISerializerStorage	42
ISerializerStream	43
ITwoWayBinding	44
ITypeResolver	44
IUFSerializable	45
IView	45
IViewModelObserver Potential future use.	46
JSONArray	46
JSONClass	47
JSONData	48
JSONLazyCreator	49
JSONNode	49

2.1 Class List 7

JsonStream	51
KeyBinding  A component that will process a key binding as well as provide a key binding instance to the source view. Note. Even when adding this binding via code the component will still be added because a component is needed to process a keypress	52
LevelLoaderSceneManager	53
ModelCollection < T > An observable collection to use in viewmodels.	53
ModelCollectionBinding < TCollectionType >	54
ModelCollectionChangeEvent	55
ModelCollectionChangeEventWith< T >	56
ModelCollisionEventBinding A collision binding that will trigger a command when executed. Use chaining when possible to provide additional options for this binding.	56
ModelCommandBinding A base class for binding to a ViewModel command.	58
ModelEventBinding An event binding. Basically a wrapper for a .NET event so events can be triggered by a string. They can easily be bound and is mainly for conveniance.	59
ModelInputButtonBinding	60
ModelKeyBinding Binds a key to a ViewModel command.	60
ModelMouseEventBinding	62
ModelPropertyBase A base class for model properties.	62
ModelPropertyBinding A class that contains a binding from a ViewModel to a Target	64
ModelViewModelCollectionBinding Class for a view collection binding. Binds a ViewModel collection to a set of corresponding Views	65
ModelViewPropertyBinding	66
MouseEventBinding	67
P< T >     A typed ViewModel Property Class	68
RegisteredInstance	70
SceneContext  The scene context keeps track of view-models based on their identifiers when a view has checked "Save & Load"	70

SceneManager  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.	71
StateLoaderResolver	75
StringSerializerStorage	<b>75</b>
TypeInstanceCollection	76
TypeMapping	76
TypeMappingCollection	76
TypeRelation	77
TypeRelationCollection	77
UFGroup	77
UFPropertyBinding  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	78
UFRequireInstanceMethod	79
UFToggleGroup	79
View< TModel >	
A View is a visual representation of a ViewModel. For example: A UI dialog, Player, Weapon, e Template Parameters	etc
	etc
Template Parameters	etc
Template Parameters  TModel The ViewModel Type	tc
Template Parameters  TModel The ViewModel Type  80	82
Template Parameters  TModel The ViewModel Type  80  ViewBase	
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer	82 88
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer A base class for all view containers. Simply just utility methods for views and events.	82 88 88
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer	82 88
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer A base class for all view containers. Simply just utility methods for views and events.	82 88 88
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer A base class for all view containers. Simply just utility methods for views and events.  ViewEventTrigger  ViewModel	82 88 88 91
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer A base class for all view containers. Simply just utility methods for views and events.  ViewEventTrigger  ViewModel A data structure that contains information/data needed for a 'View'	82 88 88 91
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer A base class for all view containers. Simply just utility methods for views and events.  ViewEventTrigger  ViewModel A data structure that contains information/data needed for a 'View'  ViewModelCollectionBinding	82 88 88 91 91 93
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer A base class for all view containers. Simply just utility methods for views and events.  ViewEventTrigger  ViewModel A data structure that contains information/data needed for a 'View'  ViewModelCollectionBinding  ViewModelCommandInfo	82 88 88 91 91 93 94
Template Parameters  TModel The ViewModel Type  80  ViewBase The base class for a View that binds to a ViewModel  ViewComponent  ViewContainer A base class for all view containers. Simply just utility methods for views and events.  ViewEventTrigger  ViewModel A data structure that contains information/data needed for a 'View'  ViewModelCollectionBinding  ViewModelCommandInfo  ViewModelPropertyInfo	82 88 88 91 91 93 94

3 Class Documentation 9

YieldCommandWith< T >	
A coroutine command with a parameter.	96
YieldCommandWithSender< T >	
A coroutine command with a parameter.	97
YieldCommandWithSenderAndArgument < TSender, TArgument >	
A coroutine command with a parameter.	98

## 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-	The member of the ViewModel.
Name	

#### 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.lsComponent [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.

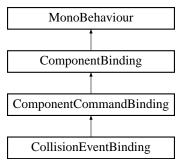
The documentation for this class was generated from the following file:

• 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 | Binding 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

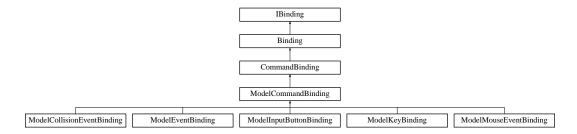
## 3.4.1 Detailed Description

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< |Command > 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.

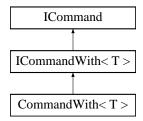
The documentation for this class was generated from the following file:

• 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** 

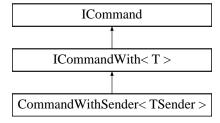
```
T | The argument parameter.
```

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, |Command oldCommand=null)
- 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**

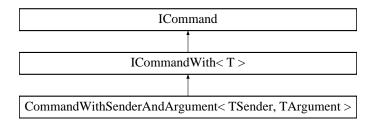
- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

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**

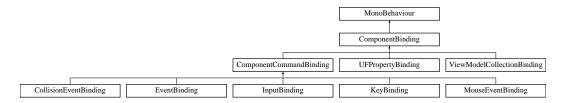
- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

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 > model-Properties)

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 < Key Value Pair < string, Model Property Base > > Component Binding. Filter Bindable Properties (
  Dictionary < string, Model Property Base > model Properties ) [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 | Binding 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, View-ModelCollectionBinding, and EventBinding.

## 3.9.3 Property Documentation

## **3.9.3.1 IBinding ComponentBinding.Binding** [get], [set]

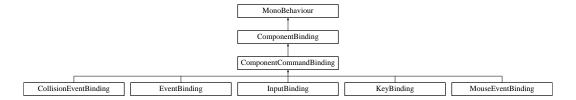
The binding that has been created for this component.

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]

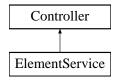
Simply a wrapper of "Binding" property cast to ModelCommandBinding

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 group of commands usually to provide an abstract level Inheritance diagram for Controller:



## **Public Member Functions**

· virtual ViewModel Create ()

Create a new ViewModel. This will generate a Unique Identifier for the VM. If this is a specific instance use the overload and pass an identifier.

virtual ViewModel Create (string identifier)

Creates a new ViewModel with a specific identifier. If it already exists in the SceneContext it will return that instead

• virtual ViewModel CreateEmpty (string identifier)

Create an empty view-model with the specified identifer. Note: This method does not wire up the view-model to this controller.

virtual ViewModel CreateEmpty ()

Create an empty view-model. Note: This method does not wire up the view-model to this controller and only instantiates an associated view-model.

- abstract void Initialize (ViewModel viewModel)
- virtual void WireCommands (ViewModel viewModel)
- void ExecuteCommand (ICommand command, object argument)
- virtual void ExecuteCommand (ICommand command)
- void ExecuteCommand < TArgument > (ICommandWith < TArgument > command, TArgument argument)
- virtual void GameEvent (string message, params object[] additionalParamters)

Send an event to our game

- UnityEngine.Coroutine StartCoroutine (IEnumerator routine)
- void StopAllCoroutines ()
- · void StopCoroutine (string name)
- ModelPropertyBinding SubscribeToProperty TViewModel, TBindingType > (TViewModel source, P < TBindingType > sourceProperty, Action < TViewModel, TBindingType > changedAction)

#### **Protected Member Functions**

Controller (SceneContext context)

Initialize this controller with a SceneContext object

void SubscribeToCommand (ICommand command, Action action)

#### **Properties**

IGameContainer Container [get, set]

The dependency container that this controller will use

SceneContext Context [get, set]

The scene context that contains the View-Models for the current scene.

## 3.11.1 Detailed Description

A controller is a group of commands usually to provide an abstract level

- 3.11.2 Constructor & Destructor Documentation
- **3.11.2.1 Controller.Controller ( SceneContext** *context* ) [protected]

Initialize this controller with a SceneContext object

Pa	ra	m	e	ŀΔ	re
гα	ı a			LC	ıa

context

3.11.3 Member Function Documentation

3.11.3.1 virtual ViewModel Controller.Create ( ) [virtual]

Create a new ViewModel. This will generate a Unique Identifier for the VM. If this is a specific instance use the overload and pass an identifier.

Returns

3.11.3.2 virtual ViewModel Controller.Create ( string identifier ) [virtual]

Creates a new ViewModel with a specific identifier. If it already exists in the SceneContext it will return that instead Parameters

identifier The identifier that will be used to check the context to see if it already exists.

Returns

3.11.3.3 virtual ViewModel Controller.CreateEmpty ( string identifier ) [virtual]

Create an empty view-model with the specified identifer. Note: This method does not wire up the view-model to this controller.

**Parameters** 

identifier

Returns

A new View-Model or the view-model found in the context with the same identifier.

**3.11.3.4 virtual ViewModel Controller.CreateEmpty ( )** [virtual]

Create an empty view-model . Note: This method does not wire up the view-model to this controller and only instantiates an associated view-model.

Returns

A new View-Model or the view-model found in the context with the same identifier.

3.11.3.5 virtual void Controller.GameEvent ( string message, params object[] additionalParamters ) [virtual]

Send an event to our game

**Parameters** 

message	
additional-	
Paramters	

## 3.11.4 Property Documentation

## **3.11.4.1 IGameContainer Controller.Container** [get], [set]

The dependency container that this controller will use

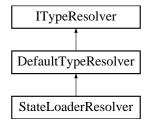
The scene context that contains the View-Models for the current scene.

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

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

# 3.12 DefaultTypeResolver Class Reference

Inheritance diagram for DefaultTypeResolver:



**Public Member Functions** 

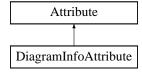
- Type GetType (string name)
- string **SetType** (Type type)
- · virtual object CreateInstance (string name, string identifier)

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

 $\bullet \ \ Assets/uFrameComplete/uFrame/Base/Serialization/DefaultTypeResolver.cs$ 

## 3.13 DiagramInfoAttribute Class Reference

Inheritance diagram for DiagramInfoAttribute:



**Public Member Functions** 

• DiagramInfoAttribute (string diagramName)

## **Properties**

• string DiagramName [get, set]

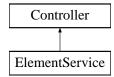
The documentation for this class was generated from the following file:

Assets/uFrameComplete/uFrame/Base/DiagramInfoAttribute.cs

## 3.14 ElementService Class Reference

Future name of controller.

