# 新平台后端开发规范

# 基础篇

## 代码结构

### 整体代码结构

分为Web网站、应用层、业务层、数据层，如下图1.1所示。



图1.1

### Web网站的代码结构

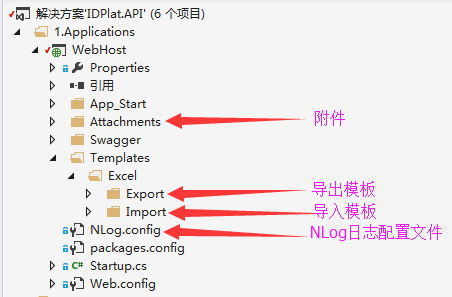


图1.2

### 应用层的代码结构

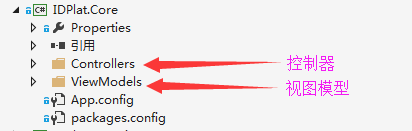


图1.3

### 业务层的代码结构

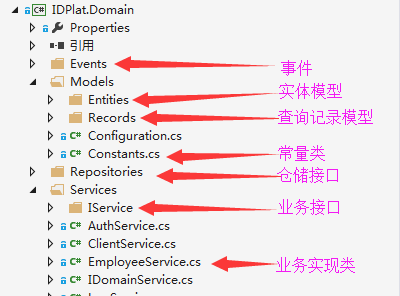


图1.4

### 数据层的代码结构

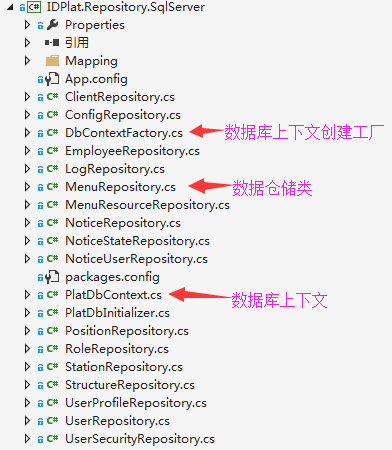


图1.5

## 规范要求

### 应用层规范

#### 路由定义

在Controller上添加属性RoutePrefix（路由前缀），并在方法上添加属性Route（具体路由），则每一个方法的访问路径通过路由前缀+具体路由组成。例如，获取员工的简要信息的访问路径为：api/employees/brief，路由前缀和具体路由分别如下图2.1和图2.2所示。

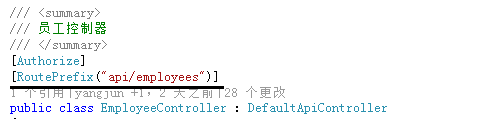


图2.1

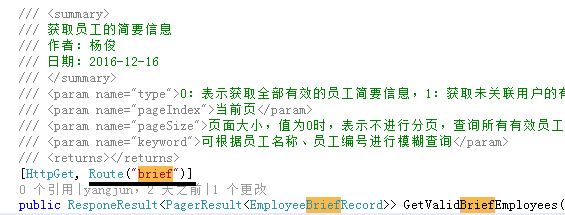


图2.2

#### 权限控制

分为用户登录权限控制、模块权限控制、用户自身权限控制、匿名访问控制四种模式。

用户登录权限控制：主要是用来验证发送请求的用户是否正常登录，只需要在Controller上添加Authorize属性就可实现，如图2.3所示。



图2.3

模块权限控制：主要是用来控制用户对其所请求的API是否拥有访问权限，只需要在方法上添加ModuleAuthorize属性就可实现，如图2.4所示。

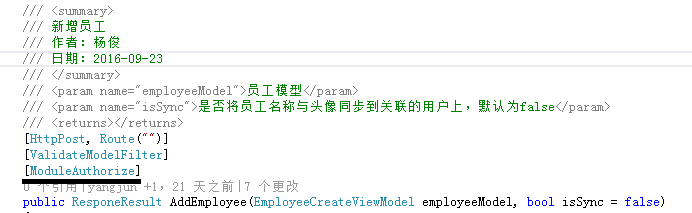


图2.4

用户自身权限控制：主要是对属于用户独有的API进行访问权限控制。需在方法上添加属性SelfAuthorize，并且Route属性上必须加上userId才可实现，userId表示当前用户Id，也是该方法的参数，如图2.5所示。

