Qomicex.Core 开发教程

概述

欢迎来到 Qomicex.Core 开发教程！本教程将帮助您了解 Qomicex.Core 的架构设计、核心功能和开发流程，为您快速上手开发基于 Qomicex.Core 的 Minecraft 启动器或相关工具提供指导。

项目简介

Qomicex.Core 是一个功能强大的 Minecraft 启动器核心库，提供了完整的启动流程管理、游戏资源管理、账户认证、Mod 管理等功能。它采用模块化设计，具有高度的可扩展性和可定制性，适用于各种规模的 Minecraft 相关应用开发。

主要特性

- **完整的启动流程管理**：从游戏安装到启动的完整流程
- **灵活的账户认证系统**：支持多种认证方式，包括 Microsoft、Yggdrasil 等
- **全面的资源管理**：游戏版本下载、Mod 管理、资源包管理等
- **强大的 Mod 支持**：支持 Forge、Fabric、Quilt 等多种 Mod 加载器
- **跨平台兼容**：支持 Windows、Linux 和 macOS 系统
- **模块化设计**：高度模块化的架构，易于扩展和定制
- **异步编程**：大量使用异步 API，提供良好的用户体验

目录

[[toc]]

在我们开始之前

在开始使用 Qomicex.Core 之前，您需要对开发环境和项目配置有一定的了解。请先阅读 在我们开始之前 章节，确保您的开发环境已正确配置。

安装并配置 Qomicex.Core

在配置好开发环境后，您需要安装并配置 Qomicex.Core 库。请阅读 安装并配置 Qomicex.Core 章节，了解如何安装和配置 Qomicex.Core。

验证模型

Qomicex.Core 提供了多种 Minecraft 验证模型，包括离线验证、Yggdrasil 验证和 Microsoft 验证。请阅读 验证模型 章节，了解如何使用这些验证模型。

安装器

Qomicex.Core 提供了多种 Mod 加载器的安装器，包括 Forge、Fabric、Quilt、LiteLoader 和 OptiFine。请阅读 安装器 章节，了解如何使用这些安装器。

资源补全器

资源补全器负责下载和管理 Minecraft 游戏所需的各种资源文件。请阅读 资源补全器 章节，了解如何使用资源补全器。

快速开始

环境要求

- .NET 8.0 或更高版本
- Visual Studio 2022 或其他兼容的 IDE
- Git 版本控制工具

创建第一个项目

```
# 创建项目
dotnet new console -n QomicexDemo
cd QomicexDemo

# 添加 NuGet 源（如果需要）
dotnet nuget add source https://api.nuget.org/v3/index.json --name nuget.org

# 添加 Qomicex.Core 依赖
dotnet add package Qomicex.Core
```

编写简单的启动器

```
using Qomicex.Core;
using Qomicex.Core.Modules.Helpers.Account.Microsoft;
using Qomicex.Core.Modules.Helpers.Resources;
using Qomicex.Core.Modules.Launcher;
using System;
using System.Threading.Tasks;

namespace QomicexDemo
{
    class Program
    {
        static async Task Main(string[] args)
        {
            Console.WriteLine("Qomicex.Core 启动器示例");
```

```
            try
            {
                // 1. 初始化核心组件
                var gameResourceHelper = new GameResourceHelper();
                var microsoftAuth = new Microsoft();

                // 2. 获取游戏版本列表
                Console.WriteLine("获取游戏版本列表...");
                var versions = await gameResourceHelper.GetMinecraftListAsync(1);
                Console.WriteLine($"可用版本数: {versions.Count}");

                // 3. 打印最新版本
                var latestVersion = versions.Find(v => v.Type == "release");
                Console.WriteLine($"最新正式版: {latestVersion?.Id}");

                // 4. 启动游戏（简化示例）
                var launcher = new Qomicex.Core.Modules.Launcher.Launcher();
                var launchParams = new LauncherParam
                {
                    GameDir = @"C:\Users\YourName\.minecraft",
                    Version = latestVersion?.Id,
                    JavaPath = "java",
                    MaxMemory = 4096,
                    Username = "Player"
                };

                Console.WriteLine("正在启动游戏...");
                var result = await launcher.StartGameAsync(launchParams);
                Console.WriteLine("游戏启动状态: " + result);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"发生错误: {ex.Message}");
            }

            Console.WriteLine("按任意键继续...");
            Console.ReadKey();
        }
    }
}
```

