关于Lind.DDD框架里API框架的技术点说明

讲解：张占岭

主要框架：Lind.DDD

目录

[关于Lind.DDD.Authorization 2](#_Toc2422)

[关于授权的原理 2](#_Toc9535)

[关于ApiValidateModelConfig 2](#_Toc12455)

[关于Lind.DDD.CacheConfigFile 3](#_Toc17239)

[如何为你的API项目注入授权模块 4](#_Toc14589)

[关于服务端收取过滤器ApiValiadateFilter 5](#_Toc10646)

[如何在客户端生产加密授权串 5](#_Toc4890)

[关于请求类与响应类 5](#_Toc21717)

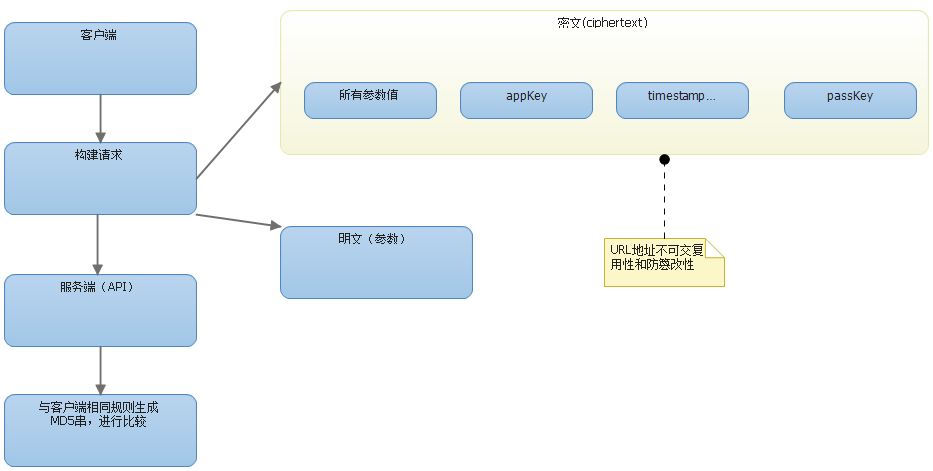
[客户端如何做分页 8](#_Toc17475)

### 关于Lind.DDD.Authorization

Lind.DDD为我们提交了强大的API校验组件，只需要在全局或者要授权的controller上添加对应的过滤器即可完成授权的过程，这样，你的API就安全多了。

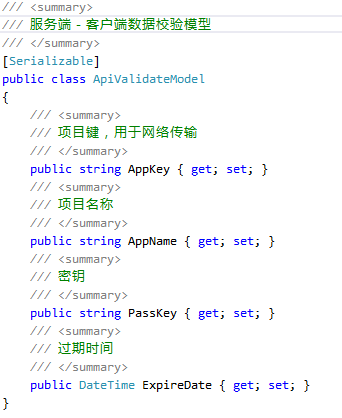
### 关于授权的原理

客户端在向API服务端获取数据时，需要先申请一个**appkey**作为自己的标识，当然这也是双方约定的，我们可以叫做公钥，而真正做数据校验的不是它，它只是一个唯一标识，对外公开，真正做数据加密的叫**passKey**，它是保存在双方内部的，不对外公开，所以叫密钥，在客户端向API服务端通讯时，需要将这个**passKey**连同参数和**appkey**传到服务端，由服务端做相同的校验码生产逻辑，最后两者进行比较，相同即验证通过。



### 关于ApiValidateModelConfig

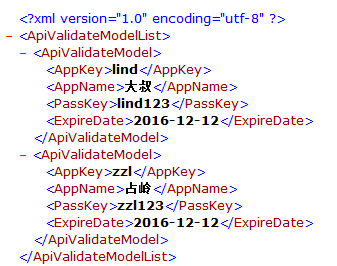
ApiValidateModelConfig主要在服务端存储所有被收取的客户端的信息，它是一个列表集合，由AppKey,AppName,PassKey,ExpireDate等元素组成，它们具体的含义如下：



而这个实体在服务端校验时，会从配置文件XML中反射出来，以遍进行比较，当然，你的配置文件如果没有修改，它会直接从内存里进行获取，这个逻辑由CacheConfig控制。

### 关于Lind.DDD.CacheConfigFile

它在早期的大叔框架里就已经出来了，主要用于做配置文件缓存的，当缓存文件被修改后，它的信息将重新被加载，否则将从内在中来进行获取，这个文件需要管理员在服务端进行维护，在添加和删除配置时，需要做修改，当然，我们也完全可以把它持久化到其它数据库里，如sqlserver,redis等介质。