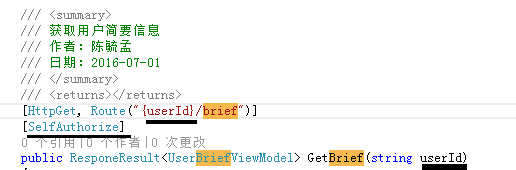


图2.5

匿名访问控制：任何用户都可以访问其所修饰的API，即该API不做任何权限控制，只需要在方法上加上属性AllowAnonymous就可实现，如图2.6所示。

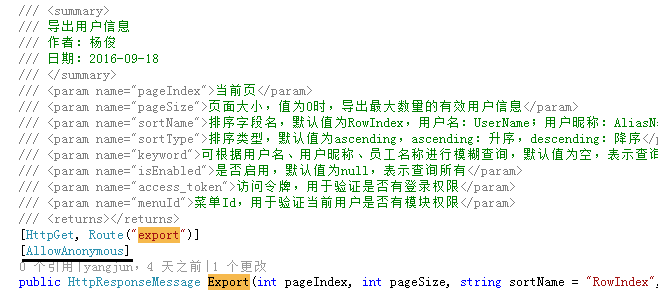


图2.6

#### 业务接口注入

通过Controller的构造函数实现对业务接口的注入，完成对所有调用接口的声明与初始化，实现方式如图2.7所示。



图2.7

#### 模型创建

模型需要放置ViewModels文件夹下，其名称由对象主体+[修饰词]+ViewModel构成，修饰词不是必须的，修饰词可以为Create、Brief、List等。其适用范围如图2.8所示。

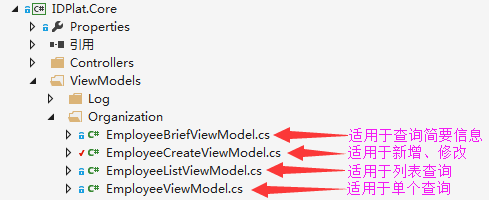


图2.8

#### 模型校验

模型校验一般是在新增、修改时进行校验，即名称中包含Create的ViewModel，其主要功能是检查Web端传递过来的数据是否符合模型的要求。可以引用System.ComponentModel.DataAnnotations类来实现对字段类型、长度、是否必填等信息进行校验。具体校验方式如图2.9所示。



图2.9

#### 模型转换

模型转换主要是在新增、修改时，将接收的ViewModel（视图模型）转换为业务层可以接收的实体模型，在查看时将从业务层获取的实体模型转换成外界所需要的视图模型。

#### 日志记录

在每次操作成功后，在返回结果之前，通过Logger对象调用Information方法来记录操作日志。调用方法如图2.10所示。



图2.10

#### 响应结果

响应结果通过类ResponeResult来定义，其主要有两个属性code（状态码）和Message（信息）。还可以通过泛型ResponseResult<T>来返回相应类型的数据。其两种使用方式如图2.11和图2.12所示。



图2.11



图2.12

#### 代码规范

1. 控制器名称：由对象主体+Controller构成，例：EmployeeController表示员工控制器。
2. Controller类需要继承Sapi.WebApi命名空间下的DefaultApiController类。

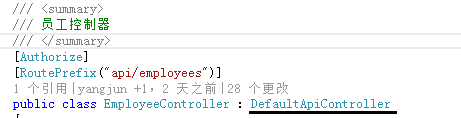


图2.13

1. 方法的摆放顺序：应按照Http方法从HttpPost（新增）、HttpPut（修改）、HttpDelete（移除）、HttpGet（查询）的顺序进行排列。
2. 方法的命名：新增方法名称以Add开头、修改方法名称以Edit开头、移除方法名称以Remove开头、查询方法名称以Get开头、导入名称为Import、导出名称为Export。
3. 对参数的合法性进行校验，例如移除员工时，校验员工ID是否为空。
4. ViewModel中集合属性在定义时，需要在构造函数中进行初始化，避免在调用该属性时出现空对象引用。其定义与初始化如图2.14和图2.15所示。



图2.14

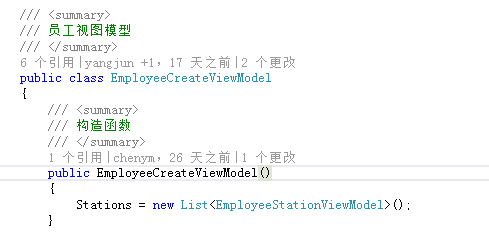


图2.15

1. 导出时，需要控制导出数据的最大数量，导出数据最大数量可以从系统配置中获取，如图2.16所示。

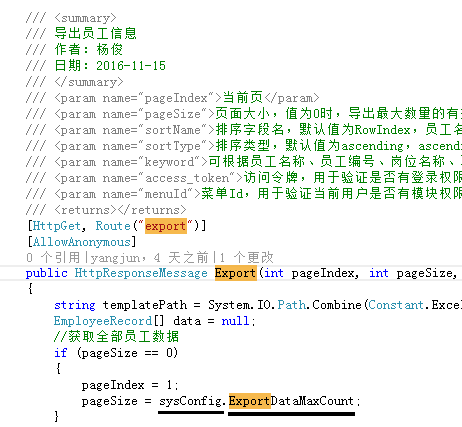


图2.16

### 业务层规范

#### 业务接口定义

业务层接口应放在IService文件夹下，需要继承命名空间Sapi.Services下的接口IBusinessService，其名称应为字母I + 对象主体+Service组成。如图2.17所示。