架构设计

核心架构

Qomicex.Core 采用三层架构设计：

1. **基础层**：包含通用工具、数据模块、日志分析等基础功能
2. **核心服务层**：提供游戏资源管理、账户认证、启动器核心功能
3. **业务逻辑层**：处理游戏启动流程、Mod 管理等复杂业务逻辑

模块架构

Qomicex.Core 采用模块化设计，每个功能模块独立实现，提供清晰的接口：

- **数据模块**：定义核心数据结构和配置
- **工具模块**：提供通用的工具方法和功能
- **账户模块**：处理用户认证和账户管理
- **资源模块**：管理游戏资源、Mod、资源包等
- **启动器模块**：负责游戏启动流程管理

核心功能开发

1. 游戏版本管理

```
using Qomicex.Core.Modules.Helpers.Resources;

var resourceHelper = new GameResourceHelper();

// 获取版本列表
var versions = await resourceHelper.GetMinecraftListAsync(1);

// 根据版本ID获取详细信息
var versionInfo = await resourceHelper.GetMinecraftVersionManifest(1);

// 下载游戏资源
var downloader = new ResourceDownloader();
await downloader.DownloadGameResourcesAsync("1.20.1", @"C:\Game\.minecraft");
```

2. 账户认证

Microsoft 账户认证

```
using Qomicex.Core.Modules.Helpers.Account.Microsoft;

var microsoftAuth = new Microsoft();

// 获取授权 URL
var authUrl = microsoftAuth.GetAuthorizationUrl();
Console.WriteLine($"请访问以下 URL 进行授权: {authUrl}");

// 获取用户输入的授权码
```

```
Console.Write("请输入授权码: ");
var code = Console.ReadLine();

try
{
    var authResult = await microsoftAuth.AuthorizeAsync(code);
    Console.WriteLine($"访问令牌: {authResult.AccessToken}");
    Console.WriteLine($"刷新令牌: {authResult.RefreshToken}");
    Console.WriteLine($"过期时间: {authResult.ExpiresIn} 秒");
}
catch (Exception ex)
{
    Console.WriteLine($"授权失败: {ex.Message}");
}
```

Yggdrasil 认证

```
using Qomicex.Core.Modules.Helpers.Account.Yggdrasil;

var yggdrasilAuth = new Yggdrasil();

// 登录
var loginResult = await yggdrasilAuth.LoginAsync("username", "password");

// 验证访问令牌
var isValid = await yggdrasilAuth.ValidateAsync(loginResult.AccessToken);

// 刷新访问令牌
var refreshResult = await yggdrasilAuth.RefreshAsync(
    loginResult.AccessToken,
    loginResult.ClientToken
);
```

3. Mod 管理

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");

// 搜索 Mod
var searchResults = await curseForgeMods.SearchAsync("sodium", "1.20.1");

// 获取 Mod 详情
var modDetails = await curseForgeMods.GetModDetailsAsync("sodium");
```

```
// 下载 Mod
await curseForgeMods.DownloadModAsync("sodium", "0.4.14",
@"C:\Game\.minecraft\mods\sodium.jar");
```

4. 游戏启动

```
using Qomicex.Core.Modules.Launcher;

var launcher = new Launcher();

var launchParams = new LauncherParam
{
    GameDir = @"C:\Game\.minecraft",
    Version = "1.20.1",
    JavaPath = @"C:\Program Files\Java\jre-17\bin\javaw.exe",
    MaxMemory = 4096,
    MinMemory = 1024,
    WindowWidth = 1280,
    WindowHeight = 720,
    Fullscreen = false,
    Server = null,
    Username = "Player",
    AuthenticationToken = "your-auth-token"
};

// 启动游戏
var result = await launcher.StartGameAsync(launchParams);

if (result)
{
    Console.WriteLine("游戏启动成功！");
}
else
{
    Console.WriteLine("游戏启动失败！");
}
```

高级主题

自定义资源源

```
using Qomicex.Core.Modules.Helpers.Resources;
```

```
// 创建自定义资源源
var customSource = new ResourceSource
{
    Name = "My Custom Source",
    BaseUrl = "https://my-custom-mirror.com/",
    GameManifestUrl = "https://my-custom-mirror.com/game_manifest.json",
    LibraryUrl = "https://my-custom-mirror.com/libraries/",
    AssetUrl = "https://my-custom-mirror.com/assets/",
    VersionUrl = "https://my-custom-mirror.com/versions/"
};

// 使用自定义资源源获取版本列表
var resourceHelper = new GameResourceHelper(customSource);
var versions = await resourceHelper.GetMinecraftListAsync();
```

扩展 Mod 支持

