基于BaGet搭建Nuget程序包服务器

BaGet简介

BaGet是一个开源的、轻量级的NuGet包服务器应用组件。BaGet源码托管地址为：https://github.com/loic-sharma/BaGet 。

BaGet是基于.NET Core开发的NuGet包服务器应用组件，因此需要运行环境安装.NET Core SDK，BaGet具有以下特性： *极速部署* 支持跨平台 *支持docker容器化部署* 支持云存储 *支持离线缓存* 支持包硬删除 * 配置持久化支持多种数据库类型

BaGet安装

> 注：以下安装在Windows操作系统中演示。

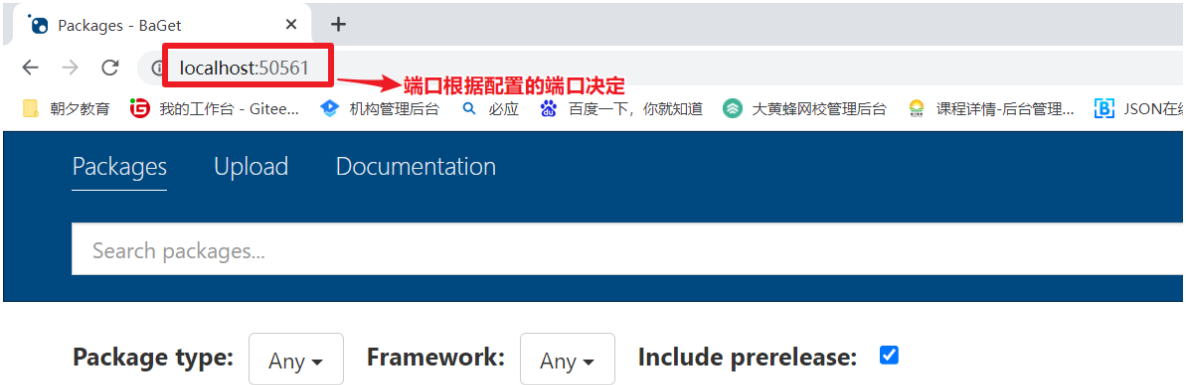
在文中已多次提到**极速安装**这个动作，那么BaGet的安装到底有那极速呢？让我们一起来体验一下吧！

1. 安装.NET Core SDK
2. 下载BaGet程序压缩包，点击这里
3. 解压刚下载的BaGet程序压缩包，打开命令行，定位到BaGet程序根目录，运行命令`dotnet BaGet.dll`
4. 在浏览器中打开地址：http://localhost:端口号/ 注意：这里端口号根据项目的配置来的

怎么样，是不是超级简单、超级快捷呢！！！

运行命令示意图：

浏览器中看到的效果如图：



从图中可以看到，BaGet监听了`50561`端口，如果你需要修改端口，则打开**launchSettings.json**配置文件，修改即可

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:50557/",
      "sslPort": 0
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development",
        "ASPNETCORE_HOSTINGSTARTUPASSEMBLIES":
"Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation"
      }
    },
    "BaGet": {
      "commandName": "Project",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development",
        "ASPNETCORE_HOSTINGSTARTUPASSEMBLIES":
"Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation"
      },
      "applicationUrl": "http://localhost:50561/"
    }
  }
}
```

特别注意：以上以`dotnet BaGet.dll`方式运行的BaGet包服务还没有设置API密钥，也就是说任何知道BaGet地址的人都是可以对其进行操作，如发布包，更新包，删除包等。所以，为了安全起见，建议为你的私有BaGet包服务配置一个API密钥。配置方式也非常简单，还是打开**appsettings.json**配置文件，修改选项，如下：

