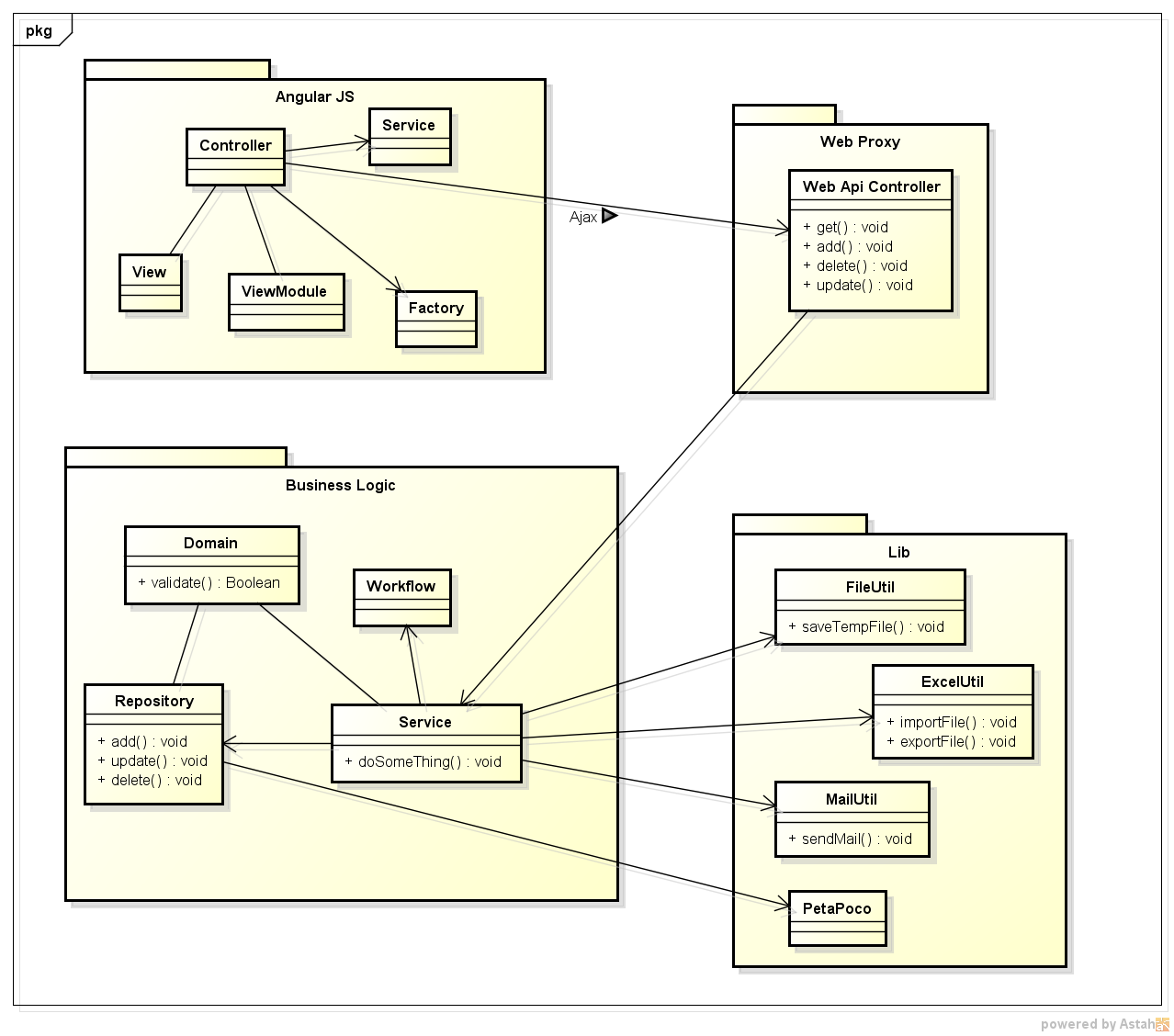
框架使用说明

1. 框架整体说明

框架分三部分：

* 1. Web层是web表现层，包含所有asp.net mvc页面,web api controller,以及js,css,html文件
  2. BLL层是业务逻辑层,包含数据库访问以及业务逻辑代码
  3. Lib层是基础设施层，包含业务无关的基础辅助类



1. ORM -- NPoco框架使用说明
   1. 简单的查询

public class User

{

public int UserId { get;set; }

public string Email { get;set; }

}

using (IDatabase db = new Database("connStringName"))

{

List<User> users = db.Fetch<User>("select userId, email from users");

}

* 1. 映射
     1. [TableName] 指定映射表名.
     2. [PrimaryKey]  指定主键
     3. [Column] 如果数据库列名和属性不一致可以制定名字。
     4. [Ignore]  会在映射的时候忽略。
     5. [ResultColumn] 列只会返回，不会新增修改.

