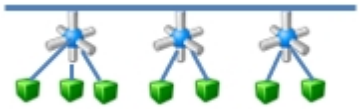
CrystalEngineAPI_ru

Created with the Freeware Edition of HelpNDoc: Full featured Help generator

Общее описание

Crystal Engine

Crystal Engine представляет собой платформу для создания серверных решений для любых интернет проектов.
С помощью визуальных редакторов приложения ProjectBuilder разработчик вводит объекты и формирует структуру серверной системы. Затем платформа компилирует серверное решение непосредственно для данного проекта.
Если сказать по простому - то CrystalEngine это компилятор, который компилирует серверный движок для каждого проекта индивидуально с учетом всех его особенностей.
На выходе получается 2 библиотеки (ServerEngine.dll и ClientEngine.dll)



Серверное решение состоит из объединенных в пиринговую (одноранговая) сеть рабочих серверов.
К каждому рабочему или PeerServer подключаются модули логики, в которых описаны функции обработки данных и операций надо объектами серверной системы.
PeerServer - это обычное windows приложение, работающее в среде .NET
LogicModule - это сборка (библиотека *.dll) .NET
Модуль логики может подключаться и отключаться от Peer сервера динамически.
К модулю логики подключается библиотека ServerEngine.dll

Хранение данных

В качестве постоянного хранилища используется база SQL или файловое хранилище.
В качестве оперативного хранилища выступает NoSQl распределенное хранилище данных, основанное на парах ключ-значение, собственной разработки. **SQL используется только для долговременного хранения данных**

Транспортная система

В качестве внутренней транспортной системы серверного решения используется динамический пул однонаправленных TCP connections. При увеличении нагрузки автоматически добавляются подключения, при уменьшении - закрываются. Данное решение используется повсеместно в ядре ОС Windows.

Обмен данными

Обмен данными производится посредством передачи команд.
Каждый модуль логики подписывается на получение определенных команд, которые он обрабатывает.
И сам может посылать команды другому модулю логики либо приложению - клиента.

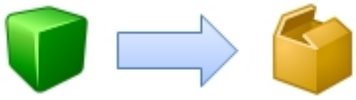
Например
отправка команды **Root**.SendCommand(**UIN** uin, **IGCommand** com);
подписывание на получение **ComLogin**.AddEvent(onLogin);
отписывание от получения **ComLogin**.RemoveEvent(onLogin);

Важно! Обязательно должно быть удаление подписки при отключении модуля.
Так, как модули подключаются и отключаются динамически, без отключения сервера.

Created with the Freeware Edition of HelpNDoc: Full featured Help generator

Модуль логики

Одним из основных элементов серверной системы является модуль логики. Он представляет собой обычную .NET сборку (*.dll) и компилируется на стороне разработчика. Затем через приложение AdminCP подключается в серверную систему. В модуле сосредоточены методы, реализующие логику проекта.
Для простоты восприятия можно представить модуль - пустой коробкой, в которую кладется та или иная ДЛЛ сборка. Привязка ДЛЛ к модулю логики производится разработчиком проекта.



Порядок создания модуля

1. Создать проект библиотеки классов в Visual Studio
2. Запустить приложение ProjectBuilder и нажать там кнопку "Build"
3. Подключить библиотеку ..ServerEngine.dll из вкладки ".NET" VisualStudio
4. Установить класс модуля как наследник интерфейса GServer.IServerModule
5. Создать 2 функции Start() и Stop() в которых произвести подключение и отключение команд, на которые данный модуль будет реагировать
6. Прописать код подключения и отключения от команд
7. Написать код реализующий логику
8. Скомпилировать сборку
9. Подключить сборку в соответствующий модуль
10. Через AdminCP запустить модуль на сервер