```
{
  "ApiKey": "zhaoxiAdvanced.NugetKey", // 这里修改成你的密钥即可(任意字符串)
  //...
}
```

更多BaGet的配置，请见BaGet官网配置说明

发布程序包到BaGet服务

访问地址：http://localhost:端口号/

如下图所示：

点击图中

①所示的**Upload**按钮，界面将切换到上传NuGet包界面，这里展示了几个重要的信息： *BaGet包服务索引的地址为图中*

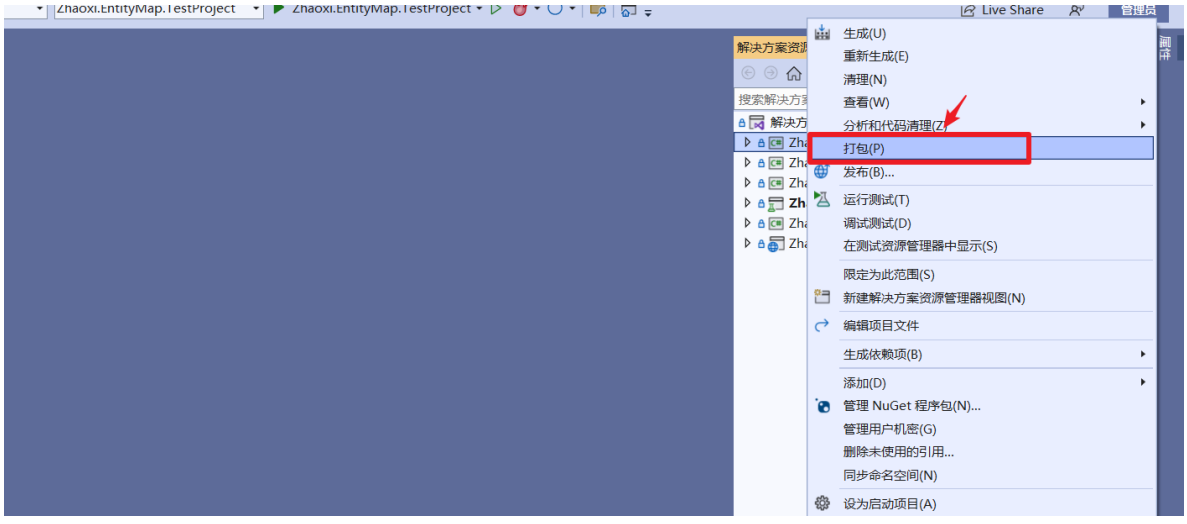
②标注的地址 使用4种不同的命令行(分别为：.NET CLI, NuGet CLI, Paket CLI, PowerShellGet)发布NuGet包

> *注：如果你不喜欢命令行的方式发布NuGet包，推荐使用**NuGet Package Explorer**发布、更新NuGet包，可以在Windows 10的应用商店下载、安装**NuGet Package Explorer**。* 使用命令行发布NuGet包的命令如标记③所示。

