Web API 操作的一些东西

1. **控制器**

1. 1 关于命名

命名就相当于给某一个事物打上标识，以便于记录与交流。

**一级独立命名法：相对独立的情况下使用，独立存在的**

例1： 用户(User)

例2： 课程（Course）

**二级索引关系命名法（二级索引导航命名法）：存在索引，二级目录关系的命名法。**

**命名采用几个词的组合，首先是索引词，然后是关系，最多为2级。**

例1: 用户课程（UserCourse）

分析：User 为索引词(表示User这个独立个体)，Course为二级关系

例2: 用户课程主题（UserCourseTheme）

分析：User 为索引词，CourseTheme 为二级关系，Course为父级,Theme 为子级，就组成CourseTheme 二级关系。

例3: 用户主题知识点（UserThemePoint）

分析：User 为索引词，ThemePoint为关系，Theme为父级, Point为子级，就组成ThemePoint二级关系。

相反可以用Theme做为子级，来找到他的上一级，如：UserCourseTheme.

二级命令法，只关联二级，清晰明了。

如果不采用这种命名法，将是：UserCourseThemePoint（用户课程主题知识点），如果在加关系的话，那么将出现概念混淆。

例4: 用户知识点段落（UserPointSection）

分析：User 为索引词，ThemePoint为关系，Point为父级, Section为子级，就组成ThemePoint二级关系。

用图表示是这样的(就相当于属性图)

User(用户) --一级独立命名

UserCourse（用户课程） --二级索引关系命名（User既是索引也是父级关系）

UserCourseTheme(用户课程主题)

UserThemePoint(用户主题知识点)

UserOrder（用户订单）

UserOrderDetail(用户订单详情)

Course(课程)

CourseDetail(课程详情)

二级关系命名，可以很清晰的理解，对应的父级关系，容易理解。

说控制器的命名，

1.尽量拆分单一，一个API 控制器只做一个事情,

2. 尽量和实体类名保持一致，因为，大多数时候，实体类名已经很单一了，还可以让一个新人快速上手，快速理解控制器的对应的意义。

**二级组合索引关系命名法（二级索引导航命名法）**

注:采用索引命名法

索引\_变量名

索引\_变量名\_描述

方法命名

Get --获得单个对象

GetList –获得集合对象

GetPage –获得带有分页

1. **关于接口**

接口定义；因为API 控制器已经明确单一了，那么对当前API控制器所有的获取都可以使用名为“Get” 的路由，

如:

/// <summary>

/// 根据Id来获得对应的信息

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public dynamic Get(int id)

{

return this.\_queryUserSubjectRecordBusiness.GetById(id);

}

--更加

/// <summary>

///

/// </summary>

/// <param name="userId"></param>

/// <returns></returns>

[HttpGet]

[Route("Get")]

public dynamic GetByUserId(int userId)

{

return this.\_queryUserSubjectRecordBusiness.GetByWhere(t => t.UserId == userId);

}

/// <summary>

/// Gets the by user identifier.

/// </summary>

/// <param name="userId">The user identifier.</param>

/// <param name="courseId">The course identifier.</param>

/// <returns></returns>

[HttpGet]

[Route("Get")]

public dynamic GetByUserId(int userId, int courseId)

{

return this.\_queryUserSubjectRecordBusiness.GetByWhere(t => t.UserId == userId && t.CourseId == courseId);

}

/// <summary>

/// Gets the by user identifier.

/// </summary>

/// <param name="userId">The user identifier.</param>

/// <param name="courseId">The course identifier.</param>

/// <param name="themeId">The theme identifier.</param>

/// <returns></returns>

[HttpGet]

[Route("Get")]

public dynamic GetByUserId(int userId, int courseId, int themeId)

{

return this.\_queryUserSubjectRecordBusiness.GetByWhere(t => t.UserId == userId && t.CourseId == courseId && t.ThemeId == themeId);

}

1. **业务接口**
2. **API控制器中的每个接口，尽量不要写业务，只写跟控制器有直接关系的代码**
3. **对一个函数，进行多个重构，根据不同的参数，返回不同的数据，但必须是有父子级关系**

**--EF**

1. **FE 查询使用Select() ,查询自己所需要的数据，返回使用c# 4.0 特性 (动态类型)**
2. **UCD 操作使用ViewModel 来接受，这样的目的是为了更方便验证数据；**