Table of Contents

Namespace ItemSystem

This is The Namespace for the ItemSystem, Assembly-CSharp

ItemSystem.Data

ItemSystem.Editor

ItemSystem.Script.Interface

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 Enum 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

```
public class ItemData : ScriptableObject
```

Fields

animator

This Variable is the animator controller that the item will use.

Declaration

```
public AnimatorOverrideController animator
```

Field Value

TYPE	DESCRIPTION
UnityEngine.AnimatorOverrideController	

canStack

This Variable determines if th items can be stacked.

Declaration

```
public bool canStack
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

itemCost

This Variable is the cost of the item.

Declaration

```
public int itemCost
```

Field Value

TYPE	DESCRIPTION
System.Int32	

itemDescription

This Variable is the description of the item.

Declaration

```
public string itemDescription
```

Field Value

TYPE	DESCRIPTION
System.String	

itemID

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

Declaration

```
public string itemID
```

Field Value

TYPE	DESCRIPTION
System.String	

itemImage

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

Declaration

```
public Sprite itemImage
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Sprite	

itemMaxStack

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

Declaration

```
public int itemMaxStack
```

Field Value

TYPE	DESCRIPTION
System.Int32	

itemName

This Variable is the name of the item.

Declaration

```
public string itemName
```

Field Value

TYPE	DESCRIPTION
System.String	

itemType

This Variable the type of item it is.

Declaration

```
public ItemType itemType
```

Field Value

TYPE	DESCRIPTION
ItemType	

itemWeight

This Variable is the weight of the item.

Declaration

```
public float itemWeight
```

Field Value

TYPE	DESCRIPTION
System.Single	

prefabWorldItem

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

Declaration

```
public GameObject prefabWorldItem
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

prefabWorldItem2D

This Variable is a prefab of the item for 2D games.

Declaration

```
public GameObject prefabWorldItem2D
```

Field Value

TYPE	DESCRIPTION
UnityEngine.GameObject	

Methods

CreateWorldObject(Transform)

This Method will Instantiate the prefabWorldItem prefab.

Declaration

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

Parameters

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

Returns

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

CreateWorldObject2D(Transform)

This Method will Instantiate the prefabWorldItem2D prefab.

Declaration

```
public GameObject CreateWorldObject2D(Transform target)
```

Parameters

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

Returns

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)

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)

Enum WeaponState

This Enum 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)

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.

Delegates

CreateItem.GiveItem

This Variable is a delegate that passes the new ItemData file.

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

createButton

This Variable is the Button 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	

itemCanStackToggle

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

Declaration

```
public Toggle itemCanStackToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Toggle	

itemCostField

This Variable is the cost that the item will have.

Declaration

```
public IntegerField itemCostField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.IntegerField	

itemDescriptionField

This Variable is the description of the item will have.

Declaration

```
public TextField itemDescriptionField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.TextField	

itemImageField

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

Declaration

```
public ObjectField itemImageField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.ObjectField	

itemMaxStackField

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

Declaration

```
public IntegerField itemMaxStackField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.IntegerField	

itemNameField

This Variable is the name of the item will have.

Declaration

```
public TextField itemNameField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.TextField	

itemtype

This Variable is the type that the new item will be.

Declaration

```
public EnumField itemtype
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.EnumField	

itemVisualSprite

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

Declaration

```
public SpriteElement itemVisualSprite
```

Field Value

TYPE	DESCRIPTION
SpriteElement	

itemWeightField

This Variable is the weight that the item will have.

Declaration

```
public FloatField itemWeightField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.FloatField	

newItemData

This Variable is the new ItemData file that is created.

Declaration

```
public ItemData newItemData
```

Field Value

TYPE	DESCRIPTION
ItemData	

Methods

CreateGUI()

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

Declaration

```
public void CreateGUI()
```

CreateItemFile()

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

Declaration

```
public void CreateItemFile()
```

GetUIElements()

This Method 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 Method 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 Method will reset the input field for the item.

Declaration

```
public void ResetField()
```

SetWindowSize(Int32, Int32)

This Method 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.

Events

OnItemChange

This Variable is the GiveItem that is subscribed to by the InventoryWindow Method 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 event CreateItem.GiveItem OnItemChange
```

Event Type

TYPE	DESCRIPTION
CreateItem.GiveItem	

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)

Delegate CreateItem.GiveItem

This Variable is a delegate that passes the new ItemData file.