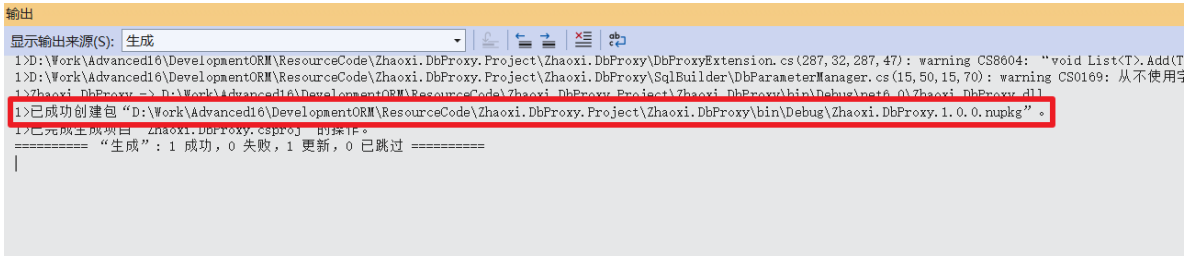
为了演示如何将自制NuGet包发布、更新到BaGet服务，我们就使用我们的自研发ORM框架，打开命令提示符工具，执行NuGet包的发布命令，如下：

推送的包从哪儿来呢？准备要发布的程序包。

再右键单击项目**Zhaoxi.DbProxy->** 打包**，如图：



打包结果如图：

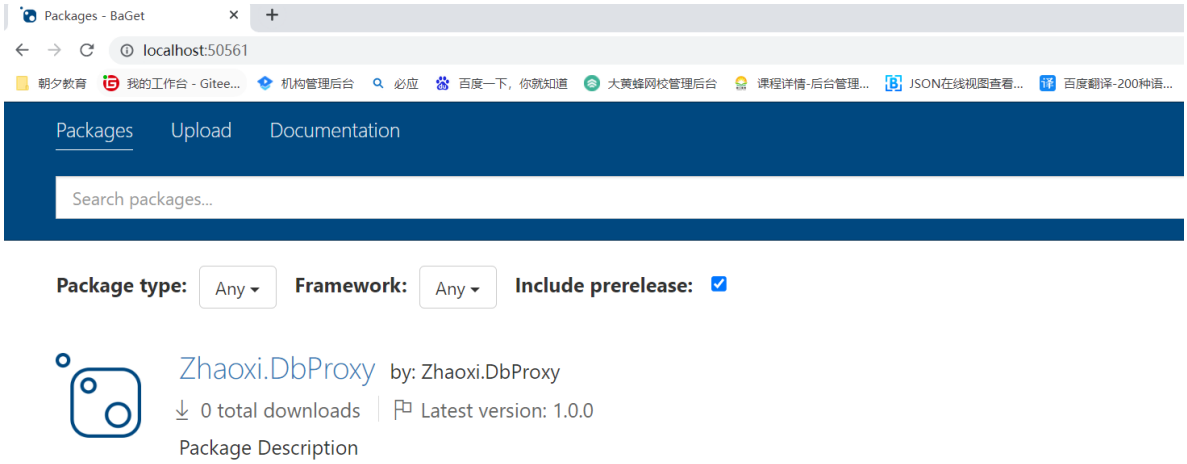


在.nupkg所在目录打开命令提示符工具，执行NuGet包的发布命令，如下：

> 注：本示例并未设置API密钥，如果你设置了BaGet的API密钥，请在发布命令中附加API密钥的参数(-k)，如：`dotnet nuget push -s http://localhost:5000/v3/index.json -k NUGET-SERVER-API-KEY WeChatPay.1.0.0.nupkg`

