|  |
| --- |
| METASHARE INC |
| WebApi开发文档 |
|  |
|  |
| **Bessie Yin** |
| **12/26/2018** |

|  |
| --- |
|  |

目录

[1 背景 3](#_Toc533598240)

[2 基本WebApi接口 3](#_Toc533598241)

[2.1 SelectAll 3](#_Toc533598242)

[2.1.1 C#调用 3](#_Toc533598243)

[2.1.2 前端js调用 3](#_Toc533598244)

[2.2 SelectAllWithChild 3](#_Toc533598245)

[2.2.1 C#调用 3](#_Toc533598246)

[2.2.2 前端js调用 4](#_Toc533598247)

[2.3 SelectById 4](#_Toc533598248)

[2.3.1 C#调用 4](#_Toc533598249)

[2.3.2 前端js调用 4](#_Toc533598250)

[2.4 SelectByIdWithChild 5](#_Toc533598251)

[2.4.1 C#调用 5](#_Toc533598252)

[2.4.2 前端js调用 5](#_Toc533598253)

[2.5 SelectBy 5](#_Toc533598254)

[2.5.1 C#调用 5](#_Toc533598255)

[2.5.2 获取调用结果： 6](#_Toc533598256)

[2.6 Insert 6](#_Toc533598257)

[2.6.1 C#调用 6](#_Toc533598258)

[2.6.2 前端js调用 7](#_Toc533598259)

[2.7 Update 7](#_Toc533598260)

[2.7.1 C#调用 7](#_Toc533598261)

[2.7.2 前端js调用 7](#_Toc533598262)

[2.8 Delete 7](#_Toc533598263)

[2.8.1 C#调用 7](#_Toc533598264)

[2.8.2 前端js调用 8](#_Toc533598265)

[2.9 SelectPagerByColumns 8](#_Toc533598266)

[2.9.1 C#调用 8](#_Toc533598267)

[2.9.2 前端js调用 9](#_Toc533598268)

[2.10 SelectAllByPager 9](#_Toc533598269)

[2.10.1 C#调用 9](#_Toc533598270)

[2.10.2 前端js调用 9](#_Toc533598271)

[2.11 SelectAllByPagerWithChild 9](#_Toc533598272)

[2.11.1 C#调用 9](#_Toc533598273)

[2.11.2 前端js调用 10](#_Toc533598274)

[3 App WebApi接口 10](#_Toc533598275)

[3.1 Add 11](#_Toc533598276)

[3.2 Edit 11](#_Toc533598277)

[3.3 UpdateFromBody 11](#_Toc533598278)

[3.4 SelectAllByColumns 11](#_Toc533598279)

[3.5 SelectByColumns 11](#_Toc533598280)

[3.6 SelectPagerByColumns 11](#_Toc533598281)

# 背景

本文档介绍MDD生成的所有WebApi接口功能以及前后端调用方式，以便开发人员使用。

注意：下文中的ApiControllerName写法：如果一个Controller类名为CourseApiController，那么在url中ApiControllerName为CourseApi。如果Controller在包下面，则url中需要包含完整的包路径。

# 基本WebApi接口

## SelectAll

### C#调用

RequestResult SelectAll() - 单表集合查询

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ApiControllerName/";

myHttpClient.BaseAddress = new Uri(url);

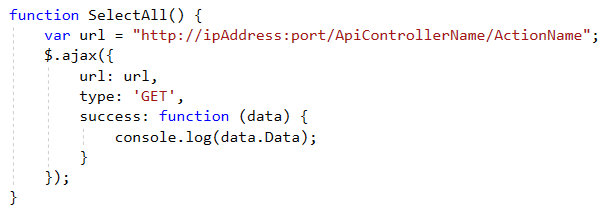
HttpResponseMessage response = myHttpClient.GetAsync("SelectAll").Result;

获取返回结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

List<T> entities = JsonConvert.DeserializeObject<List<T>>(result.Data.ToString());