Inheritance diagram for ElementService:



**Additional Inherited Members** 

## 3.14.1 Detailed Description

Future name of controller.

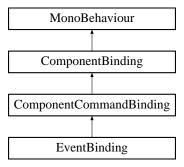
The documentation for this class was generated from the following file:

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

## 3.15 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** 

## 3.15.1 Detailed Description

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

#### 3.15.2 Member Function Documentation

## 3.15.2.1 override | Binding 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.

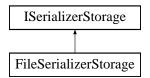
Implements ComponentBinding.

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

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

## 3.16 FileSerializerStorage Class Reference

Inheritance diagram for FileSerializerStorage:



## **Public Member Functions**

- FileSerializerStorage (string filename)
- · void Load (ISerializerStream stream)
- void Save (ISerializerStream stream)

## **Properties**

• string Filename [get, set]

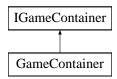
The documentation for this class was generated from the following file:

Assets/uFrameComplete/uFrame/Base/Serialization/Storage/FileSerializerStorage.cs

## 3.17 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.

IEnumerable < object > ResolveAll (Type type)

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

- void Register (Type source, Type target, string name=null)
- void RegisterInstance (Type baseType, object instance=null, bool injectNow=true)

Register a named instance

• virtual void RegisterInstance (Type baseType, object instance=null, string name=null, bool injectNow=true)

Register a named instance

- void RegisterInstance
   TBase > (TBase instance)
- void RegisterInstance < TBase > (TBase instance, bool injectNow)
- void **RegisterInstance**< **TBase** > (TBase instance, string name, bool injectNow=true)
- T Resolve < T > (string name=null, bool requireInstance=false)

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 baseType, string name=null, 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.

- TBase ResolveRelation < TBase > (Type tfor)
- void InjectAll ()

Injects everything that is registered at once

- void RegisterRelation< TFor, TBase, TConcrete > ()
- object **ResolveRelation** (Type tfor, Type tbase)
- TBase ResolveRelation < TFor, TBase > ()

## **Properties**

- TypeMappingCollection Mappings [get, set]
- TypeInstanceCollection Instances [get, set]
- TypeRelationCollection RelationshipMappings [get, set]

#### 3.17.1 Detailed Description

A ViewModel Container and a factory for Controllers and commands.

3.17.2 Member Function Documentation

3.17.2.1 void GameContainer.Clear ( )

Clears all type-mappings and instances.

Implements IGameContainer.

3.17.2.2 void GameContainer.Inject ( object obj )

Injects registered types/mappings into an object

**Parameters** 

```
obj |
```

Implements IGameContainer.

3.17.2.3 void GameContainer.InjectAll ( )

Injects everything that is registered at once

Implements IGameContainer.

3.17.2.4 void GameContainer.Register < TSource, TTarget > ( string name = null )

Register a type mapping

**Template Parameters** 

TSource	The base type.
TTarget	The concrete type

Implements IGameContainer.

3.17.2.5 void GameContainer.RegisterInstance ( Type baseType, object instance = null, bool injectNow = true )

Register a named instance

## Parameters

baseType	The type to register the instance for.
instance	The instance that will be resolved be the name
injectNow	Perform the injection immediately

3.17.2.6 virtual void GameContainer.RegisterInstance ( Type baseType, object instance = null, string name = null, bool injectNow = true ) [virtual]

Register a named instance

#### **Parameters**

baseType	The type to register the instance for.
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.

3.17.2.7 object GameContainer.Resolve ( Type baseType, string name = null, 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**

baseType	The type of instance to resolve
name	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.

3.17.2.8 T GameContainer.Resolve < T > ( string name = null, bool requireInstance = false )

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

3.17.2.9 IEnumerable < object > GameContainer.ResolveAll ( Type type )

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

**Template Parameters** 

ТТуре	The Type to resolve

## Returns

List of objects.

Implements IGameContainer.

3.17.2.10 | IEnumerable < TType > GameContainer.Resolve All < 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.

Implements IGameContainer.

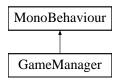
The documentation for this class was generated from the following file:

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

# 3.18 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 RegisterSceneManager (SceneManager sceneManager)

Registers a SceneManger with this game manager. This will invoke setup on the manager as well as disable it.

void ApplyRenderSettings ()

Applies the render settings specified in the inspector.

- virtual void OnEnable ()
- · void Awake ()

On awake will apply the render settings and will begin startup which will "boot" the scenemanager.

void Start ()

Checks if the gamemanager has already been loaded. If so it will copy necessary info and destroy itself. This also calls "Transition" in order to load the "Start" scene manager of the scene.

virtual void Startup ()

Startup will register every scenemanager in the scene. As well as set the "ActiveSceneManager' to the specified 'Start' scene manager specified in the inspector.

void LoadRenderSettings ()

Loads the current render settings of a scene.

void OnDestroy ()

When this is destroyed check if we are the "current instance" and set "Instance" to null. Note: This should really never happen. But in some test cases is necessary.

virtual void UnRegisterSceneManager (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 Transition< T > (Action< T > setup, UpdateProgressDelegate progress=null)
- static Coroutine SwitchGame < T > (Action < T > setup, UpdateProgressDelegate progress=null)
- static Coroutine **SwitchGameAndLevel**< **TGame** > (TGame controller, Action< TGame > setup=null, UpdateProgressDelegate progress=null)
- static void **SwitchGameAndLevel**< **T** > (SwitchLevelSettings< T > settings)
- static void **SwitchGameAndLevel**< **T** > (Action< T > setup, params string[] levels)
- static Coroutine Transition TGame > (TGame controller, Action TGame > setup=null, UpdateProgress-Delegate progress=null)

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

static void TransitionLevel
 T > (SwitchLevelSettings
 T > settings)

Transitions to another scene and loads additional scene if specified. game assuming that it will exist in the scene after loading is finished.

static void TransitionLevel< T > (Action< T > setup, params string[] levels)

Transitions to another scene and loads additional scene if specified. game assuming that it will exist in the scene after loading is finished.

• static IEnumerator Load ()

The uFrame Boot loader that willbegin the startup process

## **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

Do not use async loading on "TransitionLevel"

#### **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 > SceneManagers [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]

Is this a pro license?

## 3.18.1 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

3.18.2 Member Function Documentation

3.18.2.1 void GameManager.ApplyRenderSettings ( )

Applies the render settings specified in the inspector.

3.18.2.2 void GameManager.Awake ( )

On awake will apply the render settings and will begin startup which will "boot" the scenemanager.

**3.18.2.3** static | Enumerator GameManager.Load ( ) [static]

The uFrame Boot loader that willbegin the startup process

Returns

3.18.2.4 void GameManager.LoadRenderSettings ( )

Loads the current render settings of a scene.

3.18.2.5 void GameManager.OnDestroy ( )

When this is destroyed check if we are the "current instance" and set "Instance" to null. Note: This should really never happen. But in some test cases is necessary.

3.18.2.6 virtual void GameManager.RegisterSceneManager (SceneManager sceneManager) [virtual]

Registers a SceneManger with this game manager. This will invoke setup on the manager as well as disable it.

**Parameters** 

```
sceneManager The scene manager to register.
```

3.18.2.7 void GameManager.Start ( )

Checks if the gamemanager has already been loaded. If so it will copy necessary info and destroy itself. This also calls "Transition" in order to load the "Start" scene manager of the scene.

3.18.2.8 virtual void GameManager.Startup ( ) [virtual]

Startup will register every scenemanager in the scene. As well as set the "ActiveSceneManager' to the specified 'Start' scene manager specified in the inspector.

3.18.2.9 static Coroutine GameManager.Transition < 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

3.18.2.10 static void GameManager. Transition Level < T > ( Switch Level Settings < T > settings ) [static]

Transitions to another scene and loads additional scene if specified. game assuming that it will exist in the scene after loading is finished.

**Template Parameters** 

T	The SceneManager type that will exist in the first scene specified.

**Type Constraints** 

## T: SceneManager

3.18.2.11 static void GameManager. Transition Level < T > ( Action < T > setup, params string[] levels ) [static]

Transitions to another scene and loads additional scene if specified. game assuming that it will exist in the scene after loading is finished.

**Template Parameters** 

T	The SceneManager type that will exist in the first scene specified.

## **Parameters**

	setup	Perform additional setup when the scene has transitioned.
ſ	levels	The SceneManager type that will exist in the first scene specified.

**Type Constraints** 

## T: SceneManager

3.18.2.12 virtual void GameManager.UnRegisterSceneManager ( SceneManager sceneManager ) [virtual]

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

**Parameters** 

sceneMana	ager				

3.18.3 Member Data Documentation

3.18.3.1 bool GameManager.\_DontUseAsyncLoading = false

Do not use async loading on "TransitionLevel"

3.18.3.2 string GameManager.\_LoadingLevel

A level that displays a progress bar and message

3.18.3.3 SceneManager GameManager.\_Start

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

3.18.4 Property Documentation

**3.18.4.1 SceneManager GameManager.ActiveSceneManager** [static], [get], [set]

The current running game

**3.18.4.2 GameManager GameManager.Instance** [static], [get], [set]

The current instance of GameManager

**3.18.4.3** bool GameManager.lsPro [static], [get]

Is this a pro license?

**3.18.4.4 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.

3.18.4.5 List < SceneManager > GameManager. SceneManagers [get], [set]

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

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

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

## 3.19 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]
- bool TwoWay [get, set]

### 3.19.1 Detailed Description

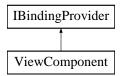
Interface for all bindings

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

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

## 3.20 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:

 $\bullet \ \ Assets/uFrameComplete/uFrame/Base/Bindings/IBindingProvider.cs$ 

## 3.21 | 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

### 3.21.1 Detailed Description

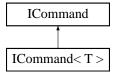
The base command interface for implementing a command in a ViewModel

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

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

## 3.22 | 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

## 3.23 | 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** 

### 3.23.1 Detailed Description

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

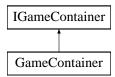
T

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

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

#### 3.24 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 InjectAll ()

Injects everything that is registered at once

void Register< TSource, TTarget > (string name=null)

Register a type mapping

- void RegisterRelation< TFor, TBase, TConcrete > ()
- void RegisterInstance < TBase > (TBase @default, bool injectNow)

Register an instance of a type.