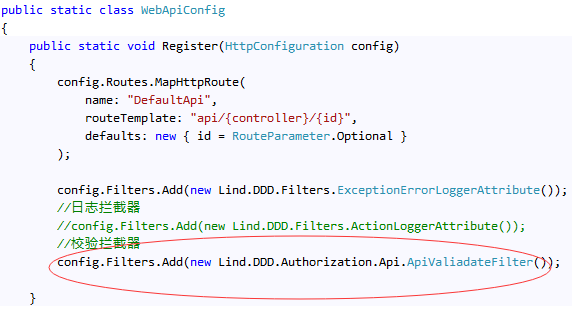


### 如何为你的API项目注入授权模块

为API项目注入授权功能很是容易，直接在对应的controller上添加过滤器Lind.DDD.Authorization.Api.ApiValiadateFilter即可。



或者在Global.asax里添加全局的过滤，也是可以的，值得注意的是，如果你添加的是全局过滤器，如果希望有一些Controller不被授权，即可以被匿名访问，那你也只需要为指定的控制器添加AllowAnonymous特性即可。



### 关于服务端收取过滤器ApiValiadateFilter

ApiValiadateFilter是服务端的收取核心组件，它会拦截指定的api控制器，然后进行授权检查，如果没有被标示AllowAnonymous，它将会进行校验，具体就是将请求参数进行排序，组件，并连同passkey(由客户端传来的appKey进行查询，得到的ApiValidateModel实体)生成新的MD5加密串，与客户端传过来的密钥进行对比，匹配即有效，否则返回403无权访问，最新的api校验的新功能如下：

1. 统一校验模块
2. 统一参数组合的生成
3. UTC时间戳的引入，参数有效性校验（1小时有效）
4. 双方约定的密钥，请求的防伪造

### 如何在客户端生产加密授权串

Lind.DDD框架为我们提供了生成请求密钥的方法，你需要做的只是将所有参数添加到字典，然后调用对应的方法即可，这对于.net开发人员来说，绝对是个福音！



### 关于请求类与响应类

API服务端与客户端约定了请求与响应的模型，它们都是抽象类，提供最基础的功能，下面简单说一下：

* 请求对象

对应于DTO请求类，它继承自抽象类RequestBase，它提供了传输标示，分页，排序，筛选字段等功能。



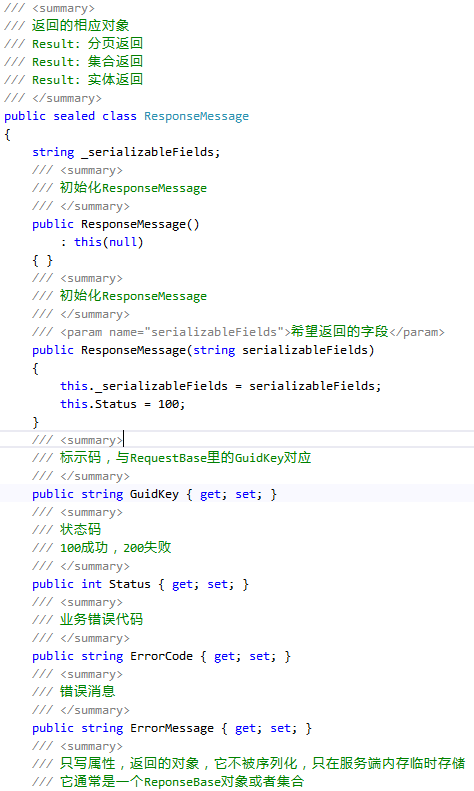
* 响应对象

对应于DTO响应类，它继承自抽象类ResponseBase，它提供了RequestBase里的传输标示，标明了是否为同一个请求，响应的字段等。



* 响应返回对象

对应于DTO的返回结果，它由密封类ResponseMessage提供，返回它的实例即可，它会提供返回状态码，唯一标识，返回对象，错误码和错误信息等。



### 客户端如何做分页

分页对于每个项目来说都是必要的，对于面向服务的API来说了是必须的，我们Lind.DDD对分页一定进行了封装，在这里为大家简单说一下。

统一的DTO请求基类RequestBase，它主要实现请求方调用时的字段过滤（ContainFields），分页控制（Page)，传输标示(GuidKey)，按字段排序时（Sort），请求方只需要传输相应的参数即可，代码如下：



在服务端进行分页方法实现时，返回统一的Lind.DDD.Paging.PagedList<T>对象，而在业务方法处理时，只需要调用请求类的GetPageParameters()方法即可拿到客户端传来的分页对象，而从数据库查出来的对象也可以使用MapToPage<T>()这个方法来映射成指的DTO对象，代码如下：