```
using Qomicex.Core.Modules.Helpers.Installers;

public class MyCustomModLoaderInstaller : InstallerBase
{
    public override Task<bool> InstallAsync(
        string versionId,
        string inheritsFromJson,
        string? para1,
        string? para2,
        string? para3,
        string? para4
    )
    {
        // 实现自定义 Mod 加载器的安装逻辑
        return Task.FromResult(true);
    }
}

// 使用自定义安装器
var customInstaller = new MyCustomModLoaderInstaller();
await customInstaller.InstallAsync(
    "1.20.1-mycustomloader-1.0",
    null,
    "1.0",
    "1.20.1",
    null,
    null
);
```

自定义启动参数

```
using Qomicex.Core.Modules.Launcher;

var launchParams = new LauncherParam
{
    GameDir = @"C:\Game\.minecraft",
    Version = "1.20.1",
    JavaPath = @"C:\Program Files\Java\jre-17\bin\javaw.exe",
    MaxMemory = 4096,
    MinMemory = 1024,
    // 自定义 JVM 参数
    JvmArgs = new List<string>
    {
        "-XX:+UseG1GC",
        "-XX:G1HeapRegionSize=4M",
        "-XX:+UnlockExperimentalVMOptions",
        "-XX:+UseCompressedOops"
    },
    // 自定义游戏参数
    GameArgs = new List<string>
    {
        "--fullscreen",
        "--server", "mc.hypixel.net",
        "--port", "25565"
    }
};
```

常见问题

1. 游戏启动失败

问题：游戏无法正常启动，提示找不到 Java 路径。

解决方案：

```
using Qomicex.Core.Modules.Helpers;

var javaHelper = new JavaHelper();

// 搜索系统中的 Java
var javaLocations = await javaHelper.SearchForJavaAsync();

if (javaLocations.Any())
{
```

```
    var firstJava = javaLocations.First();

    var launchParams = new LauncherParam
    {
        JavaPath = firstJava.Path,
        // 其他参数...
    };
}
else
{
    Console.WriteLine("未找到 Java 运行时，请确保已安装 Java。");
}
```

2. 资源下载失败

问题：游戏资源或 Mod 无法下载，网络连接失败。

解决方案：

```
using Qomicex.Core.Modules.Helpers.Resources;

var downloader = new ResourceDownloader();
downloader.MaxRetries = 3; // 增加重试次数
downloader.Timeout = TimeSpan.FromMinutes(5); // 增加超时时间

try
{
    await downloader.DownloadGameResourcesAsync("1.20.1", @"C:\Game\.minecraft");
}
catch (Exception ex)
{
    Console.WriteLine($"下载失败: {ex.Message}");
    Console.WriteLine("请检查网络连接或尝试使用其他资源源。");
}
```

性能优化

1. 资源缓存策略

```
using Qomicex.Core.Modules.Helpers.Resources;

// 配置资源缓存
var cacheOptions = new ResourceCacheOptions
{
    MaxCacheSize = 10 * 1024 * 1024 * 1024, // 10GB
```

```
    CacheExpiration = TimeSpan.FromDays(7), // 缓存过期时间
    UseDiskCache = true, // 启用磁盘缓存
    CacheLocation = @"C:\Game\.minecraft\cache" // 缓存位置
};

var resourceHelper = new GameResourceHelper(cacheOptions);

// 使用缓存获取版本信息
var versions = await resourceHelper.GetMinecraftListAsync(1);
```

2. 并发下载优化

```
using Qomicex.Core.Modules.Helpers.Downloader;

var downloader = new Downloader
{
    MaxConcurrentDownloads = 8, // 并发下载数
    ChunkSize = 4 * 1024 * 1024, // 分块大小
    RetryCount = 3 // 重试次数
};

var downloadTasks = new List<Task>
{
    downloader.DownloadAsync("http://example.com/file1.jar", "file1.jar"),
    downloader.DownloadAsync("http://example.com/file2.jar", "file2.jar"),
    downloader.DownloadAsync("http://example.com/file3.jar", "file3.jar")
};