• void RegisterInstance (Type type, object @default, bool injectNow)

Register an instance of a type.

void RegisterInstance (Type baseType, object instance=null, string name=null, bool injectNow=true)

Register a named instance

- void **RegisterInstance**< **TBase** > (TBase instance, string name, bool injectNow=true)
- void RegisterInstance < TBase > (TBase instance)
- T Resolve< T > (string name=null, bool requireInstance=false)

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

- TBase ResolveRelation < TBase > (Type tfor)
- TBase ResolveRelation < TFor, TBase > ()
- IEnumerable < TType > ResolveAll < TType > ()

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

- void **Register** (Type source, Type target, string name=null)
- IEnumerable < object > ResolveAll (Type type)

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

object Resolve (Type baseType, string name=null, 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.

object ResolveRelation (Type tfor, Type tbase)

## **Properties**

- TypeMappingCollection Mappings [get, set]
- TypeInstanceCollection Instances [get, set]
- TypeRelationCollection RelationshipMappings [get, set]
- 3.24.1 Member Function Documentation
- 3.24.1.1 void IGameContainer.Clear ( )

Clears all type mappings and instances.

Implemented in GameContainer.

3.24.1.2 void IGameContainer.Inject ( object obj )

Injects registered types/mappings into an object

D.			_ 1		
Pа	ra	m	eı	re	rs

obj	

Implemented in GameContainer.

3.24.1.3 void IGameContainer.InjectAll ( )

Injects everything that is registered at once

Implemented in GameContainer.

3.24.1.4 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.

3.24.1.5 void IGameContainer.RegisterInstance ( Type type, object @ default, bool injectNow )

Register an instance of a type.

### **Parameters**

type	
default	
injectNow	

Returns

3.24.1.6 void IGameContainer.RegisterInstance ( Type baseType, object instance = null, string name = null, bool injectNow = true )

Register a named instance

## **Parameters**

baseTyp	The type to register the instance for.
nam	The name for the instance to be resolved.
instand	The instance that will be resolved be the name
injectNo	Perform the injection immediately

Implemented in GameContainer.

 ${\it 3.24.1.7} \quad {\it void IGameContainer.RegisterInstance} {\it < TBase} > ( \ {\it TBase} @ \textit{ default, bool injectNow} \ )$ 

Register an instance of a type.

**Template Parameters** 

TBase	
•	

**Parameters** 

default	
injectNow	

Returns

**Type Constraints** 

TBase: class

3.24.1.8 object IGameContainer.Resolve ( Type baseType, string name = null, 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**

baseType	The type of instance to resolve
name	The type of instance to resolve
requireInstance	If true will return null if an instance isn't registered.

#### Returns

The/An instance of 'instanceType'

Implemented in GameContainer.

3.24.1.9 T IGameContainer.Resolve< T>( string name = null, bool requireInstance = false )

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

## **Template Parameters**

7	The type of instance to resolve
---	---------------------------------

Returns

The/An instance of 'instanceType'

Implemented in GameContainer.

**Type Constraints** 

T: class

3.24.1.10 IEnumerable < object > IGameContainer.ResolveAll ( Type type )

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

**Template Parameters** 

		ТТуре	The Type to resolve
--	--	-------	---------------------

## Returns

List of objects.

Implemented in GameContainer.

3.24.1.11 | IEnumerable < TType > | IGameContainer.ResolveAII < 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.

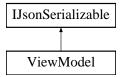
Implemented in GameContainer.

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

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

## 3.25 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

## 3.26 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**

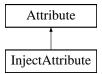
ModelCollectionChanged CollectionChanged

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

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

## 3.27 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]

### 3.27.1 Detailed Description

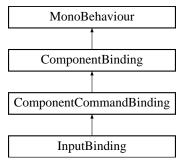
Used by the injection container to determine if a property or field should be injected.

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

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

## 3.28 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**

#### 3.28.1 Member Function Documentation

## **3.28.1.1 override | Binding | 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

## 3.29 ISerializer Interface Reference

**Public Member Functions** 

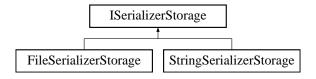
- IEnumerable < T > ReadArray < T > ()
- void WriteArray
   T > (T[] objs)
- void WriteObject (IUFSerializable obj)
- object ReadObject < T > (ISerializerStream stream)
- void SerializeField < T > (string name, T obj)
- object ReadField (string name)
- T ReadField < T > (string name)

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

• Assets/uFrameComplete/uFrame/Base/Serialization/ISerializer.cs

## 3.30 ISerializerStorage Interface Reference

Inheritance diagram for ISerializerStorage:



**Public Member Functions** 

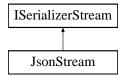
- · void Load (ISerializerStream stream)
- · void Save (ISerializerStream stream)

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

· Assets/uFrameComplete/uFrame/Base/Serialization/ISerializerStorage.cs

#### 3.31 ISerializerStream Interface Reference

Inheritance diagram for ISerializerStream:



#### **Public Member Functions**

- void **SerializeArray**< **T** > (string name, IEnumerable< T > items)
- void SerializeObjectArray (string name, IEnumerable< object > items)
- void SerializeObject (string name, object value)
- void **SerializeInt** (string name, int value)
- · void SerializeBool (string name, bool value)
- void SerializeString (string name, string value)
- void SerializeVector2 (string name, Vector2 value)
- · void SerializeVector3 (string name, Vector3 value)
- · void SerializeQuaternion (string name, Quaternion value)
- void SerializeDouble (string name, double value)
- · void SerializeFloat (string name, float value)
- · void SerializeBytes (string name, byte[] bytes)
- IEnumerable < T > DeserializeObjectArray < T > (string name)
- T **DeserializeObject** < T > (string name)
- object **DeserializeObject** (string name)
- int **DeserializeInt** (string name)
- bool **DeserializeBool** (string name)
- string **DeserializeString** (string name)
- Vector2 DeserializeVector2 (string name)
- Vector3 DeserializeVector3 (string name)
- Quaternion DeserializeQuaternion (string name)
- double DeserializeDouble (string name)
- · float DeserializeFloat (string name)
- byte[] DeserializeBytes (string name)
- void Load (byte[] readAllBytes)
- byte[] Save ()

### **Properties**

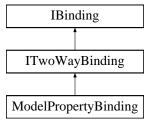
- IGameContainer DependencyContainer [get, set]
- Dictionary< string, IUFSerializable > ReferenceObjects [get, set]
   ITypeResolver TypeResolver [get, set]

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

• Assets/uFrameComplete/uFrame/Base/Serialization/ISerializerStream.cs

## 3.32 ITwoWayBinding Interface Reference

Inheritance diagram for ITwoWayBinding:



**Public Member Functions** 

void BindReverse ()
 Will be called every update frame

**Additional Inherited Members** 

- 3.32.1 Member Function Documentation
- 3.32.1.1 void ITwoWayBinding.BindReverse ( )

Will be called every update frame

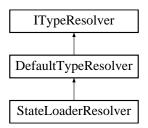
Implemented in ModelPropertyBinding.

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

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

## 3.33 ITypeResolver Interface Reference

Inheritance diagram for ITypeResolver:



**Public Member Functions** 

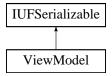
- Type **GetType** (string name)
- string **SetType** (Type type)
- object CreateInstance (string name, string identifier)

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

• Assets/uFrameComplete/uFrame/Base/Serialization/ITypeResolver.cs

## 3.34 IUFSerializable Interface Reference

Inheritance diagram for IUFSerializable:



**Public Member Functions** 

- void Write (ISerializerStream stream)
- void Read (ISerializerStream stream)

### **Properties**

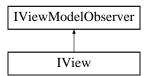
• string **Identifier** [get]

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

• Assets/uFrameComplete/uFrame/Base/Serialization/IUFSerializable.cs

## 3.35 IView Interface Reference

Inheritance diagram for IView:



## **Properties**

• 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** 

3.35.1 Property Documentation

**3.35.1.1 ViewModel IView.ViewModelObject** [get]

Gets the view model object.

The view model object.

**3.35.1.2 Type IView.ViewModelType** [get]

Gets the type of the view model.

The type of the model.

**3.35.1.3 string | View.ViewName** [get], [set]

The name of the prefab that created this view

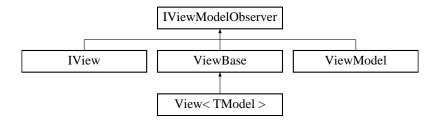
The documentation for this interface was generated from the following file:

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

### 3.36 IViewModelObserver Interface Reference

Potential future use.

Inheritance diagram for IViewModelObserver:



**Public Member Functions** 

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

### 3.36.1 Detailed Description

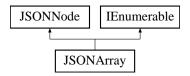
Potential future use.

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

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

## 3.37 JSONArray Class Reference

Inheritance diagram for JSONArray:



#### **Public Member Functions**

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

### **Properties**

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

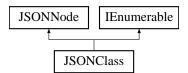
### **Additional Inherited Members**

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

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

## 3.38 JSONClass Class Reference

Inheritance diagram for JSONClass:



### **Public Member Functions**

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

### **Properties**

- override | Enumerable < JSONNode > Childs [get]
- override int Count [get]
- override JSONNode this[string aKey] [get, set]
- override JSONNode this[int alndex] [get, set]

#### **Additional Inherited Members**

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

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

## 3.39 JSONData Class Reference

Inheritance diagram for JSONData:



## **Public Member Functions**

- JSONData (Vector3 value)
- JSONData (Vector2 value)
- JSONData (Quaternion value)
- 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]

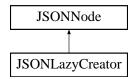
## **Additional Inherited Members**

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

Assets/uFrameComplete/uFrame/Base/SimpleJSON.cs

## 3.40 JSONLazyCreator Class Reference

Inheritance diagram for JSONLazyCreator:



#### **Public Member Functions**

- JSONLazyCreator (JSONNode aNode)
- JSONLazyCreator (JSONNode aNode, string aKey)
- override void Add (JSONNode altem)
- override void Add (string aKey, JSONNode altem)
- 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**

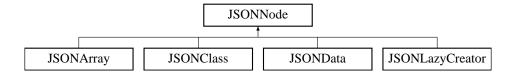
- override JSONArray AsArray [get]
- override bool AsBool [get, set]
- override double AsDouble [get, set]
- override float AsFloat [get, set]
- override int AsInt [get, set]
- override JSONClass AsObject [get]
- override JSONNode this[int alndex] [get, set]
- override JSONNode this[string aKey] [get,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

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

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

## 3.41 JSONNode Class Reference

Inheritance diagram for JSONNode:



## **Public Member Functions**

- virtual void Add (string aKey, JSONNode altem)
- virtual void Add (JSONNode altem)
- virtual JSONNode Remove (string aKey)
- virtual JSONNode Remove (int alndex)
- 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 alndex] [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]

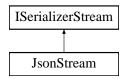
    virtual Vector4 AsVector4 [get, set]
```

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

Assets/uFrameComplete/uFrame/Base/SimpleJSON.cs

### 3.42 JsonStream Class Reference

Inheritance diagram for JsonStream:



#### **Public Member Functions**

- JsonStream (JSONNode node)
- JsonStream (string json)
- JsonStream (ITypeResolver typeResolver)
- JsonStream (ITypeResolver typeResolver, string json)
- · void Push (string name, JSONNode node)
- void Pop ()
- void Serialize (string name, object obj)
- void SerializeArray< T > (string name, IEnumerable< T > items)
- void SerializeObjectArray (string name, IEnumerable < object > items)
- void SerializeObject (string name, object value)
- · void SerializeInt (string name, int value)
- · void SerializeBool (string name, bool value)
- void **SerializeString** (string name, string value)
- void SerializeVector2 (string name, Vector2 value)
- · void SerializeVector3 (string name, Vector3 value)
- void SerializeQuaternion (string name, Quaternion value)
- · void SerializeDouble (string name, double value)
- · void SerializeFloat (string name, float value)
- void SerializeBytes (string name, byte[] bytes)
- IEnumerable < T > DeserializeObjectArray < T > (string name)
- T DeserializeObject < T > (string name)
- object **DeserializeObject** (string name)
- int **DeserializeInt** (string name)
- · bool DeserializeBool (string name)
- string **DeserializeString** (string name)
- Vector2 DeserializeVector2 (string name)
- Vector3 DeserializeVector3 (string name)
- Quaternion DeserializeQuaternion (string name)
- double DeserializeDouble (string name)
- float **DeserializeFloat** (string name)
- byte[] DeserializeBytes (string name)
- void Load (byte[] readAllBytes)
- byte[] Save ()

## **Properties**

- JSONNode RootNode [get, set]
- · Dictionary< string,
  - IUFSerializable > ReferenceObjects [get, set]
- ITypeResolver TypeResolver [get, set]
- Stack < JSONNode > NodeStack [get, set]
- JSONNode CurrentNode [get]

- IGameContainer DependencyContainer [get, set]
- bool UseReferences [get, set]

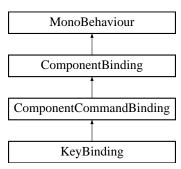
The documentation for this class was generated from the following file:

• Assets/uFrameComplete/uFrame/Base/Serialization/Json/JsonStream.cs

## 3.43 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**

### 3.43.1 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

### 3.43.2 Member Function Documentation

3.43.2.1 override | Binding 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.

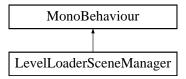
Implements ComponentBinding.

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

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

### 3.44 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

## 3.45 ModelCollection < T > Class Template Reference

An observable collection to use in viewmodels.

Inheritance diagram for ModelCollection< T >:



**Public Member Functions** 

- delegate void ModelCollectionChangedWith (ModelCollectionChangeEventWith< T > changeArgs)
- ModelCollection (List< T > value)
- void AddObject (object item)
- void RemoveObject (object item)
- ModelCollection (ViewModel owner, string propertyName)
- ModelCollection (ViewModel owner, string propertyName, IEnumerable
   T > enumerable)
- ModelCollection (IEnumerable < T > enumerable)
- virtual void Add (T item)
- override bool CanSetValue (List< T > value)
- virtual void Clear ()
- virtual bool Contains (Titem)
- 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 ()
- void AddRange (IEnumerable < T > value)
- int IndexOf (T item)
- void Insert (int index, T item)
- · void RemoveAt (int index)

### **Protected Member Functions**

- virtual void OnChangedWith (ModelCollectionChangeEventWith< T > changeargs)
- virtual void OnPropertyChanged (string propertyName)

### **Properties**

- int Count [get]
- bool **IsReadOnly** [get]
- override Type ValueType [get]
- Type ltemType [get]
- T this[int index] [get, set]

#### **Events**

- · ModelCollectionChanged CollectionChanged
- · ModelCollectionChangedWith CollectionChangedWith
- PropertyChangedEventHandler PropertyChanged

## **Additional Inherited Members**

## 3.45.1 Detailed Description

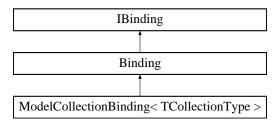
An observable collection to use in viewmodels.

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

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

## 3.46 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**

#### 3.46.1 Member Function Documentation

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

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

Reimplemented from Binding.

```
\textbf{3.46.1.2} \quad \textbf{override void ModelCollectionBinding} < \textbf{TCollectionType} > . \textbf{Unbind ( )} \quad [\texttt{virtual}]
```

Unbind this binding

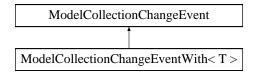
Reimplemented from Binding.

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

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

## 3.47 ModelCollectionChangeEvent Class Reference

Inheritance diagram for ModelCollectionChangeEvent:



#### **Properties**

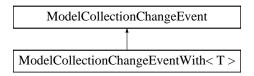
- ModelCollectionAction Action [get, set]
- object[] NewItems [get, set]
- object[] OldItems [get, set]

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

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

## 3.48 ModelCollectionChangeEventWith< T > Class Template Reference

Inheritance diagram for ModelCollectionChangeEventWith< T >:



#### **Properties**

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

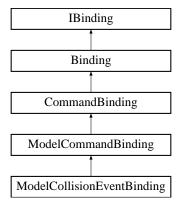
The documentation for this class was generated from the following file:

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

## 3.49 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** 

#### 3.49.1 Detailed Description

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

- 3.49.2 Member Function Documentation
- **3.49.2.1 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.

3.49.2.2 ModelCollisionEventBinding ModelCollisionEventBinding.SetParameterSelector ( Func < GameObject, object > commandArgSelector )

Set the parameter that will be passed to the command.

## Parameters

commandArg-	A selector that will select the object to pass to the command with the collider as the first
Selector	argument

Returns

**3.49.2.3** 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.

3.49.2.4 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.

### 3.49.3 Property Documentation

 $\textbf{3.49.3.1} \quad \textbf{CollisionEventType ModelCollisionEventBinding.CollisionEvent} \quad \texttt{[get], [set]}$ 

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

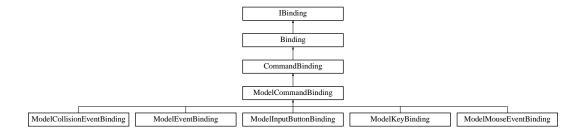
The documentation for this class was generated from the following file:

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

## 3.50 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**

## 3.50.1 Detailed Description

A base class for binding to a ViewModel command.

#### 3.50.2 Member Function Documentation

**3.50.2.1** 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.

**3.50.2.2** override void ModelCommandBinding.Unbind() [virtual]

Unbind this binding

Reimplemented from CommandBinding.

Reimplemented in ModelEventBinding.

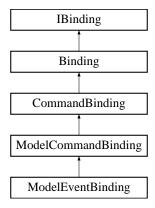
The documentation for this class was generated from the following file:

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

## 3.51 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** 

### 3.51.1 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.

### 3.51.2 Member Function Documentation

```
3.51.2.1 override void ModelEventBinding.Bind ( ) [virtual]
```

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

Reimplemented from ModelCommandBinding.

```
3.51.2.2 override void ModelEventBinding.Unbind() [virtual]
```

Unbind this binding

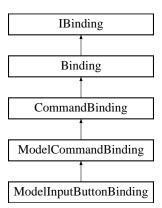
Reimplemented from ModelCommandBinding.

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

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

## 3.52 ModelInputButtonBinding Class Reference

Inheritance diagram for ModelInputButtonBinding:



## **Properties**

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

**Additional Inherited Members** 

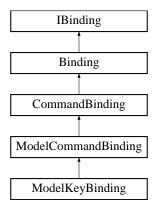
The documentation for this class was generated from the following file:

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

## 3.53 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**

### 3.53.1 Detailed Description

Binds a key to a ViewModel command.

### 3.53.2 Member Function Documentation

## 3.53.2.1 ModelKeyBinding ModelKeyBinding.RequireAlt ( )

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

### Returns

This to respect chaining.

### 3.53.2.2 ModelKeyBinding ModelKeyBinding.RequireControl ( )

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

#### Returns

This to respect chaining.

## 3.53.2.3 ModelKeyBinding ModelKeyBinding.RequireShift ( )

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

### Returns

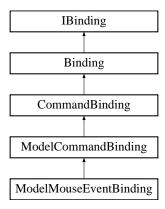
This to respect chaining.

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

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

## 3.54 ModelMouseEventBinding Class Reference

Inheritance diagram for ModelMouseEventBinding:



## **Properties**

MouseEventType EventType [get, set]

### **Additional Inherited Members**

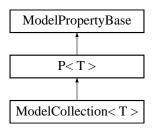
The documentation for this class was generated from the following file:

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

# 3.55 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 Member Functions** 

• ModelPropertyBase (ViewModel owner, string propertyName)

**Protected Attributes** 

· object \_value

### **Properties**

- ViewModel Owner [get, set]
- string PropertyName [get, set]
- virtual object ObjectValue [get, set]

The value of this model property

- object LastValueObject [get, set]
- virtual Type ValueType [get]

The value type of this property

#### **Events**

• PropertyChangedHandler ValueChanged

When the value has changed

3.55.1 Detailed Description

A base class for model properties.

- 3.55.2 Member Function Documentation
- 3.55.2.1 void ModelPropertyBase.QuietlySetValue (object value)

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

**Parameters** 

value

#### 3.55.3 Property Documentation

**3.55.3.1** virtual object ModelPropertyBase.ObjectValue [get], [set]

The value of this model property

**3.55.3.2** virtual Type ModelPropertyBase.ValueType [get]

The value type of this property

#### 3.55.4 Event Documentation

### 3.55.4.1 PropertyChangedHandler ModelPropertyBase.ValueChanged

When the value has changed

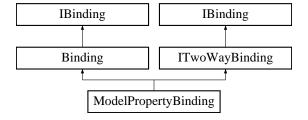
The documentation for this class was generated from the following file:

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

## 3.56 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**

### 3.56.1 Detailed Description

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

#### 3.56.2 Member Function Documentation

```
3.56.2.1 override void ModelPropertyBinding.Bind() [virtual]
```

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

Reimplemented from Binding.

