# Kalkulator statystyk postaci na potrzeby gry RPG

Generated by Doxygen 1.9.1

# Chapter 1

# Todo List

**Member Character::addItem (const Item ∗const item, itemQuantity_t quantity=1)**

make it remove item from inventory if quantity would go below or equal 0.

**Member GameData::addItem (Item ∗item)**

LOW change excpetion to it's own exception class

**Member StatModifyingEntity::addModifier (Stat::id_t, Stat::value_t by)**

Check if entity has modifier of that Stat already.

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

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

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

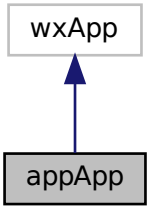
# Chapter 5

# Class Documentation

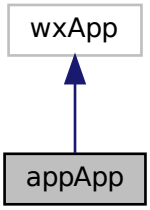
## 5.1 appApp Class Reference

`#include <appApp.hpp>`

Inheritance diagram for appApp:




Collaboration diagram for appApp:



### Public Member Functions

- virtual bool OnInit ()

### 5.1.1 Detailed Description

Definition at line 16 of file appApp.hpp.

### 5.1.2 Member Function Documentation

#### 5.1.2.1 OnInit()

```
bool appApp::OnInit ( )  [virtual]
```

Definition at line 15 of file appApp.cpp.

```
15                     {
16   appFrame *frame = new appFrame("My Application");
17   frame->Show(true);
18   return true;
19 }
```

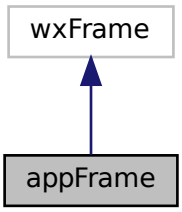
The documentation for this class was generated from the following files:

- appApp.hpp
- appApp.cpp

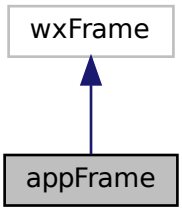
## 5.2 appFrame Class Reference

```
#include <appMain.hpp>
```

Inheritance diagram for appFrame:



Collaboration diagram for appFrame:




## Public Member Functions

- appFrame (const wxString &title)

## Private Member Functions

- void OnAddCharacter (wxCommandEvent &event)

## Private Attributes

- wxNotebook * notebook
- wxPanel * gmPanel
- wxPanel * addCharPanel
- wxTextCtrl * charNameTextCtrl

### 5.2.1 Detailed Description

Definition at line 20 of file appMain.hpp.

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 appFrame()

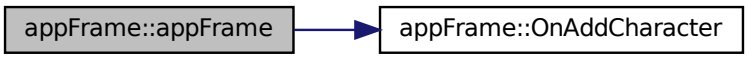
```
appFrame::appFrame (
            const wxString & title )
```

Definition at line 12 of file appMain.cpp.

```
13     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(800, 600)) {
14   wxBoxSizer *sizer = new wxBoxSizer(wxVERTICAL);
15
16   notebook = new wxNotebook(this, wxID_ANY);
17
18   gmPanel = new appGMPanel(notebook);
19   notebook->AddPage(gmPanel, "gamemaster");
20
21   addCharPanel = new wxPanel(notebook, wxID_ANY);
22   wxBoxSizer *addCharSizer = new wxBoxSizer(wxVERTICAL);
23   wxStaticText *charNameLabel =
24       new wxStaticText(addCharPanel, wxID_ANY, "Character's Name:");
25   charNameTextCtrl = new wxTextCtrl(addCharPanel, wxID_ANY);
26   wxButton *addButton = new wxButton(addCharPanel, ID_ADD_CHARACTER, "Add");
27
28   addCharSizer->Add(charNameLabel, 0, wxALIGN_LEFT | wxALL, 5);
29   addCharSizer->Add(charNameTextCtrl, 0, wxEXPAND | wxALL, 5);
30   addCharSizer->Add(addButton, 0, wxALIGN_LEFT | wxALL, 5);
31   addCharPanel->SetSizer(addCharSizer);
32
33   notebook->AddPage(addCharPanel, "Add character");
34
35   sizer->Add(notebook, 1, wxEXPAND);
36   SetSizer(sizer);
37
38   addButton->Bind(wxEVT_BUTTON, &appFrame::OnAddCharacter, this);
39 }
```

References addCharPanel, charNameTextCtrl, gmPanel, ID_ADD_CHARACTER, notebook, and OnAdd↩Character().

Here is the call graph for this function:




### 5.2.3 Member Function Documentation

#### 5.2.3.1 OnAddCharacter()

```
void appFrame::OnAddCharacter (
            wxCommandEvent & event )  [private]
```

Definition at line 41 of file appMain.cpp.

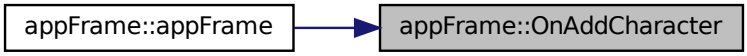
```
41                                                {
42   wxString charName = charNameTextCtrl->GetValue();
43
44   if (charName.IsEmpty()) {
45     wxMessageBox("Character's Name is empty", "Error", wxOK | wxICON_ERROR);
46     return;
47   }
```

```
48
49   wxPanel *newCharPanel = new appPlayerPanel(notebook, charName);
50   notebook->AddPage(newCharPanel, charName);
51
52   charNameTextCtrl->Clear();
53 }
```

References charNameTextCtrl, and notebook.

Referenced by appFrame().

Here is the caller graph for this function:




## 5.2.4 Member Data Documentation

### 5.2.4.1 addCharPanel

```
wxPanel* appFrame::addCharPanel  [private]
```

Definition at line 27 of file appMain.hpp.

Referenced by appFrame().

### 5.2.4.2 charNameTextCtrl

```
wxTextCtrl* appFrame::charNameTextCtrl  [private]
```

Definition at line 28 of file appMain.hpp.

Referenced by appFrame(), and OnAddCharacter().

### 5.2.4.3 gmPanel

```
wxPanel* appFrame::gmPanel  [private]
```

Definition at line 26 of file appMain.hpp.

Referenced by appFrame().

#### 5.2.4.4 notebook

```
wxNotebook* appFrame::notebook  [private]
```

Definition at line 25 of file appMain.hpp.

Referenced by appFrame(), and OnAddCharacter().

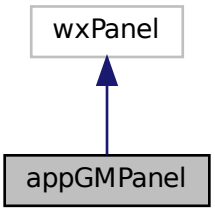
The documentation for this class was generated from the following files:

- appMain.hpp
- appMain.cpp

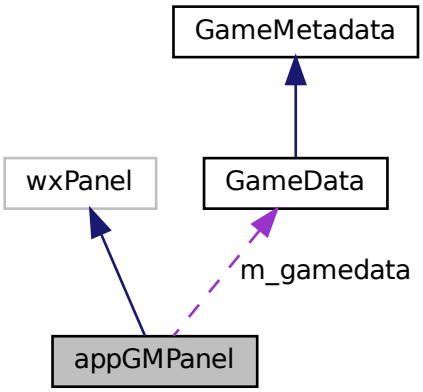
## 5.3 appGMPanel Class Reference

```
#include <appGM.hpp>
```

Inheritance diagram for appGMPanel:




Collaboration diagram for appGMPanel:



## Public Member Functions

- appGMPanel (wxNotebook *parent)

## Private Member Functions

- void OnAddStat (wxCommandEvent &event)
- void OnAddEquipment (wxCommandEvent &event)
- void OnAddSlot (wxCommandEvent &event)
- void OnAddState (wxCommandEvent &event)
- void UpdateStatsListCtrl ()
- void UpdateEqListCtrl ()
- void UpdateSlotListCtrl ()
- void UpdateStateListCtrl ()
- wxDECLARE_EVENT_TABLE ()

## Private Attributes

- wxNotebook * gmNotebook
- wxPanel * addElementPanel
- wxPanel * statsPanel
- wxPanel * eqPanel
- wxPanel * slotPanel
- wxPanel * statePanel
- wxBoxSizer * addElementSizer
- wxBoxSizer * statsSizer
- wxBoxSizer * eqSizer
- wxBoxSizer * slotSizer
- wxBoxSizer * stateSizer
- wxListCtrl * statsListCtrl
- wxListCtrl * eqListCtrl
- wxListCtrl * slotListCtrl
- wxListCtrl * stateListCtrl
- wxButton * addStatButton
- wxButton * addEquipmentButton
- wxButton * addSlotButton
- wxButton * addStateButton
- GameData * m_gamedata {nullptr}

### 5.3.1 Detailed Description

Definition at line 13 of file appGM.hpp.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 appGMPanel()

```
appGMPanel::appGMPanel (
            wxNotebook * parent )
```

### 5.3.3 Member Function Documentation

#### 5.3.3.1 OnAddEquipment()

```
void appGMPanel::OnAddEquipment (
            wxCommandEvent & event )  [private]
```

Definition at line 95 of file appGM.cpp.

```
95                                                  {
96   /*wxString name = wxGetTextFromUser("Enter name:", "Add Item");
97   wxString description = wxGetTextFromUser("Enter description:", "Add Item");
98
99   if (name.IsEmpty()) {
100    wxMessageBox("Name or description cannot be empty!", "Error",
101                 wxOK | wxICON_ERROR);
102    return;
103  }
104
105  Item* item{new Item(m_gamedata, name.ToStdString(),
106  description.ToStdString())};
107
108  if(m_gamedata == nullptr){
109        std::cerr«"you are fucking dumb\n";
110        std::abort();
111  }
112  m_gamedata->addItem(item);*/
113
114  // UpdateEqListCtrl();
115 }
```

#### 5.3.3.2 OnAddSlot()

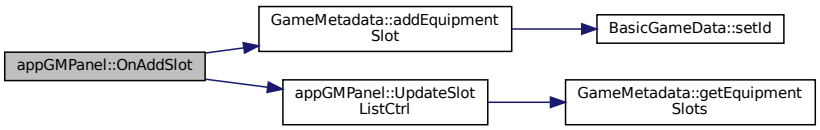
```
void appGMPanel::OnAddSlot (
            wxCommandEvent & event )  [private]
```

Definition at line 117 of file appGM.cpp.

```
117                                             {
118   wxString name = wxGetTextFromUser("Enter name:", "Add State");
119   wxString description = wxGetTextFromUser("Enter description:", "Add State");
120
121   if (name.IsEmpty()) {
122     wxMessageBox("Name or description cannot be empty!", "Error",
123                  wxOK | wxICON_ERROR);
124     return;
125   }
126
127   m_gamedata->addEquipmentSlot(
128       new EquipmentSlot(name.ToStdString(), description.ToStdString()));
129
130   UpdateSlotListCtrl();
131 }
```

References GameMetadata::addEquipmentSlot(), m_gamedata, and UpdateSlotListCtrl().

Here is the call graph for this function:




#### 5.3.3.3 OnAddStat()

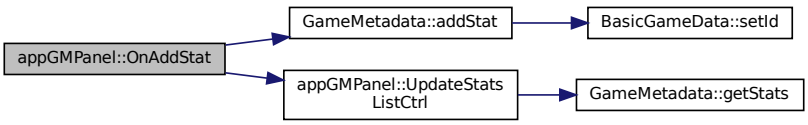
```
void appGMPanel::OnAddStat (
            wxCommandEvent & event )  [private]
```

Definition at line 80 of file appGM.cpp.

```
80                                                {
81   wxString name = wxGetTextFromUser("Enter name:", "Add Stat");
82   wxString description = wxGetTextFromUser("Enter description:", "Add Stat");
83
84   if (name.IsEmpty()) {
85     wxMessageBox("Name or description cannot be empty!", "Error",
86                  wxOK | wxICON_ERROR);
87     return;
88   }
89
90   m_gamedata->addStat(new Stat(name.ToStdString(), description.ToStdString()));
91
92   UpdateStatsListCtrl();
93 }
```

References GameMetadata::addStat(), m_gamedata, and UpdateStatsListCtrl().

Here is the call graph for this function:




#### 5.3.3.4 OnAddState()

```
void appGMPanel::OnAddState (
            wxCommandEvent & event )  [private]
```

Definition at line 133 of file appGM.cpp.

```
133                                               {
134   wxString name = wxGetTextFromUser("Enter name:", "Add State");
```

```
135   wxString description = wxGetTextFromUser("Enter description:", "Add State");
136
137   if (name.IsEmpty()) {
138     wxMessageBox("Name or description cannot be empty!", "Error",
139                  wxOK | wxICON_ERROR);
140     return;
141   }
142
143   State *state{
144       new State(m_gamedata, name.ToStdString(), description.ToStdString())};
145
146   UpdateStateListCtrl();
147 }
```

References m_gamedata, and UpdateStateListCtrl().

Here is the call graph for this function:


#### 5.3.3.5 UpdateEqListCtrl()

```
void appGMPanel::UpdateEqListCtrl ( )  [private]
```

Definition at line 159 of file appGM.cpp.

```
159                                  {
160   /*eqListCtrl->DeleteAllItems();
161
162   GameData::
163   for (size_t i = 0; i < equipment.size(); i++) {
164     Equipment *eq = equipment[i];
165     long index = eqListCtrl->InsertItem(i, eq->GetName());
166     eqListCtrl->SetItem(index, 1, eq->GetDescription());
167     eqListCtrl->SetItemData(index, reinterpret_cast<wxUIntPtr>(eq));
168   }*/
169 }
```

#### 5.3.3.6 UpdateSlotListCtrl()

```
void appGMPanel::UpdateSlotListCtrl ( )  [private]
```

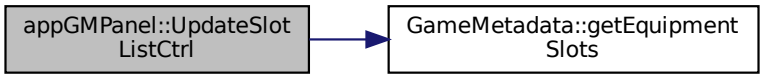
Definition at line 171 of file appGM.cpp.

```
171                                    {
172   slotListCtrl->DeleteAllItems();
173   long i{};
174   for (const auto &it : m_gamedata->getEquipmentSlots()) {
175     long index = slotListCtrl->InsertItem(i, it->getName());
176     ++i;
177   }
178 }
```

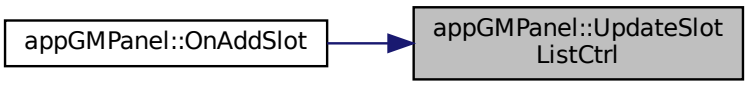
References GameMetadata::getEquipmentSlots(), m_gamedata, and slotListCtrl.

Referenced by OnAddSlot().

Here is the call graph for this function:




Here is the caller graph for this function:




#### 5.3.3.7 UpdateStateListCtrl()

```
void appGMPanel::UpdateStateListCtrl ( )  [private]
```

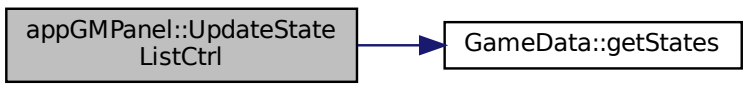
Definition at line 180 of file appGM.cpp.

```
180                                      {
181   stateListCtrl->DeleteAllItems();
182   long i{};
183   for (const auto &it : m_gamedata->getStates()) {
184     long index = stateListCtrl->InsertItem(i, it->getName());
185     ++i;
186   }
187 }
```

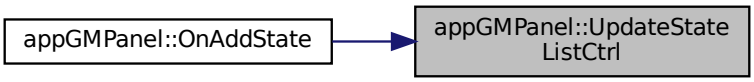
References GameData::getStates(), m_gamedata, and stateListCtrl.

Referenced by OnAddState().

Here is the call graph for this function:



Here is the caller graph for this function:




#### 5.3.3.8 UpdateStatsListCtrl()

```
void appGMPanel::UpdateStatsListCtrl ( )  [private]
```

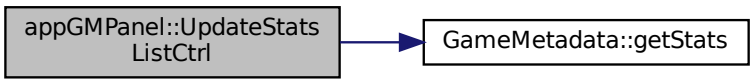
Definition at line 149 of file appGM.cpp.

```
149                                    {
150   statsListCtrl->DeleteAllItems();
151   long i{};
152   for (const auto &it : m_gamedata->getStats()) {
153     long index = statsListCtrl->InsertItem(i, it->getName());
154     statsListCtrl->SetItem(index, 1, it->getDescription());
155     ++i;
156   }
157 }
```

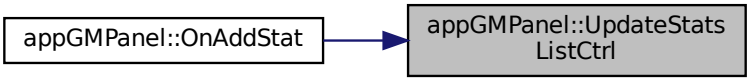
References GameMetadata::getStats(), m_gamedata, and statsListCtrl.

Referenced by OnAddStat().

Here is the call graph for this function:




Here is the caller graph for this function:



#### 5.3.3.9 wxDECLARE_EVENT_TABLE()

```
appGMPanel::wxDECLARE_EVENT_TABLE ( )  [private]
```

### 5.3.4 Member Data Documentation

#### 5.3.4.1 addElementPanel

```
wxPanel* appGMPanel::addElementPanel  [private]
```

Definition at line 20 of file appGM.hpp.

#### 5.3.4.2 addElementSizer

```
wxBoxSizer* appGMPanel::addElementSizer  [private]
```

Definition at line 26 of file appGM.hpp.

#### 5.3.4.3 addEquipmentButton

```
wxButton* appGMPanel::addEquipmentButton  [private]
```

Definition at line 38 of file appGM.hpp.

#### 5.3.4.4 addSlotButton

```
wxButton* appGMPanel::addSlotButton  [private]
```

Definition at line 39 of file appGM.hpp.

#### 5.3.4.5 addStatButton

```
wxButton* appGMPanel::addStatButton  [private]
```

Definition at line 37 of file appGM.hpp.

#### 5.3.4.6 addStateButton

```
wxButton* appGMPanel::addStateButton  [private]
```

Definition at line 40 of file appGM.hpp.

#### 5.3.4.7 eqListCtrl

```
wxListCtrl* appGMPanel::eqListCtrl  [private]
```

Definition at line 33 of file appGM.hpp.

#### 5.3.4.8 eqPanel

```
wxPanel* appGMPanel::eqPanel  [private]
```

Definition at line 22 of file appGM.hpp.

#### 5.3.4.9 eqSizer

```
wxBoxSizer* appGMPanel::eqSizer  [private]
```

Definition at line 28 of file appGM.hpp.

#### 5.3.4.10 gmNotebook

```
wxNotebook* appGMPanel::gmNotebook  [private]
```

Definition at line 18 of file appGM.hpp.

#### 5.3.4.11 m_gamedata

```
GameData* appGMPanel::m_gamedata {nullptr}  [private]
```

Definition at line 42 of file appGM.hpp.

Referenced by OnAddSlot(), OnAddStat(), OnAddState(), UpdateSlotListCtrl(), UpdateStateListCtrl(), and Update↩StatsListCtrl().

#### 5.3.4.12 slotListCtrl

```
wxListCtrl* appGMPanel::slotListCtrl  [private]
```

Definition at line 34 of file appGM.hpp.

Referenced by UpdateSlotListCtrl().

#### 5.3.4.13 slotPanel

```
wxPanel* appGMPanel::slotPanel  [private]
```

Definition at line 23 of file appGM.hpp.

#### 5.3.4.14 slotSizer

```
wxBoxSizer* appGMPanel::slotSizer  [private]
```

Definition at line 29 of file appGM.hpp.

#### 5.3.4.15 stateListCtrl

```
wxListCtrl* appGMPanel::stateListCtrl  [private]
```

Definition at line 35 of file appGM.hpp.

Referenced by UpdateStateListCtrl().

#### 5.3.4.16 statePanel

```
wxPanel* appGMPanel::statePanel  [private]
```

Definition at line 24 of file appGM.hpp.

#### 5.3.4.17 stateSizer

```
wxBoxSizer* appGMPanel::stateSizer  [private]
```

Definition at line 30 of file appGM.hpp.

#### 5.3.4.18 statsListCtrl

```
wxListCtrl* appGMPanel::statsListCtrl  [private]
```

Definition at line 32 of file appGM.hpp.

Referenced by UpdateStatsListCtrl().

#### 5.3.4.19 statsPanel

```
wxPanel* appGMPanel::statsPanel  [private]
```

Definition at line 21 of file appGM.hpp.

#### 5.3.4.20 statsSizer

```
wxBoxSizer* appGMPanel::statsSizer  [private]
```

Definition at line 27 of file appGM.hpp.

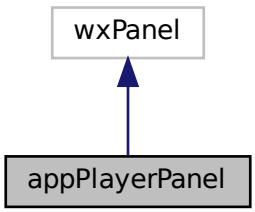
The documentation for this class was generated from the following files:

- appGM.hpp
- appGM.cpp

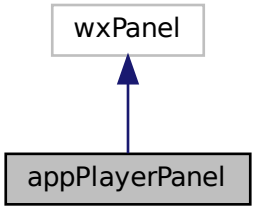
## 5.4 appPlayerPanel Class Reference

```
#include <appPlayer.hpp>
```

Inheritance diagram for appPlayerPanel:



Collaboration diagram for appPlayerPanel:




## Public Member Functions

- appPlayerPanel (wxNotebook *parent, const wxString &playerName)

## Private Member Functions

- void OnAddStat (wxCommandEvent &event)
- void OnAddEquipment (wxCommandEvent &event)
- void OnAddState (wxCommandEvent &event)
- void OnEquipEquipment (wxCommandEvent &event)
- void OnUnequipEquipment (wxCommandEvent &event)
- void UpdateStatsListCtrl ()
- void UpdateEquippedEQListCtrl ()
- void UpdateEQListCtrl ()
- void UpdateStatesListCtrl ()
- wxDECLARE_EVENT_TABLE ()

## Private Attributes

- wxNotebook * playerNotebook
- wxPanel * statsPanel
- wxPanel * equippedEQPanel
- wxPanel * eqPanel
- wxPanel * statesPanel
- wxPanel * addPanel
- wxBoxSizer * statsSizer
- wxBoxSizer * equippedEQSizer
- wxBoxSizer * eqSizer
- wxBoxSizer * statesSizer
- wxBoxSizer * addPanelSizer
- wxBoxSizer * mainSizer
- wxListCtrl * statsListCtrl
- wxListCtrl * equippedEQListCtrl
- wxListCtrl * eqListCtrl
- wxListCtrl * statesListCtrl
- wxButton * addStatButton
- wxButton * addEquipmentButton
- wxButton * addStateButton
- wxButton * equipEquipmentButton
- wxButton * unequipEquipmentButton

### 5.4.1 Detailed Description

Definition at line 19 of file appPlayer.hpp.

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 appPlayerPanel()

```
appPlayerPanel::appPlayerPanel (
            wxNotebook * parent,
            const wxString & playerName )
```

### 5.4.3 Member Function Documentation

#### 5.4.3.1 OnAddEquipment()

```
void appPlayerPanel::OnAddEquipment (
            wxCommandEvent & event )  [private]
```

Definition at line 94 of file appPlayer.cpp.

```
95 {
96     // TODO: Implement adding an equipment
97 }
```

#### 5.4.3.2 OnAddStat()

```
void appPlayerPanel::OnAddStat (
            wxCommandEvent & event )  [private]
```

Definition at line 89 of file appPlayer.cpp.

```
90 {
91     // TODO: Implement adding a stat
92 }
```

#### 5.4.3.3 OnAddState()

```
void appPlayerPanel::OnAddState (
            wxCommandEvent & event )  [private]
```

Definition at line 99 of file appPlayer.cpp.

```
100 {
101     // TODO: Implement adding a state
102 }
```

#### 5.4.3.4 OnEquipEquipment()

```
void appPlayerPanel::OnEquipEquipment (
            wxCommandEvent & event )  [private]
```

Definition at line 104 of file appPlayer.cpp.

```
105 {
106     // TODO: Implement equipping an equipment
107 }
```

#### 5.4.3.5 OnUnequipEquipment()

```
void appPlayerPanel::OnUnequipEquipment (
            wxCommandEvent & event )  [private]
```

Definition at line 109 of file appPlayer.cpp.

```
110 {
111     // TODO: Implement unequipping an equipment
112 }
```

#### 5.4.3.6 UpdateEQListCtrl()

```
void appPlayerPanel::UpdateEQListCtrl ( )  [private]
```

Definition at line 124 of file appPlayer.cpp.

```
125 {
126     // TODO: Update the equipment list control with data from eqCollection
127 }
```

#### 5.4.3.7 UpdateEquippedEQListCtrl()

```
void appPlayerPanel::UpdateEquippedEQListCtrl ( )  [private]
```

Definition at line 119 of file appPlayer.cpp.

```
120 {
121     // TODO: Update the equipped equipment list control with data from eqCollection
122 }
```

#### 5.4.3.8 UpdateStatesListCtrl()

```
void appPlayerPanel::UpdateStatesListCtrl ( )  [private]
```

Definition at line 129 of file appPlayer.cpp.

```
130 {
131     // TODO: Update the states list control with data from stateCollection
132 }
```

#### 5.4.3.9 UpdateStatsListCtrl()

```
void appPlayerPanel::UpdateStatsListCtrl ( )  [private]
```

Definition at line 114 of file appPlayer.cpp.

```
115 {
116     // TODO: Update the stats list control with data from statsCollection
117 }
```

#### 5.4.3.10 wxDECLARE_EVENT_TABLE()

```
appPlayerPanel::wxDECLARE_EVENT_TABLE ( )  [private]
```

### 5.4.4 Member Data Documentation

#### 5.4.4.1 addEquipmentButton

```
wxButton* appPlayerPanel::addEquipmentButton  [private]
```

Definition at line 46 of file appPlayer.hpp.

#### 5.4.4.2 addPanel

```
wxPanel* appPlayerPanel::addPanel  [private]
```

Definition at line 31 of file appPlayer.hpp.

#### 5.4.4.3 addPanelSizer

```
wxBoxSizer* appPlayerPanel::addPanelSizer  [private]
```

Definition at line 37 of file appPlayer.hpp.

#### 5.4.4.4 addStatButton

```
wxButton* appPlayerPanel::addStatButton  [private]
```

Definition at line 45 of file appPlayer.hpp.

#### 5.4.4.5 addStateButton

```
wxButton* appPlayerPanel::addStateButton  [private]
```

Definition at line 47 of file appPlayer.hpp.

#### 5.4.4.6 eqListCtrl

```
wxListCtrl* appPlayerPanel::eqListCtrl  [private]
```

Definition at line 42 of file appPlayer.hpp.

#### 5.4.4.7 eqPanel

```
wxPanel* appPlayerPanel::eqPanel  [private]
```

Definition at line 29 of file appPlayer.hpp.

#### 5.4.4.8 eqSizer

```
wxBoxSizer* appPlayerPanel::eqSizer  [private]
```

Definition at line 35 of file appPlayer.hpp.

#### 5.4.4.9 equipEquipmentButton

```
wxButton* appPlayerPanel::equipEquipmentButton  [private]
```

Definition at line 48 of file appPlayer.hpp.

#### 5.4.4.10 equippedEQListCtrl

```
wxListCtrl* appPlayerPanel::equippedEQListCtrl  [private]
```

Definition at line 41 of file appPlayer.hpp.

#### 5.4.4.11 equippedEQPanel

```
wxPanel* appPlayerPanel::equippedEQPanel  [private]
```

Definition at line 28 of file appPlayer.hpp.

#### 5.4.4.12 equippedEQSizer

```
wxBoxSizer* appPlayerPanel::equippedEQSizer  [private]
```

Definition at line 34 of file appPlayer.hpp.

#### 5.4.4.13 mainSizer

```
wxBoxSizer* appPlayerPanel::mainSizer  [private]
```

Definition at line 38 of file appPlayer.hpp.

#### 5.4.4.14 playerNotebook

```
wxNotebook* appPlayerPanel::playerNotebook  [private]
```

Definition at line 25 of file appPlayer.hpp.

#### 5.4.4.15 statesListCtrl

```
wxListCtrl* appPlayerPanel::statesListCtrl  [private]
```

Definition at line 43 of file appPlayer.hpp.

#### 5.4.4.16 statesPanel

```
wxPanel* appPlayerPanel::statesPanel  [private]
```

Definition at line 30 of file appPlayer.hpp.

#### 5.4.4.17 statesSizer

```
wxBoxSizer* appPlayerPanel::statesSizer  [private]
```

Definition at line 36 of file appPlayer.hpp.

