|  |
| --- |
| ItTrending |
| EFFC 3.5.3说明 |
| EFFC框架3.5.3版本调整说明 |

|  |
| --- |
| admin  2018-7-17 |

目录

[前言 2](#_Toc482027473)

[修改说明 3](#_Toc482027474)

[出现的问题 6](#_Toc482027475)

[引用文献 7](#_Toc482027476)

[历史 8](#_Toc482027477)

# 前言

本文档用于说明EFFC 3.5.3版本的设计及改动说明，用于今后作为EFFC框架下开发指导之基础。

注意：该版本需要VS2017升级到15.7.5或以上的版本，SDK需要升级到2.1.2或以上的版本

# 修改说明

1. 升级.net core 2.0.3到.net core 2.1.1
   1. 从nuget中引用需要许可证接受
   2. Nupkg的包的版本都是从1.1.0开始，低于该版本的都是3.5.2的框架
2. LinqDLR2SQL调整如下
   1. LinqDLRColumn新增substring(起始位置，截取长度)方法,参数为(起始位置，截取长度)例子如下：

from t in LinqDLRTable.New("PO", "a")  
where t.field.substring (5,2) == “PO”

* 1. LinqDLRColumn新增count(别名[可选])方法,参数为别名， 例子如下：

from t in LinqDLRTable.New("PO", "a")  
select t.field.count(“c”)

* 1. LinqDLRColumn新增avg(别名[可选])方法, 参数为别名， 例子如下：

from t in LinqDLRTable.New("PO", "a")  
select t.field.avg(“c”)

* 1. LinqDLRColumn新增ConvertToString(转化后的总长度[可选，默认50])方法, 参数为别名， 例子如下：

from t in LinqDLRTable.New("PO", "a")  
select t.field.converttostring(20)

* 1. LinqDLRColumn新增ToDateTime()方法, 用于将string类型转化成datetime类型无参数， 例子如下：

from t in LinqDLRTable.New("PO", "a")  
select t.field.ToDateTime ()

* 1. LinqDLRColumn为了配合GroupBy操作，因此调整sum，max，min，count方法，说明如下

当方法后面无参数时则生成的表达式不带别名，如：t.id.sum() 则识别为 sum(t.id),而t.id.sum(“”)会识别为sum(t.id) as sum\_xxx，而t.id.sum(“s”)会识别为sum(t.id) as s

1. EWRA针对EWRABusiModule做调整，使其具备更高的扩展性
   1. 新增RestInvokeFilterLogic用于接口执行之前进行是否要执行的判定和过滤控制，并返回StatusCode和错误信息，与AuthorizationLogic不同的是
      1. AuthorizationLogic只做权限判定和加载，返回的StatusCode是固定为401的，并且AuthorizationLogic会优先执行
      2. RestInvokeFilterLogic为自定义过滤控制器，根据路由和可执行列表来判定是否需要执行，在AuthorizationLogic之后执行

RestInvokeFilterLogic的扩展对象在OnUsed的时候会加载，通过  
RestInvokefilterLogicType来指定过滤器对象

* 1. 新增RestRouteContext抽象类，用于路由识别和搜索的扩展

1. DBExpress新增CreateTable、DropTable、AlterColumn、CopyTable、CopyData指令的支持
   1. CreateTable指令格式如下

{

"$acttype": "CreateTable",

"$table": "table的名称",

"column1": {

"$datatype": "数据类型，参见附录1",

"$precision": "数据精度，int类型，当数据类型varchar，nvarchar，numberic有效",

"$scale": "比例标准，int类型，仅numberic类型有效",

"$default": "默认值或表达式，当前支持的标准表达式为:now()(mysql,sqlite不支持),increament(整数)(oracle,sqlite不支持)",

"$isnull": "是否允许为空，为bool类型"

},

"$pk": [ "column1", "column2"], //PK设定，为字符串数组，值为上面定义的栏位名称

}

* 1. DropTable指令格式如下：

{

"$acttype": "DropTable",

"$table": "table的名称"

}

* 1. AlterColumn指令格式如下  
     {

"$acttype": "AlterColumn",

"$table": "table的名称",

"column1": {

"$alter\_action": "修改方式，默认为none，可选值为：none(不处理),rename(重命名,sqlite不支持), add（新增一个栏位,sqlite不支持）,modify（修改一个栏位,sqlite不支持）,drop（删除栏位,sqlite不支持）,drop\_add(先删除栏位，再新增,sqlite不支持)",

"$new\_name": "新的栏位名称（rename有效）",

"$datatype": "数据类型（none、drop、rename无效），目前支持类型参见附录1",

"$precision": "数据精度（none、drop、rename无效），int类型，当数据类型varchar，nvarchar，numberic，random,random\_code有效",