```
3.56.2.2 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.

```
3.56.2.3 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

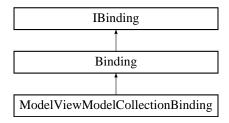
Reimplemented from Binding.

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

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

### 3.57 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]

### 3.57.1 Detailed Description

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

### 3.57.2 Member Function Documentation

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

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

Reimplemented from Binding.

**3.57.2.2** override void ModelViewModelCollectionBinding.Unbind() [virtual]

Unbind this binding

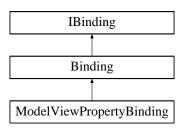
Reimplemented from Binding.

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

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

#### 3.58 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**

#### 3.58.1 Member Function Documentation

```
\textbf{3.58.1.1} \quad \textbf{override void ModelViewPropertyBinding.Bind ( )} \quad [\texttt{virtual}]
```

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

Reimplemented from Binding.

```
3.58.1.2 override void ModelViewPropertyBinding.Unbind() [virtual]
```

Unbind this binding

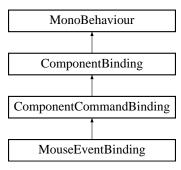
Reimplemented from Binding.

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

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

## 3.59 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**

### 3.59.1 Member Function Documentation

**3.59.1.1** override | Binding | 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.

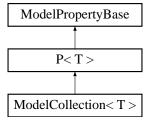
The documentation for this class was generated from the following file:

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

### 3.60 P<T> Class Template Reference

A typed ViewModel Property Class

Inheritance diagram for P< T>:



### **Public Member Functions**

- P (ViewModel owner, string propertyName)
- P (ViewModel owner, string propertyName, T value)
- 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.

- T LastValue [get]
- override Type ValueType [get]

Gets the type of the value.

**Additional Inherited Members** 

3.60.1 Detailed Description

A typed ViewModel Property Class

**Template Parameters** 

Т

3.60.2 Member Function Documentation

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

Descrialize the specified node into Value.

**Parameters** 

```
node Node.
```

Implements ModelPropertyBase.

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

Serializes this object

Implements ModelPropertyBase.

3.60.3 Property Documentation

```
3.60.3.1 TP<T>.Value [get], [set]
```

Gets or sets the value.

The value.

```
3.60.3.2 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

# 3.61 RegisteredInstance Class Reference

### **Properties**

```
Type Base [get, set]object Instance [get, set]string Name [get, set]
```

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

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

### 3.62 SceneContext Class Reference

The scene context keeps track of view-models based on their identifiers when a view has checked "Save & Load"

#### **Public Member Functions**

- SceneContext (IGameContainer gameContainer)
- TViewModel CreateViewModel < TViewModel > (Controller controller, string identifier)
- void Load (ISerializerStorage storage, ISerializerStream stream)

Load's a set of view-models from a storage medium based on a stream.

 void Save (ISerializerStorage storage, ISerializerStream stream, IEnumerable ViewModel > view-Models=null)

Saves

### **Properties**

- ViewModel this[Type type] [get, set]
- ViewModel this[string identifier] [get, set]
- IGameContainer Container [get, set]
- Dictionary< string, ViewModel > ViewModels [get, set]

The dictionary of ViewModels currently loaded in the scene that have been marked as persistant.

• Dictionary< string, ViewModel > PersitantViewModels [get, set]

### 3.62.1 Detailed Description

The scene context keeps track of view-models based on their identifiers when a view has checked "Save & Load"

### 3.62.2 Member Function Documentation

3.62.2.1 void SceneContext.Load ( ISerializerStorage storage, ISerializerStream stream )

Load's a set of view-models from a storage medium based on a stream.

### **Parameters**

storage	This is for loading the stream from a persistant medium. e.g. File, Stringetc
stream	The type of stream to serialize as. eg. Json,Xml,Binary

3.62.2.2 void SceneContext.Save ( ISerializerStorage *storage*, ISerializerStream *stream*, IEnumerable < ViewModel > viewModels = null )

Saves

#### **Parameters**

storage	
stream	
viewModels	

#### 3.62.3 Property Documentation

**3.62.3.1 Dictionary**<string, ViewModel> SceneContext.ViewModels [get], [set]

The dictionary of ViewModels currently loaded in the scene that have been marked as persistant.

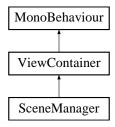
The documentation for this class was generated from the following file:

Assets/uFrameComplete/uFrame/Base/Scene/SceneContext.cs

### 3.63 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 the load function has completed

virtual void OnLoading ()

This method is called when this controller has started loading

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 ()

This method is called by the GameManager in order to register any dependencies. It is one of the first things to be invoked. This method is called before the "Load" method.

virtual void Unload ()

This method should be used to property unload a scene when transitioning to another scene.

• TViewModel SetupViewModel < TViewModel > (Controller controller, string identifier)

Used by the SceneManager when creating an instance before the scene loads. This allows a view-model instance to be ready before a view-initializes it. This is used by the uFrame generators to initialize single isntance view-models.

ViewModel RequestViewModel (ViewBase viewBase, Controller controller, string identifier)

This is method is called by each view in order to get it's view-model as well as place it in the SceneContext if the "Save & Load" option is checked in it's inspector

#### **Protected Member Functions**

virtual void Awake ()

The awake method of this scenemanager simply registers itself with the GameManager

virtual void OnDestroy ()

When this scene manager is destroy it is removed from the gamemanager.

#### **Properties**

• IGameContainer Container [get, set]

The Dependency container for this scene. If unset then it will use "GameManager.Container".

• SceneContext Context [get, set]

The scene context for the current running scene. Used for Saving and loading a scenes state.

static ISwitchLevelSettings Settings [get]

The settings at which the level will be loaded with. Used for transitioning from one scene to another.

#### 3.63.1 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.

```
3.63.2 Member Function Documentation
```

```
3.63.2.1 virtual void SceneManager.Awake ( ) [protected], [virtual]
```

The awake method of this scenemanager simply registers itself with the GameManager

```
3.63.2.2 virtual | Enumerator SceneManager.Load ( UpdateProgressDelegate progress ) [virtual]
```

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

Returns

```
3.63.2.3 virtual void SceneManager.OnDestroy ( ) [protected], [virtual]
```

When this scene manager is destroy it is removed from the gamemanager.

```
3.63.2.4 virtual void SceneManager.OnLoaded() [virtual]
```

This method is called when the load function has completed

```
3.63.2.5 virtual void SceneManager.OnLoading() [virtual]
```

This method is called when this controller has started loading

```
3.63.2.6 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

3.63.2.7 ViewModel SceneManager.RequestViewModel ( ViewBase viewBase, Controller controller, string identifier )

This is method is called by each view in order to get it's view-model as well as place it in the SceneContext if the "Save & Load" option is checked in it's inspector

#### **Parameters**

viewBase	The view that is requesting it's view-model.
controller	The controller that should be assigned to the view-model if any.
identifier	The identifier of the view-model to be created or loaded (if reloading a scenes state).

#### Returns

A new view model or the view-model with the identifier specified found in the scene context.

3.63.2.8 virtual void SceneManager.Setup ( ) [virtual]

This method is called by the GameManager in order to register any dependencies. It is one of the first things to be invoked. This method is called before the "Load" method.

3.63.2.9 TViewModel SceneManager.SetupViewModel < TViewModel > ( Controller controller, string identifier )

Used by the SceneManager when creating an instance before the scene loads. This allows a view-model instance to be ready before a view-initializes it. This is used by the uFrame generators to initialize single isntance view-models.

#### **Template Parameters**

TViewModel	The type of view-model to create.

#### **Parameters**

controller	The controller that the view-model should be initialized with
identifier	The identifier of the view-model to be created or loaded (if reloading a scenes state).

### Returns

A new view model or the view-model with the identifier specified found in the scene context.

### **Type Constraints**

TViewModel: ViewModel
TViewModel: new()

**3.63.2.10** virtual void SceneManager.Unload ( ) [virtual]

This method should be used to property unload a scene when transitioning to another scene.

3.63.3 Property Documentation

**3.63.3.1 IGameContainer SceneManager.Container** [get], [set]

The Dependency container for this scene. If unset then it will use "GameManager.Container".

**3.63.3.2 SceneContext SceneManager.Context** [get], [set]

The scene context for the current running scene. Used for Saving and loading a scenes state.

**3.63.3.3 ISwitchLevelSettings SceneManager.Settings** [static], [get]

The settings at which the level will be loaded with. Used for transitioning from one scene to another.

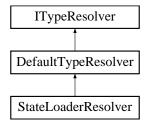
The settings.

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

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

### 3.64 StateLoaderResolver Class Reference

Inheritance diagram for StateLoaderResolver:



**Public Member Functions** 

- StateLoaderResolver (SceneContext context)
- override object CreateInstance (string name, string identifier)

### **Properties**

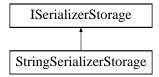
• SceneContext Context [get, set]

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

• Assets/uFrameComplete/uFrame/Base/Serialization/StateLoaderResolver.cs

# 3.65 StringSerializerStorage Class Reference

Inheritance diagram for StringSerializerStorage:



**Public Member Functions** 

- · void Load (ISerializerStream stream)
- void Save (ISerializerStream stream)
- override string ToString ()

# **Properties**

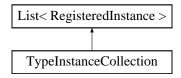
• string Result [get, set]

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

Assets/uFrameComplete/uFrame/Base/Serialization/Storage/StringSerializerStorage.cs

# 3.66 TypeInstanceCollection Class Reference

Inheritance diagram for TypeInstanceCollection:



### **Properties**

object this[Type from, string name=null] [get, set]

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

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

### 3.67 TypeMapping Class Reference

### **Properties**

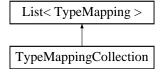
Type From [get, set]Type To [get, set]string Name [get, set]

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

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

# 3.68 TypeMappingCollection Class Reference

Inheritance diagram for TypeMappingCollection:



### **Properties**

• Type this[Type from, string name=null] [get, set]

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

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

# 3.69 TypeRelation Class Reference

**Properties** 

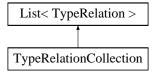
```
Type From [get, set]Type To [get, set]Type Concrete [get, set]
```

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

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

# 3.70 TypeRelationCollection Class Reference

Inheritance diagram for TypeRelationCollection:



### **Properties**

• Type this[Type from, Type to] [get, set]

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

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

# 3.71 UFGroup Class Reference

Inheritance diagram for UFGroup:



**Public Member Functions** 

• **UFGroup** (string viewModelProperties)

# **Properties**

• string Name [get, set]

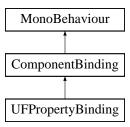
The documentation for this class was generated from the following file:

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

# 3.72 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**

#### 3.72.1 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.

### 3.72.2 Member Function Documentation

### **3.72.2.1 override lBinding 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.

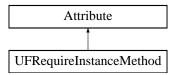
Implements ComponentBinding.

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

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

# 3.73 UFRequireInstanceMethod Class Reference

Inheritance diagram for UFRequireInstanceMethod:



**Public Member Functions** 

• UFRequireInstanceMethod (string canmovetochanged)

**Properties** 

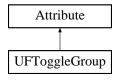
• string MethodName [get, set]

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

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

### 3.74 UFToggleGroup Class Reference

Inheritance diagram for UFToggleGroup:



**Public Member Functions** 

• **UFToggleGroup** (string checkers)

**Properties** 

• string Name [get, set]

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

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

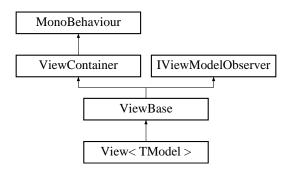
3.75	View<	<b>TModel</b>	>	Class	<b>Template</b>	Reference
------	-------	---------------	---	-------	-----------------	-----------

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

### **Template Parameters**



Inheritance diagram for View < TModel >:



#### **Protected 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.

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**

# 3.75.1 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()

### 3.75.2 Member Function Documentation

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.

3.75.2.2 virtual void View < TModel > .Initialize View Model ( TModel view Model ) [protected], [virtual]

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

**Parameters** 