#### 5.4.4.18 statsListCtrl

```
wxListCtrl* appPlayerPanel::statsListCtrl  [private]
```

Definition at line 40 of file appPlayer.hpp.

#### 5.4.4.19 statsPanel

```
wxPanel* appPlayerPanel::statsPanel  [private]
```

Definition at line 27 of file appPlayer.hpp.

#### 5.4.4.20 statsSizer

```
wxBoxSizer* appPlayerPanel::statsSizer  [private]
```

Definition at line 33 of file appPlayer.hpp.

#### 5.4.4.21 unequipEquipmentButton

```
wxButton* appPlayerPanel::unequipEquipmentButton  [private]
```

Definition at line 49 of file appPlayer.hpp.

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

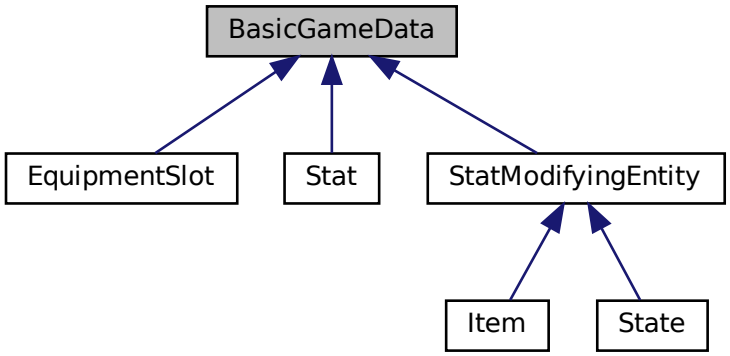
- appPlayer.hpp
- appPlayer.cpp

## 5.5 BasicGameData Class Reference

Basic information about game.

`#include <basicGamedata.hpp>`

Inheritance diagram for BasicGameData:




### Public Types

- using id_t = long long
  *Type used for ids.*

### Public Member Functions

- BasicGameData (std::string name, std::string description="")
  *Constructor creating object with given name and optional description.*
- void setName (std::string name)
  *Sets name.*
- void setDescription (std::string description)
  *Sets description.*
- std::string getName () const
  *name getter.*
- std::string getDescription () const
  *Description getter.*
- id_t getId () const
  *id getter.*
- bool isId (id_t id) const
  *Check if id is equal.*

## Static Public Attributes

- static constexpr id_t INVALID_ID {std::numeric_limits<id_t>::min()}

  *Value indicating that id is invalid.*

## Protected Member Functions

- void setId (id_t id)

  *id setter.*
- void validateIntegrity () const

  *Check itegrity of data.*

## Private Attributes

- id_t m_id {INVALID_ID}

  *id.*
- std::string m_name

  *name.*
- std::string m_description

  *description.*

## Friends

- class GameMetadata
- class GameData

### 5.5.1 Detailed Description

Basic information about game.

Definition at line 27 of file basicGamedata.hpp.

### 5.5.2 Member Typedef Documentation

#### 5.5.2.1 id_t

```
using BasicGameData::id_t = long long
```

Type used for ids.

Definition at line 33 of file basicGamedata.hpp.

### 5.5.3 Constructor & Destructor Documentation

#### 5.5.3.1 BasicGameData()

```
BasicGameData::BasicGameData (
            std::string name,
            std::string description = "" )
```

Constructor creating object with given name and optional description.

**Parameters**

| | |
|---|---|
| *name* | Name. |
| *description* | optional description. |

Definition at line 14 of file basicGamedata.cpp.

```
15     : m_name(name), m_description(description) {}
```

## 5.5.4 Member Function Documentation

### 5.5.4.1 getDescription()

```
std::string BasicGameData::getDescription ( ) const
```

Description getter.

**Returns**

Description.

Definition at line 38 of file basicGamedata.cpp.

```
38 { return m_description; };
```

References m_description.

### 5.5.4.2 getId()

```
BasicGameData::id_t BasicGameData::getId ( ) const
```

id getter.

**Returns**

id.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | When tred to get id of instance that has BasicGameData::INVALID_ID. |

Definition at line 40 of file basicGamedata.cpp.

```
40                                          {
41   if (m_id == INVALID_ID)
44     throw exceptionIllegalId();
45   return m_id;
46 };
```

References INVALID_ID, and m_id.

#### 5.5.4.3 getName()

```
std::string BasicGameData::getName ( ) const
```

name getter.

**Returns**

name

Definition at line 36 of file basicGamedata.cpp.

```
36 { return m_name; }
```

References m_name.

#### 5.5.4.4 isId()

```
bool BasicGameData::isId (
            id_t id ) const
```

Check if id is equal.

**Parameters**

| | |
|---|---|
| *id* | Id to check. |

**Returns**

True if id is same as parameter.

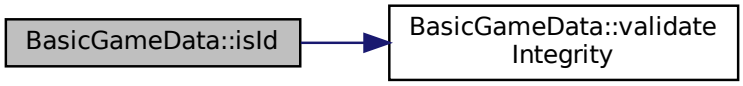
Definition at line 48 of file basicGamedata.cpp.

```
48                                          {
49   validateIntegrity();
50   bool res{m_id == id};
51   return res;
52 }
```

References m_id, and validateIntegrity().

Referenced by GameData::getItem().

Here is the call graph for this function:




Here is the caller graph for this function:


#### 5.5.4.5 setDescription()

```
void BasicGameData::setDescription (
            std::string description )
```

Sets description.

**Parameters**

| | |
|---|---|
| *description* | Description to be set. |

Definition at line 32 of file basicGamedata.cpp.

```
32                                                          {
33   m_description = description;
34 }
```

References m_description.

#### 5.5.4.6 setId()

```
void BasicGameData::setId (
            id_t id )  [protected]
```

id setter.

**Parameters**

| | |
|---|---|
| *id* | Id to set. If id to be set is INVALID_ID this will not be set. |

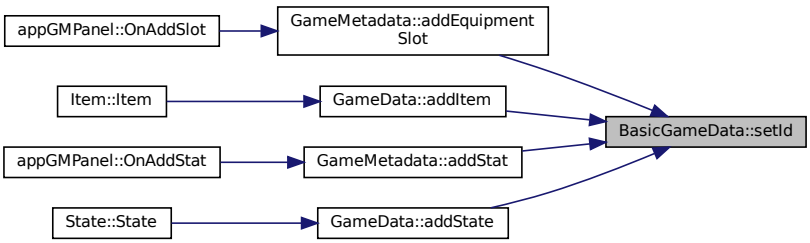
Definition at line 17 of file basicGamedata.cpp.

```
17                                      {
18   if (id == INVALID_ID)
19     return;
20   m_id = id;
21 }
```

References INVALID_ID, and m_id.

Referenced by GameMetadata::addEquipmentSlot(), GameData::addItem(), GameMetadata::addStat(), and GameData::addState().

Here is the caller graph for this function:




### 5.5.4.7 setName()

```
void BasicGameData::setName (
            std::string name )
```

Sets name.

**Parameters**

| | |
|---|---|
| *name* | name to be set. |

Definition at line 30 of file basicGamedata.cpp.

```
30 { m_name = name; }
```

References m_name.

### 5.5.4.8 validateIntegrity()

```
void BasicGameData::validateIntegrity ( ) const  [protected]
```

Check itegrity of data.

Checks if id is INVALID_ID.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | id is illegal. |

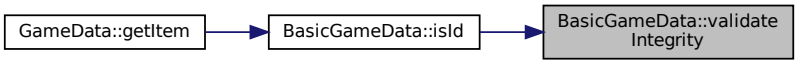
Definition at line 22 of file basicGamedata.cpp.

```
22                                                  {
24   if (m_id == INVALID_ID) {
26     throw exceptionIllegalId();
27   }
28 }
```

References INVALID_ID, and m_id.

Referenced by isId().

Here is the caller graph for this function:




### 5.5.5 Friends And Related Function Documentation

#### 5.5.5.1 GameData

```
friend class GameData  [friend]
```

Definition at line 29 of file basicGamedata.hpp.

#### 5.5.5.2 GameMetadata

```
friend class GameMetadata  [friend]
```

Definition at line 28 of file basicGamedata.hpp.

### 5.5.6 Member Data Documentation

#### 5.5.6.1 INVALID_ID

```
constexpr id_t BasicGameData::INVALID_ID {std::numeric_limits<id_t>::min()}  [static], [constexpr]
```

Value indicating that id is invalid.

Definition at line 35 of file basicGamedata.hpp.

Referenced by getId(), setId(), and validateIntegrity().

#### 5.5.6.2 m_description

```
std::string BasicGameData::m_description  [private]
```

description.

Definition at line 43 of file basicGamedata.hpp.

Referenced by getDescription(), and setDescription().

#### 5.5.6.3 m_id

```
id_t BasicGameData::m_id {INVALID_ID}  [private]
```

id.

Definition at line 39 of file basicGamedata.hpp.

Referenced by getId(), isId(), setId(), and validateIntegrity().

#### 5.5.6.4 m_name

```
std::string BasicGameData::m_name  [private]
```

name.

Definition at line 41 of file basicGamedata.hpp.

Referenced by getName(), and setName().

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

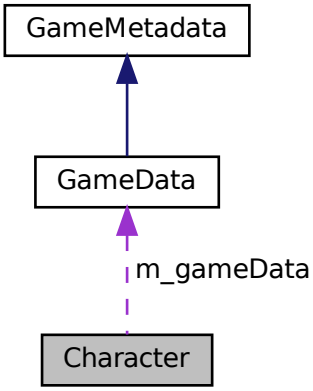
- basicGamedata.hpp
- basicGamedata.cpp

## 5.6 Character Class Reference

Represents character.

`#include <character.hpp>`

Collaboration diagram for Character:




### Public Member Functions

- Character (GameData *gameData)
  *constructor*
- void setBaseStatValue (Stat::id_t statId, Stat::value_t val)
  *Sets base stat value.*
- Stat::value_t getBaseStatValue (Stat::id_t id) const
  *Base statistics value getter.*
- statValueContrubitors_t getStatValueStatesContrubitors (Stat::id_t id) const
  *Gets collection of stat modyfiers caused by states.*
- statValueContrubitors_t getStatValueEquipmentContrubitors (Stat::id_t id) const
  *Gets collection of stat modyfiers caused by equipment.*
- statValueContrubitors_t getStatValueContrubitors (Stat::id_t id) const
  *Gets collection of stat modyfiers.*
- Stat::value_t getStatValue (Stat::id_t id) const
  *Gets final stat value.*
- const GameData *const getGameData () const
  *Gets game data used by Character.*
- void addItem (const Item *const item, itemQuantity_t quantity=1)
  *Adds item to character inventory.*
- const inventory_t & getInventory () const
  *Inventory getter.*
- const equipment_t & getEquipment () const
  *equipment getter*

- bool isEquipmentSlotUsed (EquipmentSlot::id_t eqSlotId) const

  *Check if slot is used.*
- const Item *const getEquipedItem (EquipmentSlot::id_t slotId) const

  *Get item equiped in given slot.*
- void equipItem (const Item *const item, EquipmentSlot::id_t eqSlot)

  *Equip item into eqSlot.*
- void addState (const State *state)

  *Adds State to the Character.*
- const states_t & getStates () const

  *States getter.*

## Private Types

- using statValues_t = std::map< Stat::id_t, Stat::value_t >

  *Type used for repesentation of base character stats.*
- using statValueContrubitors_t = std::vector< const StatModifyingEntity * >

  *Stat value contrubitors collection.*
- using itemQuantity_t = long long

  *Quantity of Items.*
- using inventory_t = std::map< const Item *const, itemQuantity_t >

  *Used for inventory as <Item, qunatity possed by Character>*
- using equipment_t = std::map< const EquipmentSlot *const, const Item *const >

  *Used to store equiped items of character.*
- using states_t = std::set< const State * >

  *Used to store sates affecting Character.*

## Private Member Functions

- void validateDataIntegrity (const StatModifyingEntity &entity) const

  *Validate data integrty.*
- const Item *const getEquipedItem (const EquipmentSlot *const eqSlot) const

  *Get item equiped in given slot.*

## Private Attributes

- const GameData *const m_gameData

  *Game data used by character.*
- statValues_t m_baseStatValues

  *Base values of stats.*
- inventory_t m_inventory

  *Inventorty.*
- equipment_t m_equipment

  *Equiped Items.*
- states_t m_states

  *States affecting Character.*

### 5.6.1 Detailed Description

Represents character.

Base character stats are stats that character has without any modifiers applied. If Base stat is not set it will be assumed to be 0;

Definition at line 62 of file character.hpp.

### 5.6.2 Member Typedef Documentation

#### 5.6.2.1 equipment_t

```
using Character::equipment_t = std::map<const EquipmentSlot *const, const Item *const>  [private]
```

Used to store equiped items of character.

Definition at line 75 of file character.hpp.

#### 5.6.2.2 inventory_t

```
using Character::inventory_t = std::map<const Item *const, itemQuantity_t>  [private]
```

Used for inventory as <Item, qunatity possed by Character>

Definition at line 72 of file character.hpp.

#### 5.6.2.3 itemQuantity_t

```
using Character::itemQuantity_t = long long  [private]
```

Quantity of Items.

Definition at line 69 of file character.hpp.

#### 5.6.2.4 states_t

```
using Character::states_t = std::set<const State *>  [private]
```

Used to store sates affecting Character.

Definition at line 78 of file character.hpp.

#### 5.6.2.5 statValueContrubitors_t

```
using Character::statValueContrubitors_t = std::vector<const StatModifyingEntity *>  [private]
```

Stat value contrubitors collection.

Definition at line 66 of file character.hpp.

#### 5.6.2.6 statValues_t

```
using Character::statValues_t = std::map<Stat::id_t, Stat::value_t>  [private]
```

Type used for repesentation of base character stats.

Definition at line 64 of file character.hpp.

### 5.6.3 Constructor & Destructor Documentation

#### 5.6.3.1 Character()

```
Character::Character (
            GameData * gameData )
```

constructor

**Parameters**

| | |
|---|---|
| *gameData* | Used by character.<br>`Character *character{new Character(gd)};  character->setBaseStatValue(1, 4);` |

Definition at line 30 of file character.cpp.

```
30 : m_gameData(gameData) {}
```

### 5.6.4 Member Function Documentation

#### 5.6.4.1 addItem()

```
void Character::addItem (
            const Item *const item,
            itemQuantity_t quantity = 1 )
```

Adds item to character inventory.

**Parameters**

| | |
|---|---|
| *item* | Item to add. |
| *quantity* | Qunatity of item to add. |

```
// extract item from gamedata accessed from character.
const Item *itemToAdd{character->getGameData()->getItem(1)};
character->addItem(itemToAdd);
character->addItem(itemToAdd);
character->addItem(character->getGameData()->getItem(2));
```

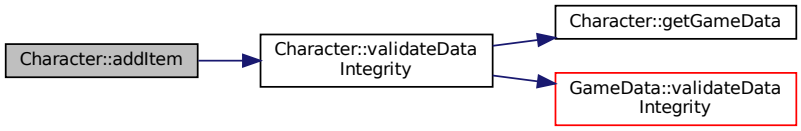
**Todo** make it remove item from inventory if quantity would go below or equal 0.

Definition at line 98 of file character.cpp.

```
98                                                                  {
99   validateDataIntegrity(*item);
100   if (quantity == 0)
101     return;
102
103   auto temp{m_inventory.insert({item, quantity})};
104
105   const bool didAdd{temp.second};
106   // If not added, change item quantity.
107   if (!didAdd) {
108     // iterator to item
109     auto itemIterator{temp.first};
110     itemQuantity_t &itemQuantity{itemIterator->second};
111     itemQuantity += quantity;
114   }
115 }
```

References m_inventory, and validateDataIntegrity().

Here is the call graph for this function:




### 5.6.4.2 addState()

```
void Character::addState (
            const State * state )
```

Adds State to the Character.

**Parameters**

| | |
|---|---|
| *state* | State to add. |

```
// Getting State which has id 2.
const State *state{character->getGameData()->getState(2)};
// Adding State to character.
```

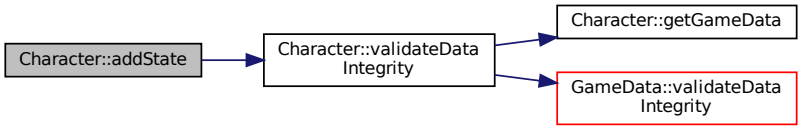
```
  character->addState(state);
```

Definition at line 160 of file character.cpp.

```
160                                         {
161   validateDataIntegrity(*state);
162   m_states.insert(state);
163 }
```

References m_states, and validateDataIntegrity().

Here is the call graph for this function:




#### 5.6.4.3 equipItem()

```
void Character::equipItem (
              const Item *const item,
              EquipmentSlot::id_t eqSlot )
```

Equip item into eqSlot.

**Parameters**

| | |
|---|---|
| *item* | Item to equip. |
| *eqSlot* | Where to equip it to. |

**Note**

It does not check whenever Character has item in invnetory.

**Exceptions**

| | |
|---|---|
| *excpetionEquipmentSlotIllegalUsage* | When attempting to equip item into slot that it can not be equiped to. |
| *exceptionEquipmentSlotOccupied* | When attempting to equip item into already occupied slot; |

**Warning**

If moving Item from inventory to equipment(as if it was take out of inventory and put on by Character). Removal of Item should be performed after this method call. If it's removed before and exception is thrown Item would have been removed from inventory and not be equiped.

```
// Item to equip.
```

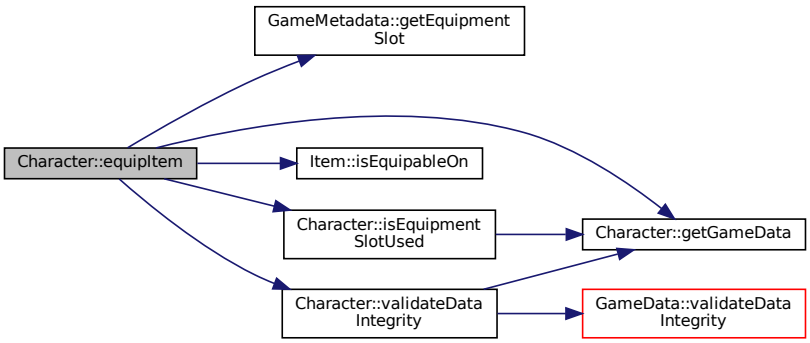
```
const Item *item{character->getGameData()->getItem(2)};
// Slot where to equip that item into.
EquipmentSlot::id_t slot{*item->getEquipableSlots().begin()};
// Optional checking if Character has that item in inventory should be here.
character->equipItem(item, slot);
// Optional removal of item from Inventory.
```

Definition at line 144 of file character.cpp.

```
144                                                                    {
145   validateDataIntegrity(*item);
146   // check if item can even be equiped into that slot.
147   if (!item->isEquipableOn(eqSlot))
148     throw excpetionEquipmentSlotIllegalUsage();
149
150   // check if desired slot is occupied.
151   if (isEquipmentSlotUsed(eqSlot))
152     throw exceptionEquipmentSlotOccupied();
153
154   // get slot pointer as it's used for internal equipment infromation.
155   const EquipmentSlot *const eqSlotPtr{getGameData()->getEquipmentSlot(eqSlot)};
156
157   m_equipment.insert({eqSlotPtr, item});
158 }
```

References GameMetadata::getEquipmentSlot(), getGameData(), Item::isEquipableOn(), isEquipmentSlotUsed(), m_equipment, and validateDataIntegrity().

Here is the call graph for this function:




#### 5.6.4.4 getBaseStatValue()

```
Stat::value_t Character::getBaseStatValue (
            Stat::id_t id ) const
```

Base statistics value getter.

**Parameters**

| | |
|---|---|
| *id* | of stat to get base value of. |

**Returns**

Base value of stat with given id.

If Character does not have base Stat value of given id 0 is returned as per class invariant.

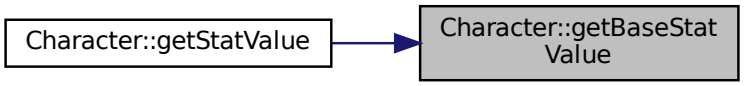
Definition at line 38 of file character.cpp.

```
38                                                          {
39   auto res{m_baseStatValues.find(id)};
40   if (res != m_baseStatValues.end())
41     return res->second;
42   return 0;
43 }
```

References m_baseStatValues.

Referenced by getStatValue().

Here is the caller graph for this function:




#### 5.6.4.5 getEquipedItem() [1/2]

```
const Item *const Character::getEquipedItem (
            const EquipmentSlot *const eqSlot ) const  [private]
```

Get item equiped in given slot.

**Parameters**

| | |
|---|---|
| *eqSlot* | EquipmentSlot to get equiped item of. |

**Exceptions**

| | |
|---|---|
| *exceptionEquipmentSlotUnused* | When trying to get equiped item of unused slot. |

**Returns**

Equiped item.

Definition at line 132 of file character.cpp.

```
132                                                         {
```

```
133   equipment_t::const_iterator r{getEquipment().find(eqSlot)};
134   if (r == getEquipment().end())
135     throw exceptionEquipmentSlotUnused();
136   return r->second;
137 }
```

References getEquipment().

Referenced by getEquipedItem().

Here is the call graph for this function:


Here is the caller graph for this function:


#### 5.6.4.6 getEquipedItem() [2/2]

```
const Item *const Character::getEquipedItem (
            EquipmentSlot::id_t slotId ) const
```

Get item equiped in given slot.

**Parameters**

| | |
|---|---|
| *slotId* | EquipmentSlot to get equiped item of. |

**Returns**

Equiped item.

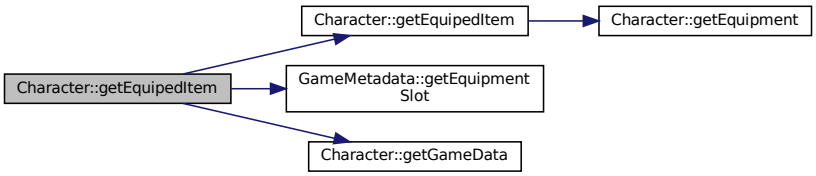
Definition at line 139 of file character.cpp.

```
139                                                                    {
140   const EquipmentSlot *const eqSlot{getGameData()->getEquipmentSlot(slotId)};
141   return getEquipedItem(eqSlot);
```

```
142 }
```

References getEquipedItem(), GameMetadata::getEquipmentSlot(), and getGameData().

Here is the call graph for this function:




#### 5.6.4.7 getEquipment()

```
const Character::equipment_t & Character::getEquipment ( ) const
```

equipment getter

**Returns**

m_equipment

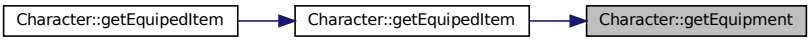
Definition at line 121 of file character.cpp.

```
121                                                          {
122   return m_equipment;
123 }
```

References m_equipment.

Referenced by getEquipedItem().

Here is the caller graph for this function:



#### 5.6.4.8 getGameData()

```
const GameData *const Character::getGameData ( ) const
```

Gets game data used by Character.

**Returns**

::m_gameData.

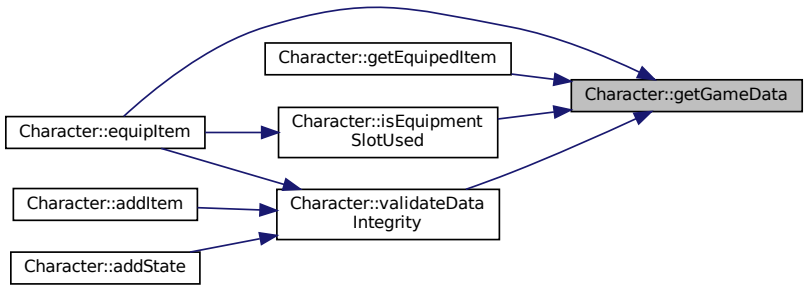
Definition at line 96 of file character.cpp.

```
96 { return m_gameData; }
```

References m_gameData.

Referenced by equipItem(), getEquipedItem(), isEquipmentSlotUsed(), and validateDataIntegrity().

Here is the caller graph for this function:




#### 5.6.4.9 getInventory()

```
const Character::inventory_t & Character::getInventory ( ) const
```

Inventory getter.

**Returns**

m_inventory

Definition at line 117 of file character.cpp.

```
117                                                 {
118   return m_inventory;
119 }
```

References m_inventory.

#### 5.6.4.10 getStates()

```
const Character::states_t & Character::getStates ( ) const
```

States getter.

**Returns**

m_states

Definition at line 165 of file character.cpp.

```
165 { return m_states; }
```

References m_states.

#### 5.6.4.11 getStatValue()

```
Stat::value_t Character::getStatValue (
            Stat::id_t id ) const
```

Gets final stat value.

**Parameters**

| | |
|---|---|
| *id* | Id of stat to get value of. |

**Returns**

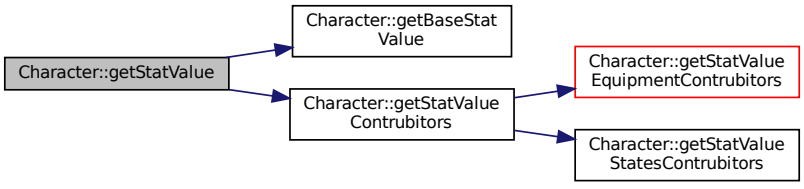
Value of stat.

Definition at line 85 of file character.cpp.

```
85                                                  {
86   statValueContrubitors_t contrubitors{getStatValueContrubitors(id)};
87
88   Stat::value_t result{getBaseStatValue(id)};
89
90   for (const StatModifyingEntity *contr : contrubitors) {
91     result += contr->getModifierValue(id);
92   }
93   return result;
94 }
```

References getBaseStatValue(), and getStatValueContrubitors().

Here is the call graph for this function:



#### 5.6.4.12 getStatValueContrubitors()

```
Character::statValueContrubitors_t Character::getStatValueContrubitors (
            Stat::id_t id ) const
```

Gets collection of stat modyfiers.

**Parameters**

| | |
|---|---|
| *id* | Id of stat. |

**Returns**

Contrubuitors to Stat value.

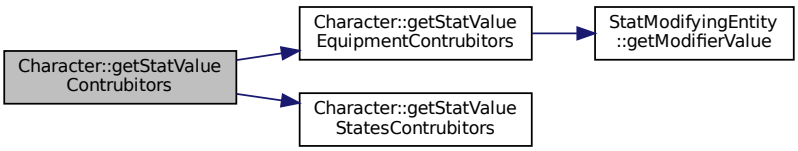
Definition at line 73 of file character.cpp.

```
73                                                  {
74   statValueContrubitors_t result{};
75   auto temp{getStatValueEquipmentContrubitors(id)};
76   // equipment contribs.
77   result.insert(result.end(), temp.begin(), temp.end());
78   temp = getStatValueStatesContrubitors(id);
79   // states contribs.
80   result.insert(result.end(), temp.begin(), temp.end());
81
82   return result;
83 }
```

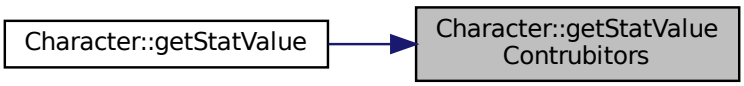
References getStatValueEquipmentContrubitors(), and getStatValueStatesContrubitors().

Referenced by getStatValue().

Here is the call graph for this function:




Here is the caller graph for this function:



#### 5.6.4.13 getStatValueEquipmentContrubitors()

```
Character::statValueContrubitors_t Character::getStatValueEquipmentContrubitors (
            Stat::id_t id ) const
```

Gets collection of stat modyfiers caused by equipment.

**Parameters**

| | |
|---|---|
| *id* | Id of stat. |

**Returns**

Contrubuitors to Stat value.