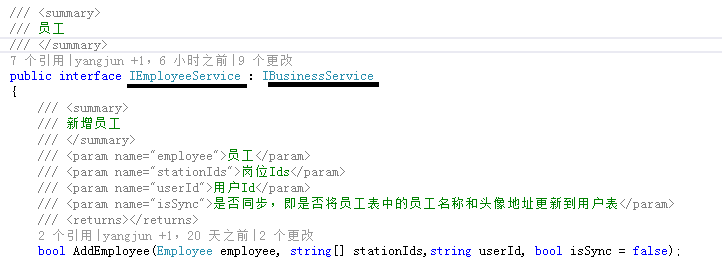


图2.17

#### 数据接口定义

数据层接口应放在Repositories文件夹下，需要继承命名空间Sapi.Data下的接口IRepository<T>，其名称应为字母I + 对象主体+Repository组成。如图2.18所示。

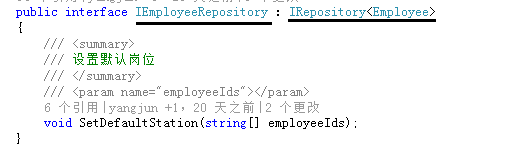


图2.18

#### 数据接口注入

通过Service的构造函数实现对数据接口的注入，完成对所有调用接口的声明与初始化，实现方式如图2.19和图2.20所示。

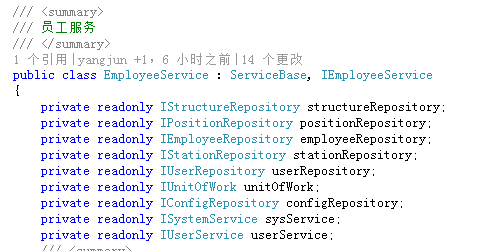


图2.19



图2.20

#### 数据接口使用

1. 新增数据，分为直接新增和在事务内新增两种情况，如图2.21所示。

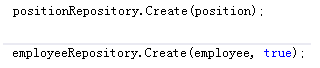


图2.21

1. 修改数据，分为直接修改和在事务内修改两种情况，如图2.22所示。



图2.22

1. 移除数据，直接将数据从数据库中删除，如图2.23所示。



图2.23

1. 查询数据，Get方法是通过主键查询单个对象的信息，Fetch方法通过拉姆达表示查询对象的集合，.Table方式得到的IQueryable对象，其可以拼接查询条件和查询结果，Linq语句查询可以实现表之间的关联查询，示例如图2.24所示。

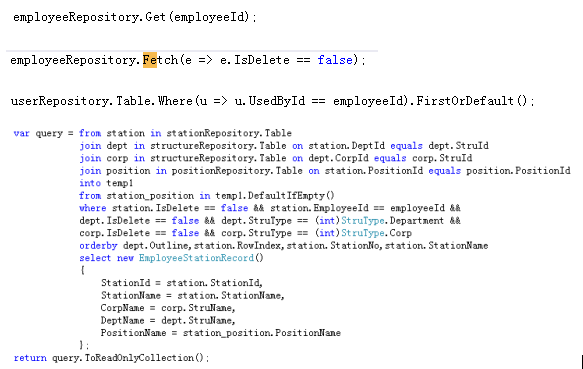


图2.24

#### 事务处理

在开启事务之前，将需要要执行的数据先准备好，在事务开启后，尽量不要执行查询语句，如图2.25所示。



图2.25

#### 代码规范

1. 服务名称：由对象主体+Service构成，例：EmployeeService表示员工服务。
2. Service类需要继承Sapi.Services命名空间下的ServiceBase类，并继承该对象所对应的接口，如图2.26所示。



图2.26

1. 方法的摆放顺序：应按照新增、修改、移除、查询的顺序进行排列。
2. 方法的命名：新增方法名称以Add开头、修改方法名称以Edit开头、移除方法名称以Remove开头、查询方法名称以Get开头。
3. 适当的使用重载功能，保证相同功能的核心代码一致。如将新增单个员工与批量新增员工进行重载，然后在新增单个员工时调用批量新增员工的方法。
4. 在新增、修改、移除时，将每次需要的验证内容放在同一个私有方法中统一进行校验。如图2.27所示。

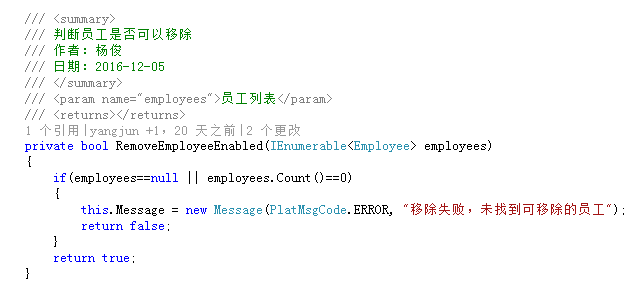


图2.27

1. 获取分页数据时，使用 PagerResult<T>类，pageSize为0是表示查询满足条件的所有数据，如图2.28所示。

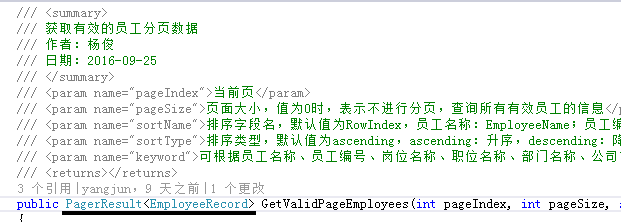


图2.28

#### 操作提醒码

操作提醒码主要是返回给前端的一个状态标识码，通过该提醒码可以知道返回的结果所代表的具体含义。其放在Constants.cs文件中，主要由PlatMsgCode.ERROR（操作错误）、PlatMsgCode.SUCCESS（操作成功）和每个业务对象特有的编码组成。三类提醒码如图2.29所示。