```
viewModel
```

### 3.75.3 Property Documentation

```
3.75.3.1 TModel View < TModel > .Model [get], [set]
```

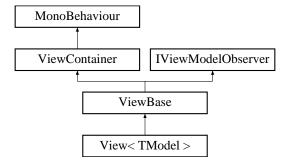
Gets or sets the ViewModel. Note: The setter will reinvoke the bind method. To set quietly use ViewModelObject The documentation for this class was generated from the following file:

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

### 3.76 ViewBase Class Reference

The base class for a View that binds to a ViewModel

Inheritance diagram for ViewBase:



# **Public Member Functions**

delegate void ViewEvent (string eventName)

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

· virtual void AfterBind ()

This method is invoked right after it has been bound

- virtual void Awake ()
- virtual void Bind ()

This method is called in order to subscribe to properties, commands, and collections.

abstract ViewModel CreateModel ()

This method is called in order to create a model for this view. In a uFrame Designer generated view it will implement this method and call the "RequestViewModel" on the scene manager.

void ExecuteCommand (ICommand command, object argument)

All of the designer generated "Execute{CommandName}" ultimately use this method. So when need to execute a command on an outside view-model(meaning not the view-model of this view) this method can be used. e.g. ExecuteCommand(command, argument)

virtual void ExecuteCommand (ICommand command)

All of the designer generated "Execute{CommandName}" ultimately use this method. So when need to execute a command on an outside view-model(meaning not the view-model of this view) this method can be used. e.g. ExecuteCommand(MyGameViewModel.MainMenuCommand)

void ExecuteCommand< TArgument > (ICommandWith< TArgument > command, ViewModel sender, T-Argument argument)

Executes a command of type ICommand.

void ExecuteCommand < TArgument > (ICommandWith < TArgument > command, TArgument argument)

Executes a command of type ICommand.

void InitializeData (ViewModel model)

A wrapper for "InitializeViewModel".

virtual void OnDestroy ()

When this view is destroy it will decrement the ViewModel's reference count. If the reference count reaches 0 it will call "Unbind" on the viewmodel properly unbinding anything subscribed to it.

- 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 Start ()

The start method approparitely initializes the "ChildViews" collection.

void AddBinding (IBinding binding)

Adds a binding to the view-model's binding dictionary for this view.

void RemoveBinding (IBinding binding)

Removes a binding from the view-models binding dictionary for this view.

virtual void Unbind ()

Unbind the current bindings.

### **Public Attributes**

bool LogEvents

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

# **Protected Member Functions**

• virtual void PreBind ()

This method is called immediately before "Bind". This method is used by uFrames designer generated code to set-up defined bindings.

• abstract void InitializeViewModel (ViewModel model)

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

• virtual void LateUpdate ()

Just calls the apply method.

· virtual void Apply ()

Overriden by the the uFrame designer to apply any two-way/reverse properties.

ViewModel RequestViewModel (Controller controller)

Request a view-model with a given controller.

• virtual ViewBase ReplaceView (ViewBase current, ViewModel value, GameObject prefab)

#### **Properties**

- List< |BindingProvider > BindingProviders [get, set]
- IEnumerable < ViewModel > ChildViewModels [get]
- List < ViewBase > ChildViews [get, set]
- bool ForceResolveViewModel [get, set]
- bool Instantiated [get, set]
- virtual bool **IsMultiInstance** [get]
- bool OverrideViewModel [get, set]
- ViewBase ParentView [get, set]
- ViewModel ParentViewModel [get]
- virtual ViewModel ViewModelObject [get, set]
- abstract Type **ViewModelType** [get]
- virtual string DefaultIdentifier [get]

This is the default identifier to use when "ResolveName" is not specified and it's a single instance. This field is automatically overriden by the uFrame designer.

• string ViewName [get, set]

The name of the prefab that created this view

• List< |Binding > Bindings [get]

A wrapper for this view's viewmodel bindings. It is a wrapper for ViewModel.Bindings[gameObject.GetInstanceId()]

• int InstanceId [get]

A lazy loaded property for "GetInstanceId" on the game-object.

• virtual string Identifier [get, set]

The identifier used for requesting a view-model. Implementation Details: -If its not a multiinstance viewmodel and the "ResolveName" is empty it will use "DefaultIdentifier" property otherwise it will use the resolve name.

- SceneManager SceneManager [get]
- bool **IsBound** [get, set]
- bool Save [get, set]

Should this view be saved in the "SceneContext"

bool InjectView [get, set]

#### **Events**

ViewEvent EventTriggered

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

3.76.1 Detailed Description

The base class for a View that binds to a ViewModel

- 3.76.2 Member Function Documentation
- 3.76.2.1 void ViewBase.AddBinding ( IBinding binding )

Adds a binding to the view-model's binding dictionary for this view.

**Parameters** 

binding

Implements IViewModelObserver.

3.76.2.2 virtual void ViewBase.AfterBind ( ) [virtual]

This method is invoked right after it has been bound

3.76.2.3 virtual void ViewBase.Apply() [protected], [virtual]

Overriden by the the uFrame designer to apply any two-way/reverse properties.

3.76.2.4 virtual void ViewBase.Bind() [virtual]

This method is called in order to subscribe to properties, commands, and collections.

3.76.2.5 abstract ViewModel ViewBase.CreateModel() [pure virtual]

This method is called in order to create a model for this view. In a uFrame Designer generated view it will implement this method and call the "RequestViewModel" on the scene manager.

#### Returns

A view model for this view to bind to

3.76.2.6 void ViewBase.ExecuteCommand ( ICommand command, object argument )

All of the designer generated "Execute{CommandName}" ultimately use this method. So when need to execute a command on an outside view-model(meaning not the view-model of this view) this method can be used. e.g. ExecuteCommand(command, argument)

#### **Parameters**

command	The command to execute e.g. MyGameViewModel.MainMenuCommand
argument	The argument to pass along if needed.

3.76.2.7 virtual void ViewBase.ExecuteCommand ( ICommand command ) [virtual]

All of the designer generated "Execute{CommandName}" ultimately use this method. So when need to execute a command on an outside view-model(meaning not the view-model of this view) this method can be used. e.g. ExecuteCommand(MyGameViewModel.MainMenuCommand)

### **Parameters**

command The command to execute e.g. MyGameViewModel.MainMenuCommand	
---	--

3.76.2.8 void ViewBase.ExecuteCommand< TArgument > ( ICommandWith< TArgument > command, ViewModel sender, TArgument argument )

Executes a command of type ICommand.

# **Template Parameters**

TArgument		
rargument	TArgument	

#### **Parameters**

command	The command instance to execute.
sender	The sender of the command.
argument	The argument required by the command.

3.76.2.9 void ViewBase.ExecuteCommand< TArgument> ( ICommandWith< TArgument> command, TArgument argument )

Executes a command of type ICommand.

### **Template Parameters**

TArgument	

#### **Parameters**

command	The command instance to execute.
argument	The argument required by the command.

3.76.2.10 void ViewBase.InitializeData ( ViewModel model )

A wrapper for "InitializeViewModel".

**Parameters** 

model

**3.76.2.11** abstract void ViewBase.InitializeViewModel ( ViewModel model ) [protected], [pure virtual]

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

#### **Parameters**

model	The model to initialize.	

Implemented in View < TModel >.

**3.76.2.12 virtual void ViewBase.LateUpdate()** [protected], [virtual]

Just calls the apply method.

3.76.2.13 virtual void ViewBase.OnDestroy() [virtual]

When this view is destroy it will decrement the ViewModel's reference count. If the reference count reaches 0 it will call "Unbind" on the viewmodel properly unbinding anything subscribed to it.

3.76.2.14 virtual void ViewBase.PreBind() [protected], [virtual]

This method is called immediately before "Bind". This method is used by uFrames designer generated code to set-up defined bindings.

3.76.2.15 void ViewBase.RemoveBinding ( IBinding binding )

Removes a binding from the view-models binding dictionary for this view.

**Parameters** 

binding

Implements IViewModelObserver.

**3.76.2.16 ViewModel ViewBase.RequestViewModel ( Controller controller )** [protected]

Request a view-model with a given controller.

**Parameters** 

controller

Returns

```
3.76.2.17 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.

```
3.76.2.18 virtual void ViewBase.Start() [virtual]
```

The start method approparitely initializes the "ChildViews" collection.

```
3.76.2.19 virtual void ViewBase.Unbind() [virtual]
```

Unbind the current bindings.

Implements IViewModelObserver.

3.76.2.20 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.
```

3.76.3 Member Data Documentation

3.76.3.1 bool ViewBase. LogEvents

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

3.76.4 Property Documentation

```
3.76.4.1 List<IBinding> ViewBase.Bindings [get]
```

A wrapper for this view's viewmodel bindings. It is a wrapper for ViewModel.Bindings[gameObject.GetInstanceId()]

```
3.76.4.2 virtual string ViewBase.DefaultIdentifier [get]
```

This is the default identifier to use when "ResolveName" is not specified and it's a single instance. This field is automatically overriden by the uFrame designer.