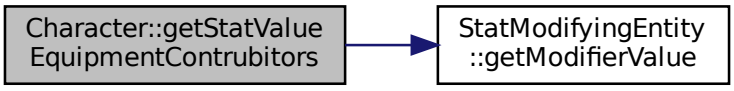
Definition at line 58 of file character.cpp.

```
58                                                                    {
59   statValueContrubitors_t result{};
60   // For each equipment piece
61   for (auto it : m_equipment) {
62     const StatModifyingEntity *modifier{it.second};
63     // Get modifier of stat caused by that piece.
64     Stat::value_t modv{modifier->getModifierValue(id)};
65     // if it modifies stat add it to result.
66     if (modv != 0)
67       result.push_back(modifier);
68   }
69   return result;
70 }
```

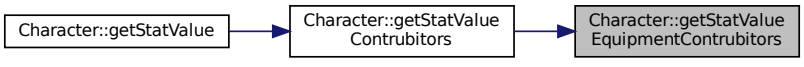
References StatModifyingEntity::getModifierValue(), and m_equipment.

Referenced by getStatValueContrubitors().

Here is the call graph for this function:




Here is the caller graph for this function:



#### 5.6.4.14 getStatValueStatesContrubitors()

```
Character::statValueContrubitors_t Character::getStatValueStatesContrubitors (
            Stat::id_t id ) const
```

Gets collection of stat modyfiers caused by states.

**Parameters**

| | |
|---|---|
| *id* | Id of stat. |

**Returns**

Contrubuitors to Stat value.

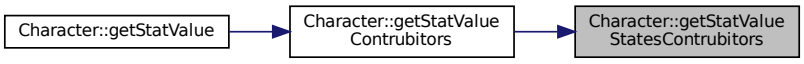
Definition at line 46 of file character.cpp.

```
46                                                                  {
47   statValueContrubitors_t result{};
48   for (auto it : m_states) {
49     Stat::value_t modv{it->getModifierValue(id)};
50     // if it modifies stat add it to result.
51     if (modv != 0)
52       result.push_back(it);
53   }
54   return result;
55 }
```

References m_states.

Referenced by getStatValueContrubitors().

Here is the caller graph for this function:




#### 5.6.4.15 isEquipmentSlotUsed()

```
bool Character::isEquipmentSlotUsed (
            EquipmentSlot::id_t eqSlotId ) const
```

Check if slot is used.

**Parameters**

| | |
|---|---|
| *eq↩ SlotId* | id of eq slot to check. |

**Returns**

True if slot is used. False if it is not used.

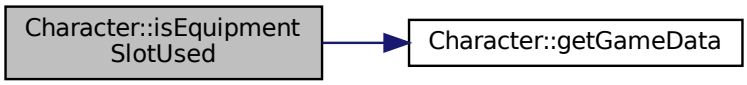
Definition at line 125 of file character.cpp.

```
125                                                                 {
126   bool result{m_equipment.find(getGameData()->getEquipmentSlot(eqSlotId)) !=
127               m_equipment.end()};
128   return result;
129 }
```

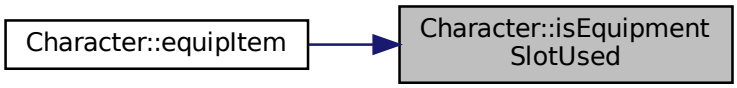
References getGameData(), and m_equipment.

Referenced by equipItem().

Here is the call graph for this function:




Here is the caller graph for this function:




#### 5.6.4.16 setBaseStatValue()

```
void Character::setBaseStatValue (
            Stat::id_t statId,
            Stat::value_t val )
```

Sets base stat value.

**Parameters**

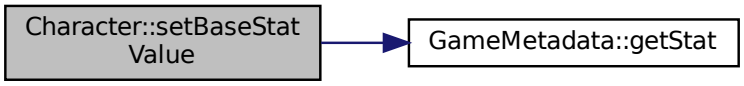
| | |
|---|---|
| *statId* | Id to set value of. |
| *val* | Value to set. |

Definition at line 32 of file character.cpp.

```
32                                                                  {
33   m_gameData->getStat(statId);
34
35   m_baseStatValues[statId] = val;
36 }
```

References GameMetadata::getStat(), m_baseStatValues, and m_gameData.

Here is the call graph for this function:




#### 5.6.4.17 validateDataIntegrity()

```
void Character::validateDataIntegrity (
            const StatModifyingEntity & entity ) const  [private]
```

Validate data integrty.

**Parameters**

| | |
|---|---|
| *entity* | Entity to validate agints. |

**Exceptions**

| | |
|---|---|
| *excpetionGameDataMissmatch* | When Character and item do not use same GameData. |

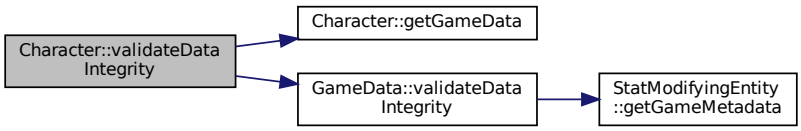
Definition at line 26 of file character.cpp.

```
26                                                                  {
27   getGameData()->validateDataIntegrity(entity);
28 }
```

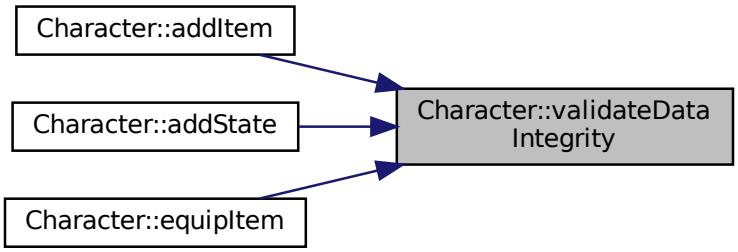
References getGameData(), and GameData::validateDataIntegrity().

Referenced by addItem(), addState(), and equipItem().

Here is the call graph for this function:




Here is the caller graph for this function:




### 5.6.5 Member Data Documentation

#### 5.6.5.1 m_baseStatValues

```
statValues_t Character::m_baseStatValues  [private]
```

Base values of stats.

Definition at line 83 of file character.hpp.

Referenced by getBaseStatValue(), and setBaseStatValue().

#### 5.6.5.2 m_equipment

```
equipment_t Character::m_equipment  [private]
```

Equiped Items.

Definition at line 87 of file character.hpp.

Referenced by equipItem(), getEquipment(), getStatValueEquipmentContrubitors(), and isEquipmentSlotUsed().

#### 5.6.5.3 m_gameData

```
const GameData* const Character::m_gameData  [private]
```

Game data used by character.

Definition at line 81 of file character.hpp.

Referenced by getGameData(), and setBaseStatValue().

#### 5.6.5.4 m_inventory

```
inventory_t Character::m_inventory  [private]
```

Inventorty.

Definition at line 85 of file character.hpp.

Referenced by addItem(), and getInventory().

#### 5.6.5.5 m_states

```
states_t Character::m_states  [private]
```

States affecting Character.

Definition at line 89 of file character.hpp.

Referenced by addState(), getStates(), and getStatValueStatesContrubitors().

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

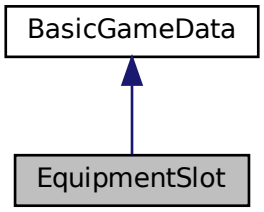
- character.hpp
- character.cpp

## 5.7 EquipmentSlot Class Reference

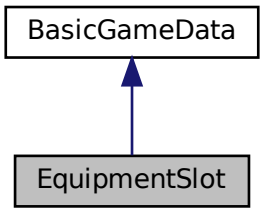
Equipment slot repesentation.

`#include <equipmentSlot.hpp>`

Inheritance diagram for EquipmentSlot:




Collaboration diagram for EquipmentSlot:




### Public Types

- using id_t = long long
  *Type used for ids.*

### Public Member Functions

- EquipmentSlot (std::string name, std::string description="")
  *Constructor.*
- void setName (std::string name)
  *Sets name.*
- void setDescription (std::string description)
  *Sets description.*

- std::string getName () const
  *name getter.*
- std::string getDescription () const
  *Description getter.*
- id_t getId () const
  *id getter.*
- bool isId (id_t id) const
  *Check if id is equal.*

### Static Public Attributes

- static constexpr id_t INVALID_ID {std::numeric_limits<id_t>::min()}
  *Value indicating that id is invalid.*

### Protected Member Functions

- void setId (id_t id)
  *id setter.*
- void validateIntegrity () const
  *Check itegrity of data.*

### Private Attributes

- id_t m_id {INVALID_ID}
  *id.*
- std::string m_name
  *name.*
- std::string m_description
  *description.*

## 5.7.1 Detailed Description

Equipment slot repesentation.

Definition at line 14 of file equipmentSlot.hpp.

## 5.7.2 Member Typedef Documentation

#### 5.7.2.1 id_t

```
using BasicGameData::id_t = long long  [inherited]
```

Type used for ids.

Definition at line 33 of file basicGamedata.hpp.

## 5.7.3 Constructor & Destructor Documentation

### 5.7.3.1 EquipmentSlot()

```
EquipmentSlot::EquipmentSlot (
            std::string name,
            std::string description = "" )
```

Constructor.

**Parameters**

| | |
|---|---|
| *name* | Name. |
| *description* | Optional description. |

Definition at line 8 of file equipmentSlot.cpp.

```
9     : BasicGameData(name, description) {}
```

## 5.7.4 Member Function Documentation

### 5.7.4.1 getDescription()

```
std::string BasicGameData::getDescription ( ) const  [inherited]
```

Description getter.

**Returns**

Description.

Definition at line 38 of file basicGamedata.cpp.

```
38 { return m_description; };
```

References BasicGameData::m_description.

### 5.7.4.2 getId()

```
BasicGameData::id_t BasicGameData::getId ( ) const  [inherited]
```

id getter.

**Returns**

id.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | When tred to get id of instance that has BasicGameData::INVALID_ID. |

Definition at line 40 of file basicGamedata.cpp.

```
40                                          {
41   if (m_id == INVALID_ID)
44     throw exceptionIllegalId();
45   return m_id;
46 };
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

#### 5.7.4.3 getName()

```
std::string BasicGameData::getName ( ) const  [inherited]
```

name getter.

**Returns**

name

Definition at line 36 of file basicGamedata.cpp.

```
36 { return m_name; }
```

References BasicGameData::m_name.

#### 5.7.4.4 isId()

```
bool BasicGameData::isId (
            id_t id ) const  [inherited]
```

Check if id is equal.

**Parameters**

| | |
|---|---|
| *id* | Id to check. |

**Returns**

True if id is same as parameter.

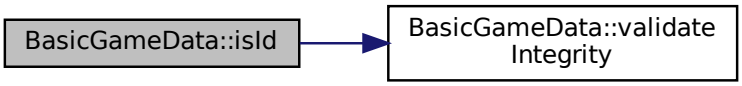
Definition at line 48 of file basicGamedata.cpp.

```
48                                          {
49   validateIntegrity();
50   bool res{m_id == id};
51   return res;
52 }
```

References BasicGameData::m_id, and BasicGameData::validateIntegrity().

Referenced by GameData::getItem().

Here is the call graph for this function:




Here is the caller graph for this function:


#### 5.7.4.5 setDescription()

```
void BasicGameData::setDescription (
            std::string description )  [inherited]
```

Sets description.

**Parameters**

| | |
|---|---|
| *description* | Description to be set. |

Definition at line 32 of file basicGamedata.cpp.

```
32                                                          {
33   m_description = description;
34 }
```

References BasicGameData::m_description.

#### 5.7.4.6 setId()

```
void BasicGameData::setId (
            id_t id )  [protected], [inherited]
```

id setter.

**Parameters**

| | |
|---|---|
| *id* | Id to set. If id to be set is INVALID_ID this will not be set. |

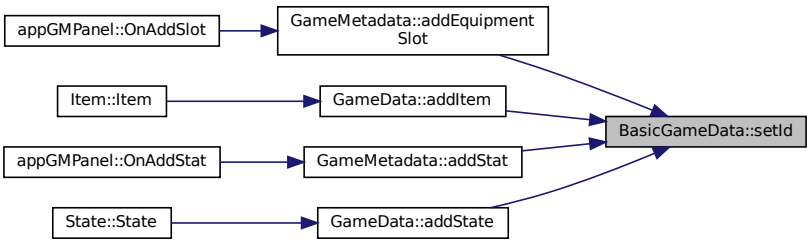
Definition at line 17 of file basicGamedata.cpp.

```
17                                  {
18   if (id == INVALID_ID)
19     return;
20   m_id = id;
21 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by GameMetadata::addEquipmentSlot(), GameData::addItem(), GameMetadata::addStat(), and GameData::addState().

Here is the caller graph for this function:




#### 5.7.4.7 setName()

```
void BasicGameData::setName (
            std::string name )  [inherited]
```

Sets name.

**Parameters**

| | |
|---|---|
| *name* | name to be set. |

Definition at line 30 of file basicGamedata.cpp.

```
30 { m_name = name; }
```

References BasicGameData::m_name.

#### 5.7.4.8 validateIntegrity()

```
void BasicGameData::validateIntegrity ( ) const  [protected], [inherited]
```

Check itegrity of data.

Checks if id is INVALID_ID.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | id is illegal. |

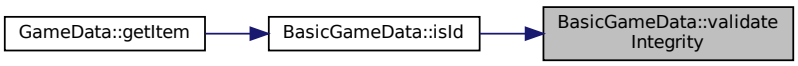
Definition at line 22 of file basicGamedata.cpp.

```
22                                                {
24   if (m_id == INVALID_ID) {
26     throw exceptionIllegalId();
27   }
28 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by BasicGameData::isId().

Here is the caller graph for this function:




### 5.7.5 Member Data Documentation

#### 5.7.5.1 INVALID_ID

```
constexpr id_t BasicGameData::INVALID_ID {std::numeric_limits<id_t>::min()}  [static], [constexpr],
[inherited]
```

Value indicating that id is invalid.

Definition at line 35 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::setId(), and BasicGameData::validateIntegrity().

#### 5.7.5.2 m_description

```
std::string BasicGameData::m_description  [private], [inherited]
```

description.

Definition at line 43 of file basicGamedata.hpp.

Referenced by BasicGameData::getDescription(), and BasicGameData::setDescription().

#### 5.7.5.3 m_id

```
id_t BasicGameData::m_id {INVALID_ID}  [private], [inherited]
```

id.

Definition at line 39 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::isId(), BasicGameData::setId(), and BasicGameData↩ ::validateIntegrity().

#### 5.7.5.4 m_name

```
std::string BasicGameData::m_name  [private], [inherited]
```

name.

Definition at line 41 of file basicGamedata.hpp.

Referenced by BasicGameData::getName(), and BasicGameData::setName().

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

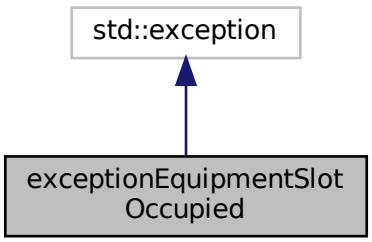
- equipmentSlot.hpp
- equipmentSlot.cpp

## 5.8 exceptionEquipmentSlotOccupied Class Reference

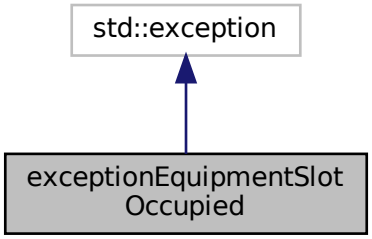
Tried to perform operation on occupied already slot.

`#include <character.hpp>`

Inheritance diagram for exceptionEquipmentSlotOccupied:




Collaboration diagram for exceptionEquipmentSlotOccupied:




### Private Member Functions

- std::string what ()
  *What.*

### 5.8.1 Detailed Description

Tried to perform operation on occupied already slot.

Definition at line 25 of file character.hpp.

### 5.8.2 Member Function Documentation

#### 5.8.2.1 what()

```
std::string exceptionEquipmentSlotOccupied::what ( )  [private]
```

What.

**Returns**

Message.

Definition at line 14 of file character.cpp.

```
14                                                  {
15   return "tried to perform operation on occupied equipment slot";
16 }
```

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

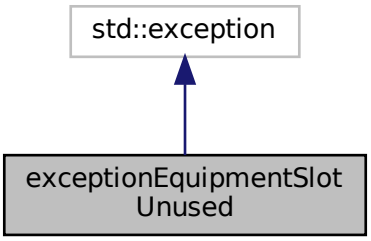
- character.hpp
- character.cpp

## 5.9 exceptionEquipmentSlotUnused Class Reference

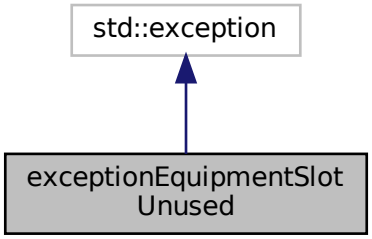
Tried to extract data from unused slot.

```
#include <character.hpp>
```

Inheritance diagram for exceptionEquipmentSlotUnused:



Collaboration diagram for exceptionEquipmentSlotUnused:




## Private Member Functions

- std::string what ()

  *What.*

### 5.9.1 Detailed Description

Tried to extract data from unused slot.

Definition at line 36 of file character.hpp.

### 5.9.2 Member Function Documentation

#### 5.9.2.1 what()

```
std::string exceptionEquipmentSlotUnused::what ( )  [private]
```

What.

**Returns**

Message.

Definition at line 18 of file character.cpp.

```
18                                                {
19   return "Tried to extract data from unused slot";
20 }
```

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

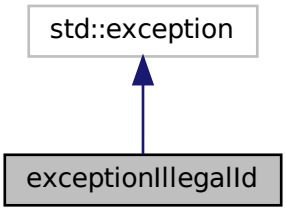
- character.hpp
- character.cpp

## 5.10 exceptionIllegalId Class Reference

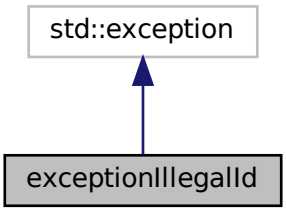
Illegal id exception.

`#include <basicGamedata.hpp>`

Inheritance diagram for exceptionIllegalId:




Collaboration diagram for exceptionIllegalId:




### Private Member Functions

- std::string what ()
  *What.*

### 5.10.1 Detailed Description

Illegal id exception.

**See also**

BasicGameData::INVALID_ID

Definition at line 17 of file basicGamedata.hpp.

### 5.10.2 Member Function Documentation

#### 5.10.2.1 what()

```
std::string exceptionIllegalId::what ( )  [private]
```

What.

**Returns**

Info string.

Definition at line 10 of file basicGamedata.cpp.

```
10                                      {
11   return "Tried to perform operations on instance with invalid id";
12 }
```

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

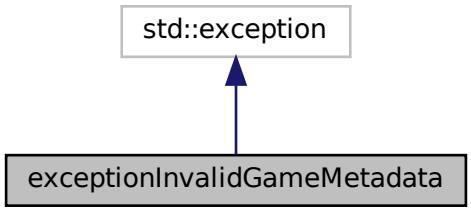
- basicGamedata.hpp
- basicGamedata.cpp

## 5.11 exceptionInvalidGameMetadata Class Reference

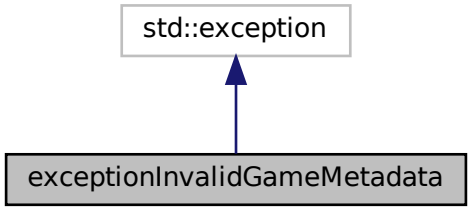
Invalid game data.

```
#include <statModifyingEntity.hpp>
```

Inheritance diagram for exceptionInvalidGameMetadata:



Collaboration diagram for exceptionInvalidGameMetadata:




## Private Member Functions

- std::string what ()

  *What.*

### 5.11.1 Detailed Description

Invalid game data.

Definition at line 18 of file statModifyingEntity.hpp.

### 5.11.2 Member Function Documentation

#### 5.11.2.1 what()

```
std::string exceptionInvalidGameMetadata::what ( )  [private]
```

What.

**Returns**

Messaage.

Definition at line 10 of file statModifyingEntity.cpp.

```
10                                                {
11   return "Invalid Game metadata";
12 }
```

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

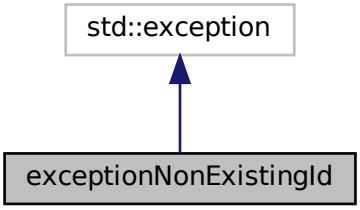
- statModifyingEntity.hpp
- statModifyingEntity.cpp

## 5.12 exceptionNonExistingId Class Reference

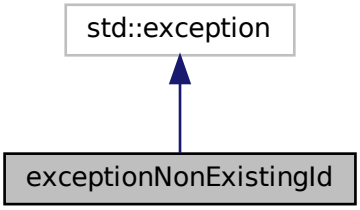
Exception.

`#include <gameMetadata.hpp>`

Inheritance diagram for exceptionNonExistingId:




Collaboration diagram for exceptionNonExistingId:




### Public Member Functions

- virtual std::string what ()
  *What.*

### 5.12.1 Detailed Description

Exception.

Definition at line 19 of file gameMetadata.hpp.

### 5.12.2 Member Function Documentation

#### 5.12.2.1 what()

```
std::string exceptionNonExistingId::what ( )  [virtual]
```

What.

**Returns**

message.

Definition at line 11 of file gameMetadata.cpp.

```
11 { return "Not found with given ID"; }
```

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

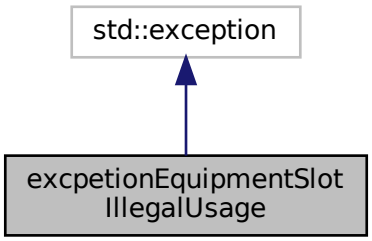
- gameMetadata.hpp
- gameMetadata.cpp

## 5.13 excpetionEquipmentSlotIllegalUsage Class Reference

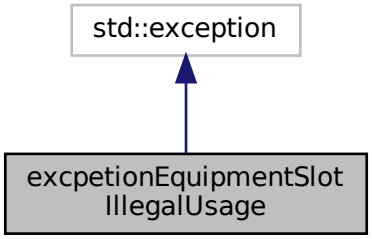
Tried to put stuff where it is not suppsoed to go.

```
#include <character.hpp>
```

Inheritance diagram for excpetionEquipmentSlotIllegalUsage:



Collaboration diagram for excpetionEquipmentSlotIllegalUsage:




## Private Member Functions

- std::string what ()
  *What.*

### 5.13.1 Detailed Description

Tried to put stuff where it is not suppsoed to go.

Definition at line 47 of file character.hpp.

### 5.13.2 Member Function Documentation

#### 5.13.2.1 what()

```
std::string excpetionEquipmentSlotIllegalUsage::what ( )  [private]
```

What.

**Returns**

Message.

Definition at line 22 of file character.cpp.

```
22                                                  {
23   return "Illegal equipment slot usage";
24 }
```

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

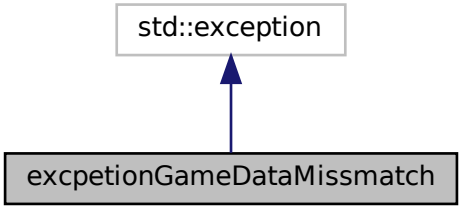
- character.hpp
- character.cpp

## 5.14 excpetionGameDataMissmatch Class Reference

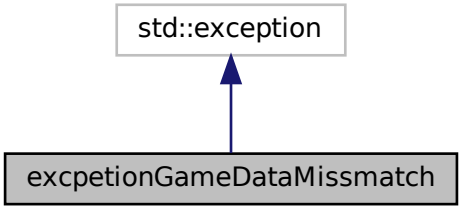
Thrown when attempted to do operation that requires 2 objects to use common GameData but different were used.

`#include <gameData.hpp>`

Inheritance diagram for excpetionGameDataMissmatch:




Collaboration diagram for excpetionGameDataMissmatch:




### Private Member Functions

- std::string what ()

  *what*

### 5.14.1 Detailed Description

Thrown when attempted to do operation that requires 2 objects to use common GameData but different were used.

Definition at line 19 of file gameData.hpp.

### 5.14.2 Member Function Documentation

#### 5.14.2.1 what()

```
std::string excpetionGameDataMissmatch::what ( )  [private]
```

what

**Returns**

message

Definition at line 11 of file gameData.cpp.

```
11 { return "Game data missmatch"; }
```

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

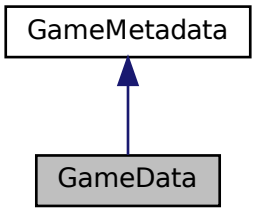
- gameData.hpp
- gameData.cpp

## 5.15 GameData Class Reference

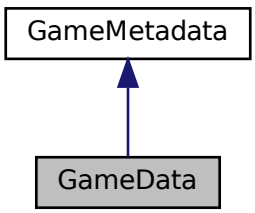
On top of what GameMetadata does. Holds items cataloge.

```
#include <gameData.hpp>
```

Inheritance diagram for GameData:




Collaboration diagram for GameData:



## Public Types

- using itemcollection_t = std::set< Item * >

  *Collection used to hold items.*
- using stateCollcetion_t = std::set< State * >

  *Colection used to hold states.*
- using statsCollection_t = std::vector< Stat * >

  *Type used for stats collection.*
- using equiptmentSlotsCollection_t = std::vector< EquipmentSlot * >

  *Type used for stroing equpment slots collection.*

## Public Member Functions

- ~GameData ()

  *Desctructor.*
- const itemcollection_t & getItems () const

  *Getter.*
- const Item *const getItem (Item::id_t id) const

  *Get Item.*
- void validateDataIntegrity (const StatModifyingEntity &entity) const

  *Checks if Item uses this GameData insnace as it's metadata.*
- void addState (State *state)

  *Adds State to collection and sets it's id.*
- const stateCollcetion_t & getStates () const

  *States getter.*
- const State *const getState (State::id_t id) const

  *State getter.*
- void addStat (Stat *stat)

  *Add given stat.*
- void addEquipmentSlot (EquipmentSlot *eqSlot)

  *Add given EquipmentSlot.*
- Stat * getStat (Stat::id_t id) const

  *Getter for Stat based on id.*
- const statsCollection_t & getStats () const

  *Stats getter.*
- EquipmentSlot * getEquipmentSlot (Stat::id_t id) const

  *Getter for EquipmentSlot based on id.*
- const equiptmentSlotsCollection_t & getEquipmentSlots () const

  *Equipmentslots getter.*

## Private Member Functions

- void addItem (Item *item)

  *Adds item to collection and sets it's id.*

## Private Attributes

- friend Item
- itemcollection_t m_items

  *Collection of items that exist in game collection.*
- Item::id_t m_nextItemId {1}

  *Id that will be given to next item added.*
- stateCollcetion_t m_states

  *Collection of states that exist in game.*
- State::id_t m_nextStateId {1}

  *Id that will be given to next state added.*
- statsCollection_t m_stats

  *Collection of Stat added.*
- equiptmentSlotsCollection_t m_equipmentSlots

  *Collection of EquipmentSlot added.*
- BasicGameData::id_t m_nextStatId {1}

  *Id that will be set to the next Stat added.*
- BasicGameData::id_t m_nextEquipmentSlotId {1}

  *Id that will be set to the next EquipmentSlot added.*

### 5.15.1 Detailed Description

On top of what GameMetadata does. Holds items cataloge.

**See also**

GameMetadata

Definition at line 31 of file gameData.hpp.

### 5.15.2 Member Typedef Documentation

#### 5.15.2.1 equiptmentSlotsCollection_t

```
using GameMetadata::equiptmentSlotsCollection_t = std::vector<EquipmentSlot *>  [inherited]
```

Type used for stroing equpment slots collection.

Definition at line 41 of file gameMetadata.hpp.

#### 5.15.2.2 itemcollection_t

```
using GameData::itemcollection_t = std::set<Item *>
```

Collection used to hold items.

Definition at line 36 of file gameData.hpp.

#### 5.15.2.3 stateCollcetion_t

```
using GameData::stateCollcetion_t = std::set<State *>
```

Colection used to hold states.

Definition at line 39 of file gameData.hpp.

#### 5.15.2.4 statsCollection_t

```
using GameMetadata::statsCollection_t = std::vector<Stat *>  [inherited]
```

Type used for stats collection.

Definition at line 39 of file gameMetadata.hpp.

### 5.15.3 Constructor & Destructor Documentation

#### 5.15.3.1 ~GameData()

```
GameData::~GameData ( )
```

Desctructor.

Definition at line 13 of file gameData.cpp.