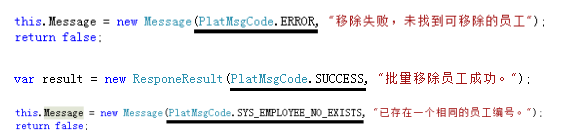


图2.29

### 数据层规范

#### 实体模型映射

实体模型映射主要是将实体模型的属性与数据库中相应表的字段，以及实体间的关系与对应表之间的关系进行映射。其命名为对象名称+Map组成，用关键字internal修饰，并继承System.Data.Entity.ModelConfiguration命名空间下的类EntityTypeConfiguration<T>，T表示实体模型，如图2.30所示。

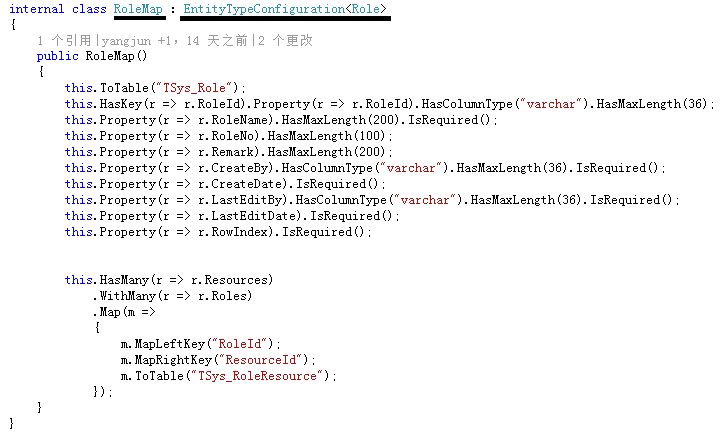


图2.30

#### 代码规范

1. 仓储名称：由对象主体+Repository构成，例：EmployeeRepository表示员工仓储。
2. Repository类需要继承Sapi.Data.EntityFramework命名空间下的Repository<T>类，并实现该对象所对应的数据接口，如图2.31所示。

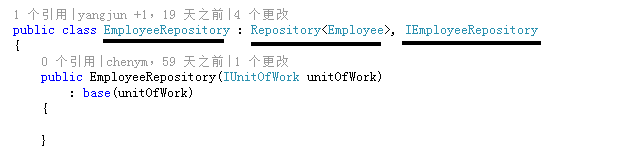


图2.31

1. 使用this.Context.Database对象执行SQL语句是应尽量使用参数化传值，降低SQL注入的风险，如图2.32所示。

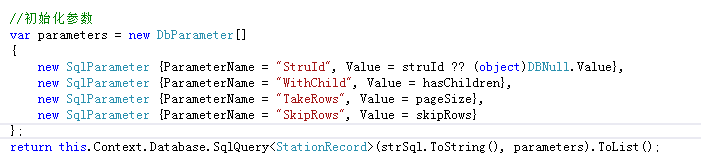


图2.32

## 参考示例

以员工对象为例。

### 应用层示例

#### 新增员工

使用HttpPost方法，用ValidateModelFilter属性开启模型校验功能，用ModuleAuthorize属性开启模块权限控制。其主要是将视图模型EmployeeCreateViewModel转换成实体模型Employee，并获取员工拥有的岗位Ids，再调用业务层的新增员工方法，实现员工的新增。如果成功，则记录操作成功日志，反之返回失败信息，示例代码如图3.1和图3.2所示。



图3.1



图3.2

#### 导入员工

使用HttpPost方法，具体路由Route为import，用ModuleAuthorize属性开启模块权限控制，并且用异步方式接收文件。其主要是将接收的Excel文件转成实体模型Employee数组，然后再调用业务层的批量新增方法，实现员工的导入功能。如果成功，则记录操作成功日志，反之返回失败信息，示例代码如图3.3和图3.4所示。