### 前端js调用



## SelectAllWithChild

RequestResult SelectAllWithChild ()-集合附加子对象查询

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

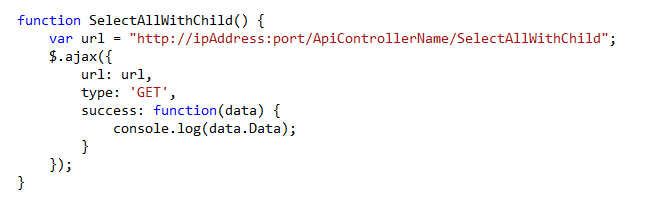
HttpResponseMessage response= myHttpClient.GetAsync("SelectAllWithChild").Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

List<T> entities = JsonConvert.DeserializeObject<List<T>>(result.Data.ToString());

### 前端js调用



## SelectById

RequestResult SelectById(int id) –单表单个查询

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

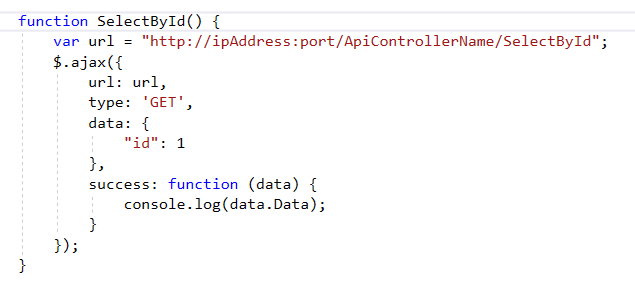
HttpResponseMessage response = myHttpClient. GetAsync("SelectById?id="+ id).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

T entity = JsonConvert.DeserializeObject<T>(result.Data.ToString());

### 前端js调用



## SelectByIdWithChild

RequestResult SelectByIdWithChild(int id, bool isAggregatedChildren) –附加子对象单个查询

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

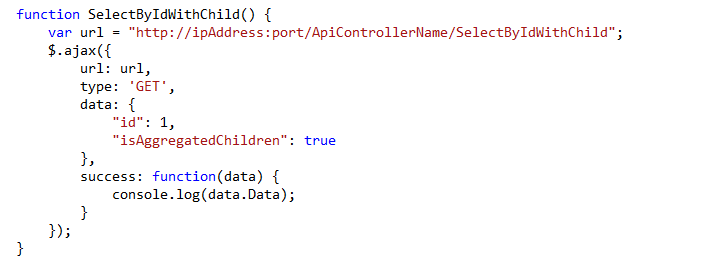
HttpResponseMessage response = myHttpClient. GetAsync("SelectByIdWithChild?id="+ id+"&isAggregatedChildren=true").Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

T entity = JsonConvert.DeserializeObject<T>(result.Data.ToString());

### 前端js调用



## SelectBy

RequestResult SelectBy(RequestData requestData)-单表结合查询

### C#调用

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

List<string> columnNames = new List<string>()

{

"Name"

};

T entity = new T()

{

Name = "name"

};

Dictionary<string, string> requestData = (new RequestData<T> {Entity = entity, Columns = columnNames }).Serialize();

FormUrlEncodedContent content = new FormUrlEncodedContent(requestData);

HttpResponseMessage response = myHttpClient.PostAsync("SelectBy ",content).Result;

### 获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

List<T> entities = JsonConvert.DeserializeObject<List<T>>(result.Data.ToString());

前端js调用



## Insert

RequestResult Insert([FromBody] TEntity entity)-单表新增

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

T entity = new T ();

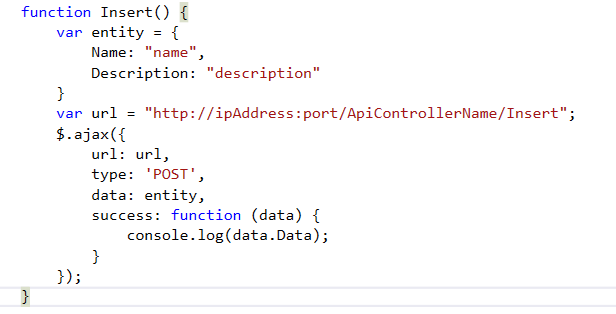
HttpResponseMessage response = myHttpClient.PostAsJsonAsync("Insert", entity).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

