DarkRift

LoginPlugin Manual

Introduction

The LoginPlugin gives a basic outline of creating a login system for your game. It allows the user to Login, Logout or Create an account and exposes a variety of events from your plugins and games to interface with. It also highlights some of the key new features in DarkRift 1.2: the database integration and the new player data systems on the ConnectionServices.

To install, drag the LoginPlugin.dll file into the Plugins directory of your DarkRift server. If you want to change the tags and subjects used by the plugin then (once the server has been run) you will find a subdirectory in the Plugins directory names “LoginPlugin” inside here will be a settings.cnf file containing the configuration and other options. If you do not have a Plugins directory run the server and it will be created.

You will need a database plugin for DarkRift installed as well and a database setup to store the logins in.

**If you don’t have DarkRift it is available from:**<http://u3d.as/7eY>

Reference (Server)

# LoginPlugin : Plugin

This is the server plugin and the part which runs all the database queries and manages the logins.

# Variables

**public** **override** **string** *name;*

*Desc:*

Returns the name of this plugin for use when getting data or in other references.

**public** **override** **string** *version;*

*Desc:*

Returns the version of this plugin.

**public** **override** Command[] *commands*

*Desc:*

Returns the commands this server uses.

**public** **override** **string** *author;*

*Desc:*

Returns the author of this plugin (me!).

**public** **override** **string** *supportEmail*

*Desc:*

Returns the support email for this plugin.

# Delegates

**public** **delegate** **void** SuccessfulLoginEventHandler(**int** userID, ConnectionService con);

*Desc:*

Used in the onSuccessfulLogin event.

**public** **delegate** **void** UnsuccessfulLoginEventHandler(ConnectionService con);

*Desc:*

Used in the onUnsuccessful Login event.

**public** **delegate** **void** LogoutEventHandler(**int** userID, ConnectionService con);

*Desc:*

Used in the onLogout event.

**public** **delegate** **void** SuccessfulAddUserEventHandler(**int** userID, **string** username, ConnectionService con);

*Desc:*

Used in the onAddUser event.

**public** **delegate** **void** UnsuccessfulAddUserEventHandler(**string** username, ConnectionService con);

*Desc:*

Used in the onAddUserFailed event.

# Events

**public** **event** SuccessfulLoginEventHandler *onSuccessfulLogin*;

*Desc: Occurs when a login is sucessful*

Called when a person logs in successfully.

**public** **event** UnsuccessfulLoginEventHandler *onUnsucessfulLogin*;

*Desc: Occurs when a login is unsucessful.*

Called when someone tries to login but fails. Probably because the user’s username or password was wrong but could be due to an error.

**public** **event** LogoutEventHandler *onLogout*;

*Desc: Occurs when a player logs out.*

Called when a player successfully logs out.

**public** **event** SuccessfulAddUserEventHandler *onAddUser*;

*Desc: Occurs when a player is added remotely.*

Called when a user is successfully created.

**public** **event** UnsuccessfulAddUserEventHandler *onAddUserFailed*;          *Desc: Occurs when a new player wasn't able to be added.*

Called when a user wasn’t created successfully. Probably because the username is already taken but could be due to as error.

# Constructors

**public** LoginPlugin ()

*Desc:*

Dont call this... It’s not sensible...

Reference (Unity) (DarkRift.LoginPlugin)

# LoginManager

The LoginManager is the only class in the plugin and does everything.

# Data

This is a dictionary of the data stored on each ConnectionService using SetData.

**IsLoggedIn**

This will be true if the client on this ConnectionService is logged in.

**UserID**

This will be set to the UserID of the client on this ConnectionService if they are logged in.

# Variables

**public** **static** **int** *userID*

*Desc: The user ID when logged in.*

When logged in this refers to the player’s user ID.

**public** **static** **bool** *isLoggedIn*

*Desc: Are we logged in to a server?*

True if you’re logged in.

**public** **static** DarkRiftConnection *connection*;

*Desc: The connection to transmit over, if null DarkRiftAPI will be used instead.*

If you’re using a DarkRiftConnection then set this to it, if you want to use DarkRiftAPI then leave this null.

**public** **static** HashType *hashType* = HashType.*MD5*;

*Desc: The hash algorithm to use, don't change this once records are set.*

This is the hashing algorithm that will be used to obscure the password for security before it is sent. As the password is stored in the database hashed you should not change this once you have users created or they won’t be able to log on!

# Delegates

**public** **delegate** **void** SuccessfulLoginEventHandler(**int** userID);

*Desc:*

This is used in the onSuccessfulLogin event.

**public** **delegate** **void** UnsuccessfulLoginEventHandler();

*Desc:*

This is used in the onUnsuccessfulLogin event.

**public** **delegate** **void** LogoutEventHandler();

*Desc:*

This is used in the onLogout event.

**public** **delegate** **void** SuccessfulAddUserEventHandler(**int** userID);

*Desc:*

This is used in the onAddUser event.

**public** **delegate** **void** UnsuccessfulAddUserEventHandler();

*Desc:*

This is used in the onAddUserFailed envent.

# Events

**public** **event** SuccessfulLoginEventHandler *onSuccessfulLogin*;

*Desc: Occurs when a login is sucessful*

Called when a person logs in successfully.

**public** **event** UnsuccessfulLoginEventHandler *onUnsucessfulLogin*;

*Desc: Occurs when a login is unsucessful.*

Called when someone tries to login but fails. Probably because the user’s username or password was wrong but could be due to an error.

**public** **event** LogoutEventHandler *onLogout*;

*Desc: Occurs when a player logs out.*

Called when a player successfully logs out.

**public** **event** SuccessfulAddUserEventHandler *onAddUser*;

*Desc: Occurs when a player is added remotely.*

Called when a user is successfully created.

**public** **event** UnsuccessfulAddUserEventHandler *onAddUserFailed*;          *Desc: Occurs when a new player wasn't able to be added.*

Called when a user wasn’t created successfully. Probably because the username is already taken but could be due to as error.

# Methods

**public** **static** **void** *Login*(**string** username, **string** password)

*Desc: Login with the specified username and password.*

Tries to login using the specified username and password. onSuccessfulLogin or onUnsuccessfulLogin will be triggered when depending on the result confirmation is received.

**public** **static** **void** *Logout*()

*Desc: Logs out the user from the server.*

Tries to logout. onLogout will be called when confirmation is received.

**public** **static** **void** *AddUser*(**string** username, **string** password)

Desc: Asks the server to add a new user to the database.

Tries to create a new user in the database with the username and password supplied. onAddUser or onAddUserFailed will be triggered when confirmation is received depending on the result.