Table of Contents

Namespace ItemSystem

This is The Namespace for the ItemSystem, Assembly-CSharp

ItemSystem.Data

ItemSystem.Editor

Namespace ItemSystem.Data

Classes

ItemData

This class is a ScriptableObject and is used to represent an item that can be used in the world or in the inventory.

Enums

ItemType

This Enum is used to determine the type of the Item. To use this enum in UI Builder

WeaponState

This is used to determine the Current State of the Weapon is in. To use this enum in UI Builder

Class ItemData

This class is a ScriptableObject and is used to represent an item that can be used in the world or in the inventory.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.ScriptableObject
ItemData

Namespace: ItemSystem.Data

Assembly: Assembly-CSharp.dll

Syntax

```
[Serializable]
public class ItemData : ScriptableObject
```

Fields

cost

This is the cost of the item.

Declaration

```
public int cost
```

Field Value

TYPE	DESCRIPTION
System.Int32	

description

This is the description of the item.

Declaration

```
public string description
```

Field Value

TYPE	DESCRIPTION
System.String	

IID

This is a unique id for the item. This will be used to identify the InventoryItem and Items in the world.

Declaration

```
public string IID
```

Field Value

TYPE	DESCRIPTION
System.String	

image

This is the Image of the item for the inventory icons.

Declaration

```
public Sprite image
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

isEquipable

This will determine if the items can be equipped.

Declaration

```
public bool isEquipable
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

itemName

This is the name of the item.

Declaration

```
public string itemName
```

Field Value

TYPE	DESCRIPTION
System.String	

prefabWorldItem

This is a prefab of the item for 3D games.

Declaration

```
public GameObject prefabWorldItem
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

stackable

This determines if the items can be stacked.

Declaration

```
public bool stackable
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

stackSize

This is how many of the item and fit into an inventory stack.

Declaration

```
public int stackSize
```

Field Value

TYPE	DESCRIPTION
System.Int32	

type

This the type of item it is.

Declaration

```
public ItemType type
```

Field Value

TYPE	DESCRIPTION
ItemType	

weight

This is the weight of the item.

Declaration

```
public float weight
```

Field Value

TYPE	DESCRIPTION
System.Single	

Methods

CreateWorldObject(Transform)

This will Instantiate the prefabWorldItem prefab.

Declaration

```
public GameObject CreateWorldObject(Transform target = null)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Transform	target	The parent of this object.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
UnityEngine.GameObject	The game object of the item.

Extension Methods

RichText.UpperCaseText(Object)
RichText.LowerCaseText(Object)
RichText.BoldText(Object)
RichText.ItalicText(Object)
RichText.Text(Object)
RichText.UnderLineText(Object)
RichText.StrikeText(Object)
RichText.SupText(Object)
RichText.SubText(Object)
RichText.PositionText(Object, Single)
RichText.SizeText(Object, Int32)
RichText.ColoredText(Object, String)
RichText.ColoredText(Object, Color)
RichText.BooleanText(Object, Boolean)
RichText.BooleanText(Object, Boolean, Color)

Enum ItemType

This Enum is used to determine the type of the Item. To use this enum in UI Builder

Namespace: ItemSystem.Data

Assembly: Assembly-CSharp.dll

Syntax

```
public enum ItemType
```

Remarks

ItemSystem.Data.ItemType, Assembly-CSharp

Fields

NAME	DESCRIPTION
Armour	
Consumable	
Materials	
Misc	
Weapon	

Extension Methods

RichText.UpperCaseText()
RichText.LowerCaseText()
RichText.BoldText()
RichText.ItalicText()
RichText.Text()
RichText.UnderLineText()
RichText.StrikeText()
RichText.SupText()
RichText.SubText()
RichText.PositionText(Single)
RichText.SizeText(Int32)
RichText.ColoredText(String)
RichText.ColoredText(Color)
RichText.BooleanText(Boolean)
RichText.BooleanText(Boolean, Color)

Enum WeaponState

This is used to determine the Current State of the Weapon is in. To use this enum in UI Builder

Namespace: ItemSystem.Data

Assembly: Assembly-CSharp.dll

Syntax

