DarkRift

MySQLConnector Manual

Introduction

The MySQLConnector allows DarkRift plugins to talk to a MySQL database through the standard, database independent interface on the server. This allows users to use or change to different databases without having to change code in their plugins.

Your plugins can easily access the database plugin via DarkRiftServer.database.

To install simply copy the MySQLConnector.dll file and the Lib folder into the Plugins directory of the server and run the server. A MySQLConnector subdirectory will be generated and a settings.cnf file will be created allowing you to configure the plugin to your database.

If you haven't got DarkRift then it's available here:
http://u3d.as/7eY

MySQL is an Oracle product and I do not own the rights to it, I have included the licence and readme for MySQL inside the Lib folder with the MySQL libraries that are required by the plugin. If there is any reason that I should not have included these libraries in this package then I will happily remove them and provide a link to where they can be downloaded instead.

Reference

Connector : DarkRift.Storage.Database

This is the main class of the MySQL connector and is where queries can be made from.

Variables

```
public override string name
```

Desc:

The name of this plugin for use when referencing it.

```
public override string version
```

Desc:

The version of this plugin.

```
public override Command[] commands
```

Desc:

The commands used by this plugin.

```
public override string author
```

Desc:

The author of the plugin (me!).

```
public override string supportEmail
```

Desc:

The support email for this plugin.

Constructors

```
public Connector()
```

Desc:

Don’t call this... Bad things will happen...

Methods

```
public override void ExecuteNonQuery (string query, QueryParameter[] parameters)
```

Desc: Execute a query on the server with no return values.

Executes a query on the database with no returns.

```
public override object ExecuteScalar(string query,        QueryParameter[] parameters)
```

Desc: Executes a query on the database with a scalar return.

Executes a database query with a scalar return value.

```
public override DatabaseRow[] ExecuteQuery (string query, QueryParameter[] parameters)
```

Desc: Executes a query on the database returning an array of rows.

Executes a query on the database that returns rows. The rows will be returned as an array of DatabaseRows.

```
public override string EscapeString(string s)
```

Desc: Removes any characters that could allow SQL injection.

Removes any characters from the string that could allow SQL injection and returns the string. It's generally better to use QueryParameters though.

```
public override void Dispose()
```

Desc: Releases all resource used by the object.

This will do nothing.

public void SendMessageToID(ushort targetID, byte tag, ushort subject, object data)

Desc: Sends data, tag and subject to a specific ID.

Sends a message to the client with the specified ID with tag, subject and data.

public void SendMessageToAll(byte tag, ushort subject, object data)

Desc: Sends data, tag and subject to all clients and the server.

Sends a message everyone (including the server) with tag, subject and data.

public void SendMessageToOthers(byte tag, ushort subject, object data)

Desc: Sends data, tag and subject to all other clients.

Sends a message everyone but the sender (and the server) with tag, subject and data.

public void Disconnect()

Desc: Disconnect from the server.

Disconnects form the specified server, this should always be called in OnApplicationQuit() to stop the server trying to sending data to it.

Delegates

public delegate void DataEvent(byte tag, ushort subject, object data);

Desc:

This is used in the onData event, it is used for basic data transmission.

public delegate void DetailedDataEvent(ushort sender, byte tag, ushort subject, object data);

Desc:

This is used in the onDataDetailed event, it's similar to the DataEvent delegate but also passes the sender's ID

public delegate void ConnectionEvent(ushort id);

Desc:

This is used in the onPlayerDisconnected event to give details about which ID disconnected. It will also be used in the onPlayerConnected event [future].

Events

public event DataEvent onData;

Desc: Occurs when data is recieved but only gives tag, subject and data.

This event is fired when data is received; it passes the tag, subject and data to the function.

public event DetailedDataEvent onDataDetailed;

Desc: Occurs when data is recieved but also passes the sender ID.

Like onData this is called when data received but also passes the ID of the sending client.

public event ConnectionEvent onPlayerDisconnected;

Desc: Occurs when a player has disconnected.

This is called when a player disconnects from the server allowing your game to remove their objects/data/etc; you will be passed their ID.

DarkRiftAPI.DarkRiftAPI

This is the static way to connect to servers, this can easily be called from any script. This should really be used by default.

Variables

public static bool isConnected

Desc: Are we connected to a server?

Determines if the API has connected to a server or not, use Connect() to connect.

Methods

public static bool Connect(string ip)

Desc: Connect to the specified IP.

Tries to connect to the sever at IP on port 4296 (the default) returning true if sucessful or false if not.

public static bool Connect(string ip, int port)

Desc: Connect to the specified I through the specified port.

Tries to connect to the sever at IP on the specified port returning true if successful or false if not.

public static void Recieve()

Desc: Processes all the data received and fires the events.

This is the function that processes, decodes and distributes any messages received from the server. Usually this would be called from the Update routine or from the DarkRiftReciever.cs script but if you need it, it's here.

public static void SendMessageToServer(byte tag, ushort subject, object data)

Desc: Sends data, tag and subject to the server.

Sends a message to the server only with tag, subject and data. This is used to talk to plugins on the server.

public static void SendMessageToID(ushort targetID, byte tag, ushort subject, object data)

Desc: Sends data, tag and subject to a specific ID.

Sends a message to the client with the specified ID with tag, subject and data.

public static void SendMessageToAll(byte tag, ushort subject, object data)

Desc: Sends data, tag and subject to all clients and the server.

Sends a message everyone (including the server) with tag, subject and data.

public static void SendMessageToOthers(byte tag, ushort subject, object data)

Desc: Sends data, tag and subject to all other clients.

Sends a message everyone but the sender (and the server) with tag, subject and data.

public static void Disconnect()

Desc: Disconnect from the server.

Disconnects form the specified server, this should always be called in OnApplicationQuit() to stop the server trying to sending data to it.

Delegates

public delegate void DataEvent(byte tag, ushort subject, object data);

Desc:

This is used in the onData event, it is used for basic data transmission.

public delegate void DetailedDataEvent(ushort sender, byte tag, ushort subject, object data);

Desc:

This is used in the onDataDetailed event, it's similar to the DataEvent delegate but also passes the sender's ID

public delegate void ConnectionEvent(ushort id);

Desc:

This is used in the onPlayerDisconnected event to give details about which ID disconnected. It will also be used in the onPlayerConnected event [future].

Events

public static event DataEvent onData;

Desc: Occurs when data is recieved but only gives tag, subject and data.

This event is fired when data is received; it passes the tag, subject and data to the function.

public static event DetailedDataEvent onDataDetailed;

Desc: Occurs when data is recieved but also passes the sender ID.

Like onData this is called when data received but also passes the ID of the sending client.

public static event ConnectionEvent onPlayerDisconnected;

Desc: Occurs when a player has disconnected.

This is called when a player disconnects from the server allowing your game to remove their objects/data/etc; you will be passed their ID.

Exceptions

ConnectionFailedException

The server couldn't be connected to, possibly because of a wrong IP or the server rejected us. See details in the error for more information.

NotConnectedException

You're trying to do something that requires you to be connected but you're not connected.

InvalidDataException

Data can't be sent because it was invalid. Usually because you are using 255 as a tag (it's reserved for inside use).