图3.3

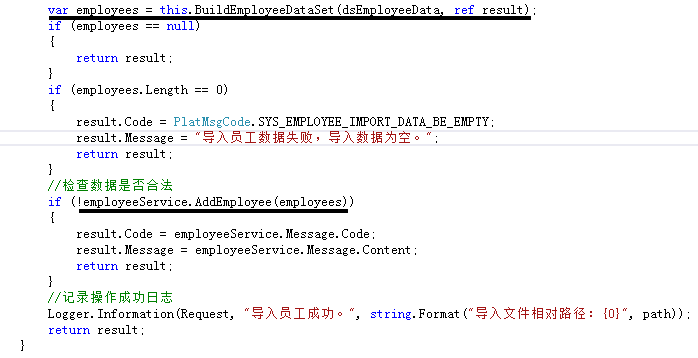


图3.4

#### 修改员工

使用HttpPut方法，用ValidateModelFilter属性开启模型校验功能，用ModuleAuthorize属性开启模块权限控制。其主要是通过员工Id获取员工实体模型Employee，然后用视图模型EmployeeCreateViewModel中数据修改Employee，并获取员工拥有的岗位Ids，再调用业务层的修改员工方法，实现员工的修改。如果成功，则记录操作成功日志，反之返回失败信息，示例代码如图3.5和图3.6所示。



图3.5

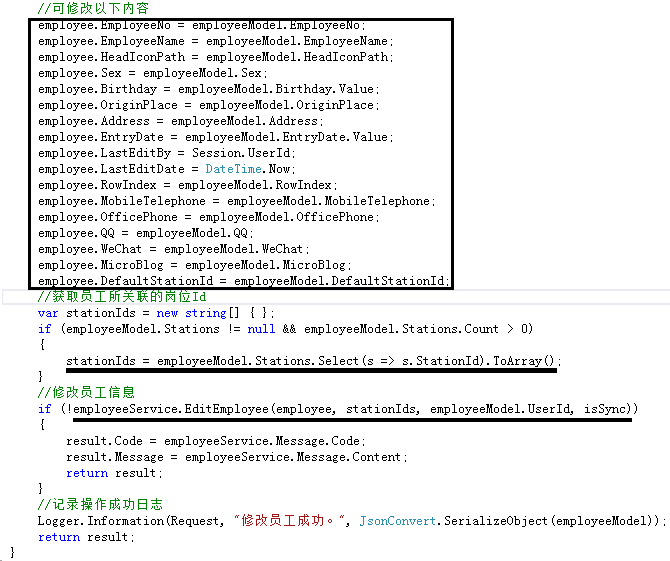


图3.6

#### 移除员工

使用HttpDelete方法，用ModuleAuthorize属性开启模块权限控制。其主要是验证参数employeeId格式是否符合要求，再调用业务层的移除员工方法，实现员工的移除。如果成功，则记录操作成功日志，反之返回失败信息，示例代码如图3.7所示。

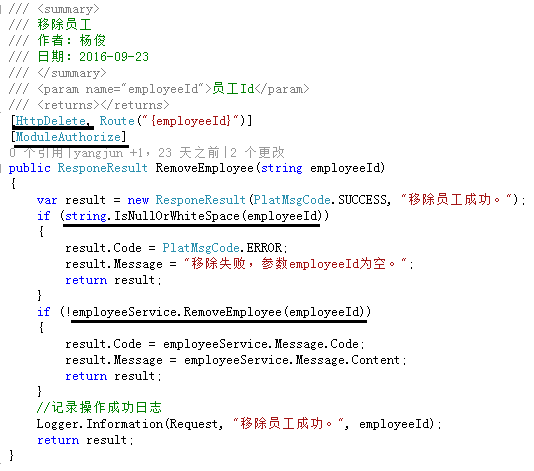


图3.7

#### 获取员工信息

1. 获取单个员工信息

使用HttpGet方法，根据员工Id获取员工实体模型Employee，并将其他信息准备好，如用户Id、用户名、创建人名称等，然后将实体模型Employee转换为视图模型EmployeeViewModel。示例代码如图3.8、图3.9和图3.10所示。



图3.8

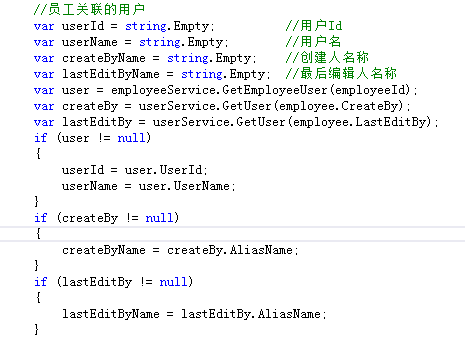


图3.9

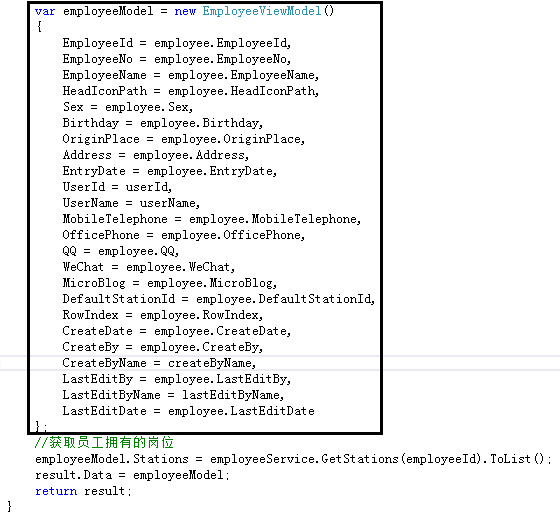


图3.10

1. 获取员工列表信息

使用HttpGet方法，用PagerResult类来定义分页类型，用ModuleAuthorize属性开启模块权限控制，具体路由Route用参数pageIndex来定义。通过调用业务层的GetValidPageEmployees方法来获取员工的有效分页数据。示例代码如图3.11所示。