await Task.WhenAll(downloadTasks);
```

贡献指南

我们欢迎开发者为 Qomicex.Core 项目做出贡献！以下是贡献步骤：

1. **Fork 项目**：在 GitHub 上 Fork 项目到自己的仓库
2. **克隆项目**：克隆自己的仓库到本地
3. **创建分支**：创建一个新的分支用于开发
4. **开发功能**：实现新功能或修复 bug
5. **编写测试**：为新功能编写测试代码
6. **提交修改**：提交修改到本地仓库
7. **创建 PR**：在 GitHub 上创建 Pull Request
8. **等待审核**：等待项目维护者审核您的 PR

许可证

Qomicex.Core 项目采用 MIT 许可证，允许您自由地使用、修改和分发项目代码。详细信息请查看项目根目录下的 LICENSE 文件。

联系方式

- **项目仓库**：https://github.com/TheMyceliumOfAntan/Qomicex.Avalonia
- **问题反馈**：https://github.com/TheMyceliumOfAntan/Qomicex.Avalonia/issues
- **讨论交流**：https://github.com/TheMyceliumOfAntan/Qomicex.Avalonia/discussions

其他资源

- **API 文档**：http://localhost:8083/api/
- **示例代码**：https://github.com/TheMyceliumOfAntan/Qomicex.Avalonia/tree/main/Examples
- **变更日志**：
  https://github.com/TheMyceliumOfAntan/Qomicex.Avalonia/blob/main/CHANGELOG.md

在我们开始之前

在我们正式开始使用 Qomicex.Core，您需要对您的项目属性做一下必要的检查和调整以满足 Qomicex.Core 的运行需要。

[[toc]]

运行时要求

您需要保证您项目的 .NET 框架至少运行在 .NET 6.0 及更高的版本当中。

目前受支持的 .NET 版本：

- .NET 6.0
- .NET 7.0
- .NET 8.0
- .NET 10.0（推荐）

::: warning Qomicex.Core 已不再支持 .NET 5.0 及更早版本。 :::

项目属性

由于 Windows 系统机制，您需要在项目属性中关闭 **首选 32 位** 的生成选项。否则您在使用 Qomicex.Core 部分组件时会出现预料之外的结果。

您需要在 Visual Studio 中切换到项目的属性页面，并找到 **首选 32 位** 的勾选框，并将其取消勾选。

32 位系统支持

::: warning Qomicex.Core 从项目立项开始就决定放弃对 32 位系统的全部支持，因为它已经过时了。 :::

开发环境要求

Visual Studio

- 推荐使用 Visual Studio 2022 或更高版本
- 确保已安装 .NET 桌面开发工作负载
- 安装 .NET 8.0 或更高版本的 SDK

Rider

- 推荐使用 Rider 2023 或更高版本
- 确保已安装 .NET 8.0 或更高版本的 SDK

其他开发工具

- .NET SDK 8.0 或更高版本
- Git 版本控制工具
- NuGet 包管理器

开发前准备

项目结构

建议您的项目结构如下：

```
MyMinecraftLauncher/
├── MyMinecraftLauncher.csproj
├── Program.cs
├── App.xaml
├── App.xaml.cs
├── Views/
│   ├── MainWindow.xaml
│   └── MainWindow.xaml.cs
└── ViewModels/
    └── MainWindowViewModel.cs
```

依赖项

在开始使用 Qomicex.Core 之前，您需要确保项目已经安装了以下 NuGet 包：

- `Qomicex.Core` - 主核心库
- `Newtonsoft.Json` - JSON 解析库（Qomicex.Core 依赖）
- `System.Net.Http` - HTTP 请求库（Qomicex.Core 依赖）

快速测试

在开始正式开发之前，您可以创建一个简单的测试项目来验证 Qomicex.Core 是否正常工作：

```
using System;
using System.Threading.Tasks;
using Qomicex.Core.Modules.Helpers.Resources;

class Program
{
    static async Task Main(string[] args)
    {
        try
        {
            Console.WriteLine("正在测试 Qomicex.Core...");
```

```
            // 测试游戏资源助手
            var resourceHelper = new GameResourceHelper();
            Console.WriteLine("正在获取 Minecraft 版本列表...");

            var versions = await resourceHelper.GetMinecraftListAsync(1); // 使用官方源
            Console.WriteLine($"成功获取到 {versions.Count} 个 Minecraft 版本");

            if (versions.Count > 0)
            {
                var latestVersion = versions[0];
                Console.WriteLine($"最新版本: {latestVersion.Id}");
                Console.WriteLine($"类型: {latestVersion.Type}");
                Console.WriteLine($"发布时间: {latestVersion.ReleaseTime}");
            }

            Console.WriteLine("\nQomicex.Core 测试成功! ");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"测试失败: {ex.Message}");
            if (ex.InnerException != null)
                Console.WriteLine($"内部错误: {ex.InnerException.Message}");
        }