```
13                      {
14   for (auto it : m_items) {
15     delete it;
16   }
17
18   for (auto it : m_states) {
19     delete it;
20   }
21 }
```

References m_items, and m_states.

### 5.15.4 Member Function Documentation

#### 5.15.4.1 addEquipmentSlot()

```
void GameMetadata::addEquipmentSlot (
            EquipmentSlot * eqSlot )  [inherited]
```

Add given EquipmentSlot.

**Parameters**

| | |
|---|---|
| *eqSlot* | EquipmentSlot to add..<br>`gameMetadata->addEquipmentSlot(new EquipmentSlot("Leg", "Leg is leg"));`<br>`gameMetadata->addEquipmentSlot(new EquipmentSlot("Head"));`<br>`gameMetadata->addEquipmentSlot(`<br>`new EquipmentSlot("Hand", "For gloves or smth"));` |

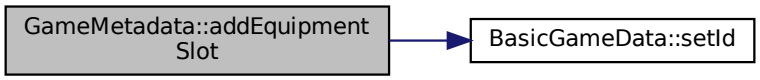
Definition at line 26 of file gameMetadata.cpp.

```
26                                                  {
27   m_equipmentSlots.push_back(eqSlot);
28   eqSlot->setId(m_nextEquipmentSlotId);
29   ++m_nextEquipmentSlotId;
30 }
```

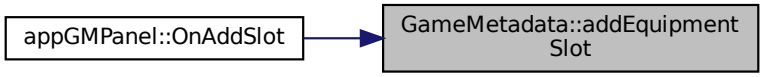
References GameMetadata::m_equipmentSlots, GameMetadata::m_nextEquipmentSlotId, and BasicGameData::setId().

Referenced by appGMPanel::OnAddSlot().

Here is the call graph for this function:




Here is the caller graph for this function:




### 5.15.4.2 addItem()

```
void GameData::addItem (
             Item * item )  [private]
```

Adds item to collection and sets it's id.

**Parameters**

| | |
|---|---|
| *item* | Item to add. |

```
Item *item{new Item(gameData, "Stick")};
item->addModifier(1, 3);
item->setEquipableOn(3);
Item *item2{new Item(gameData, "Sunglasses", "They protect from sun")};
item2->addModifier(2, 100);
item2->setEquipableOn(2);
```

**Note**

It does not validate item.

**Todo** LOW change excpetion to it's own exception class

**Exceptions**

| | |
|---|---|
| *std::__throw_runtime_error* | "Item was not added to GameData" When item was not added to the game some reason. |

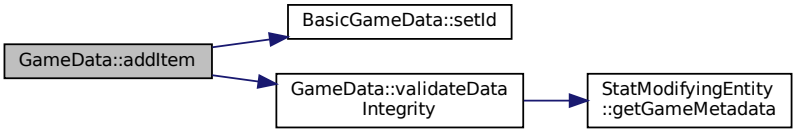
Definition at line 23 of file gameData.cpp.

```
23                                  {
24   validateDataIntegrity(*item);
25   item->setId(m_nextItemId);
26   ++m_nextItemId;
27   auto res{m_items.insert(item)};
28   if (!res.second)
32     std::__throw_runtime_error("Item was not added to GameData");
33 }
```

References m_items, m_nextItemId, BasicGameData::setId(), and validateDataIntegrity().

Referenced by Item::Item().

Here is the call graph for this function:




Here is the caller graph for this function:

#### 5.15.4.3 addStat()

```
void GameMetadata::addStat (
            Stat * stat )  [inherited]
```

Add given stat.

**Parameters**

| | |
|---|---|
| *stat* | Stat to add.<br>`gameMetadata->addStat(new Stat("strength", "just strength"));  gameMetadata->addStat(new Stat("chadness", "Only chads have this"));`<br>`gameMetadata->addStat(new Stat("speed"));` |

Definition at line 20 of file gameMetadata.cpp.

```
20                                      {
21   m_stats.push_back(stat);
22   stat->setId(m_nextStatId);
23   ++m_nextStatId;
24 }
```

References GameMetadata::m_nextStatId, GameMetadata::m_stats, and BasicGameData::setId().

Referenced by appGMPanel::OnAddStat().

Here is the call graph for this function:


Here is the caller graph for this function:


#### 5.15.4.4 addState()

```
void GameData::addState (
            State * state )
```

Adds State to collection and sets it's id.

**Parameters**

| | |
|---|---|
| *state* | State to add.<br>`State *inz{new State(gameData, "inż")}; inz->addModifier(1, 2); inz->addModifier(2, 1);` |

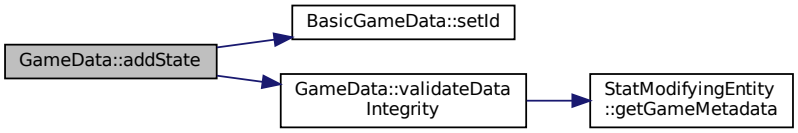
Definition at line 53 of file gameData.cpp.

```
53                                          {
54   validateDataIntegrity(*state);
55   m_states.insert(state);
56   state->setId(m_nextStateId);
57   ++m_nextStateId;
58 }
```

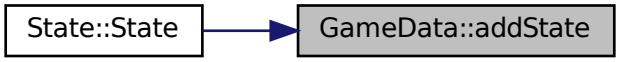
References m_nextStateId, m_states, BasicGameData::setId(), and validateDataIntegrity().

Referenced by State::State().

Here is the call graph for this function:




Here is the caller graph for this function:




#### 5.15.4.5 getEquipmentSlot()

```
EquipmentSlot * GameMetadata::getEquipmentSlot (
            Stat::id_t id ) const  [inherited]
```

Getter for EquipmentSlot based on id.

**Parameters**

| | |
|---|---|
| *id* | Id of EquipmentSlot to fetch |

#### Returns

EquipmentSlot with given id.

#### Exceptions

| | |
|---|---|
| *exceptionNonExistingId* | When tried to fetch insance with id that doesn't exist. |

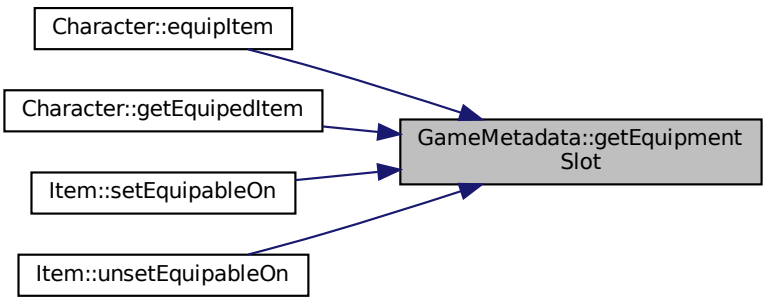
Definition at line 44 of file gameMetadata.cpp.

```
44                                                                  {
45   for (EquipmentSlot *eq : m_equipmentSlots) {
46     if (eq->m_id == id)
47       return eq;
48   }
49   throw exceptionNonExistingId();
50 }
```

References GameMetadata::m_equipmentSlots.

Referenced by Character::equipItem(), Character::getEquipedItem(), Item::setEquipableOn(), and Item::unset↩EquipableOn().

Here is the caller graph for this function:




### 5.15.4.6 getEquipmentSlots()

```
const GameMetadata::equiptmentSlotsCollection_t & GameMetadata::getEquipmentSlots ( ) const
[inherited]
```

Equipmentslots getter.

**Returns**

Equipment slots

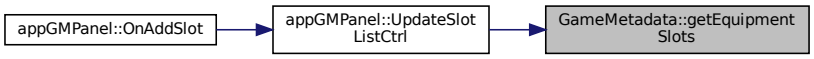
Definition at line 53 of file gameMetadata.cpp.

```
53                                          {
54   return m_equipmentSlots;
55 }
```

References GameMetadata::m_equipmentSlots.

Referenced by appGMPanel::UpdateSlotListCtrl().

Here is the caller graph for this function:




#### 5.15.4.7 getItem()

```
const Item *const GameData::getItem (
            Item::id_t id ) const
```

Get Item.

**Parameters**

| | |
|---|---|
| *id* | Id of Item to fetch. |

**Returns**

Item with given id.

**Exceptions**

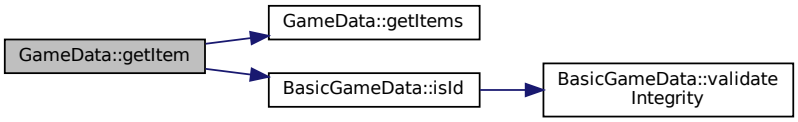
| | |
|---|---|
| *exceptionNonExistingId* | Tried to get item that does not exist. |

Definition at line 37 of file gameData.cpp.

```
37                                                  {
38   auto lookup{std::find_if(getItems().begin(), getItems().end(),
39                            [=](const Item *const it) { return it->isId(id); })};
40
41   if (lookup == getItems().end())
42     throw exceptionNonExistingId();
43   return *lookup;
44 }
```

References getItems(), and BasicGameData::isId().

Here is the call graph for this function:




#### 5.15.4.8 getItems()

```
const GameData::itemcollection_t & GameData::getItems ( ) const
```

Getter.

**Returns**

::m_items.

Definition at line 35 of file gameData.cpp.

```
35 { return m_items; }
```

References m_items.

Referenced by getItem().

Here is the caller graph for this function:


#### 5.15.4.9 getStat()

```
Stat * GameMetadata::getStat (
            Stat::id_t id ) const  [inherited]
```

Getter for Stat based on id.

**Parameters**

| | |
|---|---|
| *id* | Id of Stat to fetch |

**Returns**

Stat with given id.

**Exceptions**

| | |
|---|---|
| *exceptionNonExistingId* | When tried to fetch insance with id that doesn't exist. |

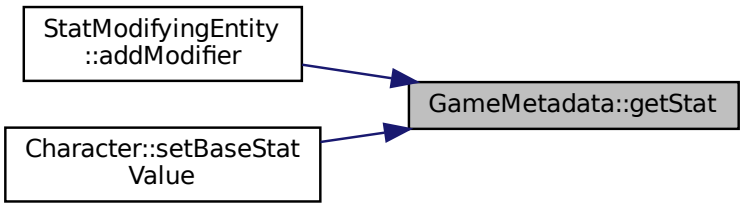
Definition at line 32 of file gameMetadata.cpp.

```
32                                                  {
33   for (Stat *st : m_stats) {
34     if (st->m_id == id)
35       return st;
36   }
37   throw exceptionNonExistingId();
38 }
```

References GameMetadata::m_stats.

Referenced by StatModifyingEntity::addModifier(), and Character::setBaseStatValue().

Here is the caller graph for this function:




### 5.15.4.10 getState()

```
const State *const GameData::getState (
            State::id_t id ) const
```

State getter.

**Parameters**

| | |
|---|---|
| *id* | Id of State to get. |

#### Returns

state with id asked.

#### Exceptions

| | |
|---|---|
| *exceptionNonExistingId* | Tried to get State that does not exist. |

Definition at line 64 of file gameData.cpp.

```
64                                                          {
65   auto lookup{
66       std::find_if(m_states.begin(), m_states.end(),
67                    [=](const State *const state) { return state->isId(id); })};
68
69   if (lookup == m_states.end()) {
70     throw exceptionNonExistingId();
71   }
72
73   return *lookup;
74 }
```

References m_states.

### 5.15.4.11 getStates()

```
const GameData::stateCollcetion_t & GameData::getStates ( ) const
```

States getter.

#### Returns

m_states

Definition at line 60 of file gameData.cpp.

```
60                                                      {
61   return m_states;
62 }
```

References m_states.

Referenced by appGMPanel::UpdateStateListCtrl().

Here is the caller graph for this function:

#### 5.15.4.12 getStats()

```
const GameMetadata::statsCollection_t & GameMetadata::getStats ( ) const  [inherited]
```

Stats getter.

**Returns**

m_stats.

Definition at line 40 of file gameMetadata.cpp.

```
40                                                                  {
41   return m_stats;
42 }
```

References GameMetadata::m_stats.

Referenced by appGMPanel::UpdateStatsListCtrl().

Here is the caller graph for this function:


#### 5.15.4.13 validateDataIntegrity()

```
void GameData::validateDataIntegrity (
            const StatModifyingEntity & entity ) const
```

Checks if Item uses this GameData insnace as it's metadata.

**Parameters**

| | |
|---|---|
| *entity* | Entity to check. |

**Exceptions**

| | |
|---|---|
| *excpetionGameDataMissmatch* | When entity does not use this as it's game data. |

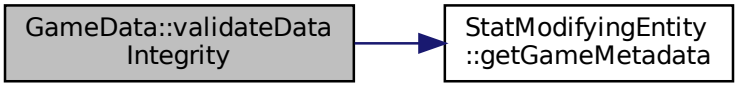
Definition at line 46 of file gameData.cpp.

```
46                                                          {
47   if (this != entity.getGameMetadata())
50     throw excpetionGameDataMissmatch();
51 }
```

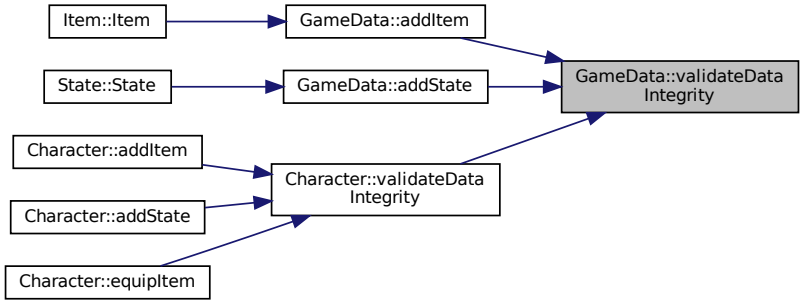
References StatModifyingEntity::getGameMetadata().

Referenced by addItem(), addState(), and Character::validateDataIntegrity().

Here is the call graph for this function:




Here is the caller graph for this function:




## 5.15.5 Member Data Documentation

### 5.15.5.1 Item

```
friend GameData::Item  [private]
```

Definition at line 32 of file gameData.hpp.

### 5.15.5.2 m_equipmentSlots

```
equiptmentSlotsCollection_t GameMetadata::m_equipmentSlots  [private], [inherited]
```

Collection of EquipmentSlot added.

Definition at line 47 of file gameMetadata.hpp.

Referenced by GameMetadata::addEquipmentSlot(), GameMetadata::getEquipmentSlot(), GameMetadata::get↩EquipmentSlots(), and GameMetadata::∼GameMetadata().

#### 5.15.5.3 m_items

```
itemcollection_t GameData::m_items  [private]
```

Collection of items that exist in game collection.

Definition at line 43 of file gameData.hpp.

Referenced by addItem(), getItems(), and ~GameData().

#### 5.15.5.4 m_nextEquipmentSlotId

```
BasicGameData::id_t GameMetadata::m_nextEquipmentSlotId {1}  [private], [inherited]
```

Id that will be set to the next EquipmentSlot added.

Definition at line 52 of file gameMetadata.hpp.

Referenced by GameMetadata::addEquipmentSlot().

#### 5.15.5.5 m_nextItemId

```
Item::id_t GameData::m_nextItemId {1}  [private]
```

Id that will be given to next item added.

Definition at line 45 of file gameData.hpp.

Referenced by addItem().

#### 5.15.5.6 m_nextStateId

```
State::id_t GameData::m_nextStateId {1}  [private]
```

Id that will be given to next state added.

Definition at line 50 of file gameData.hpp.

Referenced by addState().

#### 5.15.5.7 m_nextStatId

```
BasicGameData::id_t GameMetadata::m_nextStatId {1}  [private], [inherited]
```

Id that will be set to the next Stat added.

Definition at line 50 of file gameMetadata.hpp.

Referenced by GameMetadata::addStat().

#### 5.15.5.8 m_states

```
stateCollcetion_t GameData::m_states  [private]
```

Collection of states that exist in game.

Definition at line 48 of file gameData.hpp.

Referenced by addState(), getState(), getStates(), and ~GameData().

#### 5.15.5.9 m_stats

```
statsCollection_t GameMetadata::m_stats  [private], [inherited]
```

Collection of Stat added.

Definition at line 45 of file gameMetadata.hpp.

Referenced by GameMetadata::addStat(), GameMetadata::getStat(), GameMetadata::getStats(), and GameMetadata::~GameMetadata().

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

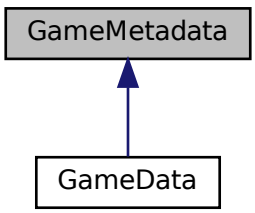
- gameData.hpp
- gameData.cpp

## 5.16 GameMetadata Class Reference

Holds game metadata. That is what statistics exist and what equipable slots exist.

```
#include <gameMetadata.hpp>
```

Inheritance diagram for GameMetadata:



## Public Types

- using statsCollection_t = std::vector< Stat * >

  *Type used for stats collection.*
- using equiptmentSlotsCollection_t = std::vector< EquipmentSlot * >

  *Type used for stroing equpment slots collection.*

## Public Member Functions

- ~GameMetadata ()

  *Destructor.*
- void addStat (Stat *stat)

  *Add given stat.*
- void addEquipmentSlot (EquipmentSlot *eqSlot)

  *Add given EquipmentSlot.*
- Stat * getStat (Stat::id_t id) const

  *Getter for Stat based on id.*
- const statsCollection_t & getStats () const

  *Stats getter.*
- EquipmentSlot * getEquipmentSlot (Stat::id_t id) const

  *Getter for EquipmentSlot based on id.*
- const equiptmentSlotsCollection_t & getEquipmentSlots () const

  *Equipmentslots getter.*

## Private Attributes

- statsCollection_t m_stats

  *Collection of Stat added.*
- equiptmentSlotsCollection_t m_equipmentSlots

  *Collection of EquipmentSlot added.*
- BasicGameData::id_t m_nextStatId {1}

  *Id that will be set to the next Stat added.*
- BasicGameData::id_t m_nextEquipmentSlotId {1}

  *Id that will be set to the next EquipmentSlot added.*

### 5.16.1 Detailed Description

Holds game metadata. That is what statistics exist and what equipable slots exist.

It will deallocate Stat and EquipmentSlot added to it so those should not be deallcated manually.

Definition at line 36 of file gameMetadata.hpp.

### 5.16.2 Member Typedef Documentation

#### 5.16.2.1 equiptmentSlotsCollection_t

```
using GameMetadata::equiptmentSlotsCollection_t = std::vector<EquipmentSlot *>
```

Type used for stroing equpment slots collection.

Definition at line 41 of file gameMetadata.hpp.

#### 5.16.2.2 statsCollection_t

```
using GameMetadata::statsCollection_t = std::vector<Stat *>
```

Type used for stats collection.

Definition at line 39 of file gameMetadata.hpp.

### 5.16.3 Constructor & Destructor Documentation

#### 5.16.3.1 ~GameMetadata()

```
GameMetadata::~GameMetadata ( )
```

Destructor.

Deallocates all added equipmentslots and stats.

Definition at line 13 of file gameMetadata.cpp.

```
13                                  {
14   for (Stat *&it : m_stats)
15     delete it;
16   for (EquipmentSlot *&it : m_equipmentSlots)
17     delete it;
18 }
```

References m_equipmentSlots, and m_stats.

### 5.16.4 Member Function Documentation

#### 5.16.4.1 addEquipmentSlot()

```
void GameMetadata::addEquipmentSlot (
            EquipmentSlot * eqSlot )
```

Add given EquipmentSlot.

**Parameters**

| | |
|---|---|
| *eqSlot* | EquipmentSlot to add..<br>`gameMetadata->addEquipmentSlot(new EquipmentSlot("Leg", "Leg is leg"));`<br>`gameMetadata->addEquipmentSlot(new EquipmentSlot("Head"));`<br>`gameMetadata->addEquipmentSlot(`<br>`new EquipmentSlot("Hand", "For gloves or smth"));` |

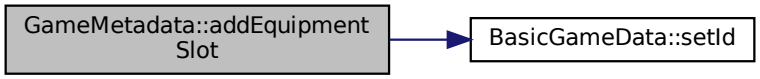
Definition at line 26 of file gameMetadata.cpp.

```
26                                                          {
27   m_equipmentSlots.push_back(eqSlot);
28   eqSlot->setId(m_nextEquipmentSlotId);
29   ++m_nextEquipmentSlotId;
30 }
```

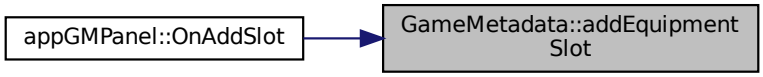
References m_equipmentSlots, m_nextEquipmentSlotId, and BasicGameData::setId().

Referenced by appGMPanel::OnAddSlot().

Here is the call graph for this function:




Here is the caller graph for this function:




#### 5.16.4.2 addStat()

```
void GameMetadata::addStat (
            Stat * stat )
```

Add given stat.

**Parameters**

| | |
|---|---|
| *stat* | Stat to add.<br>`gameMetadata->addStat(new Stat("strength", "just strength")); gameMetadata->addStat(new Stat("chadness", "Only chads have this")); gameMetadata->addStat(new Stat("speed"));` |

Definition at line 20 of file gameMetadata.cpp.

```
20                                          {
21   m_stats.push_back(stat);
22   stat->setId(m_nextStatId);
23   ++m_nextStatId;
24 }
```

References m_nextStatId, m_stats, and BasicGameData::setId().

Referenced by appGMPanel::OnAddStat().

Here is the call graph for this function:


Here is the caller graph for this function:


### 5.16.4.3 getEquipmentSlot()

```
EquipmentSlot * GameMetadata::getEquipmentSlot (
            Stat::id_t id ) const
```

Getter for EquipmentSlot based on id.

**Parameters**

| | |
|---|---|
| *id* | Id of EquipmentSlot to fetch |

**Returns**

EquipmentSlot with given id.

**Exceptions**

| | |
|---|---|
| *exceptionNonExistingId* | When tried to fetch insance with id that doesn't exist. |

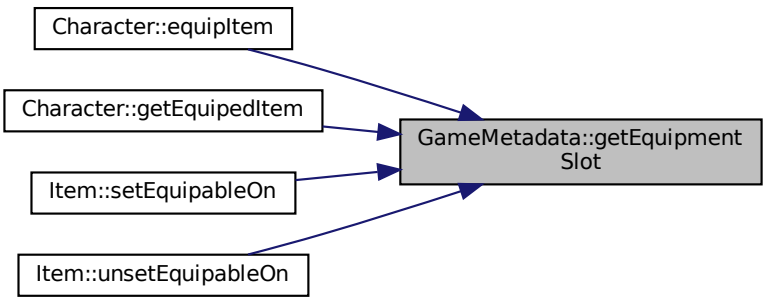
Definition at line 44 of file gameMetadata.cpp.

```
44                                                                  {
45   for (EquipmentSlot *eq : m_equipmentSlots) {
46     if (eq->m_id == id)
47       return eq;
48   }
49   throw exceptionNonExistingId();
50 }
```

References m_equipmentSlots.

Referenced by Character::equipItem(), Character::getEquipedItem(), Item::setEquipableOn(), and Item::unset↩EquipableOn().

Here is the caller graph for this function:




#### 5.16.4.4 getEquipmentSlots()

```
const GameMetadata::equiptmentSlotsCollection_t & GameMetadata::getEquipmentSlots ( ) const
```

Equipmentslots getter.

**Returns**

Equipment slots

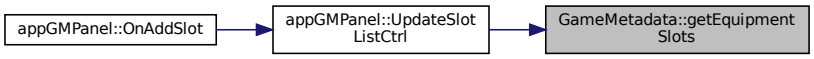
Definition at line 53 of file gameMetadata.cpp.

```
53                                                {
54   return m_equipmentSlots;
55 }
```

References m_equipmentSlots.

Referenced by appGMPanel::UpdateSlotListCtrl().

Here is the caller graph for this function:




#### 5.16.4.5 getStat()

```
Stat * GameMetadata::getStat (
            Stat::id_t id ) const
```

Getter for Stat based on id.

**Parameters**

| | |
|---|---|
| *id* | Id of Stat to fetch |

**Returns**

Stat with given id.

**Exceptions**

| | |
|---|---|
| *exceptionNonExistingId* | When tried to fetch insance with id that doesn't exist. |

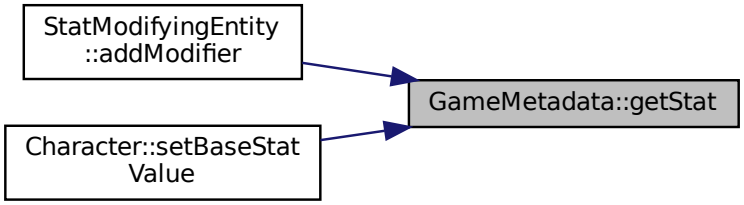
Definition at line 32 of file gameMetadata.cpp.

```
32                                                {
33   for (Stat *st : m_stats) {
34     if (st->m_id == id)
35       return st;
36   }
37   throw exceptionNonExistingId();
38 }
```

References m_stats.

Referenced by StatModifyingEntity::addModifier(), and Character::setBaseStatValue().

Here is the caller graph for this function:




#### 5.16.4.6 getStats()

```
const GameMetadata::statsCollection_t & GameMetadata::getStats ( ) const
```

Stats getter.

**Returns**

m_stats.

Definition at line 40 of file gameMetadata.cpp.