T entity = JsonConvert.DeserializeObject<T>(result.Data.ToString());

### 前端js调用



## Update

RequestResult Update([FromBody] TEntity entity) -单表修改

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

T currentEntity = new T ();

HttpResponseMessage response = myHttpClient.PostAsJsonAsync("Update",currentEntity).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

T entity = JsonConvert.DeserializeObject<T>(result.Data.ToString());

### 前端js调用

和Insert方法类似

## Delete

RequestResult Delete (int id)-单表删除

RequestResult Delete(TEntity entity)-单表查询

调用方式以Delete(int id)举例，Delete(TEntity entity)调用方式可参照update。

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

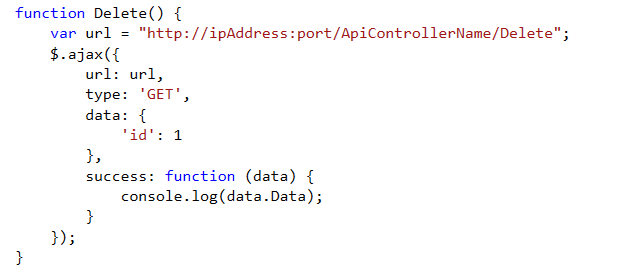
HttpResponseMessage response = myHttpClient.GetAsync("Delete?id="+ id).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

T entity = JsonConvert.DeserializeObject<T>(result.Data.ToString());

### 前端js调用



## SelectPagerByColumns

RequestResult SelectPagerByColumns(RequestData requestData) -分页单表集合查询

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

Pager pager = new Pager()

{

PageIndex = 1,

PageSize = 1

};

List<string> columnNames = new List<string>()

{

"Name"

};

T entity = new T()

{

Name = "name"

};

Dictionary<string, string> requestData = (new RequestData<T> { Pager = pager, Entity = entity, Columns = columnNames }).Serialize();

FormUrlEncodedContent content = new FormUrlEncodedContent(requestData);

HttpResponseMessage response = myHttpClient.PostAsync("SelectPagerByColumns",content).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

TargetPager<T> targetPager = JsonConvert.DeserializeObject<TargetPager<T>>(entity.Data.ToString());

### 前端js调用

和Selectby调用方式相同，只是需要写上PagerJsonString的内容PageIndex，PageSize。

## SelectAllByPager

RequestResult SelectAllByPager(RequestData requestData) – 单表集合分页查询

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

Pager pager = new Pager()

{

PageIndex = 1,

PageSize = 1

};

Dictionary<string, string> requestData = (new RequestData<T> { Pager = pager }).Serialize();

FormUrlEncodedContent content = new FormUrlEncodedContent(requestData);

HttpResponseMessage response = myHttpClient.PostAsync("SelectAllByPager",content).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

TargetPager<T> targetPager = JsonConvert.DeserializeObject<TargetPager<T>>(entity.Data.ToString());

### 前端js调用

和Selectby调用方式相同，只是需要写上PagerJsonString的内容PageIndex，PageSize，

EntityJsonString和ColumnNamesJsonString可赋值为null

## SelectAllByPagerWithChild

RequestResult SelectAllByPagerWithChild(RequestData requestData)-附加子对象集合分页查询

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

Pager pager = new Pager()

{

PageIndex = 1,

PageSize = 1

};

Dictionary<string, string> requestData = (new RequestData<T> { Pager = pager }).Serialize();

FormUrlEncodedContent content = new FormUrlEncodedContent(requestData);

HttpResponseMessage response = myHttpClient.PostAsync("SelectAllByPagerWithChild",content).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

TargetPager<T> targetPager = JsonConvert.DeserializeObject<TargetPager<T>>(entity.Data.ToString());