Ниже приведен пример простого модуля логики

```
using System;
using GServer;

namespace TestModule1 {

  public class Module1 : IServerModule {

    public void Start() {
      try {
        ComLogin.AddEvent(onLogin);
      }catch(Exception e) {
        GError.AddError(e);
      }
    }

    public void Stop() {
      try {
        ComLogin.RemoveEvent(onLogin);
      }catch(Exception e) {
        GError.AddError(e);
      }
    }

    private void onLogin(ComLogin com) {
      try {
        UIN id = Root.SQL.CheckLogin.Execute(com.Login, com.Password);
        if(id!=UIN.EmptyUin) {
          Root.Users[id].Logined = true;
          com.Client.SendCommand(new ComLoginAnswer(){Result = "OK"});
```

```
        }else {
          com.Client.SendCommand(new ComLoginAnswer() { Result = "NO" });
        }
      }catch(Exception e) {
        GError.AddError(e);
      }
    }

  }
}
```

Команды

Команды

Передача информации в серверной системе осуществляется посредством команд.
Модули логики общаются друг с другом и с внешними клиентами через команды. Поэтому на создание структуры команд нами было уделено много времени.
Команды представляют собой структуры данных различных типов. Описание структур команд есть на стороне сервера и на стороне клиента - поэтому в пакетах передаются только значения переменных и минимум служебной информации. Упаковка и распаковка команд происходит очень быстро, основываясь на собственном методе, построенном на указателях.
Помимо структуры данных команда содержит делегат срабатывания.
На команду можно подписаться и отписаться.
При подписывании на команду можно указать UIN-адрес и команда сработает только если придет пакет с таким адресом. Это позволяет отправлять и получать персонифицированые команды.
Например в модуле можно подписаться на получение команды для объекта A (aID)

```
ComLogin.AddEvent(onLogin,aID);
```

модуль B должен послать команду для объекта A

```
Root.SendCommand(aID, new ComLoginAnswer(){Login = "Test", Password = "Test"});
```

при этом эту команду получат только те модули, которые подписались на нее именно с таким ID.

Можно послать команду без адреса и ее получат ВСЕ, кто подписался на эту команду.

```
ComLogin.AddEvent(onLogin); - подписывание на получение
ComLogin.RemoveEvent(onLogin); - отписывание от получения

Root.SendCommand(new ComLoginAnswer(){Login = "Test", Password = "Test"}); - отправка команду
```

Важно! Обязательно должно быть удаление подписки на команду при отключении модуля. Так, как модули подключаются и отключаются динамически, без отключения сервера.

Created with the Freeware Edition of HelpNDoc: Full featured Help generator

Работа с SQL

Создание запросов к базе SQL осуществляется в редакторе ProjectBuilder. Тамже происходит тестирование на работоспособность. На основе проверенного запроса генерируется SQL сущность (класс) для каждого запроса и в коде разработчик оперирует уже готовым классом и структурой данных.
Доступ к классам запросов происходит через статический класс **Root.SQL.**название класса
Например

```
UIN id = Root.SQL.CheckLogin.Execute(com.Login, com.Password);
```

данная сущность производит выборку и возвращает одно значение.
Сущности могут возвращать одно поле, несколько полей, либо несколько записей

Created with the Freeware Edition of HelpNDoc: Full featured Help generator

Server Engine

Root

Статический класс Root

Класс **Root** представляет собой статический класс для доступа ко всем объектам серверной системы, отправки командб и доступа к SQL сущностям.
Все объекты, которые вводятся через ProjectBuilder доступны через класс **Root**

Например доступ к объекту User: **Root**.Users[UserID]

Отправка команд

- **Root**.SendCommand(**UIN** uin, **IGCommand** com);
  где uin - идентификатор объекта или клиента, которому необходимо отправить команду.
  com - непосредственно экземпляр класса команды.
  Например: **Root**.SendCommand(ClientID, **new** **ComLogin**{Login = "Test",Password = "Test"});
  Root.SendCommand(**IGCommand** com);
  отправляет команду всем, кто подписан на ее получение во всей серверной системе
  Root.SendAllUserCommand(**IGCommand** com);
  отправляет команду всем подключенным клиентам TCP сервера данного пира
  byte[] **Root**.SendRequest(**UIN** uin, **IGRequest** req);
  отправляет запрос и дожидается ответа

Досступ к SQL

Root.SQL Доступ к SQL осуществляется через статический класс.