```
public enum WeaponState
```

Remarks

ItemSystem.Data.WeaponState, Assembly-CSharp

Fields

NAME	DESCRIPTION
Attack	
Defend	
Idle	
Parry	

Extension Methods

RichText.UpperCaseText()
RichText.LowerCaseText()
RichText.BoldText()
RichText.ItalicText()
RichText.Text()
RichText.UnderLineText()
RichText.StrikeText()
RichText.SupText()
RichText.SubText()
RichText.PositionText(Single)
RichText.SizeText(Int32)
RichText.ColoredText(String)
RichText.ColoredText(Color)
RichText.BooleanText(Boolean)
RichText.BooleanText(Boolean, Color)

Namespace ItemSystem.Editor

Classes

CreateItem

This class is an EditorWindow used to create an ItemData file.

EditItem

This class is an EditorWindow used to edit any ItemData file.

ItemIDScrubber

This class is an EditorWindow used to check all ItemData in the Resource folder and make sure the ID's are not conflicted.

Class CreateItem

This class is an EditorWindow used to create an ItemData file.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.ScriptableObject
UnityEditor.EditorWindow
CreateItem

Namespace: ItemSystem.Editor

Assembly: Assembly-CSharp-Editor.dll

Syntax

```
public class CreateItem : EditorWindow
```

Fields

costField

This is the cost that the item will have.

Declaration

```
public IntegerField costField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.IntegerField	

createButton

This is the element that will be used to Create the ItemData file and pass in the data from the input fields.

Declaration

```
public Button createButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Button	

descriptionField

This is the description of the item will have.

Declaration

```
public TextField descriptionField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.TextField	

imageField

This is the image that the item will use in the inventory.

Declaration

```
public ObjectField imageField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.ObjectField	

itemCanStackToggle

This is to determine if the item will be able to stack.

Declaration

```
public Toggle itemCanStackToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Toggle	

itemNameField

This is the name of the item will have.

Declaration

```
public TextField itemNameField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.TextField	

itemVisualSprite

This is a display of the image that the item will use.

Declaration

```
public SpriteElement itemVisualSprite
```

Field Value

TYPE	DESCRIPTION
SpriteElement	

newItemData

This is the new ItemData file that is created.

Declaration

```
public ItemData newItemData
```

Field Value

TYPE	DESCRIPTION
ItemData	

OnItemChange

This is the GiveItem that is subscribed to by the InventoryWindow AddItemData. when the InventoryWindow is open it and an ItemData file is created it will add the new ItemData file ot the AddItemData Method.

Declaration

```
public static Action<ItemData> OnItemChange
```

Field Value

TYPE	DESCRIPTION
System.Action<ItemData>	

stackSizeField

This is the max stack size that the item will have.

Declaration

```
public IntegerField stackSizeField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.IntegerField	

type

This is the type that the new item will be.

Declaration

```
public EnumField type
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.EnumField	

weightField

This is the weight that the item will have.

Declaration

```
public FloatField weightField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.FloatField	

Methods

CreateGUI()

This is a unity method, that will be run when the window is created.

Declaration

```
public void CreateGUI()
```

CreateItemFile()

This Will create the item from the inputs you have made.

Declaration

```
public void CreateItemFile()
```

GetUIElements()

This will get the Elements in the UI. And set RegisterValueChangedCallback if any other element's values are changed.

Declaration

```
public void GetUIElements()
```

OpenWindow()

This static will open the CreateItem. Though the UnityEditor Toolbar: Tools/DownUnder Studios/Item System/Tools/Create Item
Or by the shortcut key CTRl + SHIFT + ALT + C

Declaration

```
[MenuItem("Tools/DownUnder Studios/Item System/Tools/Create Item %#&C")]
public static void OpenWindow()
```

ResetField()

This will reset the input field for the item.