### 前端js调用

和Selectby调用方式相同，只是需要写上PagerJsonString的内容PageIndex，PageSize，

EntityJsonString和ColumnNamesJsonString可赋值为null

## Add

RequestResult Add(TEntity entity) -单表新增

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

Dictionary<string, string> entityDictionary = new Dictionary<string, string>();

entityDictionary.Add("Id", "124");

entityDictionary.Add("Name", "ClassA");

FormUrlEncodedContent content = new FormUrlEncodedContent(entityDictionary);

//System.Net.Http.HttpResponseMessage response = myHttpClient.PostAsync(“Add”, content).Result;

System.Net.Http.HttpResponseMessage response = myHttpClient.PostAsync(baseurl + "/ClassAApi/Edit", content).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

T entity = JsonConvert.DeserializeObject<T>(result.Data.ToString());

### 前端js调用

var url= "http://ipAddress:port/ApiControllerName/Add";

var data = "Id=1&Name=ClassName";

$.ajax({

url: url,

type: "post",

contentType: 'application/x-www-form-urlencoded',

data: data,

success: function (result) {

alert(result.message);

},

error: function (result) {

JSON.stringify(result);

}

});

## Edit

RequestResult Edit(TEntity entity) -单表修改

### C#调用

同Add

### 前端js调用

同Add

## UpdateFromBody

RequestResult UpdateFromBody(([FromBody] dynamic entity)-单表修改，支持传入的部分属性修改。

### C#调用

HttpClient myHttpClient = new HttpClient();

string url = "http://ipAddress:port/ControllerName/";

myHttpClient.BaseAddress = new Uri(url);

object currentEntity = new { Id = 1, Name = "Class1",};

HttpResponseMessage response = myHttpClient.PostAsJsonAsync("Update",currentEntity).Result;

获取调用结果：

RequestResult result = response.Content.ReadAsAsync<RequestResult>().Result;

T entity = JsonConvert.DeserializeObject<T>(result.Data.ToString());

### 前端js调用

同Update

## SelectAllByColumns

RequestResult SelectAllByColumns(TEntity entity)-单表集合查询

前端调用方式：

## SelectByColumns

RequestResult SelectByColumns(TEntity entity)-单表单个查询

前端调用方式：

## SelectPagerByColumns

RequestResult SelectPagerByColumns(dynamic pagerSearch)-单表集合分页查询

前端调用方式：

注意：一次只修改部分属性的值,目前App WebApi支持两种实体的部分属性的更新的方法，

1.public virtual RequestResult Edit(TEntity entity)

2.public virtual RequestResult UpdateFromBody([FromBody] dynamic entity)

方法一的使用：

目前方法一的更新只支持实体对应字段的更新，如果该实体有非引用数据的子类，并且需要同时更新子类，那么需要在查找时将子类的信息组装好，并且在更新时需要同时更新子类。如果有额外的逻辑可以重写基类方法的protected virtual void AdditionalBusinessLogic(TEntity entity, TEntity newEntity)。

方法二的使用：

与注意事项3相同，此处传回的json对象键值需要和相应实体的属性名保持一致且不区分大小写（在实体的属性中不能有大小写转换后相同改的属性名）。

注意事项：目前方法二的更新只支持实体对应字段的更新，如果该实体有非引用数据的子类，并且需要同时更新子类，那么需要在查找时将子类的信息组装好，并且在更新时需要同时更新子类。如果有额外的逻辑可以重写基类方法的protected virtual void AdditionalBusinessLogic(dynamic entity, TEntity newEntity)。

方法AdditionalBusinessLogic：

参数一entity：代表从前端接收的数据，newEntity代表经过处理后将所有与属性对相应的字段都赋好值的实体，如果没有额外逻辑，子类不需要重写，否则需要重写该方法。

备注

1，fromBody:在cation方法传入参数前添加[frombody]属性，参数将以一个整体的josn对象的形式传递。

2，不加fromBody:在cation方法传入参数前不添加[frombody]属性，参数将以表单的形式提交。