```
3.76.4.3 virtual string ViewBase.ldentifier [get], [set]
```

The identifier used for requesting a view-model. Implementation Details: -If its not a multiinstance viewmodel and the "ResolveName" is empty it will use "DefaultIdentifier" property otherwise it will use the resolve name.

- If it's a multiinstance viewmodel and the resolvename is specified it will use that.
- If the Use Hashcode as identifier is checked it will use this views hashcode.

Note: If using a prefab that is placed in the Unity editor in various places around a scene and it still needs to be unique every scene load (for scene loading and saving) you will want to override this property and supply a identifier that makes it unique.

```
3.76.4.4 int ViewBase.InstanceId [get]
```

A lazy loaded property for "GetInstanceId" on the game-object.

```
3.76.4.5 bool ViewBase.Save [get], [set]
```

Should this view be saved in the "SceneContext"

**3.76.4.6** string ViewBase.ViewName [get], [set]

The name of the prefab that created this view

#### 3.76.5 Event Documentation

### 3.76.5.1 ViewEvent ViewBase.EventTriggered

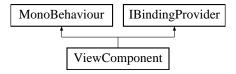
An event that is invoked whe calling Event("MyEvent")

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

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

# 3.77 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]

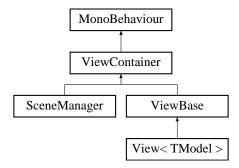
The documentation for this class was generated from the following file:

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

#### 3.78 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, string identifier=null)
- ViewBase InstantiateView (string viewName, ViewModel model, string identifier=null)

Instantiates a view.

• ViewBase InstantiateView (string viewName, Vector3 position, string identifier=null)

Instantiates a view.

ViewBase InstantiateView (string viewName, ViewModel model, Vector3 position, string identifier=null)

Instantiates a view.

 ViewBase InstantiateView (string viewName, ViewModel model, Vector3 position, Quaternion rotation, string identifier=null)

Instantiates a view.

 ViewBase InstantiateView (GameObject prefab, ViewModel model, Vector3 position, Quaternion rotation, string identifier=null)

Instantiates a view.

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

### 3.78.1 Detailed Description

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

### 3.78.2 Member Function Documentation

3.78.2.1 ViewBase ViewContainer.InstantiateView ( string viewName, ViewModel model, string identifier = null)

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

3.78.2.2 ViewBase ViewContainer.InstantiateView ( string viewName, Vector3 position, string identifier = null)

Instantiates a view.

**Parameters** 

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

### Returns

The instantiated view

3.78.2.3 ViewBase ViewContainer.InstantiateView ( string viewName, ViewModel model, Vector3 position, string identifier = null)

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

3.78.2.4 ViewBase ViewContainer.InstantiateView ( string *viewName*, ViewModel *model*, Vector3 *position*, Quaternion *rotation*, string *identifier* = null)

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

3.78.2.5 ViewBase ViewContainer.InstantiateView ( GameObject *prefab*, ViewModel *model*, Vector3 *position*, Quaternion *rotation*, string *identifier* = null)

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

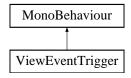
The instantiated view

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

Assets/uFrameComplete/uFrame/Base/ViewContainer.cs

# 3.79 ViewEventTrigger Class Reference

Inheritance diagram for ViewEventTrigger:

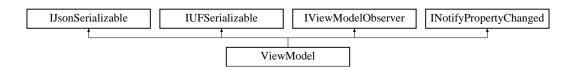


The documentation for this class was generated from the following file:

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

### 3.80 ViewModel Class Reference

A data structure that contains information/data needed for a 'View' Inheritance diagram for ViewModel:



# **Public Member Functions**

- void AddBinding (IBinding binding)
- void RemoveBinding (IBinding binding)
- virtual void Unbind ()
- virtual void **Deserialize** (JSONNode node)
- · virtual IEnumerable
  - < ModelPropertyBase > GetProperties ()

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

- virtual JSONNode Serialize ()
- override string ToString ()
- virtual void Write (ISerializerStream stream)
- virtual void Read (ISerializerStream stream)
- virtual void **OnPropertyChanged** (string propertyName)
- List< ViewModelPropertyInfo > GetViewModelProperties ()
- List< ViewModelCommandInfo > GetViewModelCommands ()

# **Static Public Member Functions**

static Dictionary< string,</li>

PropertyInfo > GetReflectedCommands (Type modelType)

Grabs all the commands available for a viewmodel type

static Dictionary< string,</li>

FieldInfo > GetReflectedModelProperties (Type modelType)

Grab the bindable properties for the view-model

#### **Protected Member Functions**

- ICommand Command (Action command)
- ICommand Command (Func< IEnumerator > command)
- virtual void WireCommands (Controller controller)
- virtual void FillProperties (List< ViewModelPropertyInfo > list)
- virtual void FillCommands (List< ViewModelCommandInfo > list)

#### **Properties**

- Dictionary < int, List < IBinding > > Bindings [get, set]
- int References [get, set]
- virtual string **Identifier** [get, set]
- 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

- Dictionary < string, |Command > Commands [get]
- · Dictionary< string,

```
ModelPropertyBase > Properties [get]
```

- Controller Controller [get, set]
- bool Dirty [get, set]

#### **Events**

• PropertyChangedEventHandler PropertyChanged

### 3.80.1 Detailed Description

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

3.80.2 Member Function Documentation

```
3.80.2.1 virtual | Enumerable < Model | Property | Base > View | Model | Get | Properties ( ) [virtual]
```

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

Returns

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

Grabs all the commands available for a viewmodel type

**Parameters** 

```
modelType |
```

Returns

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

Grab the bindable properties for the view-model

**Parameters** 

modelType	
-----------	--

Returns

### 3.80.3 Property Documentation

### 3.80.3.1 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** 

bindingProperty-	The name of the property/field to access
Name	

#### Returns

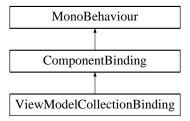
ModelPropertyBase The Model Property class. Use value to get the value of the property

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

• Assets/uFrameComplete/uFrame/Base/ViewModels/ViewModel.cs

### 3.81 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**

### 3.81.1 Member Function Documentation

```
3.81.1.1 override | Binding 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

### 3.82 ViewModelCommandInfo Class Reference

**Public Member Functions** 

- ViewModelCommandInfo (string name, ICommand command)
- ViewModelCommandInfo (Type parameterType, string name, ICommand command)

#### **Properties**

```
Type ParameterType [get, set]string Name [get, set]ICommand Command [get, set]
```

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

Assets/uFrameComplete/uFrame/Base/ViewModels/ViewModel.cs

# 3.83 ViewModelPropertyInfo Class Reference

**Public Member Functions** 

ViewModelPropertyInfo (ModelPropertyBase property, bool isElementProperty, bool isCollectionProperty, bool isEnum)

### **Properties**

```
bool IsEnum [get, set]
ModelPropertyBase Property [get, set]
bool IsElementProperty [get, set]
bool IsCollectionProperty [get, set]
```

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

Assets/uFrameComplete/uFrame/Base/ViewModels/ViewModel.cs

### 3.84 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

#### 3.84.1 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.

### 3.84.2 Member Function Documentation

3.84.2.1 virtual GameObject ViewResolver.FindView ( ViewModel model ) [virtual]

Provides a prefab

**Parameters** 

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

Returns

**3.84.2.2 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
----------	--

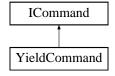
Returns

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

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

### 3.85 YieldCommand Class Reference

Inheritance diagram for YieldCommand:



#### **Public Member Functions**

- YieldCommand (Func< IEnumerator > enumerator Delegate)
- IEnumerator Execute ()

#### **Protected Member Functions**

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

#### **Properties**

- object Sender [get, set]
- object Parameter [get, set]
- Func< |Enumerator > Enumerator Delegate [get, set]

#### **Events**

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

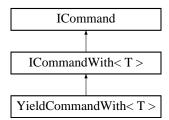
The documentation for this class was generated from the following file:

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

# 3.86 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 > Enumerator Delegate [get, set]

#### **Events**

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

### 3.86.1 Detailed Description

A coroutine command with a parameter.

**Template Parameters** 

T

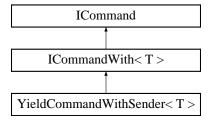
The documentation for this class was generated from the following file:

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

### 3.87 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 > Enumerator Delegate [get, set]

#### **Events**

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

### 3.87.1 Detailed Description

A coroutine command with a parameter.

**Template Parameters** 



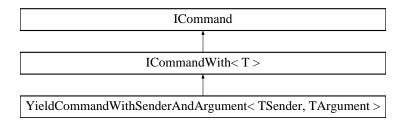
The documentation for this class was generated from the following file:

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

# 3.88 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 > enumerator-Delegate)
- 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,</li>
   IEnumerator > Enumerator Delegate [get, set]

#### **Events**

- CommandEvent OnCommandExecuted
- CommandEvent OnCommandExecuting

# 3.88.1 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

_DontUseAsyncLoading	CanTwoWayBind
GameManager, 32	Binding, 11
_LoadingLevel	Clear
GameManager, 32	GameContainer, 27
_LogEvents	IGameContainer, 36
ViewBase, 87	CollisionEvent
	ModelCollisionEventBinding, 58
_Start	CollisionEventBinding, 12
GameManager, 33	GetBinding, 12
A still a Coolean Adama a say	Command, 13
ActiveSceneManager	
GameManager, 33	CommandBinding, 14
AddBinding	Bind, 15
ViewBase, 84	ComponentCommandBinding, 19
AfterBind	Unbind, 15
ViewBase, 84	CommandWith $< T > 15$
Apply	CommandWithSender < TSender >, 16
ViewBase, 84	CommandWithSenderAndArgument< TSender, T-
ApplyRenderSettings	Argument >, 17
GameManager, 31	ComponentBinding, 17
Awake	Binding, 19
GameManager, 31	FilterBindableProperties, 18
SceneManager, 72	GetBinding, 18
-	ComponentCommandBinding, 19
Bind	CommandBinding, 19
Binding, 11	Container
CommandBinding, 15	Controller, 23
ModelCollectionBinding< TCollectionType >, 55	SceneManager, 74
ModelCommandBinding, 59	Context
ModelEventBinding, 60	Controller, 23
ModelPropertyBinding, 65	SceneManager, 74
ModelViewModelCollectionBinding, 66	Controller, 19
ModelViewPropertyBinding, 67	Container, 23
ViewBase, 85	Context, 23
BindReverse	Controller, 21
ITwoWayBinding, 44	Create, 22
	CreateEmpty, 22
ModelPropertyBinding, 65	GameEvent, 22
BindableProperty, 9	Create
Value, 9	Controller, 22
Binding, 9	CreateEmpty
Bind, 11	Controller, 22
Binding, 10	CreateModel
CanTwoWayBind, 11	ViewBase, 85
ComponentBinding, 19	•
GetTargetValueDelegate, 11	DefaultIdentifier
IsComponent, 11	ViewBase, 87
ModelMemberName, 11	DefaultTypeResolver, 23
ModelProperty, 11	Deserialize
ModelPropertySelector, 11	P < T >, 69
SetTargetValueDelegate, 11	DiagramInfoAttribute, 23
Source, 11	,
SourceValue, 11	ElementService, 24
TwoWay, 11	EventBinding, 24
Unbind, 11	GetBinding, 25
Bindings	EventTriggered
ViewBase, 87	ViewBase, 88