Declaration

```
public void ResetField()
```

SetWindowSize(Int32, Int32)

This will set the Editor Window min & max size to a fix size. With both params being the width & height respectively.

Declaration

```
public void SetWindowSize(int width, int height)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	width	The width of the Window.
System.Int32	height	The height of the Window.

Extension Methods

RichText.UpperCaseText(Object)
RichText.LowerCaseText(Object)
RichText.BoldText(Object)
RichText.ItalicText(Object)
RichText.Text(Object)
RichText.UnderLineText(Object)
RichText.StrikeText(Object)
RichText.SupText(Object)
RichText.SubText(Object)
RichText.PositionText(Object, Single)
RichText.SizeText(Object, Int32)
RichText.ColoredText(Object, String)
RichText.ColoredText(Object, Color)
RichText.BooleanText(Object, Boolean)
RichText.BooleanText(Object, Boolean, Color)

Class EditItem

This class is an EditorWindow used to edit any ItemData file.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.ScriptableObject
UnityEditor.EditorWindow
EditItem

Namespace: ItemSystem.Editor

Assembly: Assembly-CSharp-Editor.dll

Syntax

```
public class EditItem : EditorWindow
```

Fields

costField

This is the element that will be used to get the cost for the item.

Declaration

```
public IntegerField costField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.IntegerField	

descriptionField

This is the element that will be used to get the description for the item.

Declaration

```
public TextField descriptionField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.TextField	

imageField

This is the element that will be used to get the Sprite for the item.

Declaration

```
public ObjectField imageField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.ObjectField	

itemCanStackToggle

This is the element that will be used to determine if the item can stack.

Declaration

```
public Toggle itemCanStackToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Toggle	

itemNameField

This is the element that will be used to get the name for the item.

Declaration

```
public TextField itemNameField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.TextField	

itemVisualSprite

This is the element that will show the item image tat will be used by the inventory.

Declaration

```
public SpriteElement itemVisualSprite
```

Field Value

TYPE	DESCRIPTION
SpriteElement	

stackSizeField

This is the element that will be used to get the max stack size for the item.

Declaration

```
public IntegerField stackSizeField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.IntegerField	

targetItemData

This is the ItemData file that will be edited.

Declaration

```
public static ItemData targetItemData
```

Field Value

TYPE	DESCRIPTION
ItemData	

type

This is the element that will be used to get the type of the item.

Declaration

```
public EnumField type
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.EnumField	

weightField

This is the element that will be used to get the weight of the item.

Declaration

```
public FloatField weightField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.FloatField	

Methods

CreateGUI()

This is a unity method, that will be run when the window is created.

Declaration

```
public void CreateGUI()
```

GetUIElements()

This will get the Elements in the UI. And set RegisterValueChangedCallback if any other element's values are changed.

Declaration

```
public void GetUIElements()
```

OnOpenAsset(Int32, Int32)

This will open the EditItem window when an ItemData file is double clicked.

Declaration

```
[OnOpenAsset]
public static bool OnOpenAsset(int instanceID, int line)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	instanceID	not used.
System.Int32	line	not used.

Returns

TYPE	DESCRIPTION
System.Boolean	not used.

OpenWindow()

This will open the EditItem. The window can be open by double clicking an ItemData file.

Declaration

```
public static void OpenWindow()
```

SetItemData(ItemData)

This will set the target ItemData. and open the window.

Declaration

```
public static bool SetItemData(ItemData item)
```

Parameters

TYPE	NAME	DESCRIPTION
ItemData	item	The ItemData that will be edited.

Returns

TYPE	DESCRIPTION
System.Boolean	Return true if targetItemData is set and window open.

SetWindowSize(Int32, Int32)

This will set the Editor Window min & max size to a fix size. With both params being the width & height respectively.