        Console.WriteLine("\n按任意键退出...");
        Console.ReadKey();
    }
}
```

如果您能看到输出的 Minecraft 版本信息，说明 Qomicex.Core 已经正常工作了。

下一步

现在您已经准备好了开发环境，可以继续学习如何使用 Qomicex.Core 的其他功能：

- 安装并配置 Qomicex.Core
- 验证模型
- 安装器
- 资源补全器

安装并配置 Qomicex.Core

[[toc]]

从 NuGet 安装

您可以方便地从 NuGet 上搜索并下载 Qomicex.Core 的软件包。

Visual Studio 包管理器

在 Visual Studio 的包管理器中搜索 `Qomicex.Core` 并将其添加到您的项目中。

.NET CLI

要通过 .NET CLI 来安装 Qomicex.Core，您只需要将终端切换到包含 .csproj 文件的项目目录，并在终端中执行：

```
dotnet add package Qomicex.Core --version 1.0.0
```

PackageReference

您只需在项目的 **[项目名].csproj** 文件中添加：

```
<PackageReference Include="Qomicex.Core" Version="1.0.0" />
```

::: tip 其中，**1.0.0** 为 Qomicex.Core 的版本号，您可以将其替换为其他的版本号， 所有的发行版本都可以在 Qomicex.Core - Nuget 中查看。 :::

从源码引用

另外一种使用 Qomicex.Core 的方法是直接添加代码仓库到您的项目引用。

::: info 在执行下面的命令前，您可能需要先安装 Git CLI :::

克隆 Qomicex.Avalonia 仓库

使用命令行切换到项目解决方案的根目录，并在命令行中执行下面的代码来完成仓库的克隆：

```
git clone https://github.com/TheMyceliumOfAntan/Qomicex.Avalonia.git
```

使用 Git Submodule（子模块）的方式拉取 Qomicex.Core

::: tip Git Submodule 是一个非常实用的功能，在这里我们只展示了其最基本的用例。 在 Git 官方文档 中您可以查看到更多的使用案例。 :::

使用命令行切换到项目解决方案的根目录，并在命令行中执行下面的代码来完成仓库的克隆：

```
git submodule add https://github.com/TheMyceliumOfAntan/Qomicex.Avalonia.git
```

添加对 Qomicex.Core 的引用

接下来，在 Visual Studio 的 **解决方案资源管理器** 视图中，右键点击位于树状图顶层的解决方案名称。并选择 **“添加”-“现有项目”**，并在文件浏览窗口中找到刚刚克隆的 Qomicex.Avalonia 项目文件夹中的 **Qomicex.Core.csproj**。

接着，在 **解决方案资源管理器** 找到您需要引用 Qomicex.Core 的项目，并右键单击，选择 **“添加”-“项目引用”**。 最后在弹出窗口中勾选 Qomicex.Core 即可完成对其的引用。

使用前配置

修改默认连接数

在使用 Qomicex.Core 之前，您需要在程序的入口点（通常是 **App.xaml.cs** 或 **Program.cs**） 中添加一些代码来初始化 Qomicex.Core 的相关服务。

由于 .NET 运行时默认的最大连接数限制，在使用 Qomicex.Core 下载模块时可能会遭遇性能瓶颈。 因此，您需要在入口处添加下面的代码来修改默认的最大连接数：

```
using System.Net;

ServicePointManager.DefaultConnectionLimit = 512;
```

初始化下载服务

在初始化下载服务时您可以选择自定义请求时所使用的 User Agent（默认为 "QomicexCore"）。

```
// 可选：配置下载服务
HttpClientHelper.Ua = "MyCustomUserAgent";
```

配置微软登录验证器

关于 Azure Active Directory 应用具体的注册方法请参考 Microsoft 官方文档。

::: tip 在配置微软验证器前，您需要在 Azure 注册您的应用，并对其进行正确的配置。 在您完成配置之后，您会获得一串 Client ID。

相关资料：

- Azure 官网
- 设备代码流

:::

::: warning CLIENT ID 为敏感的个人凭据，请妥善保存 CLIENT ID 并不要将其泄露给其他人。 :::

```
using Qomicex.Core.Modules.Helpers.Account.Microsoft;