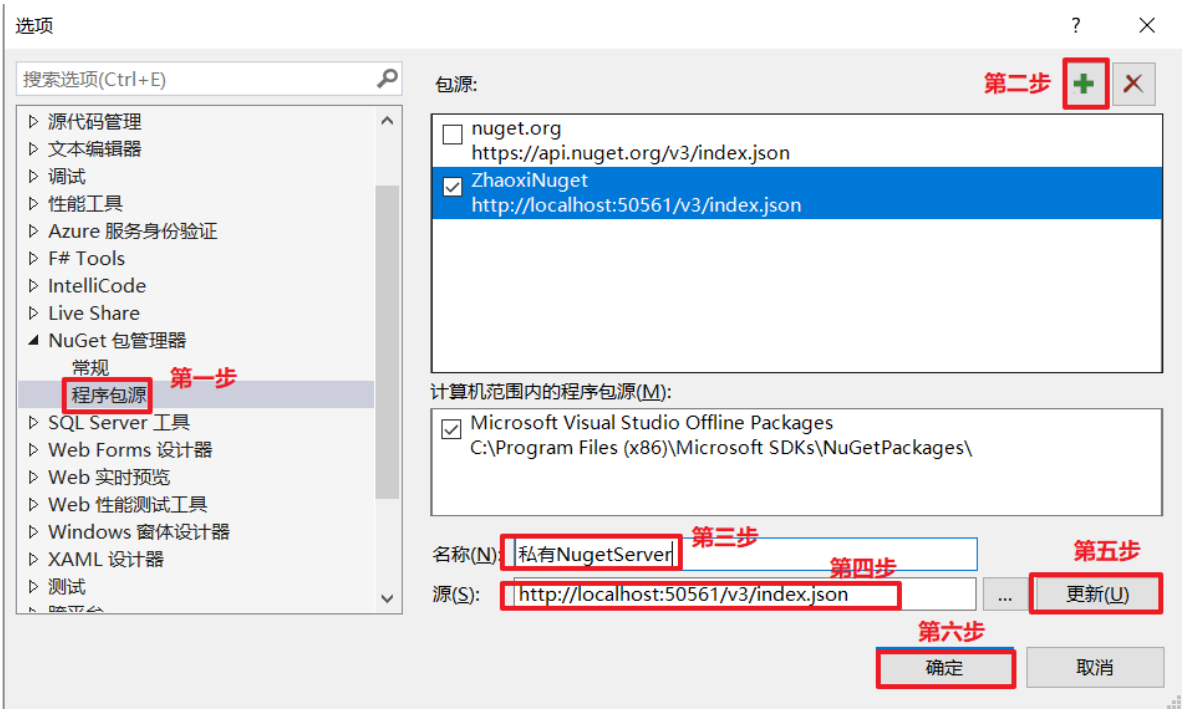
看到图中所示的“已推送包”表示NuGet包成功发布到了刚才用BaGet搭建的私有NuGet包服务器。

刷新浏览器地址：http://localhost:50561/ ，NuGet包列表中出现了WeChatPay这个NuGet包，如图：



安装私有NuGet包

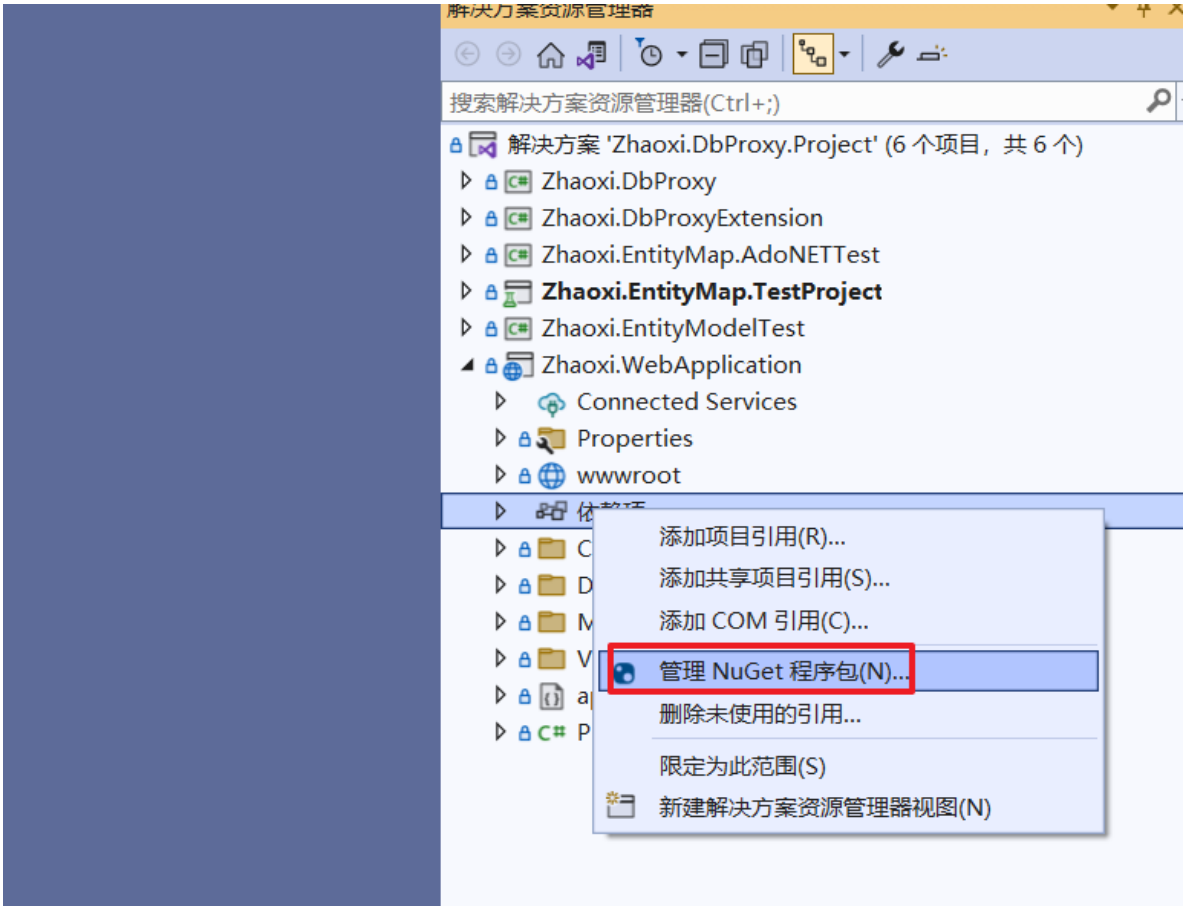
将NuGet包成功上传到BaGet搭建的私有服务器之后，便可下载安装和使用了，接下来我们来配置Visual Studio的Nuget包源以添加私有BaGet包源地址。打开Visual Studio的**选项** -> **NuGet包管理器** -> **程序包源**，依次完成下图中的操作：



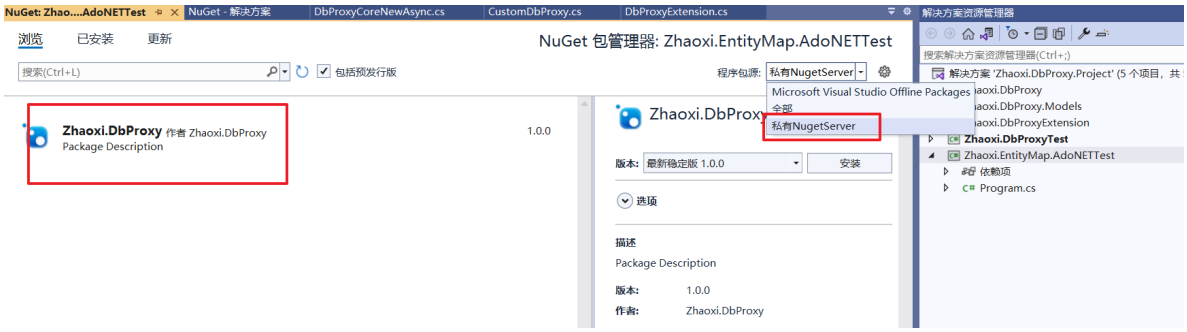