ExecuteCommand	ViewModel, 92
ViewBase, 85	GetTargetValueDelegate
ExecuteCommand< TArgument >	Binding, 11
ViewBase, 85	
	IBinding, 33
FileSerializerStorage, 25	IBindingProvider, 34
FilterBindableProperties	ICommand, 34
ComponentBinding, 18	ICommand $<$ T $>$ , 35
FindView	ICommandWith< T>, 35
ViewResolver, 95	IGameContainer, 35
	Clear, 36
GameContainer, 26	Inject, 36
Clear, 27	InjectAll, 38
Inject, 27	Register < TSource, TTarget >, 38
InjectAll, 27	RegisterInstance, 38
Register< TSource, TTarget >, 27	RegisterInstance< TBase >, 38
RegisterInstance, 27	_
	Resolve, 39
Resolve, 27	Resolve < T >, 39
Resolve < T >, 28	ResolveAll, 39
ResolveAll, 28	ResolveAll< TType >, 39
ResolveAll< TType >, 28	IJsonSerializable, 40
GameEvent	IModelCollection, 40
Controller, 22	ISerializer, 42
GameManager, 29	ISerializerStorage, 42
_DontUseAsyncLoading, 32	ISerializerStream, 43
_LoadingLevel, 32	ITwoWayBinding, 44
Start, 33	BindReverse, 44
ActiveSceneManager, 33	ITypeResolver, 44
ApplyRenderSettings, 31	IUFSerializable, 45
Awake, 31	IView, 45
Instance, 33	ViewModelObject, 46
IsPro, 33	ViewModelType, 46
Load, 31	ViewName, 46
LoadRenderSettings, 31	IViewModelObserver, 46
LoadingViewModel, 33	Identifier
OnDestroy, 31	ViewBase, 87
RegisterSceneManager, 31	InitializeData
SceneManagers, 33	ViewBase, 86
Start, 31	InitializeViewModel
Startup, 31	View < TModel >, 81, 82
Transition < TGame >, 31	ViewBase, 86
TransitionLevel $<$ T $>$ , 32	Inject
UnRegisterSceneManager, 32	GameContainer, 27
GetArgument	IGameContainer, 36
ModelCollisionEventBinding, 57	InjectAll
GetBinding	GameContainer, 27
CollisionEventBinding, 12	IGameContainer, 38
ComponentBinding, 18	InjectAttribute, 41
EventBinding, 25	InputBinding, 41
InputBinding, 42	GetBinding, 42
KeyBinding, 52	Instance
MouseEventBinding, 68	
<u>.</u>	GameManager, 33
UFPropertyBinding, 78	InstanceId
ViewModelCollectionBinding, 94	ViewBase, 87
GetProperties	InstantiateView
ViewModel, 92	ViewContainer, 89, 90
GetReflectedCommands	IsComponent
ViewModel, 92	Binding, 11
GetReflectedModelProperties	IsPro

GameManager, 33	ModelPropertyBinding, 64 Bind, 65
JSONArray, 46	BindReverse, 65
JSONClass, 47	Unbind, 65
JSONData, 48	ModelPropertySelector
JSONLazyCreator, 49	Binding, 11
JSONNode, 49	ModelViewModelCollectionBinding, 65
JsonStream, 51	Bind, 66
, .	Unbind, 66
KeyBinding, 52	ModelViewPropertyBinding, 66
GetBinding, 52	Bind, 67
3, 1	Unbind, 67
LateUpdate	•
ViewBase, 86	MouseEventBinding, 67
LevelLoaderSceneManager, 53	GetBinding, 68
Load	Object) / close
GameManager, 31	ObjectValue
SceneContext, 70	ModelPropertyBase, 64
SceneManager, 72	OnDestroy
<b>G</b> .	GameManager, 31
LoadRenderSettings	SceneManager, 72
GameManager, 31	ViewBase, 86
LoadingViewModel	OnLoaded
GameManager, 33	SceneManager, 72
	OnLoading
Model	SceneManager, 72
View< TModel >, 82	
ModelCollection < T >, 53	P < T >, 68
ModelCollectionBinding< TCollectionType >, 54	Deserialize, 69
Bind, 55	Serialize, 69
Unbind, 55	Value, 69
ModelCollectionChangeEvent, 55	ValueType, 69
ModelCollectionChangeEventWith< T >, 56	PreBind
ModelCollisionEventBinding, 56	ViewBase, 86
CollisionEvent, 58	
GetArgument, 57	QuietlySetValue
SetParameterSelector, 57	ModelPropertyBase, 63
Subscribe, 57	, ,
When, 57	Register< TSource, TTarget >
ModelCommandBinding, 58	GameContainer, 27
Bind, 59	IGameContainer, 38
Unbind, 59	RegisterInstance
ModelEventBinding, 59	GameContainer, 27
Bind, 60	IGameContainer, 38
Unbind, 60	RegisterInstance < TBase >
ModelInputButtonBinding, 60	IGameContainer, 38
ModelKeyBinding, 60	RegisterSceneManager
RequireAlt, 61	GameManager, 31
RequireControl, 61	RegisteredInstance, 70
RequireShift, 61	Reload
ModelMemberName	SceneManager, 72
Binding, 11	RemoveBinding
ModelMouseEventBinding, 62	ViewBase, 86
ModelProperty	RequestViewModel
Binding, 11	SceneManager, 72
ModelPropertyBase, 62	ViewBase, 86
ObjectValue, 64	RequireAlt
QuietlySetValue, 63	ModelKeyBinding, 61
ValueChanged, 64	RequireControl
ValueType, 64	ModelKeyBinding, 61
	· •

RequireShift	Startup
ModelKeyBinding, 61	GameManager, 31
Resolve	StateLoaderResolver, 75
GameContainer, 27	StringSerializerStorage, 75
IGameContainer, 39	Subscribe
Resolve < T >	ModelCollisionEventBinding, 57
GameContainer, 28	3,
IGameContainer, 39	Transition < TGame >
ResolveAll	GameManager, 31
GameContainer, 28	TransitionLevel < T >
IGameContainer, 39	GameManager, 32
ResolveAll < TType >	TwoWay
GameContainer, 28	Binding, 11
IGameContainer, 39	TypeInstanceCollection, 76
Touris Container, Co	TypeMapping, 76
Save	TypeMappingCollection, 76
SceneContext, 70	TypeRelation, 77
ViewBase, 87	TypeRelationCollection, 77
SceneContext, 70	Typortolation collection, 77
Load, 70	UFGroup, 77
Save, 70	UFPropertyBinding, 78
ViewModels, 71	GetBinding, 78
SceneManager, 71	UFRequireInstanceMethod, 79
Awake, 72	UFToggleGroup, 79
Container, 74	UnRegisterSceneManager
Context, 74	GameManager, 32
Load, 72	Unbind
OnDestroy, 72	Binding, 11
OnLoaded, 72	CommandBinding, 15
OnLoading, 72	ModelCollectionBinding< TCollectionType >, 55
Reload, 72	ModelCommandBinding, 59
RequestViewModel, 72	ModelEventBinding, 60
Settings, 74	ModelPropertyBinding, 65
Setup, 74	ModelViewModelCollectionBinding, 66
SetupViewModel < TViewModel >, 74	ModelViewPropertyBinding, 67
Unload, 74	ViewBase, 87
SceneManagers	Unload
GameManager, 33	SceneManager, 74
Serialize	Coordinating of , 7 -
P< T >, 69	Value
SetParameterSelector	BindableProperty, 9
ModelCollisionEventBinding, 57	P < T >, 69
SetTargetValueDelegate	ValueChanged
Binding, 11	ModelPropertyBase, 64
Settings	ValueType
SceneManager, 74	ModelPropertyBase, 64
Setup	P < T > ,69
SceneManager, 74	View< TModel >, 80
SetupBindings	InitializeViewModel, 81, 82
ViewBase, 86	Model, 82
SetupViewModel < TViewModel >	ViewBase, 82
SceneManager, 74	_LogEvents, 87
Source	AddBinding, 84
Binding, 11	AfterBind, 84
SourceValue	Apply, 84
Binding, 11	Bind, 85
Start	Bindings, 87
GameManager, 31	CreateModel, 85
ViewBase, 87	DefaultIdentifier, 87

```
EventTriggered, 88
    ExecuteCommand, 85
     ExecuteCommand< TArgument >, 85
    Identifier, 87
    InitializeData, 86
    InitializeViewModel, 86
    Instanceld, 87
    LateUpdate, 86
    OnDestroy, 86
    PreBind, 86
    RemoveBinding, 86
    RequestViewModel, 86
    Save, 87
    SetupBindings, 86
    Start, 87
     Unbind, 87
     ViewEvent, 87
     ViewName, 87
ViewComponent, 88
ViewContainer, 88
    InstantiateView, 89, 90
ViewEvent
    ViewBase, 87
ViewEventTrigger, 91
ViewModel, 91
    GetProperties, 92
    GetReflectedCommands, 92
    GetReflectedModelProperties, 92
ViewModelCollectionBinding, 93
    GetBinding, 94
ViewModelCommandInfo, 94
ViewModelObject
    IView, 46
ViewModelPropertyInfo, 94
ViewModelType
    IView, 46
ViewModels
    SceneContext, 71
ViewName
    IView, 46
    ViewBase, 87
ViewResolver, 95
    FindView, 95
When
    ModelCollisionEventBinding, 57
YieldCommand, 95
YieldCommandWith< T >, 96
YieldCommandWithSender< T >, 97
YieldCommandWithSenderAndArgument< TSender, T-
         Argument >, 98
```