图3.11

#### 导出员工

使用HttpGet方法，具体路由Route为export，用AllowAnonymous属性实现无权限访问。通过模板文件Employee.xlsx和方法ExportExcel将从业务层获取的数据放入Excel文件中实现导出功能。示例代码如图3.12所示。



图3.12

### 业务层示例

#### 新增员工

AddEmployee方法进行了重载，单个新增调用批量新增，核心代码在批量新增中实现，批量新增中通过方法AddEmployeeEnabled来验证是否可以进行新增，批量新增员工时需要使用事务，故在开启事务之前要将需要处理的数据准备好。示例代码如图3.13、图3.14和图3.15所示。

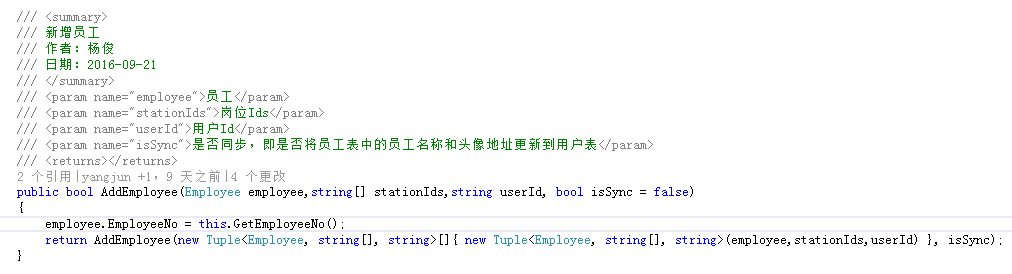


图3.13

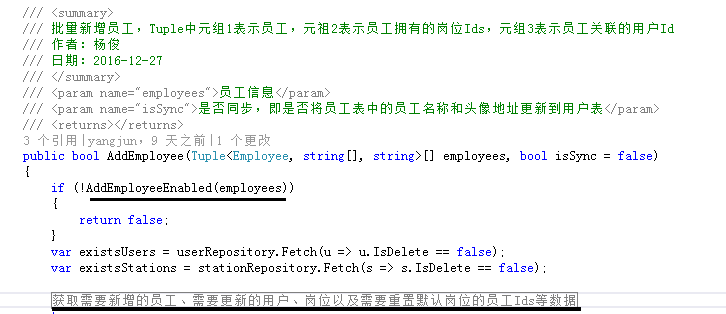


图3.14



图3.15

#### 修改员工

EditEmployee方法先通过方法EditEmployeeEnabled对实体模型Employee进行验证，然后在开启事务之前将需要处理的数据准备好。示例代码如图3.16和图3.17所示。



图3.16

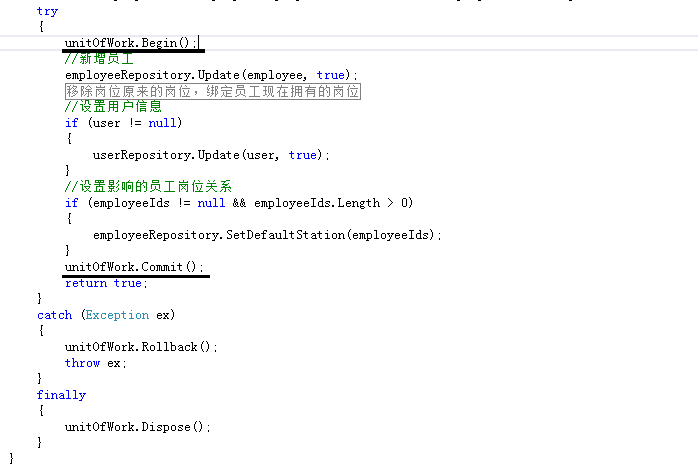


图3.17

#### 移除员工

RemoveEmployee方法进行了重载，单个移除调用批量移除，核心代码在批量移除中实现，批量移除中通过方法RemoveEmployeeEnabled来验证是否可以进行移除。示例代码如图3.18和图3.19所示。