例子:

[TableName("Users")]

[PrimaryKey("UserId")]

public class User

{

public int UserId { get;set; }

[Column("emailAddress")]

public string Email { get;set; }

[ResultColumn]

public string ExtraInfo { get;set; }

[Ignore]

public int Temp { get;set; }

}

* 1. 查询单个数据
     1. 通过Id查询:

IDatabase db = new Database("connStringName");

User u = db.SingleById<User>(3);

* + 1. 通过sql查询:

User u = db.Single<User>("where emailaddress = @0", "email@domain.com");

或者

User u = db.Single<User>("select u.\* from users u where emailaddress = @0", "email@domain.com");

注意:这个方法还有SingleOrDefault,如果不是很确认有没有对象，请使用SingleOrDefault，因为没有对象single方法会报错。而SingleOrDefault再多余一条也会报错，多余一条请使用First<T> and FirstOfDefault<T>

* 1. 新增修改数据
     1. 新增记录:

IDatabase db = new Database("connStringName");

User u = new User()

{

Email = "name@domain.com",

LastLoggedIn = DateTime.UtcNow

};

db.Insert(u);

* + 1. 修改记录:

var user = db.SingleById(1);

user.Email = "new@domain.com";

db.Update(user);

* + 1. 删除记录:

var user = db.SingleById(1);

db.Delete(user);

或者

db.Delete<User>(1);

* 1. 查询
     1. 读取所有:

List<User> users = db.Fetch<User>();

* + 1. 根据条件:

List<User> users = db.Fetch<User>("where isActive = 1");

* + 1. 根据sql:

List<User> users = db.Fetch<User>("select u.\* from users where u.isActive = 1");

* + 1. 延迟所有:

List<User> users = db.Query<User>("select u.\* from users where u.isActive = 1");

* 1. 分页查询

IDatabase db = new Database("connStringName");

Page<T> pagedUsers = db.Page<User>(2, 10, "select u.\* from users u order by userid");

* 1. 事务处理
     1. 代码1:

using (IDatabase db = new Database("connStringName"))

{

db.BeginTransaction();

//Your CRUD operation here

db.CompleteTransaction();

}

* + 1. 代码2:

using (IDatabase db = new Database("connStringName"))

{

using (var transaction = db.GetTransaction())

{

//Your CRUD operation here

transaction.Complete();

}

}

1. AngularJs框架以及相关插件使用说明
   1. 由于框架已经在app.js增加了定制拦截器,统一处理了angularjs的ajax错误,所以所有angularjs的ajax方法不用自己处理错误。
   2. 增加模块方法:

在框架内增加一个模块，

* + 1. 所有前端模块都会在App目录下，单独的模块可以新建一个文件夹，
    2. 新建的文件夹下面增加 模块名+Module的js文件，作为模块定义文件
    3. 新建的文件夹下面增加cshtml页面(即view)
    4. 新建的文件夹下面增加以Ctrl后缀的js文件(即Controller)
    5. 在模块文件下增加相应的状态(state),如下testuser是状态,

譬如:

.state('testuser', {

url: "/testuser",

templateUrl: "App/form/testuser.cshtml",

controller: "mForm.testUserCtrl"

})

* + 1. 最后Index.cshtml后面的script块增加js文件
  1. Ui-select插件—下拉框插件

基本使用:

<ui-select ng-model="selected.value">

<ui-select-match>

<span ng-bind="$select.selected.name"></span>

</ui-select-match>

<ui-select-choices repeat="item in (itemArray | filter: $select.search) track by item.id">

<span ng-bind="item.name"></span>

</ui-select-choices>

</ui-select>

angular.module('app')

.controller('ctrl', ['$scope', function ($scope){

$scope.itemArray = [

{id: 1, name: 'first'},

{id: 2, name: 'second'},

{id: 3, name: 'third'},

{id: 4, name: 'fourth'},

{id: 5, name: 'fifth'},

];

$scope.selected = { value: $scope.itemArray[0] };

}]);