> 步骤2：点击+号，新增一个程序包源项 步骤3：选中步骤2中新增的项，在3处的广西框中填入程序包源的名称(可任意取名) 步骤4：填入私有NuGet包的服务索引地址 步骤5：点击**更新**按钮，以更新程序包源的信息 步骤6：点击**确定**按钮，以保存新增的程序包源信息

进入一个需要引入Nuget包的项目

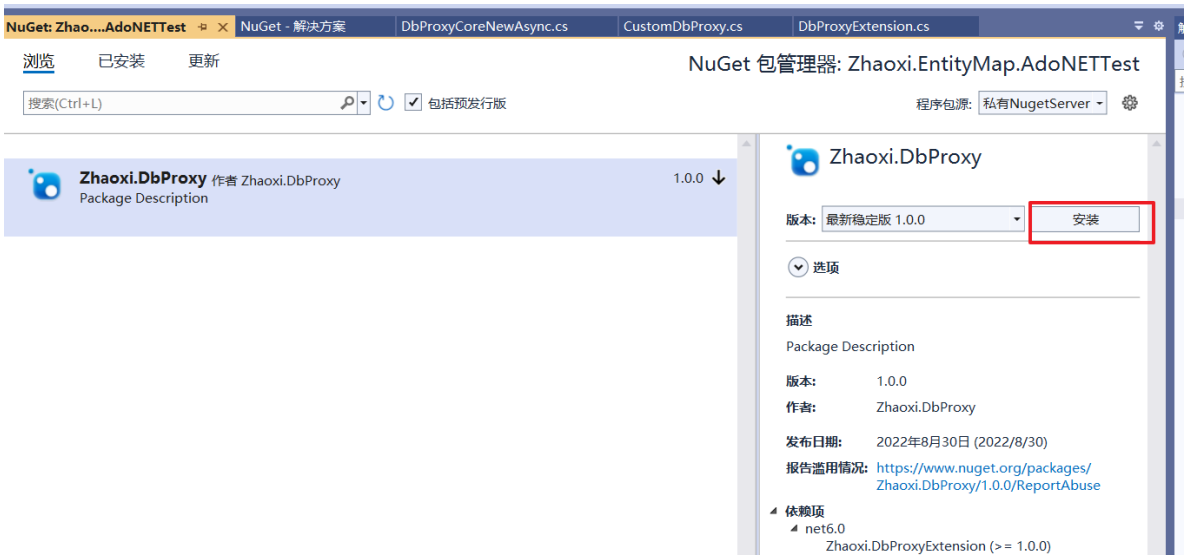
右键单击项目的**依赖项** -> **管理NuGet程序包**，如图：



在打开界面的右上角可以看到程序包源的下拉框，下拉列表中会列出上面新增的**私有NuGetServer**程序包源的选项，点击选中此项，NuGet列表会自动刷新，之后将看到我们上传的自制程序包**Zhaoxi.DbProxy**，如图：



选中**Zhaoxi.DbProxy**包，点击右侧的**安装**按钮，以在当前项目中安装此程序包，如图：



再次回到Visual Studio编辑器主界面，现在我们便可以调用**Zhaoxi.DbProxy**这个包中可访问的资源了

删除特定包