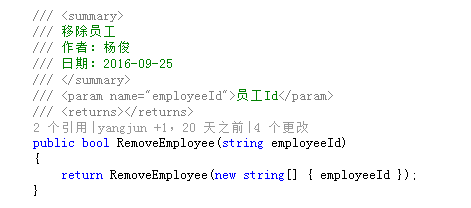


图3.18



图3.19

#### 查询员工

1. 查询单个员工

调用数据层的Get方法，通过主键获取单个员工信息。示例代码如图3.20所示。

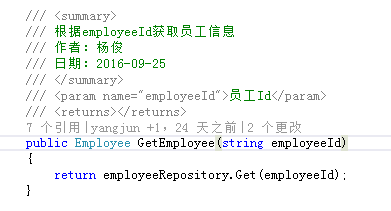


图3.20

1. 查询有效员工的分页列表

用PagerResult<EmployeeRecord>类定义GetValidPageEmployees方法的返回类型，采用Linq查询方式来实现与其它表的关联查询和实现查询结果直接模型化，并可以实现点击表头排序功能，当pageSize=0时，表示查询所有。示例如图3.21、图2.22和图2.23所示。

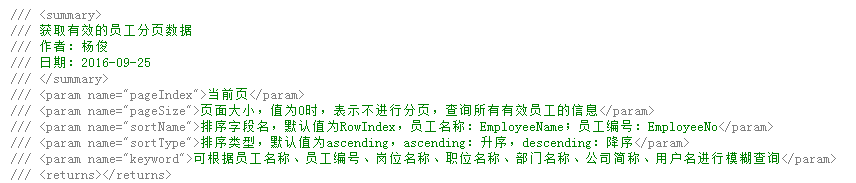


图2.21

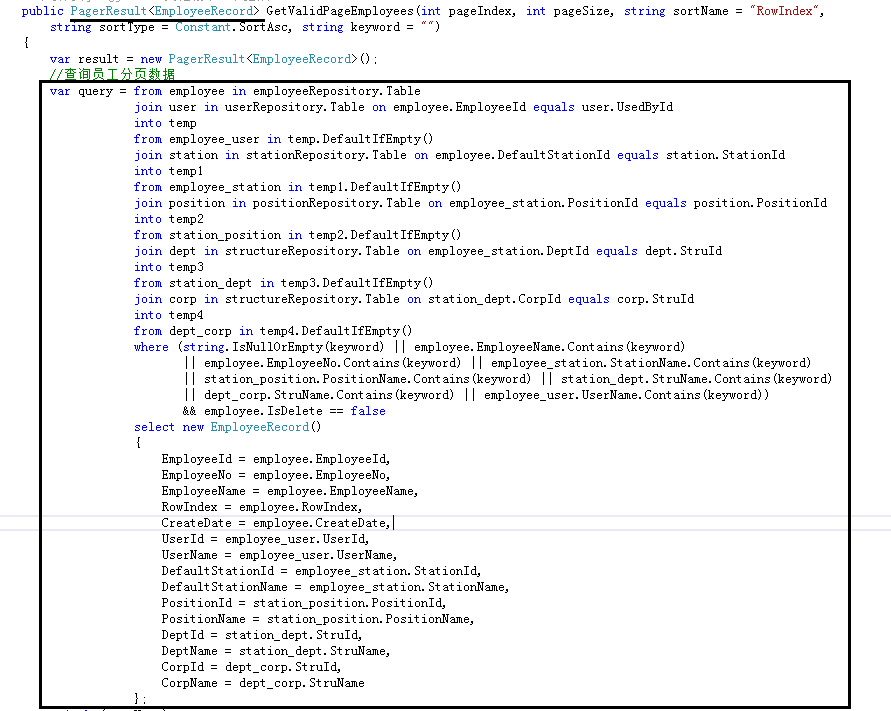


图2.22

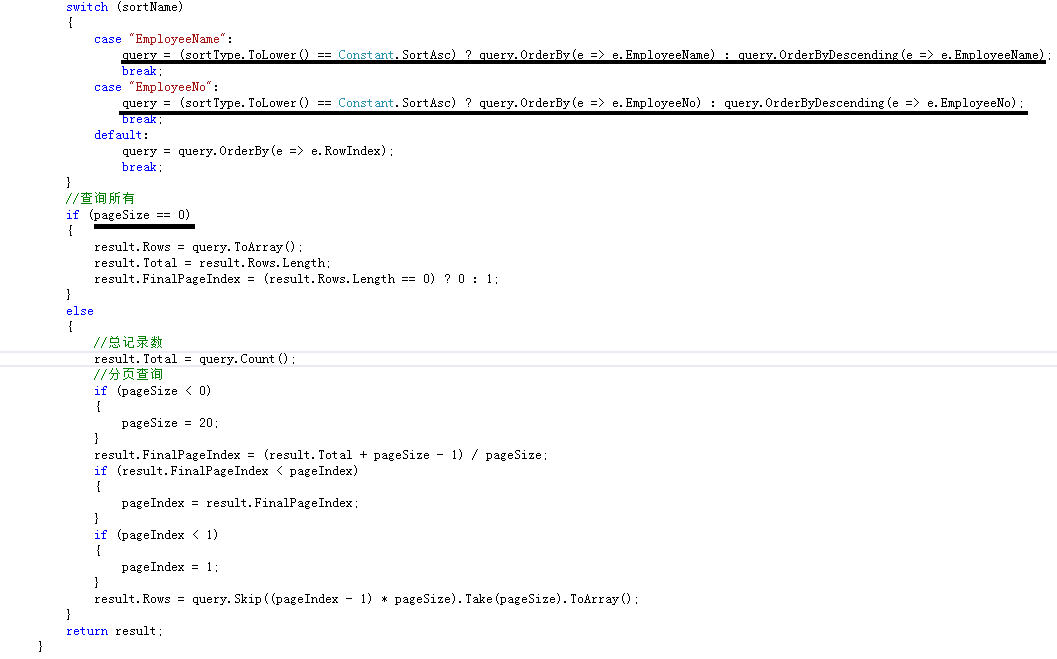


图3.23

1. 通过Fetch方法获取列表数据。示例代码如图3.24所示。