```
40                                                                  {
41   return m_stats;
42 }
```

References m_stats.

Referenced by appGMPanel::UpdateStatsListCtrl().

Here is the caller graph for this function:


### 5.16.5 Member Data Documentation

#### 5.16.5.1 m_equipmentSlots

```
equiptmentSlotsCollection_t GameMetadata::m_equipmentSlots  [private]
```

Collection of EquipmentSlot added.

Definition at line 47 of file gameMetadata.hpp.

Referenced by addEquipmentSlot(), getEquipmentSlot(), getEquipmentSlots(), and ~GameMetadata().

#### 5.16.5.2 m_nextEquipmentSlotId

```
BasicGameData::id_t GameMetadata::m_nextEquipmentSlotId {1}  [private]
```

Id that will be set to the next EquipmentSlot added.

Definition at line 52 of file gameMetadata.hpp.

Referenced by addEquipmentSlot().

#### 5.16.5.3 m_nextStatId

```
BasicGameData::id_t GameMetadata::m_nextStatId {1}  [private]
```

Id that will be set to the next Stat added.

Definition at line 50 of file gameMetadata.hpp.

Referenced by addStat().

#### 5.16.5.4 m_stats

```
statsCollection_t GameMetadata::m_stats  [private]
```

Collection of Stat added.

Definition at line 45 of file gameMetadata.hpp.

Referenced by addStat(), getStat(), getStats(), and ~GameMetadata().

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

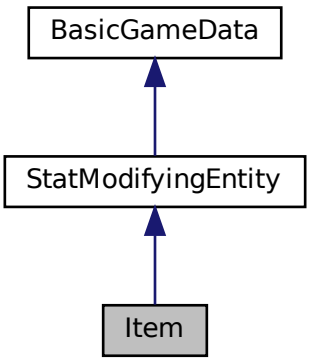
- gameMetadata.hpp
- gameMetadata.cpp

## 5.17 Item Class Reference

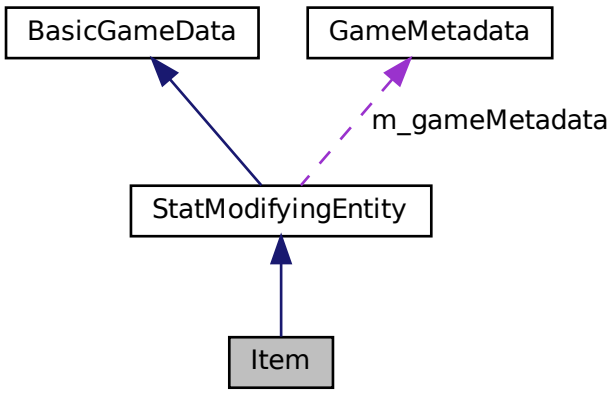
Represents an item in game.

`#include <item.hpp>`

Inheritance diagram for Item:




Collaboration diagram for Item:




### Public Types

- using equipableSlots_t = std::set< EquipmentSlot::id_t >

  *Collection.*

- using modifier_t = std::pair< Stat::id_t, Stat::value_t >

  *Repesents modification of stats <stat modyfied, value of modification>*
- using modifersCollection_t = std::vector< modifier_t >

  *Collection type used to store modifers.*
- using id_t = long long

  *Type used for ids.*

## Public Member Functions

- Item (GameData *gameData, std::string name, std::string description="")

  *Constructor.*
- void setEquipableOn (EquipmentSlot::id_t equipmentSlotId)

  *Make item equipable on given equipment slot.*
- void unsetEquipableOn (EquipmentSlot::id_t equipmentSlotId)

  *Makes item no longer possible to equip onto given slot.*
- bool isEquipableOn (EquipmentSlot::id_t equipmentSlotId) const

  *Checks whenever Item is equipable in given slot id.*
- const equipableSlots_t & getEquipableSlots () const

  *Get equipable slots.*
- void addModifier (Stat::id_t, Stat::value_t by)

  *Add modification of stats.*
- Stat::value_t getModifierValue (Stat::id_t id) const

  *Modifier value getter.*
- const GameMetadata *const getGameMetadata () const

  *GameMeatadata getter.*
- const modifersCollection_t & getModifiers () const

  *Modifiers getter.*
- void setName (std::string name)

  *Sets name.*
- void setDescription (std::string description)

  *Sets description.*
- std::string getName () const

  *name getter.*
- std::string getDescription () const

  *Description getter.*
- id_t getId () const

  *id getter.*
- bool isId (id_t id) const

  *Check if id is equal.*

## Static Public Attributes

- static constexpr id_t INVALID_ID {std::numeric_limits<id_t>::min()}

  *Value indicating that id is invalid.*

## Protected Member Functions

- void setId (id_t id)

  *id setter.*
- void validateIntegrity () const

  *Check itegrity of data.*

### Private Attributes

- equipableSlots_t m_equipableOn
  *Which equipment slots given item can be put on.*
- modifersCollection_t m_modifiers
  *Holds modifiers.*
- const GameMetadata *const m_gameMetadata
  *Game Metadata.*
- id_t m_id {INVALID_ID}
  *id.*
- std::string m_name
  *name.*
- std::string m_description
  *description.*

## 5.17.1 Detailed Description

Represents an item in game.

Should be used in GameData.

Definition at line 20 of file item.hpp.

## 5.17.2 Member Typedef Documentation

### 5.17.2.1 equipableSlots_t

```
using Item::equipableSlots_t = std::set<EquipmentSlot::id_t>
```

Collection.

Definition at line 24 of file item.hpp.

### 5.17.2.2 id_t

```
using BasicGameData::id_t = long long  [inherited]
```

Type used for ids.

Definition at line 33 of file basicGamedata.hpp.

#### 5.17.2.3 modifersCollection_t

```
using StatModifyingEntity::modifersCollection_t = std::vector<modifier_t>  [inherited]
```

Collection type used to store modifers.

Definition at line 38 of file statModifyingEntity.hpp.

#### 5.17.2.4 modifier_t

```
using StatModifyingEntity::modifier_t = std::pair<Stat::id_t, Stat::value_t>  [inherited]
```

Repesents modification of stats <stat modyfied, value of modification>

Definition at line 36 of file statModifyingEntity.hpp.

### 5.17.3 Constructor & Destructor Documentation

#### 5.17.3.1 Item()

```
Item::Item (
            GameData * gameData,
            std::string name,
            std::string description = "" )
```

Constructor.

**Parameters**

| | |
|---|---|
| *gameData* | GameData to which item is added. |
| *name* | Name of item. |
| *description* | description. |

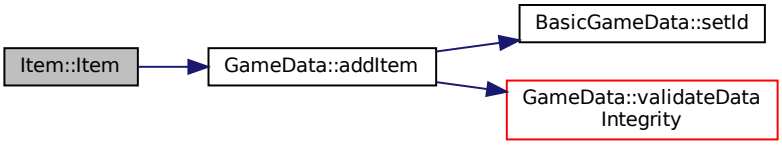
```
Item *item{new Item(gameData, "Stick")};
item->addModifier(1, 3);
item->setEquipableOn(3);
Item *item2{new Item(gameData, "Sunglasses", "They protect from sun")};
item2->addModifier(2, 100);
item2->setEquipableOn(2);
```

Definition at line 9 of file item.cpp.

```
10     : StatModifyingEntity(gameData, name, description) {
11   gameData->addItem(this);
12 }
```

References GameData::addItem().

Here is the call graph for this function:




### 5.17.4 Member Function Documentation

#### 5.17.4.1 addModifier()

```
void StatModifyingEntity::addModifier (
            Stat::id_t statModifed,
            Stat::value_t by )  [inherited]
```

Add modification of stats.

**Parameters**

| | |
|---|---|
| *statModifed* | modified. |
| *by* | Modify value. |

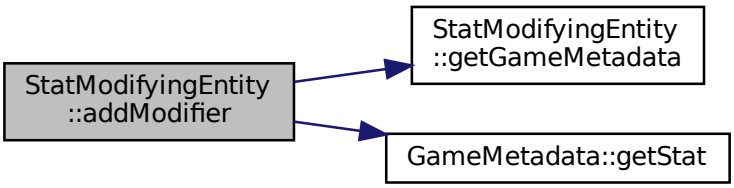
**Todo** Check if entity has modifier of that Stat already.

Definition at line 24 of file statModifyingEntity.cpp.

```
25                                                  {
26
27   getGameMetadata()->getStat(statModifed);
28   m_modifiers.push_back({statModifed, by});
29 }
```

References StatModifyingEntity::getGameMetadata(), GameMetadata::getStat(), and StatModifyingEntity::m_↩ modifiers.

Here is the call graph for this function:




#### 5.17.4.2 getDescription()

```
std::string BasicGameData::getDescription ( ) const  [inherited]
```

Description getter.

**Returns**

Description.

Definition at line 38 of file basicGamedata.cpp.

```
38 { return m_description; };
```

References BasicGameData::m_description.

#### 5.17.4.3 getEquipableSlots()

```
const Item::equipableSlots_t & Item::getEquipableSlots ( ) const
```

Get equipable slots.

**Returns**

m_equipableOn.

Definition at line 30 of file item.cpp.

```
30                                                  {
31   return m_equipableOn;
32 }
```

References m_equipableOn.

#### 5.17.4.4 getGameMetadata()

```
const GameMetadata *const StatModifyingEntity::getGameMetadata ( ) const  [inherited]
```

GameMeatadata getter.

**Returns**

GemeMetadata used by instance.

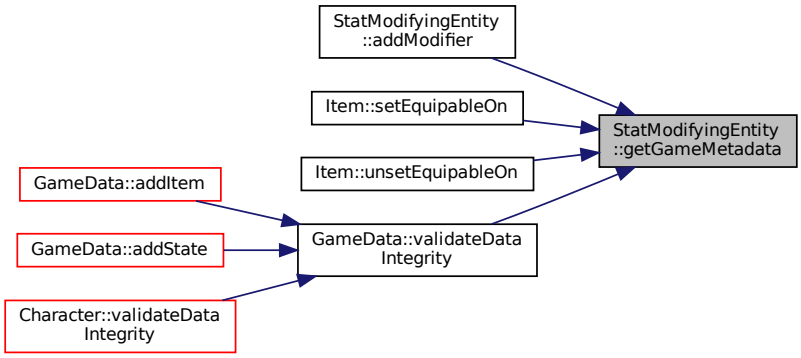
Definition at line 42 of file statModifyingEntity.cpp.

```
42                                                          {
43   return m_gameMetadata;
44 }
```

References StatModifyingEntity::m_gameMetadata.

Referenced by StatModifyingEntity::addModifier(), setEquipableOn(), unsetEquipableOn(), and GameData↩ ::validateDataIntegrity().

Here is the caller graph for this function:




#### 5.17.4.5 getId()

```
BasicGameData::id_t BasicGameData::getId ( ) const  [inherited]
```

id getter.

**Returns**

id.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | When tred to get id of instance that has BasicGameData::INVALID_ID. |

Definition at line 40 of file basicGamedata.cpp.

```
40                                          {
41   if (m_id == INVALID_ID)
44     throw exceptionIllegalId();
45   return m_id;
46 };
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

#### 5.17.4.6 getModifiers()

```
const StatModifyingEntity::modifersCollection_t & StatModifyingEntity::getModifiers ( ) const
[inherited]
```

Modifiers getter.

**Returns**

::m_modifiers

Definition at line 47 of file statModifyingEntity.cpp.

```
47                                              {
48   return m_modifiers;
49 }
```

References StatModifyingEntity::m_modifiers.

#### 5.17.4.7 getModifierValue()

```
Stat::value_t StatModifyingEntity::getModifierValue (
            Stat::id_t id ) const  [inherited]
```

Modifier value getter.

**Parameters**

| | |
|---|---|
| *id* | Id of stat to get value of modifier of. |

**Returns**

Modifier value or 0 if Instance does not modify stat asked.

Definition at line 31 of file statModifyingEntity.cpp.

```
31                                                                  {
32   auto lookup{
```

```
33       std::find_if(m_modifiers.begin(), m_modifiers.end(),
34                    [=](const modifier_t mod) { return mod.first == id; })};
35   // If not found modifier of asked stat.
36   if (lookup == m_modifiers.end())
37     return 0;
38
39   return lookup->second;
40 }
```

References StatModifyingEntity::m_modifiers.

Referenced by Character::getStatValueEquipmentContrubitors().

Here is the caller graph for this function:


#### 5.17.4.8 getName()

```
std::string BasicGameData::getName ( ) const  [inherited]
```

name getter.

**Returns**

name

Definition at line 36 of file basicGamedata.cpp.

```
36 { return m_name; }
```

References BasicGameData::m_name.

#### 5.17.4.9 isEquipableOn()

```
bool Item::isEquipableOn (
            EquipmentSlot::id_t equipmentSlotId ) const
```

Checks whenever Item is equipable in given slot id.

**Parameters**

| | |
|---|---|
| *equipment↩ SlotId* | Slot to check. |

**Returns**

True if Item is equipable in enquiered slot. False if it is not.

Definition at line 24 of file item.cpp.

```
24                                                                  {
25   auto it{m_equipableOn.find(equipmentSlotId)};
26   const bool result{it != m_equipableOn.end()};
27   return result;
28 }
```

References m_equipableOn.

Referenced by Character::equipItem().

Here is the caller graph for this function:


#### 5.17.4.10 isId()

```
bool BasicGameData::isId (
            id_t id ) const  [inherited]
```

Check if id is equal.

**Parameters**

| | |
|---|---|
| *id* | Id to check. |

**Returns**

True if id is same as parameter.

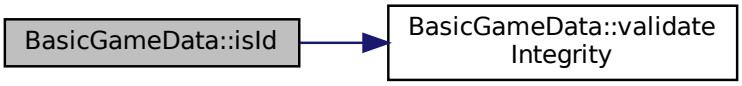
Definition at line 48 of file basicGamedata.cpp.

```
48                                          {
49   validateIntegrity();
50   bool res{m_id == id};
51   return res;
52 }
```

References BasicGameData::m_id, and BasicGameData::validateIntegrity().

Referenced by GameData::getItem().

Here is the call graph for this function:




Here is the caller graph for this function:


#### 5.17.4.11 setDescription()

```
void BasicGameData::setDescription (
            std::string description )  [inherited]
```

Sets description.

**Parameters**

| | |
|---|---|
| *description* | Description to be set. |

Definition at line 32 of file basicGamedata.cpp.

```
32                                                          {
33   m_description = description;
34 }
```

References BasicGameData::m_description.

#### 5.17.4.12 setEquipableOn()

```
void Item::setEquipableOn (
            EquipmentSlot::id_t equipmentSlotId )
```

Make item equipable on given equipment slot.

**Parameters**

| | |
|---|---|
| *equipment↩ SlotId* | On which EquipmentSlot should it be equipable. |

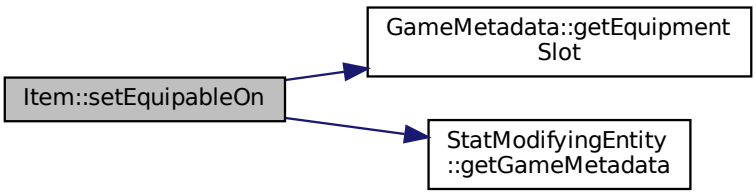
If it is already equipable on that slot nothing will be done.

Definition at line 14 of file item.cpp.

```
14                                                                  {
15   getGameMetadata()->getEquipmentSlot(equipmentSlotId);
16   m_equipableOn.insert(equipmentSlotId);
17 }
```

References GameMetadata::getEquipmentSlot(), StatModifyingEntity::getGameMetadata(), and m_equipableOn.

Here is the call graph for this function:




#### 5.17.4.13 setId()

```
void BasicGameData::setId (
              id_t id )  [protected], [inherited]
```

id setter.

**Parameters**

| | |
|---|---|
| *id* | Id to set. If id to be set is INVALID_ID this will not be set. |

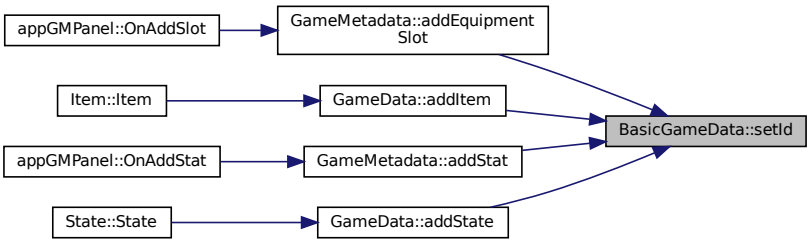
Definition at line 17 of file basicGamedata.cpp.

```
17                                      {
18   if (id == INVALID_ID)
19     return;
20   m_id = id;
21 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by GameMetadata::addEquipmentSlot(), GameData::addItem(), GameMetadata::addStat(), and GameData::addState().

Here is the caller graph for this function:




#### 5.17.4.14 setName()

```
void BasicGameData::setName (
            std::string name )  [inherited]
```

Sets name.

**Parameters**

| | |
|---|---|
| *name* | name to be set. |

Definition at line 30 of file basicGamedata.cpp.

```
30 { m_name = name; }
```

References BasicGameData::m_name.

#### 5.17.4.15 unsetEquipableOn()

```
void Item::unsetEquipableOn (
            EquipmentSlot::id_t equipmentSlotId )
```

Makes item no longer possible to equip onto given slot.

**Parameters**

| | |
|---|---|
| *equipment↩ SlotId* | equipment slot id that item will be no longer equipable on. |

If it is already not equipable on that slot nothing will happen.

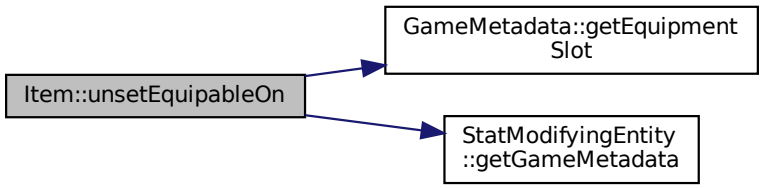
Definition at line 19 of file item.cpp.

```
19                                                  {
```

```
20   getGameMetadata()->getEquipmentSlot(equipmentSlotId);
21   m_equipableOn.erase(equipmentSlotId);
22 }
```

References GameMetadata::getEquipmentSlot(), StatModifyingEntity::getGameMetadata(), and m_equipableOn.

Here is the call graph for this function:




#### 5.17.4.16 validateIntegrity()

```
void BasicGameData::validateIntegrity ( ) const  [protected], [inherited]
```

Check itegrity of data.

Checks if id is INVALID_ID.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | id is illegal. |

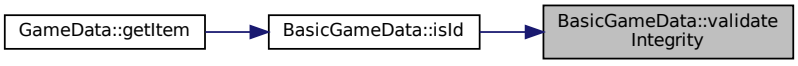
Definition at line 22 of file basicGamedata.cpp.

```
22                                          {
24   if (m_id == INVALID_ID) {
26     throw exceptionIllegalId();
27   }
28 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by BasicGameData::isId().

Here is the caller graph for this function:



### 5.17.5 Member Data Documentation

#### 5.17.5.1 INVALID_ID

```
constexpr id_t BasicGameData::INVALID_ID {std::numeric_limits<id_t>::min()}  [static], [constexpr], [inherited]
```

Value indicating that id is invalid.

Definition at line 35 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::setId(), and BasicGameData::validateIntegrity().

#### 5.17.5.2 m_description

```
std::string BasicGameData::m_description  [private], [inherited]
```

description.

Definition at line 43 of file basicGamedata.hpp.

Referenced by BasicGameData::getDescription(), and BasicGameData::setDescription().

#### 5.17.5.3 m_equipableOn

```
equipableSlots_t Item::m_equipableOn  [private]
```

Which equipment slots given item can be put on.

Definition at line 28 of file item.hpp.

Referenced by getEquipableSlots(), isEquipableOn(), setEquipableOn(), and unsetEquipableOn().

#### 5.17.5.4 m_gameMetadata

```
const GameMetadata* const StatModifyingEntity::m_gameMetadata  [private], [inherited]
```

Game Metadata.

Definition at line 45 of file statModifyingEntity.hpp.

Referenced by StatModifyingEntity::getGameMetadata().

#### 5.17.5.5 m_id

```
id_t BasicGameData::m_id {INVALID_ID}  [private], [inherited]
```

id.

Definition at line 39 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::isId(), BasicGameData::setId(), and BasicGameData↩ ::validateIntegrity().

#### 5.17.5.6 m_modifiers

```
modifersCollection_t StatModifyingEntity::m_modifiers  [private], [inherited]
```

Holds modifiers.

Definition at line 42 of file statModifyingEntity.hpp.

Referenced by StatModifyingEntity::addModifier(), StatModifyingEntity::getModifiers(), and StatModifyingEntity↩ ::getModifierValue().

#### 5.17.5.7 m_name

```
std::string BasicGameData::m_name  [private], [inherited]
```

name.

Definition at line 41 of file basicGamedata.hpp.

Referenced by BasicGameData::getName(), and BasicGameData::setName().

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

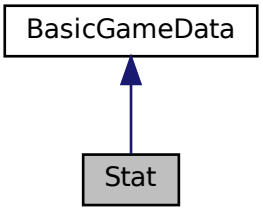
- item.hpp
- item.cpp

## 5.18 Stat Class Reference

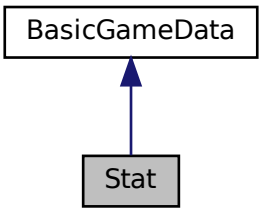
Statistics.

`#include <stat.hpp>`

Inheritance diagram for Stat:




Collaboration diagram for Stat:




### Public Types

- using value_t = long long
  *Type of stat value.*
- using id_t = long long
  *Type used for ids.*

### Public Member Functions

- Stat (std::string name, std::string description="")
  *Constructor.*
- void setName (std::string name)
  *Sets name.*

- void setDescription (std::string description)

  *Sets description.*
- std::string getName () const

  *name getter.*
- std::string getDescription () const

  *Description getter.*
- id_t getId () const

  *id getter.*
- bool isId (id_t id) const

  *Check if id is equal.*

## Static Public Attributes

- static constexpr id_t INVALID_ID {std::numeric_limits<id_t>::min()}

  *Value indicating that id is invalid.*

## Protected Member Functions

- void setId (id_t id)

  *id setter.*
- void validateIntegrity () const

  *Check itegrity of data.*

## Private Attributes

- id_t m_id {INVALID_ID}

  *id.*
- std::string m_name

  *name.*
- std::string m_description

  *description.*

### 5.18.1 Detailed Description

Statistics.

Definition at line 14 of file stat.hpp.

### 5.18.2 Member Typedef Documentation

#### 5.18.2.1 id_t

```
using BasicGameData::id_t = long long  [inherited]
```

Type used for ids.

Definition at line 33 of file basicGamedata.hpp.

#### 5.18.2.2 value_t

```
using Stat::value_t = long long
```

Type of stat value.

Definition at line 17 of file stat.hpp.

### 5.18.3 Constructor & Destructor Documentation

#### 5.18.3.1 Stat()

```
Stat::Stat (
            std::string name,
            std::string description = "" )
```

Constructor.

**Parameters**

| | |
|---|---|
| *name* | Name of statistics. |
| *description* | Optional description of statistics; |

Definition at line 9 of file stat.cpp.

```
10     : BasicGameData(name, description) {}
```

### 5.18.4 Member Function Documentation

#### 5.18.4.1 getDescription()

```
std::string BasicGameData::getDescription ( ) const  [inherited]
```

Description getter.

**Returns**

Description.

Definition at line 38 of file basicGamedata.cpp.

```
38 { return m_description; };
```

References BasicGameData::m_description.

#### 5.18.4.2 getId()

```
BasicGameData::id_t BasicGameData::getId ( ) const  [inherited]
```

id getter.

**Returns**

id.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | When tred to get id of instance that has BasicGameData::INVALID_ID. |

Definition at line 40 of file basicGamedata.cpp.

```
40                                          {
41   if (m_id == INVALID_ID)
44     throw exceptionIllegalId();
45   return m_id;
46 };
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

#### 5.18.4.3 getName()

```
std::string BasicGameData::getName ( ) const  [inherited]
```

name getter.

**Returns**

name

Definition at line 36 of file basicGamedata.cpp.

```
36 { return m_name; }
```

References BasicGameData::m_name.

#### 5.18.4.4 isId()

```
bool BasicGameData::isId (
            id_t id ) const  [inherited]
```

Check if id is equal.

**Parameters**

| | |
|---|---|
| *id* | Id to check. |

**Returns**

True if id is same as parameter.

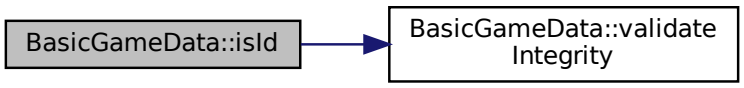
Definition at line 48 of file basicGamedata.cpp.

```
48                                      {
49   validateIntegrity();
50   bool res{m_id == id};
51   return res;
52 }
```

References BasicGameData::m_id, and BasicGameData::validateIntegrity().

Referenced by GameData::getItem().

Here is the call graph for this function:




Here is the caller graph for this function:


#### 5.18.4.5 setDescription()

```
void BasicGameData::setDescription (
            std::string description )  [inherited]
```

Sets description.

**Parameters**

| | |
|---|---|
| *description* | Description to be set. |

Definition at line 32 of file basicGamedata.cpp.

```
32                                                          {
33   m_description = description;
34 }
```

References BasicGameData::m_description.

#### 5.18.4.6 setId()

```
void BasicGameData::setId (
            id_t id )  [protected], [inherited]
```

id setter.

**Parameters**

| | |
|---|---|
| *id* | Id to set. If id to be set is INVALID_ID this will not be set. |

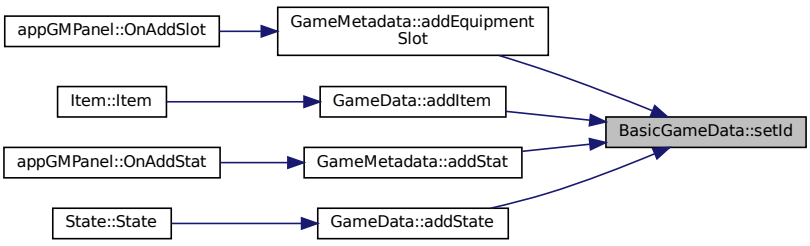
Definition at line 17 of file basicGamedata.cpp.

```
17                                  {
18   if (id == INVALID_ID)
19     return;
20   m_id = id;
21 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by GameMetadata::addEquipmentSlot(), GameData::addItem(), GameMetadata::addStat(), and GameData::addState().

Here is the caller graph for this function:



#### 5.18.4.7 setName()

```
void BasicGameData::setName (
            std::string name )  [inherited]
```

Sets name.

**Parameters**

| | |
|---|---|
| *name* | name to be set. |

Definition at line 30 of file basicGamedata.cpp.

```
30 { m_name = name; }
```

References BasicGameData::m_name.

#### 5.18.4.8 validateIntegrity()

```
void BasicGameData::validateIntegrity ( ) const  [protected], [inherited]
```

Check itegrity of data.