Namespace: ItemSystem.Editor

Assembly: Assembly-CSharp-Editor.dll

Syntax

```
public delegate void GiveItem(ItemData newItemData);
```

Parameters

TYPE	NAME	DESCRIPTION
ItemData	newItemData	

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)

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

itemCanStackToggle

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

Declaration

```
public Toggle itemCanStackToggle
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Toggle	

itemCostField

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

Declaration

```
public IntegerField itemCostField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.IntegerField	

itemDescriptionField

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

Declaration

```
public TextField itemDescriptionField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.TextField	

itemImageField

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

Declaration

```
public ObjectField itemImageField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.ObjectField	

itemMaxStackField

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

Declaration

```
public IntegerField itemMaxStackField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.IntegerField	

itemNameField

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

Declaration

```
public TextField itemNameField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.TextField	

itemtype

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

Declaration

```
public EnumField itemtype
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.EnumField	

itemVisualSprite

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

Declaration

```
public SpriteElement itemVisualSprite
```

Field Value

TYPE	DESCRIPTION
SpriteElement	

itemWeightField

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

Declaration

```
public FloatField itemWeightField
```

Field Value

TYPE	DESCRIPTION
UnityEditor.UIElements.FloatField	

targetItemData

This Variable is the ItemData file that will be edited.

Declaration

```
public static ItemData targetItemData
```

Field Value

TYPE	DESCRIPTION
ItemData	

Methods

CreateGUI()

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

Declaration

```
public void CreateGUI()
```

GetUIElements()

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

Declaration

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

SetItemData(ItemData)

This static Method 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 Method 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 Method 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)

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 Variable is the UI Button that when pressed will clear the Log.

Declaration

```
public Button clearButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Button	

currentLogNumber

This Variable is the number of logs made.

Declaration

```
public int currentLogNumber
```

Field Value

TYPE	DESCRIPTION
System.Int32	

items

This Variable is an array of all ItemData in the project.

Declaration

```
public ItemData[] items
```

Field Value

TYPE	DESCRIPTION
ItemData[]	

logDataString

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

Declaration

```
public VisualElement LogEntryParent
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.VisualElement	

SaveButton

This Variable is the UI Button that when pressed will save the Log.

Declaration

```
public Button SaveButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Button	

startButton

This Variable is the UI Button that when pressed will Start the scrubbing process.

Declaration

```
public Button startButton
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UIElements.Button	

Methods

Clear()

This Method will clear the Log.

Declaration

```
public void Clear()
```

CreateGUI()

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

Declaration

```
public void CreateGUI()
```

CreateLogEntry(String)

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

Declaration

```
public void Save()
```

SetWindowSize(Int32, Int32)

This Method 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 Static Method 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 Method 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)

Namespace ItemSystem.Script.Interface

Interfaces

IConsume

This Interface should be used to use an item.

IPickUp

This Interface should be used to pickup an item.

IUse

This Interface should be used to use an item.

IWeapon

This Interface should be used by an Weapon type item class.

Interface IConsume

This Interface should be used to use an item.

Namespace: ItemSystem.Script.Interface

Assembly: Assembly-CSharp.dll

Syntax

```
public interface IConsume
```

Properties

amount

This Variable will determine how many uses are left.

Declaration

```
int amount { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Use()

This Method will be used to when that player wants to consume an item.

Declaration

```
void Use()
```

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)

Interface IPickUp

This Interface should be used to pickup an item.

Namespace: ItemSystem.Script.Interface

Assembly: Assembly-CSharp.dll

Syntax

```
public interface IPickUp
```

Properties

canPickUp

This Variable will determine if the Item can be picked up.

Declaration

```
bool canPickUp { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

Pickup()

This Method will be used to when that player wants to pick up the item.

Declaration

```
void Pickup()
```

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)

Interface IUse

This Interface should be used to use an item.

Namespace: ItemSystem.Script.Interface

Assembly: Assembly-CSharp.dll

Syntax

```
public interface IUse
```

Properties

canUse

This Variable will determine if the Item can be used.

Declaration

```
bool canUse { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

Use()

This Method will be used to when that player wants to use an item.

Declaration

```
void Use()
```

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)

Interface IWeapon

This Interface should be used by an Weapon type item class.

Namespace: ItemSystem.Script.Interface

Assembly: Assembly-CSharp.dll

Syntax

```
public interface IWeapon
```

Fields

v

This property is the currentState of the weapon.

Declaration

```
WeaponState State { get; set; }
```

Field Value

TYPE	DESCRIPTION
WeaponState	

Properties

canAttack

This Variable determines if the weapon can attack.

Declaration

```
bool canAttack { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

canParry

This Variable determines if the weapon can parry.

Declaration

```
bool canParry { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

canDefend

This Variable determines if the weapon can defend.

Declaration

```
bool canDefend { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

Attack()

This Method is what will happen if canAttack is true.

Declaration

```
void Attack()
```

Parry()

This Method is what will happen if canParry is true.

Declaration

```
void Parry()
```

Defend()

This Method is what will happen if canDefend is true.

Declaration

```
void Defend()
```

Idle()

This Method is what will happen if the weapon is idle.

Declaration

```
void Idle()
```

SetWeaponState()

This Method is what will check if the weapon will attack, parry, defend or idle.

Declaration

```
void SetWeaponState()
```

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)