**杭州风特陪玩项目框架文档说明**

1. 文档说明……………………………………………………………………………1

二、框架核心功能说明…………………………………………………………………2

2.1、如何添加仓储[？](file:///G:\文档\Differ%20BS框架开发文档.docx#_、如果添加完数据表后同步完成领域、DTO、仓储？)

2.2、[如何在Service层调用仓储？](file:///G:\\文档\\Differ%20BS框架开发文档.docx" \l "_、当仓储基类达不到业务要求时，如何添加在Domain添加仓储接口及仓储)

2.3、[如何注册对象（依赖注入）？](file:///G:\\文档\\Differ%20BS框架开发文档.docx" \l "_、如何在Domain层运用单元调用仓储？)

2.4、 [UI层如何调用Services服务？](file:///G:\文档\Differ%20BS框架开发文档.docx#_、UI层如何调用Services服务？)

2.5、 [如何写入日志？](file:///G:\文档\Differ%20BS框架开发文档.docx#_、UI层如何调用Services服务？)

2.6、 [如何操作缓存？](file:///G:\文档\Differ%20BS框架开发文档.docx#_、如何在MVC_Controller中添加API方法？)

2.7、 [如何操作缓存？](file:///G:\文档\Differ%20BS框架开发文档.docx#_、如何操作缓存？)

2.8、[如何获取、添加配置？](file:///G:\\文档\\Differ%20BS框架开发文档.docx" \l "_、如何获取、添加配置？)

1. 文档说明
   1. 本文档是对杭州风特陪玩项目 B/S框架开发中各个功能说明；
   2. 该框架解决方案主要分为:

仓储（Repository）、

服务(Service)、

基础结构设施、包含一些通用的组件及loc容器（Common）、

实体层（Entity）

UI层（FengTe.GamePlay.Web）

* 1. 该框架基于传统三层的基础上稍作封装，基于接口编程，对仓储层，服务层分别定义了基接口（IBaseRepository,IBaseService）统一了crud的命名方式，此外也是适应不用的数据库和业务。数据持久层采用dapper(微小的orm框架，体积小，性能相对来说也不错，当然相对ado.net原生来说还是稍显略逊)。UI层采用asp.netMVc5。
  2. 操作环境：vs2015,MSsqlServer2016
  3. 关于项目框架所用技术：

后台Ui布局：layui,弹层layer

依赖注入：autofac

缓存：memcached

用于数据持久层: dapper

1. 框架核心功能说明

### 如何添加仓储？

* + - 1. 添加实体
         1. Entity(实体层)的下添加对应的实体类，类中包含对应属性；
      2. 添加仓储

2、在仓储（Repository）文件下找到仓储接口类库（IRepository）,添加分别对应的实体仓储，然后继承泛型接口IBaseRepository<T>

基接口：public interface IBaseRepository<T> where T:class,new()

如：public interface IUserRepository:IBaseRepository<Users>  
 在该仓储下除了基接口自带的方法外，如果服务层另有需求，在这里定义特殊方法。

* + - 1. 3、在仓储实现层（Repository）找到MSSQLDB文件夹添加对应实体仓储，然后继承此前添加的实体仓储接口。

如：public class UserRepository : IUserRepository

{

public bool Delete(Users entity)

{

throw new NotImplementedException();

}

public bool DeleteList(IList<Users> list)

{

throw new NotImplementedException();

}

public IEnumerable<Users> GetList()

{

using (var conn=ConnectionFactory.Connection())

{

//string sql = @"select \* from users ";

return conn.Query<Users>("select \* from users");

}

}

public Users GetModel(int id)

{

throw new NotImplementedException();

}

public IList<Users> GetQueryMultiple()

{

throw new NotImplementedException();

}

public int Insert(Users entity)

{

throw new NotImplementedException();

}

public int InsertList(IList<Users> list)

{

throw new NotImplementedException();

}

public Tuple<IEnumerable<Users>, int> LoadPageEntities(int pageIndex, int PageSize, bool isAsc, QueryParam param = null)

{

throw new NotImplementedException();

}

public bool Update(Users entity)

{

throw new NotImplementedException();

}

public bool UpdateList(IList<Users> list)

{

throw new NotImplementedException();

}

注：上层（服务层需要什么提供什么，切忌不管上层需求，有什么便提供什么样的数据）

2.2、如何在Service层调用仓储？

1、在Service文件夹下找到服务接口类库（IService）,建立对应的实体服务接口，继承基服务（IBaseService）

基服务接口：crud方法

public interface IBaseService<T> where T:class,new()

{

/// <summary>

/// 返回列表

/// </summary>

IEnumerable<T> GetList();

/// <summary>

/// 返回单个对象

/// </summary>

T GetModel(int id);

/// <summary>

/// 查询多个结果集

/// </summary>

IList<T> GetQueryMultiple();

/// <summary>

/// 分页

/// </summary>

/// <param name="pageIndex"></param>

/// <param name="PageSize"></param>

/// <param name="isAsc">是否升序</param>

/// <param name="param">模糊查询参数</param>

/// <returns></returns>

Tuple<IEnumerable<T>, int> LoadPageEntities(int pageIndex, int PageSize, bool isAsc, QueryParam param = null);

/// <summary>

/// 删除单条记录

/// </summary>

bool Delete(T entity);

/// <summary>

/// 批量删除

/// </summary>

bool DeleteList(IList<T> list);

/// <summary>

/// 新增单条记录

/// </summary>

int Insert(T entity);

/// <summary>

/// 批量新增

/// </summary>

int InsertList(IList<T> list);

/// <summary>

/// 更新单条记录

/// </summary>

bool Update(T entity);

/// <summary>

/// 批量更新

/// </summary>

bool UpdateList(IList<T> list);

}

如：public interface IUserService : IBaseService<Users>

{

}

2、服务实现在Service类库中建立对应的实体服务

如：public class UserService : IUserService

public IEnumerable<Users> GetList()

{

return IocUtils.Resolve<IUserRepository>().GetList();

}

2.3、如何注册对象（依赖注入）

1、在FengTe.GamePlay.Utility的类库下找到（IocUtils）看该类中定义的方法。

这里的loc使用的是Autofac，关于autofac不会的自行百度，另外为什么依赖注入，整个框架的搭建过程中都基于“高内聚，低耦合”的思想。说到这里。通着则通，不通百度。

2、关于服务层调用仓储层的方法，该对象也是注入生成，接下来的ui层也是以一样。

2.4、UI层如何调用Services服务

1、通过公共组件注入对象调用服务层的方法

如：

public ActionResult Index()

{

var data= IocUtils.Resolve<IUserService>().GetList();

return View(data);

}

2.5、如何操作系统日志

关于系统日志封装了通用类，在配置文件中应用了log4net(关于log4net基础知识及配置百度脑补)。需要调用时

LogHelper.WriteLog(“错误信息”);

此外在系统的global中也调用了抛异常的类

public void Application\_Error(object send, EventArgs e)

{

//记录日志

Exception ex = Server.GetLastError();

string errorMsg = ex.ToString();

LogHelper.WriteLog(errorMsg);

// //转到错误页或者跳转。

// Response.Redirect("/Home/Error");

}