Declaration

```
public void SetWindowSize(int width, int height)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	width	The width of the Window.
System.Int32	height	The height of the Window.

UpdateItemData()

This will update ItemData file with the new data from the input fields.

Declaration

```
public void UpdateItemData()
```

Extension Methods

RichText.UpperCaseText(Object)
RichText.LowerCaseText(Object)
RichText.BoldText(Object)
RichText.ItalicText(Object)
RichText.Text(Object)
RichText.UnderLineText(Object)
RichText.StrikeText(Object)
RichText.SupText(Object)
RichText.SubText(Object)
RichText.PositionText(Object, Single)
RichText.SizeText(Object, Int32)
RichText.ColoredText(Object, String)
RichText.ColoredText(Object, Color)
RichText.BooleanText(Object, Boolean)
RichText.BooleanText(Object, Boolean, Color)

Class ItemIDScrubber

This class is an EditorWindow used to check all ItemData in the Resource folder and make sure the ID's are not conflicted.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.ScriptableObject
UnityEditor.EditorWindow
ItemIDScrubber

Namespace: ItemSystem.Editor

Assembly: Assembly-CSharp-Editor.dll

Syntax

```
public class ItemIDScrubber : EditorWindow
```

Fields

clearButton

This is the element that when pressed will clear the Log.

Declaration

```
public Button clearButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Button	

currentLogNumber

This is the number of logs made.

Declaration

```
public int currentLogNumber
```

Field Value

TYPE	DESCRIPTION
System.Int32	

items

This is an array of all ItemData in the project.

Declaration

```
public ItemData[] items
```

Field Value

TYPE	DESCRIPTION
ItemData[]	

logDataString

This is the sting that hold all of the logs that will be added into a file upon saving.

Declaration

```
public string logDataString
```

Field Value

TYPE	DESCRIPTION
System.String	

LogEntryParent

This is the parent where all log elements will be parented to.

Declaration

```
public VisualElement LogEntryParent
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.VisualElement	

SaveButton

This is the element that when pressed will save the Log.

Declaration

```
public Button SaveButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Button	

startButton

This is the element that when pressed will Start the scrubbing process.

Declaration

```
public Button startButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Button	

Methods

Clear()

This will clear the Log.

Declaration

```
public void Clear()
```

CreateGUI()

This is a unity method, that will be run when the window is created.

Declaration

```
public void CreateGUI()
```

CreateLogEntry(String)

This Will create a log entry from a string log.

Declaration

```
public void CreateLogEntry(string log)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	log	this is the log that is passed in.

Save()

This will save the Log to a .txt file. (With time stamp).

Declaration

```
public void Save()
```

SetWindowSize(Int32, Int32)

This will set the Editor Window min & max size to a fix size. With both params being the width & height respectively.

Declaration

```
public void SetWindowSize(int width, int height)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	width	The width of the Window.
System.Int32	height	The height of the Window.

ShowExample()

This will open the CreateItem. Though the UnityEditor Toolbar: Tools/DownUnder Studios/Item System/Tools/Item ID Scrubber Or by the shortcut key CTRl + SHIFT + ALT + S

Declaration

```
[MenuItem("Tools/DownUnder Studios/Item System/Tools/Item ID Scrubber %#&S")]
public static void ShowExample()
```

Start()

This will cross check every ItemData file for conflicting IDs. And return a log for each ID that is changed.

Declaration

```
public void Start()
```

Extension Methods

RichText.UpperCaseText(Object)
RichText.LowerCaseText(Object)
RichText.BoldText(Object)
RichText.ItalicText(Object)
RichText.Text(Object)
RichText.UnderLineText(Object)
RichText.StrikeText(Object)
RichText.SupText(Object)
RichText.SubText(Object)
RichText.PositionText(Object, Single)
RichText.SizeText(Object, Int32)
RichText.ColoredText(Object, String)
RichText.ColoredText(Object, Color)
RichText.BooleanText(Object, Boolean)
RichText.BooleanText(Object, Boolean, Color)