Checks if id is INVALID_ID.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | id is illegal. |

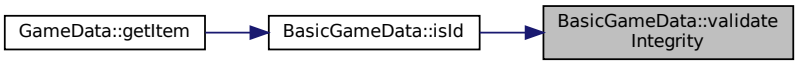
Definition at line 22 of file basicGamedata.cpp.

```
22                                          {
24   if (m_id == INVALID_ID) {
26     throw exceptionIllegalId();
27   }
28 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by BasicGameData::isId().

Here is the caller graph for this function:




### 5.18.5 Member Data Documentation

#### 5.18.5.1 INVALID_ID

```
constexpr id_t BasicGameData::INVALID_ID {std::numeric_limits<id_t>::min()}  [static], [constexpr],
[inherited]
```

Value indicating that id is invalid.

Definition at line 35 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::setId(), and BasicGameData::validateIntegrity().

#### 5.18.5.2 m_description

```
std::string BasicGameData::m_description  [private], [inherited]
```

description.

Definition at line 43 of file basicGamedata.hpp.

Referenced by BasicGameData::getDescription(), and BasicGameData::setDescription().

#### 5.18.5.3 m_id

```
id_t BasicGameData::m_id {INVALID_ID}  [private], [inherited]
```

id.

Definition at line 39 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::isId(), BasicGameData::setId(), and BasicGameData↩
::validateIntegrity().

#### 5.18.5.4 m_name

```
std::string BasicGameData::m_name  [private], [inherited]
```

name.

Definition at line 41 of file basicGamedata.hpp.

Referenced by BasicGameData::getName(), and BasicGameData::setName().

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

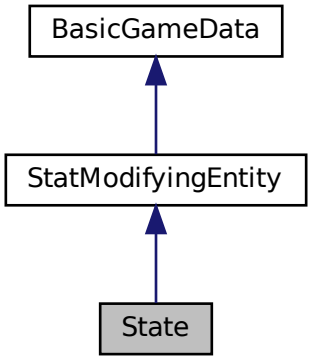
- stat.hpp
- stat.cpp

## 5.19 State Class Reference

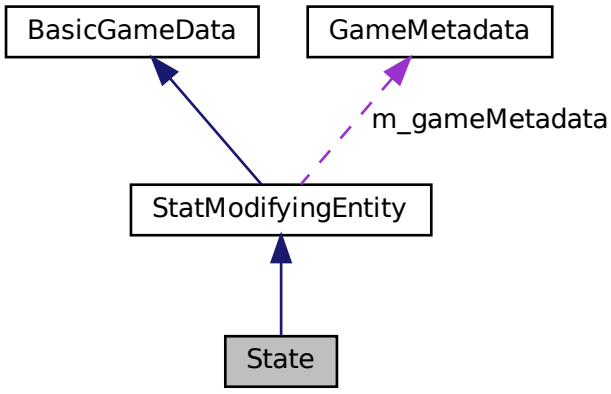
Reperesents State in game.

`#include <state.hpp>`

Inheritance diagram for State:




Collaboration diagram for State:




### Public Types

- using modifier_t = std::pair< Stat::id_t, Stat::value_t >

  *Repesents modification of stats <stat modyfied, value of modification>*
- using modifersCollection_t = std::vector< modifier_t >

  *Collection type used to store modifers.*
- using id_t = long long

  *Type used for ids.*

## Public Member Functions

- State (GameData *gameData, std::string name, std::string description="")
  *Constructor.*
- void addModifier (Stat::id_t, Stat::value_t by)
  *Add modification of stats.*
- Stat::value_t getModifierValue (Stat::id_t id) const
  *Modifier value getter.*
- const GameMetadata *const getGameMetadata () const
  *GameMeatadata getter.*
- const modifersCollection_t & getModifiers () const
  *Modifiers getter.*
- void setName (std::string name)
  *Sets name.*
- void setDescription (std::string description)
  *Sets description.*
- std::string getName () const
  *name getter.*
- std::string getDescription () const
  *Description getter.*
- id_t getId () const
  *id getter.*
- bool isId (id_t id) const
  *Check if id is equal.*

## Static Public Attributes

- static constexpr id_t INVALID_ID {std::numeric_limits<id_t>::min()}
  *Value indicating that id is invalid.*

## Protected Member Functions

- void setId (id_t id)
  *id setter.*
- void validateIntegrity () const
  *Check itegrity of data.*

## Private Attributes

- modifersCollection_t m_modifiers
  *Holds modifiers.*
- const GameMetadata *const m_gameMetadata
  *Game Metadata.*
- id_t m_id {INVALID_ID}
  *id.*
- std::string m_name
  *name.*
- std::string m_description
  *description.*

### 5.19.1 Detailed Description

Reperesents State in game.

Definition at line 16 of file state.hpp.

### 5.19.2 Member Typedef Documentation

#### 5.19.2.1 id_t

```
using BasicGameData::id_t = long long  [inherited]
```

Type used for ids.

Definition at line 33 of file basicGamedata.hpp.

#### 5.19.2.2 modifersCollection_t

```
using StatModifyingEntity::modifersCollection_t = std::vector<modifier_t>  [inherited]
```

Collection type used to store modifers.

Definition at line 38 of file statModifyingEntity.hpp.

#### 5.19.2.3 modifier_t

```
using StatModifyingEntity::modifier_t = std::pair<Stat::id_t, Stat::value_t>  [inherited]
```

Repesents modification of stats <stat modyfied, value of modification>

Definition at line 36 of file statModifyingEntity.hpp.

### 5.19.3 Constructor & Destructor Documentation

#### 5.19.3.1 State()

```
State::State (
            GameData * gameData,
            std::string name,
            std::string description = "" )
```

Constructor.

**Parameters**

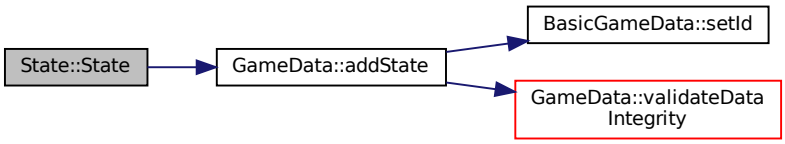
| | |
|---|---|
| *gameData* | Gamedata into which State will be added |
| *name* | Name of state. |
| *description* | Description of state. |

Definition at line 9 of file state.cpp.

```
10     : StatModifyingEntity(gameData, name, description) {
11   gameData->addState(this);
12 }
```

References GameData::addState().

Here is the call graph for this function:




## 5.19.4 Member Function Documentation

### 5.19.4.1 addModifier()

```
void StatModifyingEntity::addModifier (
            Stat::id_t statModifed,
            Stat::value_t by )  [inherited]
```

Add modification of stats.

**Parameters**

| | |
|---|---|
| *statModifed* | modified. |
| *by* | Modify value. |

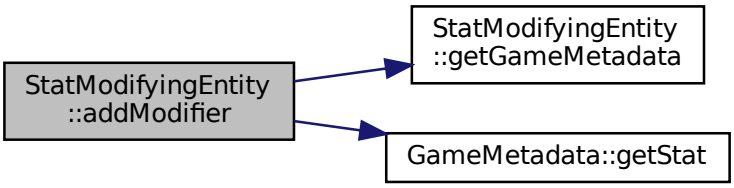
**Todo** Check if entity has modifier of that Stat already.

Definition at line 24 of file statModifyingEntity.cpp.

```
25                                                          {
26
27   getGameMetadata()->getStat(statModifed);
28   m_modifiers.push_back({statModifed, by});
29 }
```

References StatModifyingEntity::getGameMetadata(), GameMetadata::getStat(), and StatModifyingEntity::m_↩ modifiers.

Here is the call graph for this function:




#### 5.19.4.2 getDescription()

```
std::string BasicGameData::getDescription ( ) const  [inherited]
```

Description getter.

**Returns**

Description.

Definition at line 38 of file basicGamedata.cpp.

```
38 { return m_description; };
```

References BasicGameData::m_description.

#### 5.19.4.3 getGameMetadata()

```
const GameMetadata *const StatModifyingEntity::getGameMetadata ( ) const  [inherited]
```

GameMeatadata getter.

**Returns**

GemeMetadata used by instance.

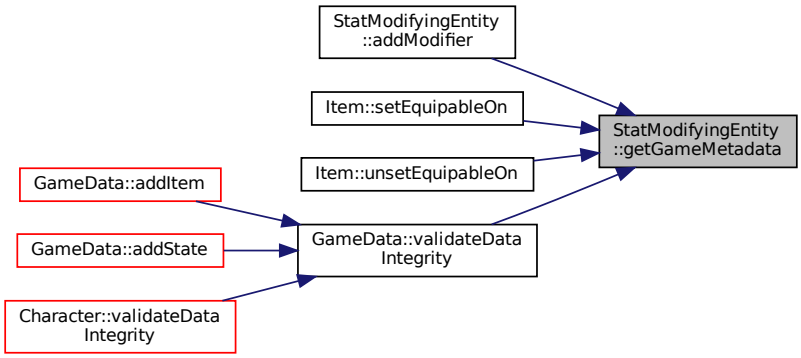
Definition at line 42 of file statModifyingEntity.cpp.

```
42                                                          {
43   return m_gameMetadata;
44 }
```

References StatModifyingEntity::m_gameMetadata.

Referenced by StatModifyingEntity::addModifier(), Item::setEquipableOn(), Item::unsetEquipableOn(), and GameData::validateDataIntegrity().

Here is the caller graph for this function:




#### 5.19.4.4 getId()

```
BasicGameData::id_t BasicGameData::getId ( ) const  [inherited]
```

id getter.

**Returns**

id.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | When tred to get id of instance that has BasicGameData::INVALID_ID. |

Definition at line 40 of file basicGamedata.cpp.

```
40                                                  {
41   if (m_id == INVALID_ID)
44     throw exceptionIllegalId();
45   return m_id;
46 };
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

#### 5.19.4.5 getModifiers()

```
const StatModifyingEntity::modifersCollection_t & StatModifyingEntity::getModifiers ( ) const
[inherited]
```

Modifiers getter.

**Returns**

::m_modifiers

Definition at line 47 of file statModifyingEntity.cpp.

```
47                                                  {
48   return m_modifiers;
49 }
```

References StatModifyingEntity::m_modifiers.

#### 5.19.4.6 getModifierValue()

```
Stat::value_t StatModifyingEntity::getModifierValue (
            Stat::id_t id ) const  [inherited]
```

Modifier value getter.

**Parameters**

| | |
|---|---|
| *id* | Id of stat to get value of modifier of. |

**Returns**

Modifier value or 0 if Instance does not modify stat asked.

Definition at line 31 of file statModifyingEntity.cpp.

```
31                                                                          {
32   auto lookup{
33       std::find_if(m_modifiers.begin(), m_modifiers.end(),
34                    [=](const modifier_t mod) { return mod.first == id; })};
35   // If not found modifier of asked stat.
36   if (lookup == m_modifiers.end())
37     return 0;
38
39   return lookup->second;
40 }
```

References StatModifyingEntity::m_modifiers.

Referenced by Character::getStatValueEquipmentContrubitors().

Here is the caller graph for this function:


#### 5.19.4.7 getName()

```
std::string BasicGameData::getName ( ) const  [inherited]
```

name getter.

**Returns**

name

Definition at line 36 of file basicGamedata.cpp.

```
36 { return m_name; }
```

References BasicGameData::m_name.

#### 5.19.4.8 isId()

```
bool BasicGameData::isId (
            id_t id ) const  [inherited]
```

Check if id is equal.

**Parameters**

| | |
|---|---|
| *id* | Id to check. |

**Returns**

True if id is same as parameter.

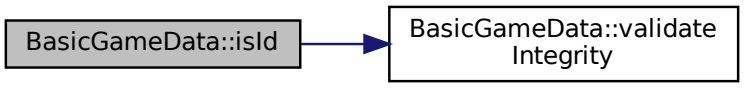
Definition at line 48 of file basicGamedata.cpp.

```
48                                  {
49   validateIntegrity();
50   bool res{m_id == id};
51   return res;
52 }
```

References BasicGameData::m_id, and BasicGameData::validateIntegrity().

Referenced by GameData::getItem().

Here is the call graph for this function:




Here is the caller graph for this function:


#### 5.19.4.9 setDescription()

```
void BasicGameData::setDescription (
            std::string description )  [inherited]
```

Sets description.

**Parameters**

| | |
|---|---|
| *description* | Description to be set. |

Definition at line 32 of file basicGamedata.cpp.

```
32                                                  {
33   m_description = description;
34 }
```

References BasicGameData::m_description.

#### 5.19.4.10 setId()

```
void BasicGameData::setId (
            id_t id )  [protected], [inherited]
```

id setter.

**Parameters**

| | |
|---|---|
| *id* | Id to set. If id to be set is INVALID_ID this will not be set. |

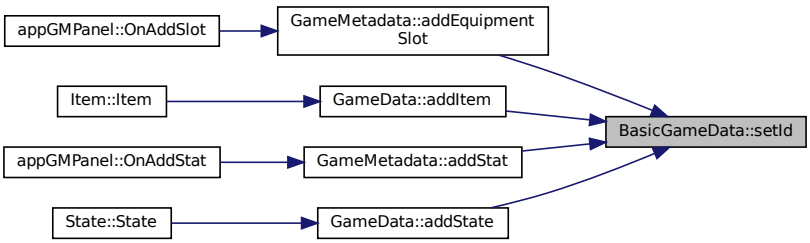
Definition at line 17 of file basicGamedata.cpp.

```
17                                  {
18   if (id == INVALID_ID)
19     return;
20   m_id = id;
21 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by GameMetadata::addEquipmentSlot(), GameData::addItem(), GameMetadata::addStat(), and GameData::addState().

Here is the caller graph for this function:




#### 5.19.4.11 setName()

```
void BasicGameData::setName (
            std::string name )  [inherited]
```

Sets name.

**Parameters**

| | |
|---|---|
| *name* | name to be set. |

Definition at line 30 of file basicGamedata.cpp.

```
30 { m_name = name; }
```

References BasicGameData::m_name.

#### 5.19.4.12 validateIntegrity()

```
void BasicGameData::validateIntegrity ( ) const  [protected], [inherited]
```

Check itegrity of data.

Checks if id is INVALID_ID.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | id is illegal. |

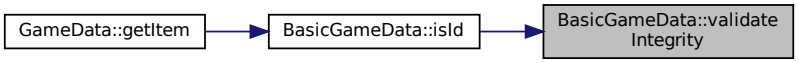
Definition at line 22 of file basicGamedata.cpp.

```
22                                                {
24   if (m_id == INVALID_ID) {
26     throw exceptionIllegalId();
27   }
28 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by BasicGameData::isId().

Here is the caller graph for this function:




## 5.19.5 Member Data Documentation

### 5.19.5.1 INVALID_ID

`constexpr id_t BasicGameData::INVALID_ID {std::numeric_limits<id_t>::min()}  [static], [constexpr], [inherited]`

Value indicating that id is invalid.

Definition at line 35 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::setId(), and BasicGameData::validateIntegrity().

### 5.19.5.2 m_description

`std::string BasicGameData::m_description  [private], [inherited]`

description.

Definition at line 43 of file basicGamedata.hpp.

Referenced by BasicGameData::getDescription(), and BasicGameData::setDescription().

#### 5.19.5.3 m_gameMetadata

```
const GameMetadata* const StatModifyingEntity::m_gameMetadata  [private], [inherited]
```

Game Metadata.

Definition at line 45 of file statModifyingEntity.hpp.

Referenced by StatModifyingEntity::getGameMetadata().

#### 5.19.5.4 m_id

```
id_t BasicGameData::m_id {INVALID_ID}  [private], [inherited]
```

id.

Definition at line 39 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::isId(), BasicGameData::setId(), and BasicGameData↩ ::validateIntegrity().

#### 5.19.5.5 m_modifiers

```
modifersCollection_t StatModifyingEntity::m_modifiers  [private], [inherited]
```

Holds modifiers.

Definition at line 42 of file statModifyingEntity.hpp.

Referenced by StatModifyingEntity::addModifier(), StatModifyingEntity::getModifiers(), and StatModifyingEntity↩ ::getModifierValue().

#### 5.19.5.6 m_name

```
std::string BasicGameData::m_name  [private], [inherited]
```

name.

Definition at line 41 of file basicGamedata.hpp.

Referenced by BasicGameData::getName(), and BasicGameData::setName().

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

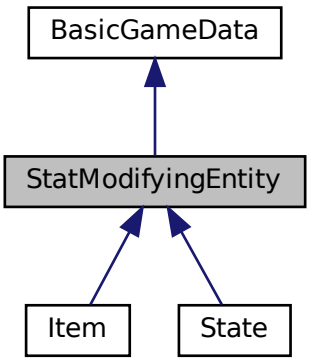
- state.hpp
- state.cpp

## 5.20 StatModifyingEntity Class Reference

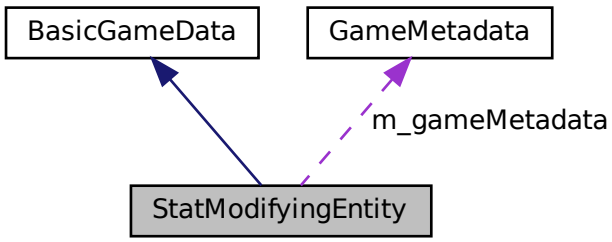
Repesents collection of stat modifiers.

`#include <statModifyingEntity.hpp>`

Inheritance diagram for StatModifyingEntity:




Collaboration diagram for StatModifyingEntity:




### Public Types

- using modifier_t = std::pair< Stat::id_t, Stat::value_t >

  *Repesents modification of stats <stat modyfied, value of modification>*
- using modifersCollection_t = std::vector< modifier_t >

  *Collection type used to store modifers.*
- using id_t = long long

  *Type used for ids.*

## Public Member Functions

- StatModifyingEntity (const GameMetadata *const gameMetadata, std::string name, std::string description="")
  *Constructor.*
- void addModifier (Stat::id_t, Stat::value_t by)
  *Add modification of stats.*
- Stat::value_t getModifierValue (Stat::id_t id) const
  *Modifier value getter.*
- const GameMetadata *const getGameMetadata () const
  *GameMeatadata getter.*
- const modifersCollection_t & getModifiers () const
  *Modifiers getter.*
- void setName (std::string name)
  *Sets name.*
- void setDescription (std::string description)
  *Sets description.*
- std::string getName () const
  *name getter.*
- std::string getDescription () const
  *Description getter.*
- id_t getId () const
  *id getter.*
- bool isId (id_t id) const
  *Check if id is equal.*

## Static Public Attributes

- static constexpr id_t INVALID_ID {std::numeric_limits<id_t>::min()}
  *Value indicating that id is invalid.*

## Protected Member Functions

- void setId (id_t id)
  *id setter.*
- void validateIntegrity () const
  *Check itegrity of data.*

## Private Attributes

- modifersCollection_t m_modifiers
  *Holds modifiers.*
- const GameMetadata *const m_gameMetadata
  *Game Metadata.*
- id_t m_id {INVALID_ID}
  *id.*
- std::string m_name
  *name.*
- std::string m_description
  *description.*

### 5.20.1 Detailed Description

Repesents collection of stat modifiers.

It modifies Stats so it needs access to information about what stats do exist. Associtates instance with game↩ Metadata.

Definition at line 33 of file statModifyingEntity.hpp.

### 5.20.2 Member Typedef Documentation

#### 5.20.2.1 id_t

```
using BasicGameData::id_t = long long  [inherited]
```

Type used for ids.

Definition at line 33 of file basicGamedata.hpp.

#### 5.20.2.2 modifersCollection_t

```
using StatModifyingEntity::modifersCollection_t = std::vector<modifier_t>
```

Collection type used to store modifers.

Definition at line 38 of file statModifyingEntity.hpp.

#### 5.20.2.3 modifier_t

```
using StatModifyingEntity::modifier_t = std::pair<Stat::id_t, Stat::value_t>
```

Repesents modification of stats <stat modyfied, value of modification>

Definition at line 36 of file statModifyingEntity.hpp.

### 5.20.3 Constructor & Destructor Documentation

#### 5.20.3.1 StatModifyingEntity()

```
StatModifyingEntity::StatModifyingEntity (
            const GameMetadata *const gameMetadata,
            std::string name,
            std::string description = "" )
```

Constructor.

**Parameters**

| | |
|---|---|
| *gameMetadata* | gameMetadata that instance is about. |
| *name* | Name |
| *description* | Description. |

**Exceptions**

| | |
|---|---|
| *exceptionInvalidGameMetadata* | |

Definition at line 14 of file statModifyingEntity.cpp.

```
17     : BasicGameData(name, description), m_gameMetadata(gameMetadata) {
19   if (gameMetadata == nullptr)
20     throw exceptionInvalidGameMetadata();
21 }
```

## 5.20.4 Member Function Documentation

### 5.20.4.1 addModifier()

```
void StatModifyingEntity::addModifier (
            Stat::id_t statModifed,
            Stat::value_t by )
```

Add modification of stats.

**Parameters**

| | |
|---|---|
| *statModifed* | modified. |
| *by* | Modify value. |

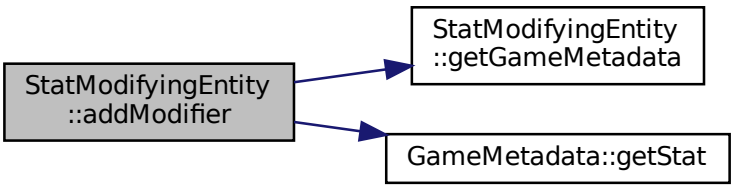
**Todo** Check if entity has modifier of that Stat already.

Definition at line 24 of file statModifyingEntity.cpp.

```
25                                                          {
26
27   getGameMetadata()->getStat(statModifed);
28   m_modifiers.push_back({statModifed, by});
29 }
```

References getGameMetadata(), GameMetadata::getStat(), and m_modifiers.

Here is the call graph for this function:




#### 5.20.4.2 getDescription()

```
std::string BasicGameData::getDescription ( ) const  [inherited]
```

Description getter.

**Returns**

Description.

Definition at line 38 of file basicGamedata.cpp.

```
38 { return m_description; };
```

References BasicGameData::m_description.

#### 5.20.4.3 getGameMetadata()

```
const GameMetadata *const StatModifyingEntity::getGameMetadata ( ) const
```

GameMeatadata getter.

**Returns**

GemeMetadata used by instance.

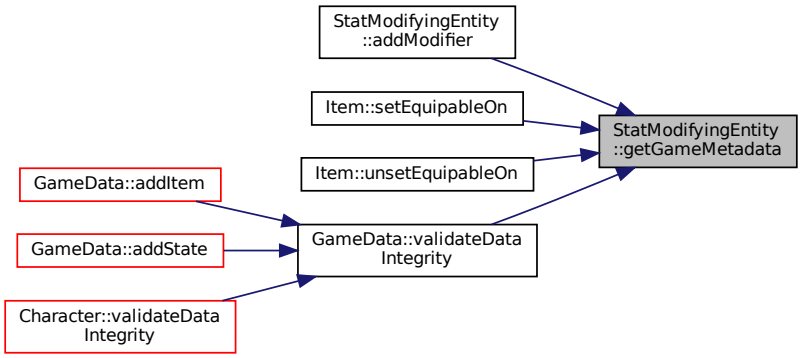
Definition at line 42 of file statModifyingEntity.cpp.

```
42                                                          {
43   return m_gameMetadata;
44 }
```

References m_gameMetadata.

Referenced by addModifier(), Item::setEquipableOn(), Item::unsetEquipableOn(), and GameData::validateData↩ Integrity().

Here is the caller graph for this function:




#### 5.20.4.4 getId()

```
BasicGameData::id_t BasicGameData::getId ( ) const  [inherited]
```

id getter.

**Returns**

id.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | When tred to get id of instance that has BasicGameData::INVALID_ID. |

Definition at line 40 of file basicGamedata.cpp.

```
40                                                  {
41   if (m_id == INVALID_ID)
44     throw exceptionIllegalId();
45   return m_id;
46 };
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

#### 5.20.4.5 getModifiers()

```
const StatModifyingEntity::modifersCollection_t & StatModifyingEntity::getModifiers ( ) const
```

Modifiers getter.

**Returns**

::m_modifiers

Definition at line 47 of file statModifyingEntity.cpp.

```
47                                                  {
48   return m_modifiers;
49 }
```

References m_modifiers.

#### 5.20.4.6 getModifierValue()

```
Stat::value_t StatModifyingEntity::getModifierValue (
            Stat::id_t id ) const
```

Modifier value getter.

**Parameters**

| | |
|---|---|
| *id* | Id of stat to get value of modifier of. |

**Returns**

Modifier value or 0 if Instance does not modify stat asked.

Definition at line 31 of file statModifyingEntity.cpp.

```
31                                                                          {
32   auto lookup{
33       std::find_if(m_modifiers.begin(), m_modifiers.end(),
34                    [=](const modifier_t mod) { return mod.first == id; })};
35   // If not found modifier of asked stat.
36   if (lookup == m_modifiers.end())
37     return 0;
38
39   return lookup->second;
40 }
```

References m_modifiers.

Referenced by Character::getStatValueEquipmentContrubitors().

Here is the caller graph for this function:


#### 5.20.4.7 getName()

```
std::string BasicGameData::getName ( ) const  [inherited]
```

name getter.

**Returns**

name

Definition at line 36 of file basicGamedata.cpp.

```
36 { return m_name; }
```

References BasicGameData::m_name.

#### 5.20.4.8 isId()

```
bool BasicGameData::isId (
            id_t id ) const  [inherited]
```

Check if id is equal.

**Parameters**

| | |
|---|---|
| *id* | Id to check. |

**Returns**

True if id is same as parameter.

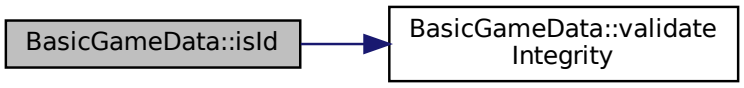
Definition at line 48 of file basicGamedata.cpp.

```
48                                      {
49   validateIntegrity();
50   bool res{m_id == id};
51   return res;
52 }
```

References BasicGameData::m_id, and BasicGameData::validateIntegrity().

Referenced by GameData::getItem().

Here is the call graph for this function:




Here is the caller graph for this function:


#### 5.20.4.9 setDescription()

```
void BasicGameData::setDescription (
            std::string description )  [inherited]
```

Sets description.

**Parameters**

| | |
|---|---|
| *description* | Description to be set. |

Definition at line 32 of file basicGamedata.cpp.

```
32                                                          {
33   m_description = description;
34 }
```

References BasicGameData::m_description.

#### 5.20.4.10 setId()

```
void BasicGameData::setId (
            id_t id )  [protected], [inherited]
```

id setter.

**Parameters**

| | |
|---|---|
| *id* | Id to set. If id to be set is INVALID_ID this will not be set. |

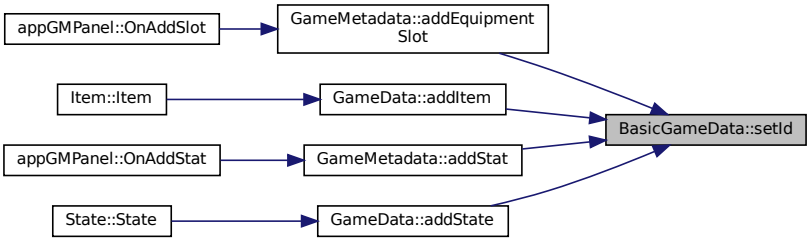
Definition at line 17 of file basicGamedata.cpp.

```
17                                  {
18   if (id == INVALID_ID)
19     return;
20   m_id = id;
21 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by GameMetadata::addEquipmentSlot(), GameData::addItem(), GameMetadata::addStat(), and GameData::addState().

Here is the caller graph for this function:




#### 5.20.4.11 setName()

```
void BasicGameData::setName (
            std::string name )  [inherited]
```

Sets name.

**Parameters**

| | |
|---|---|
| *name* | name to be set. |

Definition at line 30 of file basicGamedata.cpp.

```
30 { m_name = name; }
```

References BasicGameData::m_name.

#### 5.20.4.12 validateIntegrity()

```
void BasicGameData::validateIntegrity ( ) const  [protected], [inherited]
```

Check itegrity of data.

Checks if id is INVALID_ID.

**Exceptions**

| | |
|---|---|
| *exceptionIllegalId* | id is illegal. |

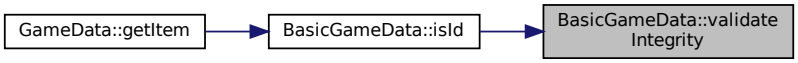
Definition at line 22 of file basicGamedata.cpp.

```
22                                                  {
24   if (m_id == INVALID_ID) {
26     throw exceptionIllegalId();
27   }
28 }
```

References BasicGameData::INVALID_ID, and BasicGameData::m_id.

Referenced by BasicGameData::isId().

Here is the caller graph for this function:




## 5.20.5 Member Data Documentation

### 5.20.5.1 INVALID_ID

`constexpr id_t BasicGameData::INVALID_ID {std::numeric_limits<id_t>::min()}  [static], [constexpr], [inherited]`

Value indicating that id is invalid.

Definition at line 35 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::setId(), and BasicGameData::validateIntegrity().

### 5.20.5.2 m_description

`std::string BasicGameData::m_description  [private], [inherited]`

description.

Definition at line 43 of file basicGamedata.hpp.

Referenced by BasicGameData::getDescription(), and BasicGameData::setDescription().

#### 5.20.5.3 m_gameMetadata

```
const GameMetadata* const StatModifyingEntity::m_gameMetadata  [private]
```

Game Metadata.

Definition at line 45 of file statModifyingEntity.hpp.

Referenced by getGameMetadata().

#### 5.20.5.4 m_id

```
id_t BasicGameData::m_id {INVALID_ID}  [private], [inherited]
```

id.

Definition at line 39 of file basicGamedata.hpp.

Referenced by BasicGameData::getId(), BasicGameData::isId(), BasicGameData::setId(), and BasicGameData↩ ::validateIntegrity().

#### 5.20.5.5 m_modifiers

```
modifersCollection_t StatModifyingEntity::m_modifiers  [private]
```

Holds modifiers.

Definition at line 42 of file statModifyingEntity.hpp.

Referenced by addModifier(), getModifiers(), and getModifierValue().

#### 5.20.5.6 m_name

```
std::string BasicGameData::m_name  [private], [inherited]
```

name.

Definition at line 41 of file basicGamedata.hpp.

Referenced by BasicGameData::getName(), and BasicGameData::setName().