MicrosoftAuthenticator.Configure(new MicrosoftAuthenticatorAPISettings
{
    ClientId = "[YOUR CLIENT ID]",
    TenentId = "consumers",
    Scopes = new[] { "XboxLive.signin", "offline_access", "openid", "profile", "email" }
});
```

在取得 Client ID 后，将 **[YOUR CLIENT ID]** 替换为您的 Client ID。

配置 CurseForge API 服务（可选）

该服务为可选项目，如果您没有使用任何 CurseForge 相关服务，您可以忽略这个步骤。

::: tip 在使用 CurseForge 相关服务前，您需要准备 CurseForge 官方下发的 API KEY。 如果您还没有，请前往 申请页面 - CurseForge 来获得您的 API KEY。 :::

::: warning API KEY 为敏感的个人凭据，请妥善保存 API KEY 并不要将其泄露给其他人。 :::

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

CurseForgeAPIHelper.SetApiKey("[YOUR API KEY]");
```

将 **[YOUR API KEY]** 替换为您从 CurseForge 官方获取的 API KEY。

项目配置

应用程序清单文件

为了确保 Qomicex.Core 能够正常运行，您需要确保应用程序清单文件（app.manifest）包含以下配置：

```
<?xml version="1.0" encoding="utf-8"?>
<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1">
  <assemblyIdentity version="1.0.0.0" name="MyMinecraftLauncher"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
```

```
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <requestedExecutionLevel level="requireAdministrator" uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>
  <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
    <application>
      <supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}"/>
    </application>
  </compatibility>
</assembly>
```

构建配置

建议您在项目属性中进行以下配置：

1. **目标平台**：选择 "Any CPU" 并取消勾选 "Prefer 32-bit"
2. **目标框架**：选择 ".NET 8.0" 或更高版本
3. **输出类型**：选择 "Windows 应用程序"（如果是 GUI 应用）或 "控制台应用程序"（如果是命令行工具）

测试配置

为了确保您的应用程序能够正常工作，建议您在开发过程中进行以下测试：

1. 测试网络连接
2. 测试资源下载功能
3. 测试游戏启动功能
4. 测试不同版本的 Minecraft 兼容性
5. 测试不同操作系统的兼容性

常见问题

无法找到 Qomicex.Core 命名空间

确保您已经正确安装了 Qomicex.Core NuGet 包，并且项目引用已正确添加。

程序崩溃或无法启动

检查以下几个方面：

1. 确保您的应用程序具有足够的权限
2. 确保您的网络连接正常
3. 检查应用程序日志文件以获取详细错误信息

4. 尝试以管理员身份运行应用程序

资源下载失败

检查以下几个方面：

1. 确保您的网络连接正常
2. 尝试使用不同的资源源
3. 检查防火墙或安全软件是否阻止了下载请求
4. 尝试调整下载超时时间

游戏启动失败

检查以下几个方面：

1. 确保您已经正确安装了 Java
2. 检查 Java 路径是否正确
3. 检查游戏资源是否完整
4. 尝试调整内存分配

验证模型

在 Qomicex.Core 中，我们已经为开发者实现了您在 Minecraft 开发过程中可能遇到的所有的验证情形。

支持列表

模型名称	适用情形
OfflineAuthenticator	离线验证模型
YggdrasilAuthenticator	适用于旧版本的登录模型
MicrosoftAuthenticator	新版的微软验证模型

[[toc]]

OfflineAuthenticator（离线验证模型）

概述

离线验证模型是 Qomicex.Core 提供的最简单的验证方式，它不需要任何网络连接或用户凭证。使用该模型可以快速启动游戏，但只能进行单人游戏或局域网游戏。

使用方法

```
using Qomicex.Core.Modules.Helpers.Account;
using Qomicex.Core.Modules.Helpers.Account.Offline;

// 创建离线验证器实例
var authenticator = new OfflineAuthenticator
{
    Username = "Player" // 自定义玩家名称
};

try
{
    // 执行验证
    var result = await authenticator.AuthenticateAsync();

    if (result.IsSuccess)
    {
        Console.WriteLine("离线验证成功！");
        Console.WriteLine($"玩家名称: {result.PlayerName}");
        Console.WriteLine($"UUID: {result.UUID}");
```

```
    }
    else
    {
        Console.WriteLine($"验证失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"验证过程中发生错误: {ex.Message}");
}
```