"$scale": "比例标准（none、drop、rename），int类型，仅numberic类型有效",

"$default": "默认值或表达式（none、drop、modify、rename无效），当前支持的标准表达式为:now()(mysql,sqlite不支持),increament(整数)(oracle,sqlite不支持)",

"$isnull": "是否允许为空（none、drop、rename），为bool类型"

}

* 1. CopyTable指令，受各个DB的约束设定，复制的table可能不包含原表的PK等约束，格式如下：

{

"$acttype": "CopyTable",

"$table": "table的名称",

"$to\_table": "目标表",

"$with\_data": "是否包含数据"

",

}

* 1. CopyData指令，不同于CopyTable，该指令必须是两个table都存在的情况下才能执行，格式如下：

{

"$acttype": "CopyData",

"$table": "table的名称",

"$to\_table": "目标表",

"$if\_not\_exists": true,//为true的时候添加not exists的where条件，否则不过滤，注意：该过滤条件不是根据每笔资料进行比对的，而是根据目标表是否存在数据来作为判定依据，因此会出现只要目标表存在资料，无论与来源数据是否相同都不会执行写入操作

"column1 ": “目标表的栏位”,//指定来源表的栏位，可以为多个，值为目标表的栏位

……

"columnN": “目标表的栏位”,

"$where": “where表达式，可以参看EFFC3.0说明.txt中DBExpress的说明（$where部分），该where针对的是$table中的栏位”

}

1. DBExpress中的Table中，为了允许多个同名table（目前的table名作为key导致不能出现重复的情况），新增指代符号”~”，该符号不参与sql的组织，只是一种符号标记，如下情况：

Select a.\*,t11.value,t12.value from a

left join t1 t11 on t11.id=a.id

left join t1 t12 on t12.id=a.id2

Express的表达式如下

"$table":{

"a":"a",

"t1~1":{

"$as":"t11",

"$joinl":{

"$by":"a",

"$on":{

"id":"a.id"

}

}

},

"t1~2":{

"$as":"t12",

"$joinl":{

"$by":"a",

"$on":{

"id":"a.id2"

}

}

}

}

这样通过~1和~2使得DBExpress表达式可以容纳多个同名table

1. DBExpress新增datatype:guid,random,random\_code
   1. guid:无需设定精度
   2. random：随机数字码，需要设定精度
   3. random\_code：随机码，数字+大小写字母，需要设定精度
2. LinqDLR2SQL新增扩展DaoSqlGenerator，达成以下目标
   1. 可以生成Update，Insert，Delete，Query，QueryByPage的相关sql，并直接做DB操作
   2. 新增DoQuery，可以直接返回数据结果
   3. 对应的DaoLinqDLR2SqlExtend用于扩展出LinqDLR2Sql<TSource>的相关方法
   4. LamdaSQLObject新增column方法，该方法作用是将参数中的名称转为LinqDLRColumn，操作语法如下：

from t in LinqDLRTable.New("PO", "a")  
where t.column(“date”) == DateTime.Now.ToString(“yyyyMMdd”)  
select t;

1. EFFCWebMiddleWare新增一个请求资源Options，继承或重载该类可以实现以下操作
   1. ConvertExtTo：根据request的类型，进行转化，比如可以将html的静态请求类型转化成go这种动态类型（框架默认go为动态请求类型）
   2. IsStaticType：判定requestExtType是否为静态资源请求类型
   3. IsOpenStaticFileType：判定requestExtType是否为可以对外访问的静态资源请求类型
   4. StaticFileRootPath：静态资源的根路径
   5. PagePath4Forbidden：403个性化页面的路径，~为站点根路径
   6. PagePath4NotFound：404个性化页面的路径，~为站点根路径

# 计划升级的内容

1. 添加对SSE请求的处理逻辑
   1. 流程图如下



WebGo根据content-type执行sse判定，如果是sse则呼叫business模块调用logic中的sse\_init方法执行初始化，并决定采用何种执行方案；

WebGo根据回传的数据信息来决定执行方案，然后开始正式调用对应的logic执行逻辑

# 附录

1. DBExpress支持的数据类型如下：
   1. varchar：单字节字符串
   2. nvarchar：双字节字符串
   3. int:整数类型
   4. numberic：数字浮点数类型
   5. bit:布尔值类型，1和0来表示
   6. datetime:时间类型
   7. text：多行文本
   8. file：存放文件路径
   9. picture：存放图片路径
   10. guid：自动生成guid串，DB中的实际类型为varchar(50)
   11. random: 随机数字码，需要设定精度
   12. random\_code: 随机码，数字+大小写字母，需要设定精度
   13. map\_xy:存放地图经纬度
   14. pic\_xy：存放图片上XY轴的坐标
   15. json：存放JSON串

# 出现的问题

# 引用文献

1. RestFUL架构参考：http://www.ruanyifeng.com/blog/2014/05/restful\_api.html

# 历史