UIN

Класс UIN

Класс **UIN** реализует 16 байтный Уникальный Идентификатор серверного решения (аналог Guid с дополнениями)

Свойство возвращает Guid
```
    public Guid AsGuid;
```

Свойство возвращает пустой UIN 00000000-0000-0000-0000-000000000000
```
    public static UIN EmptyUin;
```

Свойство возвращает новый уникальный UIN например 05C7A3AE-E2AE-48CB-A3A6-2EB67F90EBA1
```
    public static UIN NewUin;
```

Конструктор с параметром Guid
```
    public UIN(Guid value);
```

Конструктор с параметром **byte**[] - получает (десериализует) UIN из 16 байтов
```
    public UIN(byte[] value)
```

Возможность присвоения **byte**[] = **UIN** (UIN преобразуется в массив байтов 16 байт)

Возможность присвоения **string** = **UIN** (UIN преобразуется в строку вида 05C7A3AE-E2AE-48CB-A3A6-2EB67F90EBA1)

Created with the Freeware Edition of HelpNDoc: Full featured Help generator

Возможность присвоения **Guid** = **UIN** (UIN преобразуется в Guid)

Возможность присвоения **UIN** = **Guid** (Guid преобразуется в UIN)

Сложение двух UIN **UIN** = **UIN** + **UIN** (сумма на основе XOR)

Сравнение двух UIN **UIN** == **UIN** (на основе сравнения Guid)

Загрузка из строки вида 05C7A3AE-E2AE-48CB-A3A6-2EB67F90EBA1

```
    public static UIN LoadFromString(string s);
```

Загрузка из строки массива байтов 16

```
    public static UIN LoadFromBytes(byte[] b);
```

Получение UIN как MD5 хеш файла

```
    public static UIN MD5File(string fileName);
```

Получение UIN как MD5 хеш строки

```
    public static UIN MD5String(string value);
```

Получение UIN как MD5 хеш массива байтов

```
    public static UIN MD5Array(byte[] value);
```

Получение строки вида 05C7A3AE-E2AE-48CB-A3A6-2EB67F90EBA1

```
    public string ToString();
```

Получение строки вида 05C7A3AEE2AE48CBA3A62EB67F90EBA1

```
    public string ToHexString();
```

Получение массива байтов **byte**[16]

```
    public byte[] ToArray();
```

Convert

Статический класс GConvert

Класс **GConvert** представляет собой набор функций быстрой конвертации данных

Функции получения массива байтов из типов данных

```
    public static byte[] GetBytes(object value);
    public static byte[] GetBytes(string value);
    public static byte[] GetBytes(UIN value);
    public static byte[] GetBytes(Guid value);
    public static byte[] GetBytes(int value);
    public static byte[] GetBytes(uint value);
    public static byte[] GetBytes(short value);
    public static byte[] GetBytes(ushort value);
    public static byte[] GetBytes(long value);
    public static byte[] GetBytes(ulong value);
    public static byte[] GetBytes(byte value);
    public static byte[] GetBytes(sbyte value);
    public static byte[] GetBytes(bool value);
    public static byte[] GetBytes(float value);
    public static byte[] GetBytes(double value);
    public static byte[] GetBytes(DateTime value);
```

Функция получения значения определенного типа из массива байтов

```
    public static T ToObject<T>(byte[] data);
```

Функция получения значения определенного типа (пустого) из массива байтов

Created with the Freeware Edition of HelpNDoc: Full featured Help generator

например при ToDefault<string>() == "" пустая строка
```
    public static T ToDefault<T>();
```