特点

- 不需要网络连接
- 不需要用户凭证
- 验证过程非常快速
- 支持自定义玩家名称
- 可以快速启动游戏
- 适用于开发和测试环境

YggdrasilAuthenticator（旧版登录模型）

概述

Yggdrasil 验证模型是 Mojang 官方提供的验证方式，适用于使用 Mojang 账号登录的用户。该验证模型需要网络连接，并使用用户名和密码进行验证。

使用方法

```
using Qomicex.Core.Modules.Helpers.Account.Yggdrasil;

// 创建 Yggdrasil 验证器实例
var authenticator = new YggdrasilAuthenticator
{
    Username = "your-email@example.com", // 您的 Mojang 账号
    Password = "your-password" // 您的密码
};

try
{
    // 执行验证
    var result = await authenticator.AuthenticateAsync();

    if (result.IsSuccess)
    {
```

```
        Console.WriteLine("Yggdrasil 验证成功！");
        Console.WriteLine($"玩家名称: {result.PlayerName}");
        Console.WriteLine($"UUID: {result.UUID}");
        Console.WriteLine($"访问令牌: {result.AccessToken}");
        Console.WriteLine($"客户端令牌: {result.ClientToken}");
    }
    else
    {
        Console.WriteLine($"验证失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"验证过程中发生错误: {ex.Message}");
}
```

特点

- 支持 Mojang 账号登录
- 需要网络连接
- 支持离线模式（通过访问令牌）
- 支持刷新访问令牌
- 支持验证访问令牌有效性

高级功能

```
using Qomicex.Core.Modules.Helpers.Account.Yggdrasil;

// 创建验证器实例
var authenticator = new YggdrasilAuthenticator
{
    Username = "your-email@example.com",
    Password = "your-password"
};

// 验证
var result = await authenticator.AuthenticateAsync();

if (result.IsSuccess)
{
    // 验证访问令牌
    var validateResult = await authenticator.ValidateTokenAsync(result.AccessToken,
result.ClientToken);
    Console.WriteLine($"访问令牌有效: {validateResult.IsSuccess}");
```

```
    // 刷新访问令牌
    var refreshResult = await authenticator.RefreshTokenAsync(result.AccessToken,
result.ClientToken);
    if (refreshResult.IsSuccess)
    {
        Console.WriteLine($"刷新访问令牌成功！");
        Console.WriteLine($"新访问令牌：{refreshResult.AccessToken}");
    }

    // 失效访问令牌
    var invalidateResult = await authenticator.InvalidateTokenAsync(result.AccessToken,
result.ClientToken);
    Console.WriteLine($"失效访问令牌成功：{invalidateResult.IsSuccess}");
}
```

MicrosoftAuthenticator（新版微软验证模型）

概述

Microsoft 验证模型是 Mojang 官方推荐的验证方式，适用于使用 Microsoft 账号登录的用户。该验证模型需要网络连接，并使用 Microsoft 账号进行验证。

配置

在使用 Microsoft 验证模型之前，您需要先配置您的应用程序：

```
using Qomicex.Core.Modules.Helpers.Account.Microsoft;

// 配置微软验证器
MicrosoftAuthenticator.Configure(new MicrosoftAuthenticatorAPISettings
{
    ClientId = "your-client-id",
    TenentId = "consumers",
    Scopes = new[] { "XboxLive.signin", "offline_access", "openid", "profile", "email" }
});
```

使用方法

```
using Qomicex.Core.Modules.Helpers.Account.Microsoft;

// 创建微软验证器实例
var authenticator = new MicrosoftAuthenticator();

try
{
```

```
    // 获取授权 URL
    var authUrl = authenticator.GetAuthorizationUrl();
    Console.WriteLine($"请访问以下 URL 进行授权: {authUrl}");

    // 获取用户输入的授权码
    Console.Write("请输入授权码: ");
    var code = Console.ReadLine();

    // 执行验证
    var result = await authenticator.AuthorizeAsync(code);

    if (result.IsSuccess)
    {
        Console.WriteLine("Microsoft 验证成功! ");
        Console.WriteLine($"玩家名称: {result.PlayerName}");
        Console.WriteLine($"UUID: {result.UUID}");
        Console.WriteLine($"访问令牌: {result.AccessToken}");
        Console.WriteLine($"刷新令牌: {result.RefreshToken}");
        Console.WriteLine($"过期时间: {result.ExpiresIn} 秒");
    }
    else
    {
        Console.WriteLine($"验证失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"验证过程中发生错误: {ex.Message}");
}
```