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

- statModifyingEntity.hpp
- statModifyingEntity.cpp

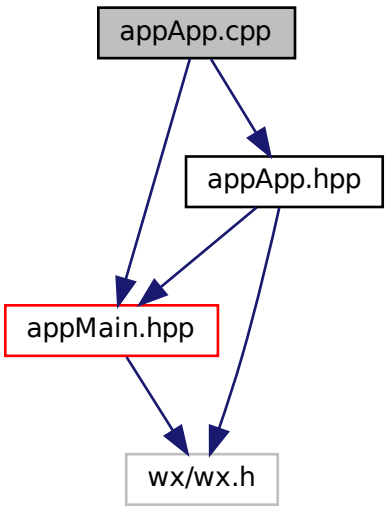
# Chapter 6

# File Documentation

## 6.1 appApp.cpp File Reference

```
#include "appApp.hpp"
#include "appMain.hpp"
```

Include dependency graph for appApp.cpp:




### Functions

- wxIMPLEMENT_APP (appApp)

### 6.1.1 Function Documentation

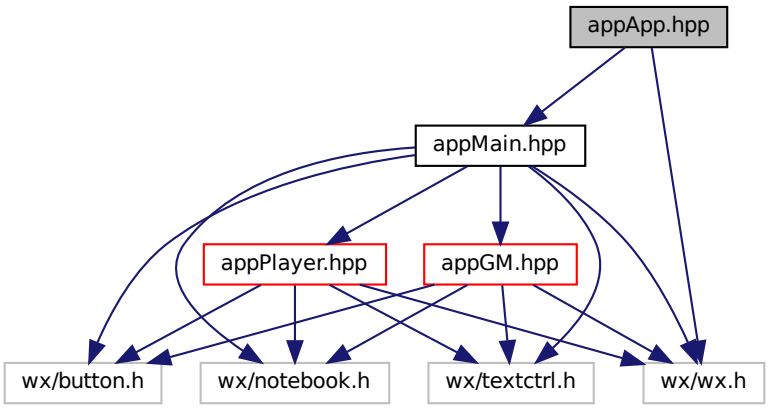
#### 6.1.1.1 wxIMPLEMENT_APP()

```
wxIMPLEMENT_APP (
              appApp   )
```

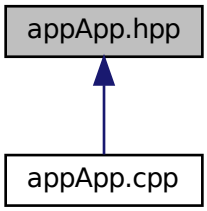
## 6.2 appApp.hpp File Reference

```
#include "appMain.hpp"
#include <wx/wx.h>
```

Include dependency graph for appApp.hpp:




This graph shows which files directly or indirectly include this file:

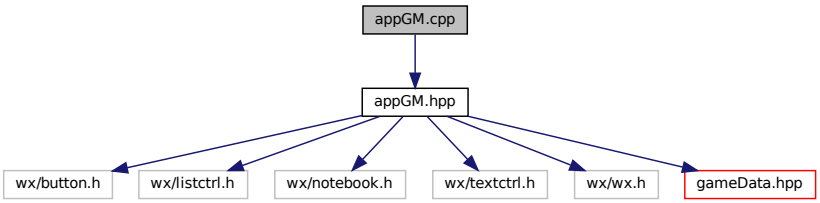



### Classes

- class appApp

# 6.3 appGM.cpp File Reference

`#include "appGM.hpp"`
Include dependency graph for appGM.cpp:




## Functions

- wxBEGIN_EVENT_TABLE (appGMPanel, wxPanel) wxEND_EVENT_TABLE() appGMPanel

## 6.3.1 Function Documentation

### 6.3.1.1 wxBEGIN_EVENT_TABLE()

```
wxBEGIN_EVENT_TABLE (
            appGMPanel ,
            wxPanel  )
```

Definition at line 3 of file appGM.cpp.

```
11     : wxPanel(parent, wxID_ANY), m_gamedata(new GameData) {
12   gmNotebook = new wxNotebook(this, wxID_ANY);
13
14   addElementPanel = new wxPanel(gmNotebook, wxID_ANY);
15   addElementSizer = new wxBoxSizer(wxVERTICAL);
16
17   addStatButton = new wxButton(addElementPanel, ID_ADD_STAT, "Add stat");
18   addEquipmentButton =
19       new wxButton(addElementPanel, ID_ADD_EQUIPMENT, "Add item");
20   addSlotButton = new wxButton(addElementPanel, ID_ADD_SLOT, "Add slot");
21   addStateButton = new wxButton(addElementPanel, ID_ADD_STATE, "Add state");
22
23   addElementSizer->Add(addStatButton, 0, wxALL, 5);
24   addElementSizer->Add(addEquipmentButton, 0, wxALL, 5);
25   addElementSizer->Add(addSlotButton, 0, wxALL, 5);
26   addElementSizer->Add(addStateButton, 0, wxALL, 5);
27
28   addElementPanel->SetSizer(addElementSizer);
29
30   statsPanel = new wxPanel(gmNotebook, wxID_ANY);
31   statsSizer = new wxBoxSizer(wxVERTICAL);
32   statsListCtrl = new wxListCtrl(statsPanel, wxID_ANY, wxDefaultPosition,
33                                  wxDefaultSize, wxLC_REPORT | wxLC_SINGLE_SEL);
34   statsListCtrl->InsertColumn(0, "Name");
35   statsListCtrl->InsertColumn(1, "Description");
36   statsListCtrl->SetColumnWidth(0, 150);
37   statsListCtrl->SetColumnWidth(1, 400);
38   statsSizer->Add(statsListCtrl, 1, wxEXPAND | wxALL, 5);
39   statsPanel->SetSizer(statsSizer);
40
41   eqPanel = new wxPanel(gmNotebook, wxID_ANY);
```

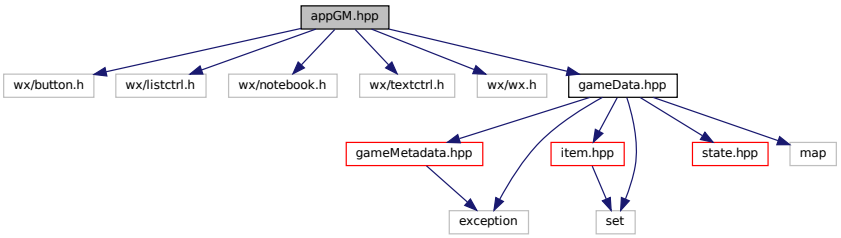
```
42   eqSizer = new wxBoxSizer(wxVERTICAL);
43   eqListCtrl = new wxListCtrl(eqPanel, wxID_ANY, wxDefaultPosition,
44                               wxDefaultSize, wxLC_REPORT | wxLC_SINGLE_SEL);
45   eqListCtrl->InsertColumn(0, "Name");
46   eqListCtrl->SetColumnWidth(0, 150);
47   eqSizer->Add(eqListCtrl, 1, wxEXPAND | wxALL, 5);
48   eqPanel->SetSizer(eqSizer);
49
50   slotPanel = new wxPanel(gmNotebook, wxID_ANY);
51   slotSizer = new wxBoxSizer(wxVERTICAL);
52   slotListCtrl = new wxListCtrl(slotPanel, wxID_ANY, wxDefaultPosition,
53                                 wxDefaultSize, wxLC_REPORT | wxLC_SINGLE_SEL);
54   slotListCtrl->InsertColumn(0, "Name");
55   slotListCtrl->SetColumnWidth(0, 150);
56   slotSizer->Add(slotListCtrl, 1, wxEXPAND | wxALL, 5);
57   slotPanel->SetSizer(slotSizer);
58
59   statePanel = new wxPanel(gmNotebook, wxID_ANY);
60   stateSizer = new wxBoxSizer(wxVERTICAL);
61   stateListCtrl = new wxListCtrl(statePanel, wxID_ANY, wxDefaultPosition,
62                                  wxDefaultSize, wxLC_REPORT | wxLC_SINGLE_SEL);
63   stateListCtrl->InsertColumn(0, "Name");
64   stateListCtrl->SetColumnWidth(0, 150);
65   stateSizer->Add(stateListCtrl, 1, wxEXPAND | wxALL, 5);
66   statePanel->SetSizer(stateSizer);
67
68   gmNotebook->AddPage(addElementPanel, "Add");
69   gmNotebook->AddPage(statsPanel, "Stats");
70   gmNotebook->AddPage(eqPanel, "Items");
71   gmNotebook->AddPage(slotPanel, "Slots");
72   gmNotebook->AddPage(statePanel, "States");
73
74   wxBoxSizer *mainSizer = new wxBoxSizer(wxVERTICAL);
75   mainSizer->Add(gmNotebook, 1, wxEXPAND | wxALL, 5);
76
77   SetSizer(mainSizer);
78 }
```

References ID_ADD_EQUIPMENT, ID_ADD_SLOT, ID_ADD_STAT, and ID_ADD_STATE.

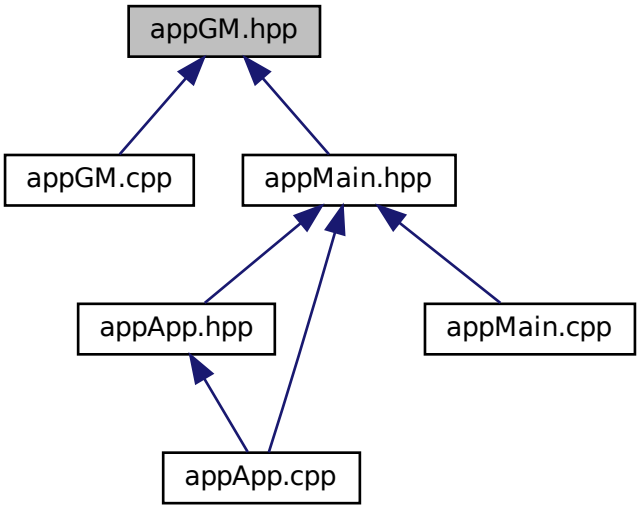
## 6.4 appGM.hpp File Reference

```
#include <wx/button.h>
#include <wx/listctrl.h>
#include <wx/notebook.h>
#include <wx/textctrl.h>
#include <wx/wx.h>
#include "gameData.hpp"
```

Include dependency graph for appGM.hpp:



This graph shows which files directly or indirectly include this file:




## Classes

- class appGMPanel

## Enumerations

- enum { ID_ADD_STAT = wxID_HIGHEST + 1 , ID_ADD_EQUIPMENT , ID_ADD_SLOT , ID_ADD_STATE }

### 6.4.1 Enumeration Type Documentation

#### 6.4.1.1 anonymous enum

```
anonymous enum
```

**Enumerator**

| | |
|---|---|
| ID_ADD_STAT | |
| ID_ADD_EQUIPMENT | |
| ID_ADD_SLOT | |
| ID_ADD_STATE | |

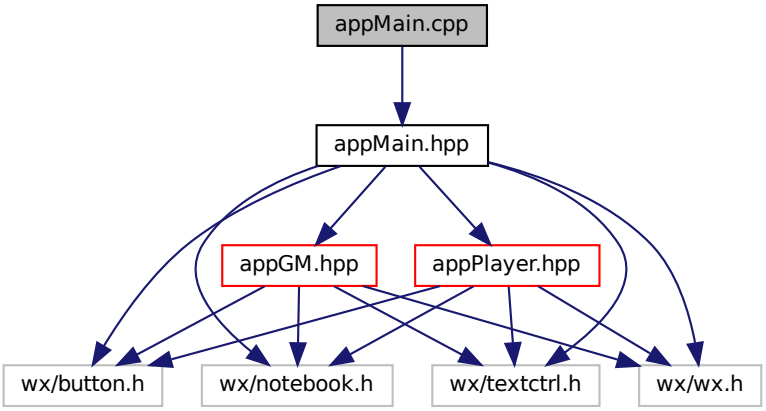
Definition at line 11 of file appGM.hpp.

```
11 { ID_ADD_STAT = wxID_HIGHEST + 1, ID_ADD_EQUIPMENT, ID_ADD_SLOT, ID_ADD_STATE };
```

## 6.5 appMain.cpp File Reference

```
#include "appMain.hpp"
```

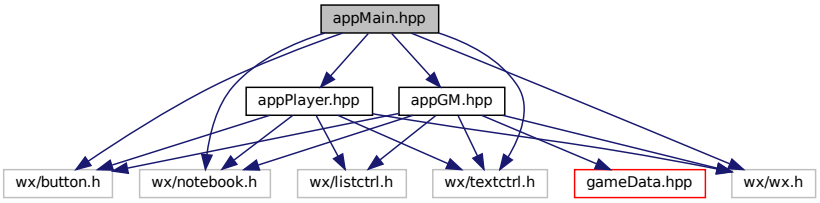
Include dependency graph for appMain.cpp:



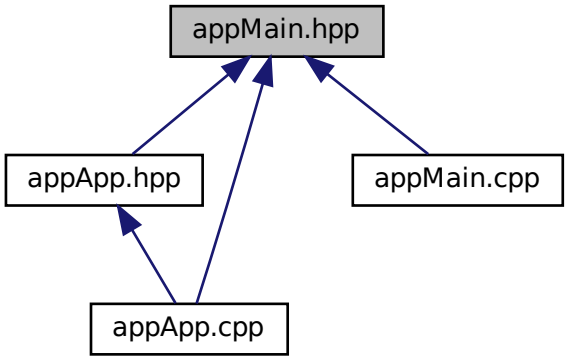

## 6.6 appMain.hpp File Reference

```
#include "appGM.hpp"
#include "appPlayer.hpp"
#include <wx/button.h>
#include <wx/notebook.h>
#include <wx/textctrl.h>
#include <wx/wx.h>
```

Include dependency graph for appMain.hpp:



This graph shows which files directly or indirectly include this file:




## Classes

- class appFrame

## Enumerations

- enum { ID_ADD_CHARACTER = 1 }

### 6.6.1 Enumeration Type Documentation

#### 6.6.1.1 anonymous enum

```
anonymous enum
```

**Enumerator**

| | |
|---|---|
| ID_ADD_CHARACTER | |

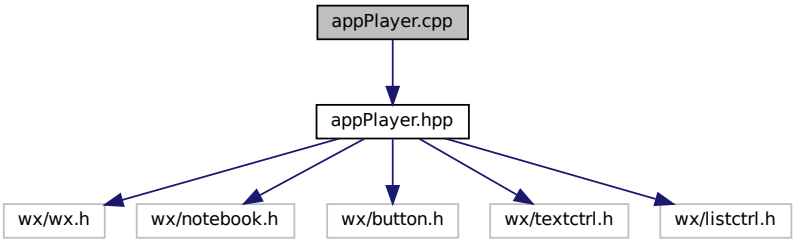
Definition at line 33 of file appMain.hpp.

```
33 { ID_ADD_CHARACTER = 1 };
```

## 6.7 appPlayer.cpp File Reference

```
#include "appPlayer.hpp"
```

Include dependency graph for appPlayer.cpp:




## Functions

- wxBEGIN_EVENT_TABLE (appPlayerPanel, wxPanel) wxEND_EVENT_TABLE() appPlayerPanel

### 6.7.1 Function Documentation

#### 6.7.1.1 wxBEGIN_EVENT_TABLE()

```
wxBEGIN_EVENT_TABLE (
            appPlayerPanel ,
            wxPanel  )
```

Definition at line 3 of file appPlayer.cpp.

```
12     : wxPanel(parent, wxID_ANY)
13 {
14     playerNotebook = new wxNotebook(this, wxID_ANY);
15 
16     statsPanel = new wxPanel(playerNotebook, wxID_ANY);
17     statsSizer = new wxBoxSizer(wxVERTICAL);
18     statsListCtrl = new wxListCtrl(statsPanel, wxID_ANY, wxDefaultPosition, wxDefaultSize, wxLC_REPORT |
       wxLC_SINGLE_SEL);
19     statsListCtrl->InsertColumn(0, "Name");
20     statsListCtrl->InsertColumn(1, "Description");
21     statsListCtrl->InsertColumn(2, "Value");
22     statsListCtrl->SetColumnWidth(0, 150);
23     statsListCtrl->SetColumnWidth(1, 400);
24     statsListCtrl->SetColumnWidth(2, 50);
25     statsSizer->Add(statsListCtrl, 1, wxEXPAND | wxALL, 5);
26     statsPanel->SetSizer(statsSizer);
27 
28     equippedEQPanel = new wxPanel(playerNotebook, wxID_ANY);
29     equippedEQSizer = new wxBoxSizer(wxVERTICAL);
30     equippedEQListCtrl = new wxListCtrl(equippedEQPanel, wxID_ANY, wxDefaultPosition, wxDefaultSize,
       wxLC_REPORT | wxLC_SINGLE_SEL);
31     equippedEQListCtrl->InsertColumn(0, "Name");
32     equippedEQListCtrl->InsertColumn(1, "Description");
33     equippedEQListCtrl->SetColumnWidth(0, 150);
34     equippedEQListCtrl->SetColumnWidth(1, 400);
35     equippedEQSizer->Add(equippedEQListCtrl, 1, wxEXPAND | wxALL, 5);
36 
37     // Dodajemy przycisk do panelu equippedEQPanel
38     unequipEquipmentButton = new wxButton(equippedEQPanel, ID_CHAR_UNEQUIP_EQUIPMENT, "Unequip");
39     equippedEQSizer->Add(unequipEquipmentButton, 0, wxALIGN_RIGHT | wxALL, 5);
40 
41     equippedEQPanel->SetSizer(equippedEQSizer);
42 
43     eqPanel = new wxPanel(playerNotebook, wxID_ANY);
```

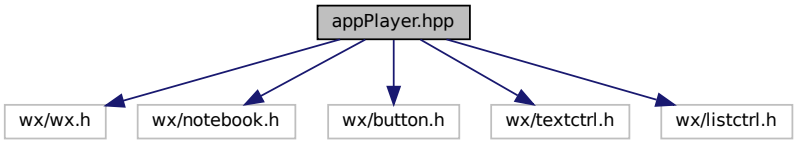
```
44      eqSizer = new wxBoxSizer(wxVERTICAL);
45      eqListCtrl = new wxListCtrl(eqPanel, wxID_ANY, wxDefaultPosition, wxDefaultSize, wxLC_REPORT |
        wxLC_SINGLE_SEL);
46      eqListCtrl->InsertColumn(0, "Name");
47      eqListCtrl->InsertColumn(1, "Description");
48      eqListCtrl->SetColumnWidth(0, 150);
49      eqListCtrl->SetColumnWidth(1, 400);
50      eqSizer->Add(eqListCtrl, 1, wxEXPAND | wxALL, 5);
51
52      // Dodajemy przycisk do panelu eqPanel
53      equipEquipmentButton = new wxButton(eqPanel, ID_CHAR_EQUIP_EQUIPMENT, "Equip");
54      eqSizer->Add(equipEquipmentButton, 0, wxALIGN_RIGHT | wxALL, 5);
55
56      eqPanel->SetSizer(eqSizer);
57
58      statesPanel = new wxPanel(playerNotebook, wxID_ANY);
59      statesSizer = new wxBoxSizer(wxVERTICAL);
60      statesListCtrl = new wxListCtrl(statesPanel, wxID_ANY, wxDefaultPosition, wxDefaultSize, wxLC_REPORT
        | wxLC_SINGLE_SEL);
61      statesListCtrl->InsertColumn(0, "Name");
62      statesListCtrl->InsertColumn(1, "Description");
63      statesListCtrl->SetColumnWidth(0, 150);
64      statesListCtrl->SetColumnWidth(1, 400);
65      statesSizer->Add(statesListCtrl, 1, wxEXPAND | wxALL, 5);
66      statesPanel->SetSizer(statesSizer);
67
68      addPanel = new wxPanel(playerNotebook, wxID_ANY);
69      addPanelSizer = new wxBoxSizer(wxVERTICAL);
70      addStatButton = new wxButton(addPanel, ID_CHAR_ADD_STAT, "Add Stat");
71      addEquipmentButton = new wxButton(addPanel, ID_CHAR_ADD_EQUIPMENT, "Add Equipment");
72      addStateButton = new wxButton(addPanel, ID_CHAR_ADD_STATE, "Add State");
73      addPanelSizer->Add(addStatButton, 0, wxALIGN_LEFT | wxALL, 5);
74      addPanelSizer->Add(addEquipmentButton, 0, wxALIGN_LEFT | wxALL, 5);
75      addPanelSizer->Add(addStateButton, 0, wxALIGN_LEFT | wxALL, 5);
76      addPanel->SetSizer(addPanelSizer);
77
78      playerNotebook->AddPage(statsPanel, "Stats");
79      playerNotebook->AddPage(equippedEQPanel, "Equipped EQ");
80      playerNotebook->AddPage(eqPanel, "EQ");
81      playerNotebook->AddPage(statesPanel, "States");
82      playerNotebook->AddPage(addPanel, "Add");
83
84      mainSizer = new wxBoxSizer(wxVERTICAL);
85      mainSizer->Add(playerNotebook, 1, wxEXPAND | wxALL, 5);
86      SetSizerAndFit(mainSizer);
87  }
```

References ID_CHAR_ADD_EQUIPMENT, ID_CHAR_ADD_STAT, ID_CHAR_ADD_STATE, ID_CHAR_EQUIP_↩ EQUIPMENT, and ID_CHAR_UNEQUIP_EQUIPMENT.

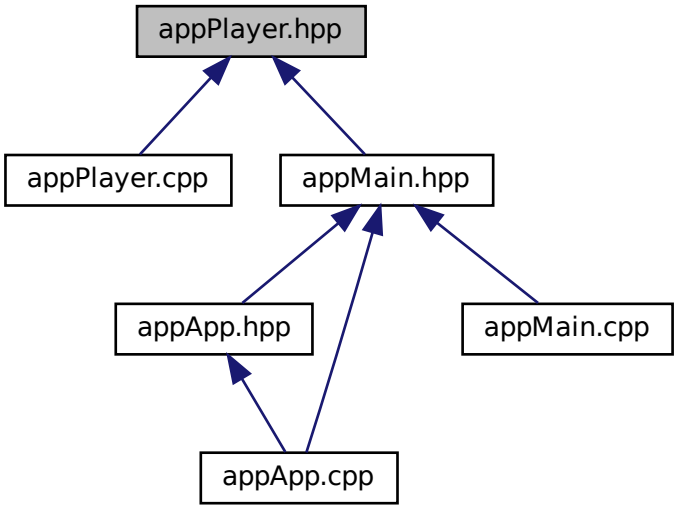
## 6.8 appPlayer.hpp File Reference

```
#include <wx/wx.h>
#include <wx/notebook.h>
#include <wx/button.h>
#include <wx/textctrl.h>
#include <wx/listctrl.h>
```

Include dependency graph for appPlayer.hpp:



This graph shows which files directly or indirectly include this file:




## Classes

- class appPlayerPanel

## Enumerations

- enum {
  ID_CHAR_ADD_STAT = wxID_HIGHEST + 1 , ID_CHAR_ADD_EQUIPMENT , ID_CHAR_ADD_STATE ,
  ID_CHAR_EQUIP_EQUIPMENT ,
  ID_CHAR_UNEQUIP_EQUIPMENT }

### 6.8.1 Enumeration Type Documentation

#### 6.8.1.1 anonymous enum

anonymous enum

**Enumerator**

| | |
|---|---|
| ID_CHAR_ADD_STAT | |
| ID_CHAR_ADD_EQUIPMENT | |
| ID_CHAR_ADD_STATE | |
| ID_CHAR_EQUIP_EQUIPMENT | |
| ID_CHAR_UNEQUIP_EQUIPMENT | |

Definition at line 10 of file appPlayer.hpp.

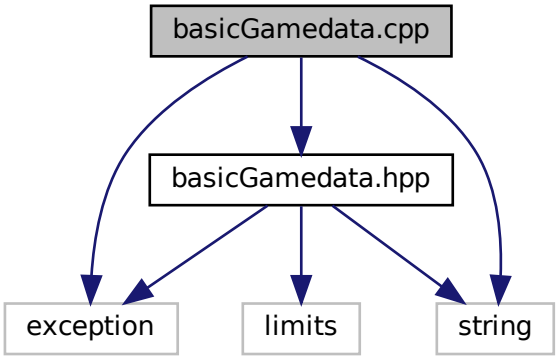
```
11 {
12     ID_CHAR_ADD_STAT = wxID_HIGHEST + 1,
13     ID_CHAR_ADD_EQUIPMENT,
14     ID_CHAR_ADD_STATE,
15     ID_CHAR_EQUIP_EQUIPMENT,
16     ID_CHAR_UNEQUIP_EQUIPMENT
17 };
```

## 6.9 basicGamedata.cpp File Reference

BasicGameData implementation.

```
#include "basicGamedata.hpp"
#include <exception>
#include <string>
```

Include dependency graph for basicGamedata.cpp:




### 6.9.1 Detailed Description

BasicGameData implementation.

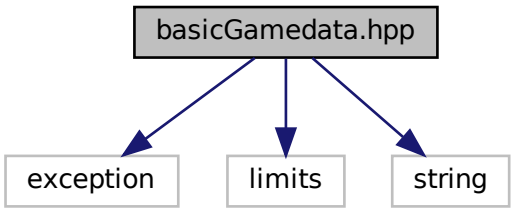
## 6.10 basicGamedata.hpp File Reference

BasicGameData interface.

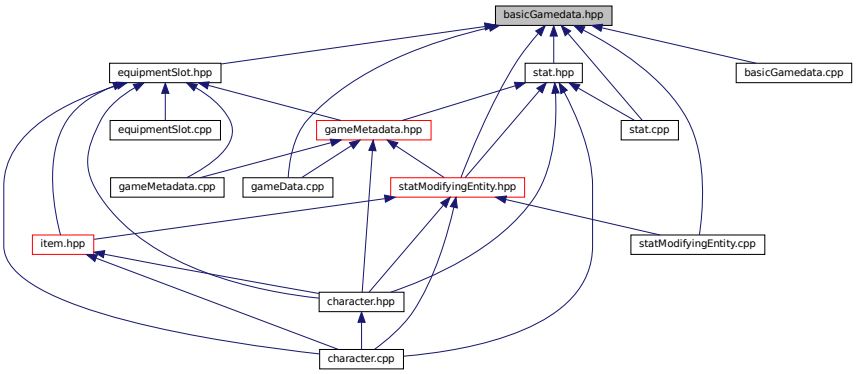
```
#include <exception>
#include <limits>
```

```
#include <string>
```

Include dependency graph for basicGamedata.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class exceptionIllegalId

  *Illegal id exception.*
- class BasicGameData

  *Basic information about game.*

### 6.10.1 Detailed Description

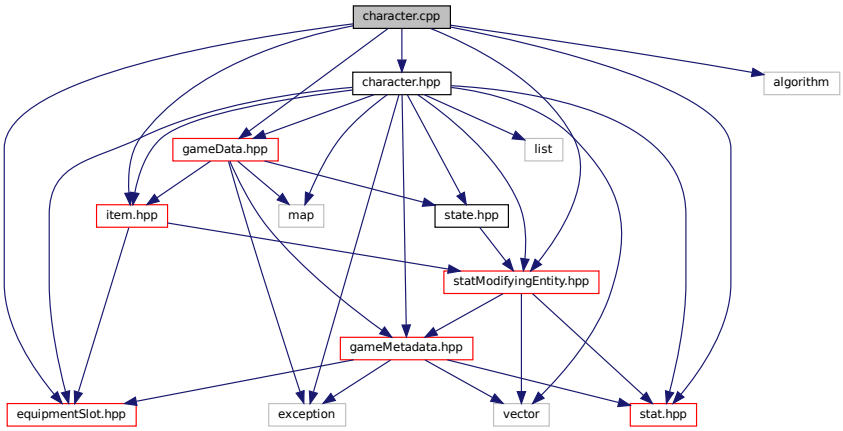
BasicGameData interface.

## 6.11 character.cpp File Reference

Character implementation.

```
#include "character.hpp"
#include "equipmentSlot.hpp"
#include "gameData.hpp"
#include "item.hpp"
#include "stat.hpp"
#include "statModifyingEntity.hpp"
#include <algorithm>
```

Include dependency graph for character.cpp:




### 6.11.1 Detailed Description

Character implementation.