图3.24

#### 数据校验

以员工新增校验为例，在AddEmployeeEnabled方法中分为单个员工和多个员工两种情况进行验证，在多个员工情况下，在校验之前将数据库中满足条件的查询出来，然后在校验时使用，避免在循环校验时不断请求数据库，示例代码如图3.25和图3.26所示。



图3.25

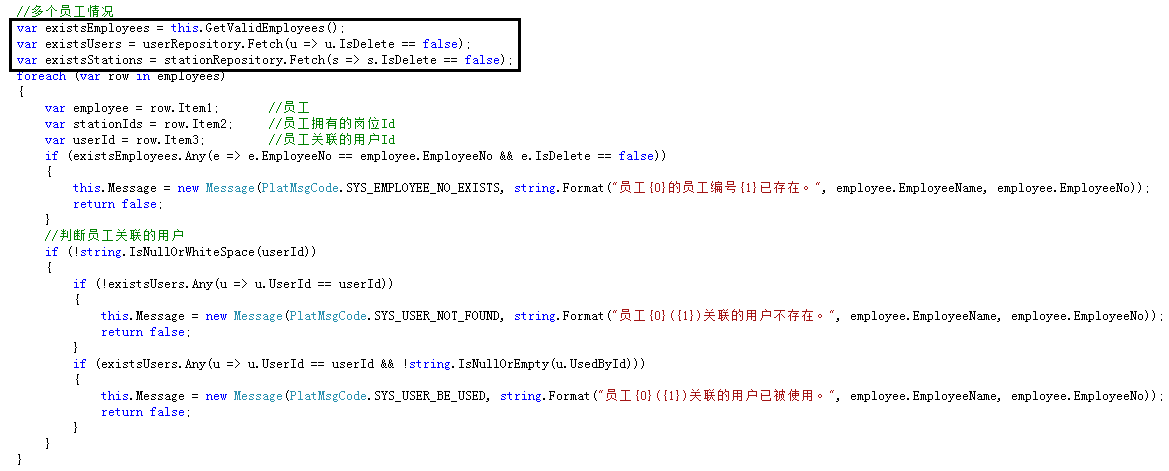


图3.26

### 数据层示例

#### ExecuteSqlCommand方法

在执行不需要返回值的SQL语句时，一般使用ExecuteSqlCommand方法。示例如图3.27所示。

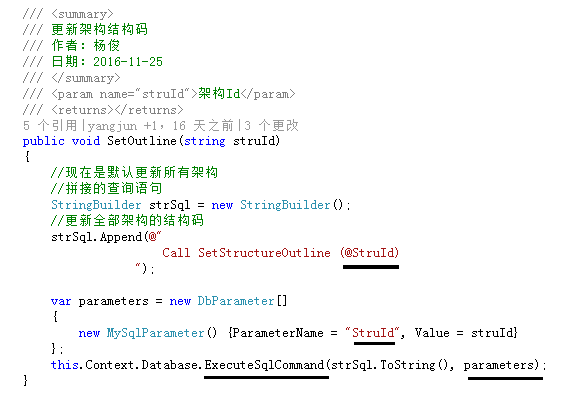


图3.27

#### SqlQuery<T>方法

SqlQuery<T>方法一般用于返回一个结果集，T为返回结果的类型，可以为int、string和自定义类型等。示例如图3.28所示。

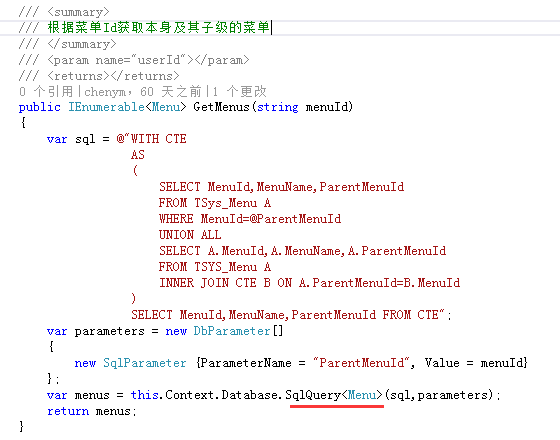


图3.28

# 高级篇

## 1.缓存应用

## 2.事件应用

## 3.参考示例