Функция получения значения определенного типа из массива байтов
```
    public static string ToString(byte[] data);
    public static bool ToBool(byte[] data);
    public static int ToInt32(byte[] data);
    public static uint ToUInt32(byte[] data);
    public static long ToInt64(byte[] data);
    public static ulong ToUInt64(byte[] data);
    public static short ToInt16(byte[] data);
    public static ushort ToUInt16(byte[] data);
    public static byte ToByte(byte[] data);
    public static sbyte ToSByte(byte[] data);
    public static float ToFloat(byte[] data);
    public static double ToDouble(byte[] data);
    public static DateTime ToDateTime(byte[] data);
    public static UIN ToUIN(byte[] data);
    public static UIN ToUIN(byte[] data, int pos);
```

IOClasses

Класс GBufferReader

Класс **GBufferReader** реализует чтение из буфера байтов (аналог BinaryReader но в 10 раз быстрее).

Конструктор для чтения из массива байтов
```
    public GBufferReader(byte[] buf)
```

Конструктор для чтения из файла
```
    public GBufferReader(string fileName)
```

Чтение указанного типа данных (только определенных ниже типов)
```
    public T Read<T>();
```

Чтение определенного типа данных
```
    public bool ReadBoolean();
    public int ReadInt32();
    public short ReadInt16();
    public long ReadInt64();
    public byte ReadByte();
    public sbyte ReadSByte();
    public uint ReadUInt32();
    public ushort ReadUInt16();
    public ulong ReadUInt64();
    public float ReadSingle();
    public double ReadDouble();
    public string ReadString();
    public byte[] ReadBytes();
    public int[] ReadIntArray();
    public UIN ReadUIN();
    public Guid ReadGuid();
    public DateTime ReadDateTime();
```

Пример использования

```
 public GObject Deserialize(byte[] data){
   using (GBufferReader br = new GBufferReader(data)) {
     GObject obj = new GObject();
```

Created with the Freeware Edition of HelpNDoc: Full featured Help generator

```
      obj.ClassUin = br.ReadUIN();
      obj.Name = br.ReadString();
      obj.TypeID = br.ReadInt32();
      obj.Global = br.ReadBoolean();
      return obj;
    }
  }
```

GBufferWriter

Класс GBufferWriter

Класс **GBufferWriter** реализует запись в буфер байтов (аналог BinaryWriter но в 10 раз быстрее).

Запись объекта (только определенных ниже типов)

```
    public void Write(object value);
```

Запись определенного типа значения в буфер байтов

```
    public void Write(bool n);
    public void Write(int n);
    public void Write(long n);
    public void Write(short n);
    public void Write(byte n);
    public void Write(uint n);
    public void Write(ulong n);
    public void Write(ushort n);
    public void Write(sbyte n);
    public void Write(float n);
    public void Write(double n);
    public void Write(byte[] n);
    public void Write(int[] n);
    public void Write(string n);
    public void Write(UIN n);
    public void Write(Guid n);
    public void Write(DateTime n);
```

Получение записанного буфера байтов

```
    public byte[] ToArray();
```

Пример использования

```
  public byte[] Serialize(GObject obj){
    using (GBufferWriter bw = new GBufferWriter()) {
      bw.Write(obj.ClassUin);
      bw.Write(obj.Name);
      bw.Write(obj.TypeID);
      bw.Write(obj.Global);
      return bw.ToArray();
    }
  }
```

GError

Класс GError

Класс **GError** реализует отправку ошибок на AdminCP

Created with the Freeware Edition of HelpNDoc: Full featured Help generator

Отправка ошибки строкой
```
  public static void AddError(string err);
```

Отправка ошибки класса Exception
```
  public static void AddError(Exception ee);
```