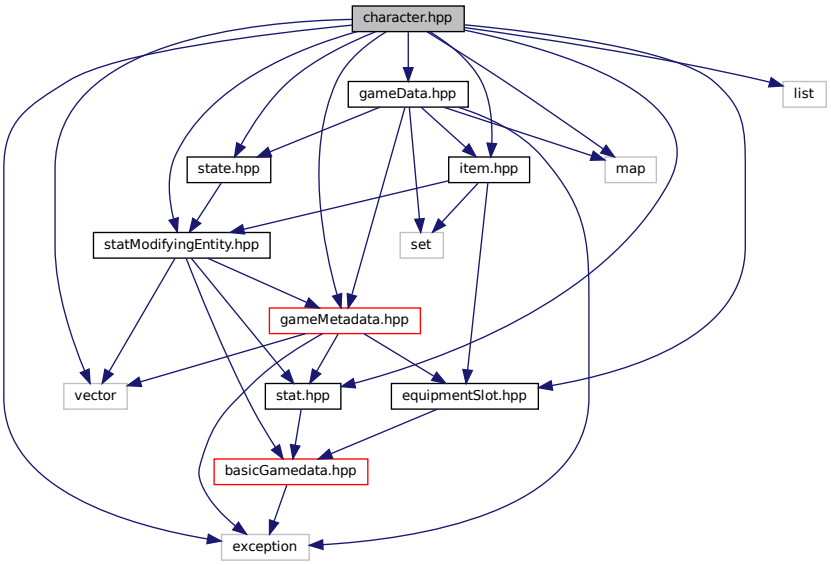
## 6.12 character.hpp File Reference

Character interface.

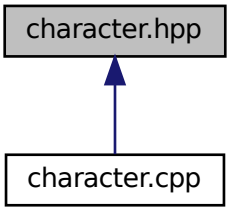
```
#include "equipmentSlot.hpp"
#include "gameData.hpp"
#include "gameMetadata.hpp"
#include "item.hpp"
#include "stat.hpp"
#include "statModifyingEntity.hpp"
#include "state.hpp"
#include <exception>
#include <list>
#include <map>
```

```
#include <vector>
```

Include dependency graph for character.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class exceptionEquipmentSlotOccupied

  *Tried to perform operation on occupied already slot.*
- class exceptionEquipmentSlotUnused

  *Tried to extract data from unused slot.*
- class excpetionEquipmentSlotIllegalUsage

  *Tried to put stuff where it is not suppsoed to go.*
- class Character

  *Represents character.*

### 6.12.1 Detailed Description

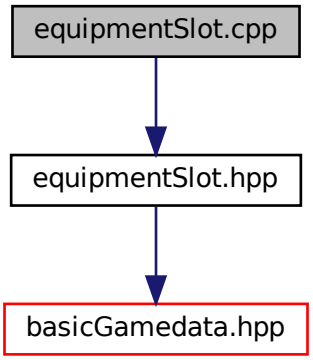
Character interface.

## 6.13 equipmentSlot.cpp File Reference

EquipmentSlot implementation.

```
#include "equipmentSlot.hpp"
```

Include dependency graph for equipmentSlot.cpp:




### 6.13.1 Detailed Description

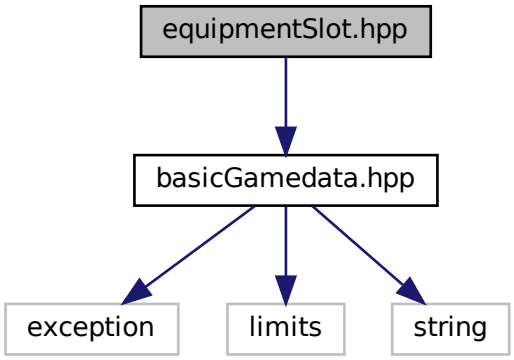
EquipmentSlot implementation.

## 6.14 equipmentSlot.hpp File Reference

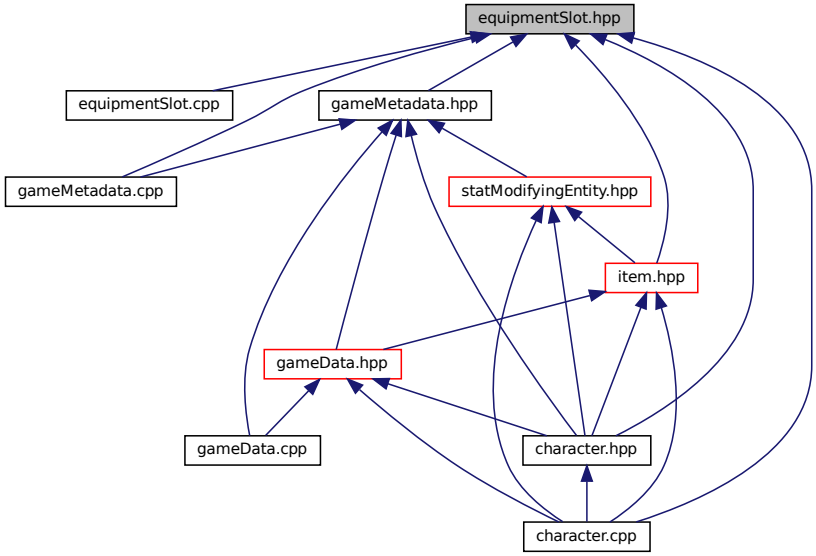
EquipmentSlot interface.

```
#include "basicGamedata.hpp"
```

Include dependency graph for equipmentSlot.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class EquipmentSlot

  *Equipment slot repesentation.*

### 6.14.1 Detailed Description

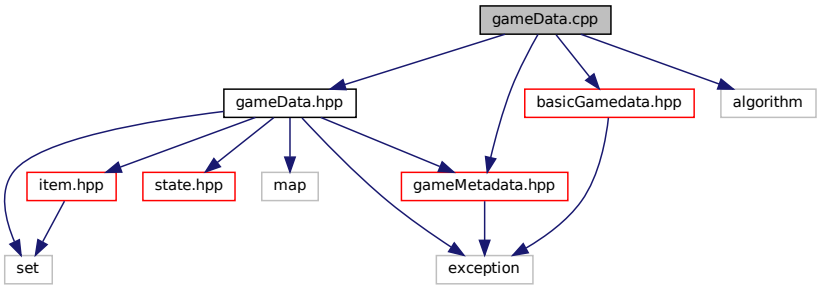
EquipmentSlot interface.

## 6.15 gameData.cpp File Reference

GameData implementation.

```
#include "gameData.hpp"
#include "basicGamedata.hpp"
#include "gameMetadata.hpp"
#include <algorithm>
```

Include dependency graph for gameData.cpp:




### 6.15.1 Detailed Description

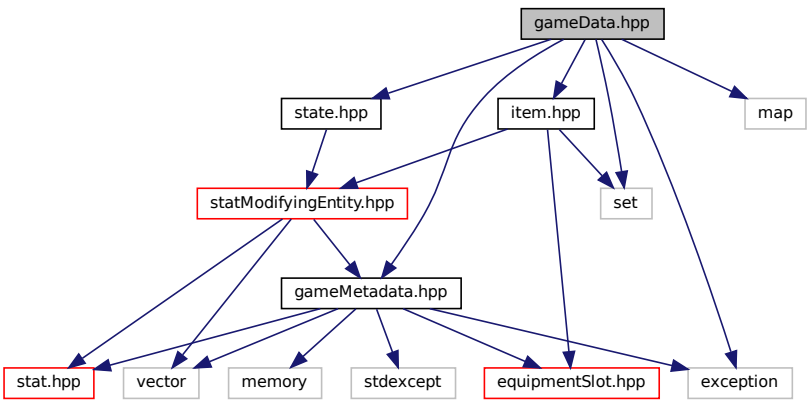
GameData implementation.

## 6.16 gameData.hpp File Reference

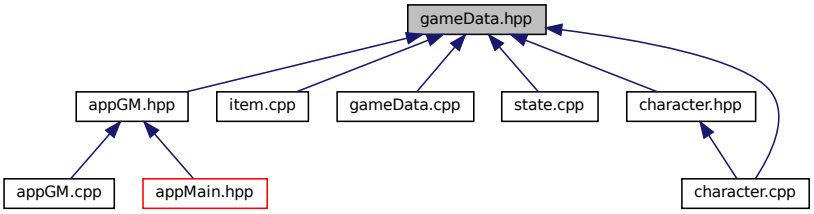
GameData interface.

```
#include "gameMetadata.hpp"
#include "item.hpp"
#include "state.hpp"
#include <exception>
#include <map>
#include <set>
```

Include dependency graph for gameData.hpp:



This graph shows which files directly or indirectly include this file:




## Classes

- class excpetionGameDataMissmatch

  *Thrown when attempted to do operation that requires 2 objects to use common GameData but different were used.*
- class GameData

  *On top of what GameMetadata does. Holds items cataloge.*

### 6.16.1 Detailed Description

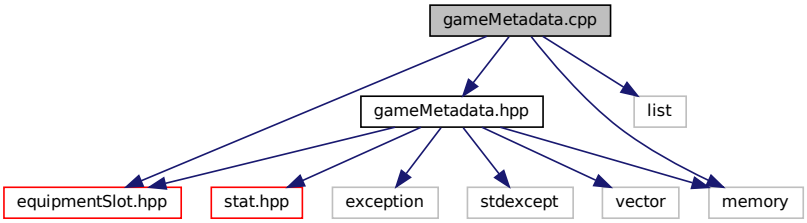
GameData interface.

## 6.17 gameMetadata.cpp File Reference

GameMetadata implementation.

```
#include "gameMetadata.hpp"
#include "equipmentSlot.hpp"
#include <list>
#include <memory>
```

Include dependency graph for gameMetadata.cpp:




### 6.17.1 Detailed Description

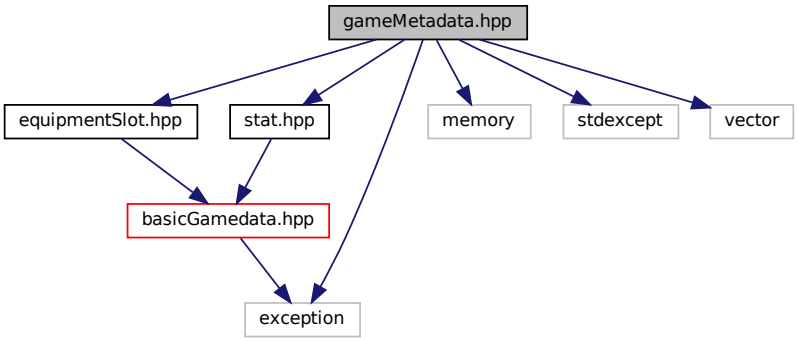
GameMetadata implementation.

# 6.18 gameMetadata.hpp File Reference

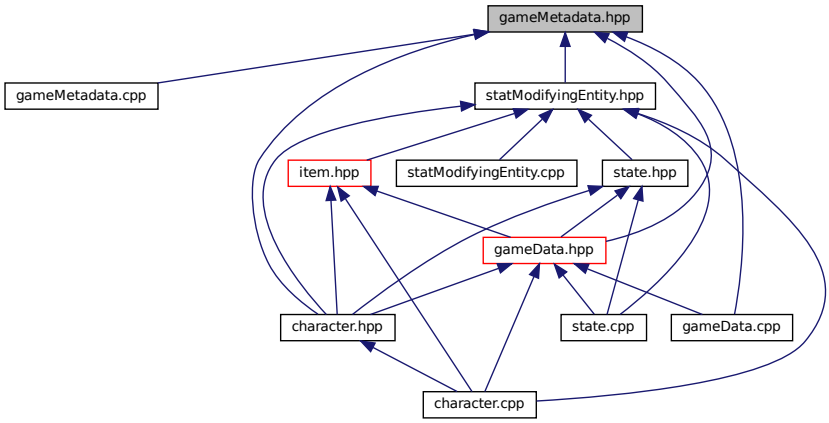
GameMetadata interface.

```
#include "equipmentSlot.hpp"
#include "stat.hpp"
#include <exception>
#include <memory>
#include <stdexcept>
#include <vector>
```

Include dependency graph for gameMetadata.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class exceptionNonExistingId

  *Exception.*
- class GameMetadata

  *Holds game metadata. That is what statistics exist and what equipable slots exist.*

### 6.18.1 Detailed Description

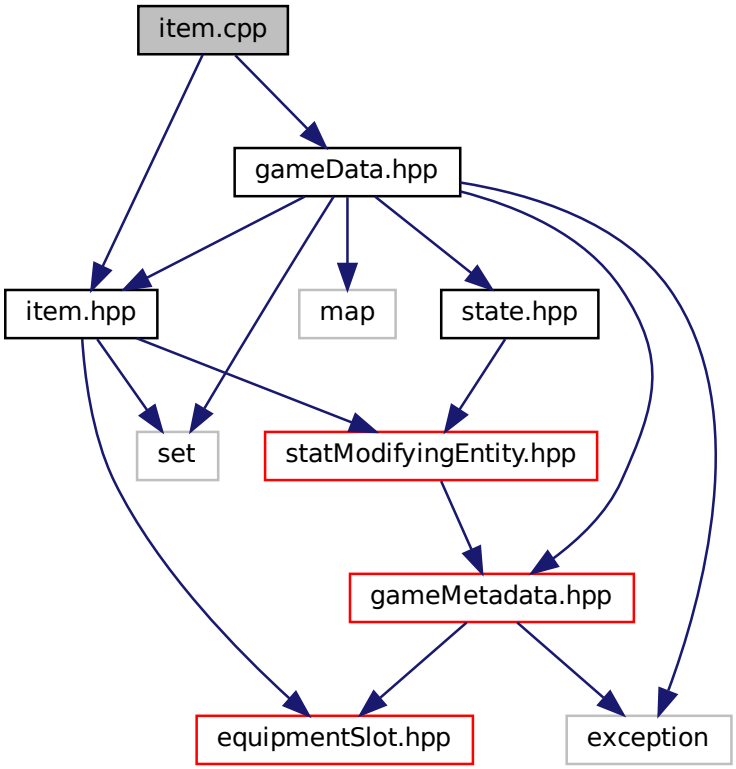
GameMetadata interface.

## 6.19 item.cpp File Reference

Item implementation.

```
#include "item.hpp"
#include "gameData.hpp"
```

Include dependency graph for item.cpp:




### 6.19.1 Detailed Description

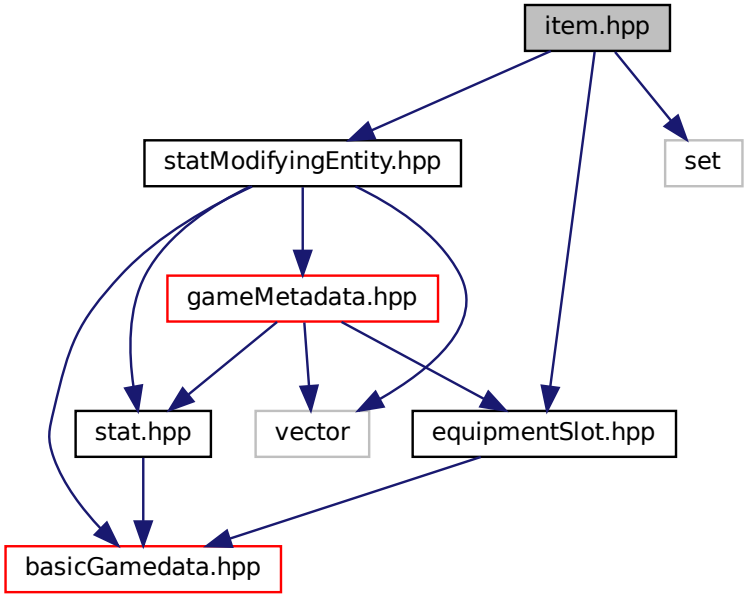
Item implementation.

## 6.20 item.hpp File Reference

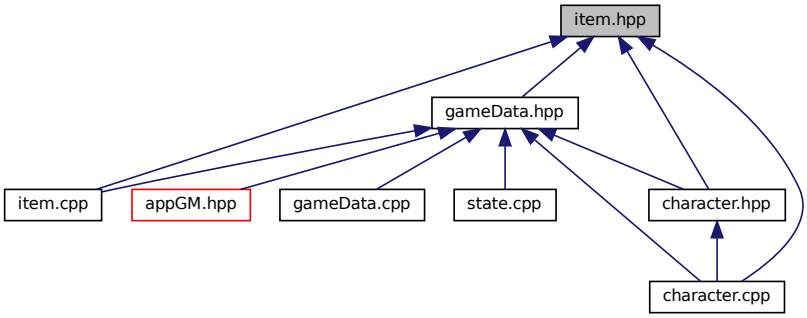
Item interface.

```
#include "equipmentSlot.hpp"
#include "statModifyingEntity.hpp"
#include <set>
```

Include dependency graph for item.hpp:




This graph shows which files directly or indirectly include this file:




### Classes

- class Item

  *Represents an item in game.*

### 6.20.1 Detailed Description

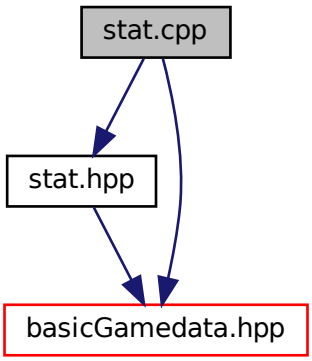
Item interface.

## 6.21 stat.cpp File Reference

Stat implementation.

```
#include "stat.hpp"
#include "basicGamedata.hpp"
```

Include dependency graph for stat.cpp:




### 6.21.1 Detailed Description

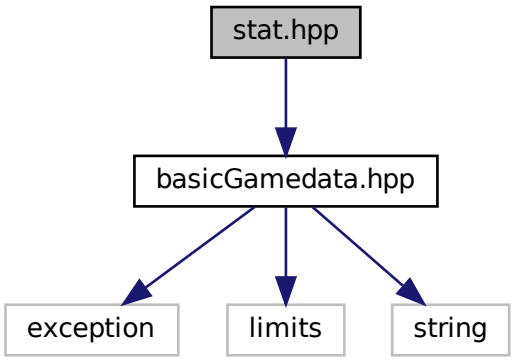
Stat implementation.

## 6.22 stat.hpp File Reference

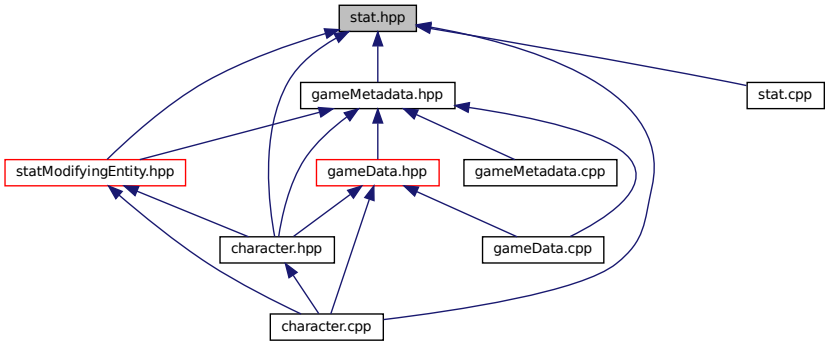
Stat interface.

```
#include "basicGamedata.hpp"
```

Include dependency graph for stat.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class Stat

  *Statistics.*

### 6.22.1 Detailed Description

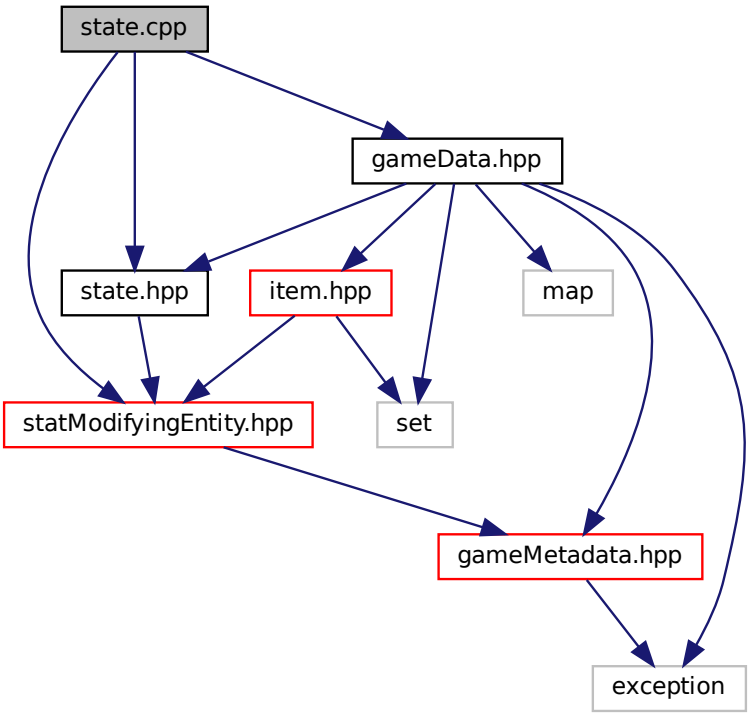
Stat interface.

## 6.23 state.cpp File Reference

State implementation.

```
#include "state.hpp"
#include "gameData.hpp"
#include "statModifyingEntity.hpp"
```

Include dependency graph for state.cpp:




### 6.23.1 Detailed Description

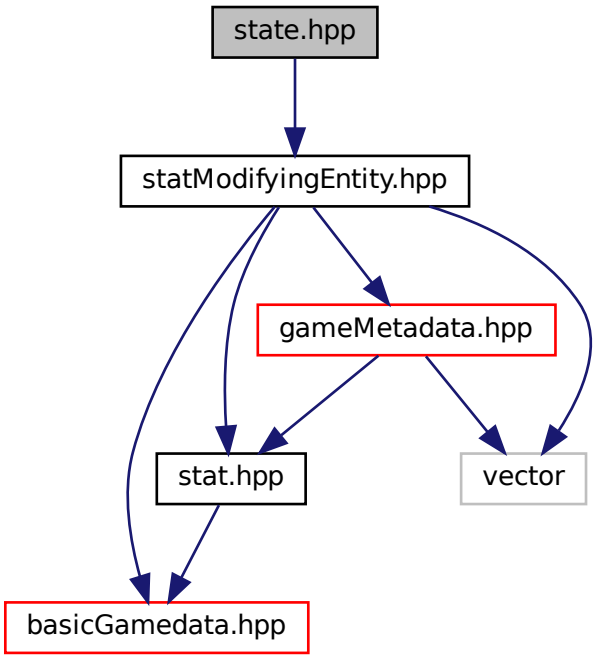
State implementation.

## 6.24 state.hpp File Reference

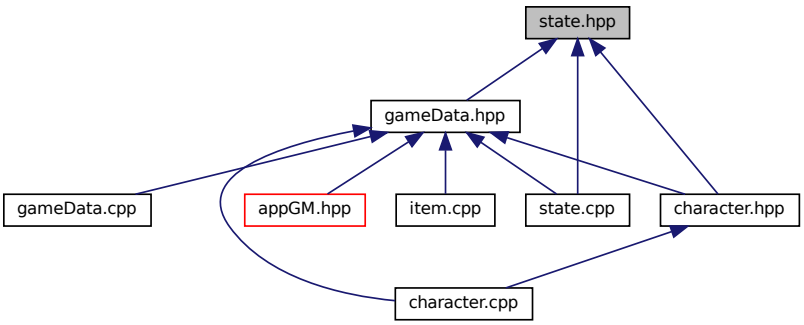
State interface.

```
#include "statModifyingEntity.hpp"
```

Include dependency graph for state.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class State

  *Reperesents State in game.*

### 6.24.1 Detailed Description

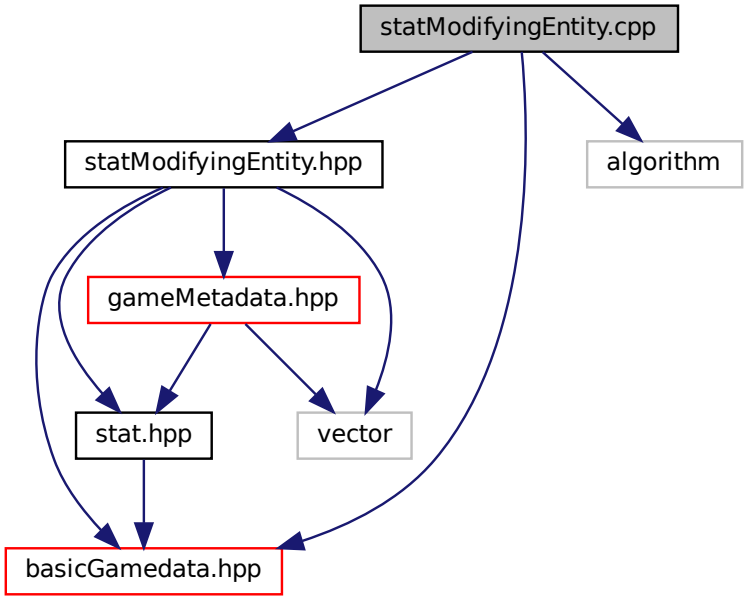
State interface.

## 6.25 statModifyingEntity.cpp File Reference

StatModifyingEntity implementation.

```
#include "statModifyingEntity.hpp"
#include "basicGamedata.hpp"
#include <algorithm>
```

Include dependency graph for statModifyingEntity.cpp:




### 6.25.1 Detailed Description

StatModifyingEntity implementation.

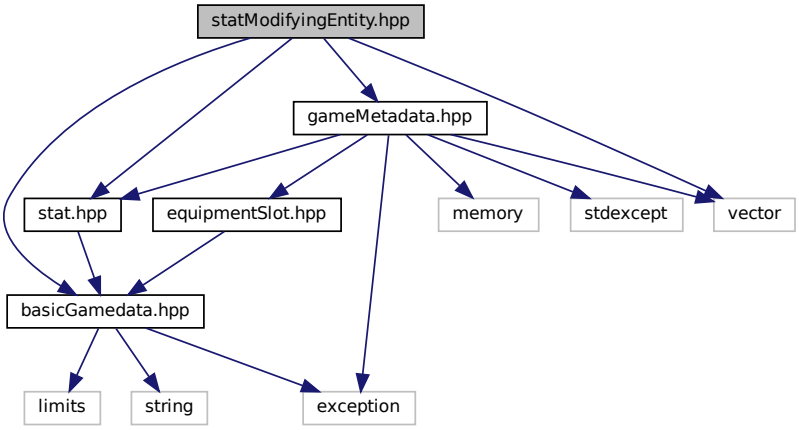
## 6.26 statModifyingEntity.hpp File Reference

StatModifyingEntity interface.

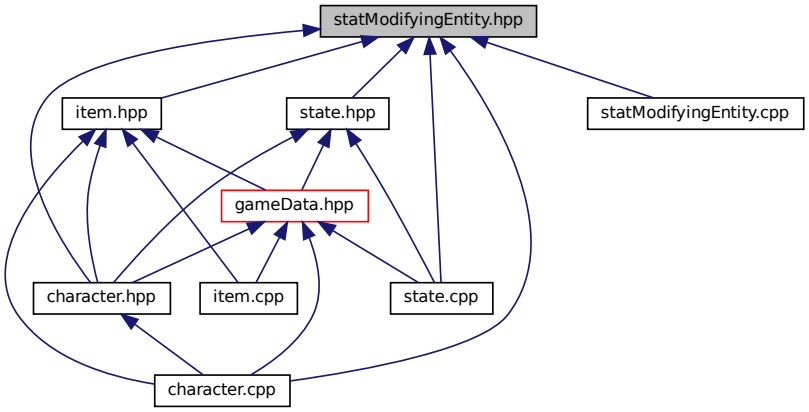
```
#include "basicGamedata.hpp"
#include "gameMetadata.hpp"
#include "stat.hpp"
```

```
#include <vector>
```

Include dependency graph for statModifyingEntity.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class exceptionInvalidGameMetadata

  *Invalid game data.*
- class StatModifyingEntity

  *Repesents collection of stat modifiers.*

### 6.26.1 Detailed Description

StatModifyingEntity interface.

# Index