* 1. ngTable—表格插件

<http://ng-table.com/#/>

如下带有分页以及排序的表格，调用api/user/pageuser

$scope.tableParams = new NgTableParams(

{

page: 1, // show first page

count: 10, // count per page

sorting: {

badge: 'asc' // initial sorting

}

},

{

total: 0, // length of data

getData: function (params) {

var d = $q.defer();

var obj = \_.extend($scope.qInfor,

{

Page: params.page(),

ItemsPerPage: params.count(),

OrderBy: params.orderBy()

});

$http.post("api/user/pageuser", obj).success(

function (data) {

params.total(data.total);

d.resolve(data.result);

console.log(data.result);

});

return d.promise;

}

});

* 1. AngularStrap—模态对话框插件

var myOtherModal = $modal({scope: $scope, template: 'modal/docs/modal.demo.tpl.html', show: false});

// Show when some event occurs (use $promise property to ensure the template has been loaded)

$scope.showModal = function() {

myOtherModal.$promise.then(myOtherModal.show);

};

* 1. AngularStrap—Tab插件

<div class="btn btn-default" ng-click="toggleThirdTab()">

{{ tabs[2].disabled ? 'Enable' : 'Disable' }} third tab

</div>

<div class="btn btn-default" ng-click="tabs.activeTab='Profile'">

Select "Profile" tab

</div>

<hr>

<!-- bsActivePane is optional -->

<div bs-active-pane="tabs.activeTab" bs-tabs>

<div ng-repeat="tab in tabs" data-title="{{ tab.title }}" name="{{ tab.title }}" disabled="{{ tab.disabled }}" ng-bind="tab.content" bs-pane>

</div>

</div>

* 1. AngularStrap—日期时间选择插件

<input type="text" size="10" class="form-control" ng-model="sharedDate" data-autoclose="1" placeholder="Date" bs-datepicker>

<input type="text" size="8" class="form-control" ng-model="sharedDate" data-time-format="h:mm:ss a" data-autoclose="1" placeholder="Time" bs-timepicker>

* 1. ngUpload 简单文件上传插件

该插件是简单封装了jquery.fileupload ，data-url是上传服务端地址

注意该插件只支持一次上传一个文件

<input type="file" ng-upload data-url="api/util/PostFile" />

* 1. JsDraw—结构式插件

<div js-draw="{idMode:'static'}" id="mol1" viewonly popup ng-model='mol'></div>

* 1. userPick—用户选择插件

<user-pick ng-model="user1.selected"></user-pick>

注意必须绑定对象，所以后台可能需要建立一个viewmodel

* 1. 权限设置

在cshtml中使用

@if (HasRoles("Admin"))

{

<div>Admin权限才能显示</div>

}

来控制权限

1. IOC -- NInject框架使用说明
   1. Ninject是一个轻量级的基于.Net平台的依赖注入（IOC）框架。所谓的IOC，即控制反转（Inversion of Control），它是一个经典的面向对象编程法则，它的作用主要是用来帮助应用程序解耦，并把程序分离成一个个松耦合高内聚的模块。控制反转还有一个名字叫依赖注入（Dependency Injection），简称DI。
   2. 推荐构造函数依赖注入。如：

public class HomeController : Controller

{

private readonly SiteMapService \_mss;

public HomeController(SiteMapService mss)

{

\_mss = mss;

}

* 1. 在Web项目下\App\_Start\ NinjectWebCommon.cs的

private static void RegisterServices(IKernel kernel)

快内注册需要依赖注入的类和接口

可以单个注册，如:

Bind<ILogService>().To<DbLogService>();

也可以自动搜索注册,以下是把所有dll里面所有service和repository为后缀名的类自动注册

kernel.Bind(x => x.FromAssembliesMatching("\*.dll")

.Select(f => f.Name.EndsWith("Repository")|| f.Name.EndsWith("Service"))

.BindToSelf().Configure(binding => binding.InRequestScope())

);

* 1. 在Web项目下\App\_Start\ NinjectWebCommon.cs的

private static void RegisterServices(IKernel kernel)

主数据库链接已经注册,如下

kernel.Bind<Database>().ToConstructor(

context => new Database("UserConnection")

).InRequestScope();

InRequestScope将保证一个请求只有一个数据库连接

数据库链接在webconfig 下面：

<connectionStrings>

<add name="UserConnection" providerName="System.Data.SqlClient" connectionString="server=168.150.32.216;uid=wuxiappsa;pwd=wuxiappsa@2012;database=publicDB" />

</connectionStrings>

1. Web API 说用说明
   1. 由于框架修改了默认的WebAPI路由配置，所以没有按照restful风格的以：api/{controller}/{action}/{id} 配置路由。
   2. 所有Ajax处理都推荐在WebAPI处理。