Пример

```
    private static void onBuildProject(ComProjectBuild com) {
      try {
        Code....
      } catch (Exception ee) {
        GError.AddError(ee);
      }
    }
```

Возможно самому отлавливать все ошибки в своем коде, добавив обработчик на делегат
```
OnErrorEvent(string err);
```

Пример

```
GError.OnError = onError;

private void onError(string err) {
  myErrors += err;
}
```

IGConnection

Класс интерфейса IGConnection

Класс IGConnection реализует сокет - клиента TCP или UDP

```
public interface IGConnection {
    object User { get; set; } - Свободное поле заполняется разработчиком
    object Tag { get; set; } - Свободное поле заполняется разработчиком
    UIN ID { get; set; } - Возвращает ID подключения
    UIN UserID { get; set; } - Свободное поле заполняется разработчиком
    UIN ObjectID { get; set; } - Свободное поле заполняется разработчиком
    bool Logined { get; set; } - Свободное поле заполняется разработчиком
    int ConnectTick { get; set; }
    string Name { get; set; } - Свободное поле заполняется разработчиком
    int Status { get; set; } - Свободное поле заполняется разработчиком
    int Level { get; set; } - Свободное поле заполняется разработчиком
    string PeerIP { get; } - Возвращает IP адрес пира
    int PeerPort { get; } - Возвращает Port пира
    string RemoteIP { get; } - Возвращает IP адрес клиента подключения
    int RemotePort { get; } - Возвращает Port адрес клиента подключения
    GClientType ClientType { get; } - Возвращает Тип подключенияIP адрес клиента
подключения
                                        enum GClientType { ClientNone, ClientTcp,
ClientUdp }
    void Close(); - Закрывает подключение
    void SendCommand(IGCommand com); - Отправляет команду
    void SendPacket(GCommandPacket p); - Отправляет внешний пакет команд
    void SendPacket(); - Отправляет внутренний пакет команд
    void AddPacket(IGCommand com); - Добавляет команду во внутренний пакет команд
  }
```

Client Engine

GTCPClient

Класс GTcpClient

Класс **GTCPClient** реализует TCP сокет-клиент для подключения к серверу

подключение производится через функцию

```
public void Connect(string addr, int port);
```

Пример реализации клиента

```
using System;
using System.Windows.Forms;
using GServer;

namespace TestModuleClient {
  public partial class Form1 : Form {

    GTCPClient client = new GTCPClient();


    public Form1() {
      InitializeComponent();
//подключение к событиям коннект \ дисконнект
      ComSocketConnect.AddEvent(onConnect);
      ComSocketDisconnect.AddEvent(onDisconnect);
    }


    private void onConnect(ComSocketConnect com) {
//все функции, которые обращаются к визуальным компонентам,
  требуется синхронизировать с основным потоком программы
      Invoke((Action) delegate {
        label1.Text = @"Connected";
        label3.Text = client.PeerIP + @":" + client.PeerPort;
      });
    }

    private void onDisconnect(ComSocketDisconnect com) {
      try {
        Invoke((Action) delegate { label1.Text = @"Disconnected"; });
      }catch(Exception) {

      }
    }

    private void button1_Click(object sender, EventArgs e) {
      client.Connect(ipEdit.Text, Int32.Parse(portEdit.Text));//@"192.168.0.100", 9985);
    }

    private void button2_Click(object sender, EventArgs e) {
      client.Close();
    }

    private void button3_Click(object sender, EventArgs e) {
      client.SendCommand(new ComLogin { Login = textBox2.Text, Password = textBox1.Text
```

```
});
    }
```

GTCPClientUnity

Класс GTcpClientUnity

Класс **GTCPCLientUnity** реализует TCP сокет-клиент для подключения к серверу из клиента на движке Unity3D

все аналогично обычному клиенту за исключением того, что для Пингов требуется подключение внешнего таймера
Внешний таймер требуется подключить к функции время 1000 мс

```
public void OnPingTimer();
```