```
dotnet nuget delete -s http://localhost:50561/v3/index.json Zhaoxi.DbProxy 1.0.0
-k 123
dotnet nuget delete -s http://localhost:50561/v3/index.json
Zhaoxi.DbProxyExtension 1.0.0 -k 123
```

常见问题

删除 `nuget` 包后，同版本号 `nuget` 包无法进行安装。

```
 >dotnet nuget push -s http://localhost:5000/v3/index.json NugetTool.1.0.0.nupkg
-k "ggcyadmin@@"
 正在将 NugetTool.1.0.0.nupkg 推送到 'http://localhost:5000/api/v2/package'...
  PUT http://localhost:5000/api/v2/package/
  Conflict http://localhost:5000/api/v2/package/ 400 毫秒
 要跳过已发布的包，请使用 --skip-duplicate 选项
 error: Response status code does not indicate success: 409 (Conflict).
```

对应 nuget 默认目录中 `packages`，文件仍然存在。

> 启用包硬删除
>
> 为了防止“左垫”问题，BaGet 的默认配置不允许删除包。每当 BaGet 收到包删除请求时，它将改为“取消列出”该包。未列出的包是不可发现的，但如果您知道包的 id 和版本，仍然可以下载。您可以通过设置`PackageDeletionBehavior`

```
{ "PackageDeletionBehavior" : "HardDelete" , }
```

> 启用包覆盖
>
> 通常，如果 id 和 version 已被占用，BaGet 将拒绝包上传。您可以通过设置将 BaGet 配置为覆盖已经存在的包`AllowPackageOverwrites`

```
{ "AllowPackageOverwrites" : true , }
```

重启服务后，再次进行上传操作，提示成功。