特点

- 支持 Microsoft 账号登录
- 需要网络连接
- 支持多因素认证
- 支持刷新访问令牌
- 支持设备代码流（Device Code Flow）

设备代码流（Device Code Flow）

```
using Qomicex.Core.Modules.Helpers.Account.Microsoft;

var authenticator = new MicrosoftAuthenticator();

try
```

```
{
    // 启动设备代码流
    var deviceCode = await authenticator.StartDeviceCodeFlowAsync();

    Console.WriteLine($"请访问: {deviceCode.VerificationUri}");
    Console.WriteLine($"输入代码: {deviceCode.UserCode}");
    Console.WriteLine($"过期时间: {deviceCode.ExpiresIn} 秒");
    Console.WriteLine($"轮询间隔: {deviceCode.Interval} 秒");

    // 等待用户授权
    var result = await authenticator.WaitForDeviceCodeFlowCompletionAsync(deviceCode);

    if (result.IsSuccess)
    {
        Console.WriteLine("Microsoft 验证成功! ");
        Console.WriteLine($"玩家名称: {result.PlayerName}");
        Console.WriteLine($"UUID: {result.UUID}");
        Console.WriteLine($"访问令牌: {result.AccessToken}");
        Console.WriteLine($"刷新令牌: {result.RefreshToken}");
    }
    else
    {
        Console.WriteLine($"验证失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"验证过程中发生错误: {ex.Message}");
}
```

验证结果处理

通用验证结果

所有验证器都会返回一个通用的验证结果对象：

```
public class AuthenticationResult
{
    public bool IsSuccess { get; set; }
    public string PlayerName { get; set; }
    public string UUID { get; set; }
    public string AccessToken { get; set; }
    public string ClientToken { get; set; }
    public string RefreshToken { get; set; }
    public int ExpiresIn { get; set; }
```

```
    public string ErrorMessage { get; set; }
}
```

错误处理

在使用验证器时，您需要处理可能的错误：

```
using Qomicex.Core.Modules.Helpers.Account;

try
{
    var result = await authenticator.AuthenticateAsync();

    if (result.IsSuccess)
    {
        // 验证成功，处理结果
    }
    else
    {
        // 验证失败，显示错误信息
        Console.WriteLine($"验证失败: {result.ErrorMessage}");
    }
}
catch (AuthenticationException ex)
{
    // 处理认证相关异常
    Console.WriteLine($"认证异常: {ex.Message}");
}
catch (NetworkException ex)
{
    // 处理网络异常
    Console.WriteLine($"网络异常: {ex.Message}");
}
catch (Exception ex)
{
    // 处理其他异常
    Console.WriteLine($"其他异常: {ex.Message}");
}
```

最佳实践

安全存储凭证

在实际应用中，您应该妥善存储用户的凭证信息：

```
using Qomicex.Core.Modules.Helpers.Account;
using System.Security.Cryptography;
using System.Text;

// 加密凭证信息
public static string Encrypt(string text, string key)
{
    // 使用 AES 加密算法
    using var aes = Aes.Create();
    aes.Key = Encoding.UTF8.GetBytes(key.PadRight(32).Substring(0, 32));
    aes.IV = Encoding.UTF8.GetBytes(key.PadRight(16).Substring(0, 16));

    using var encryptor = aes.CreateEncryptor();
    var bytes = Encoding.UTF8.GetBytes(text);
    var encrypted = encryptor.TransformFinalBlock(bytes, 0, bytes.Length);

    return Convert.ToBase64String(encrypted);
}

// 解密凭证信息
public static string Decrypt(string encryptedText, string key)
{
    using var aes = Aes.Create();
    aes.Key = Encoding.UTF8.GetBytes(key.PadRight(32).Substring(0, 32));
    aes.IV = Encoding.UTF8.GetBytes(key.PadRight(16).Substring(0, 16));

    using var decryptor = aes.CreateDecryptor();
    var bytes = Convert.FromBase64String(encryptedText);
    var decrypted = decryptor.TransformFinalBlock(bytes, 0, bytes.Length);

    return Encoding.UTF8.GetString(decrypted);
}

// 使用示例
var encryptedPassword = Encrypt("your-password", "your-secret-key");
// 存储 encryptedPassword

var decryptedPassword = Decrypt(encryptedPassword, "your-secret-key");
// 使用 decryptedPassword 进行验证
```

会话管理

您应该正确管理用户的会话：

```
using Qomicex.Core.Modules.Helpers.Account;

// 保存会话信息
public static void SaveSession(AuthenticationResult result)
{
    var session = new
    {
        PlayerName = result.PlayerName,
        UUID = result.UUID,
        AccessToken = result.AccessToken,
        ClientToken = result.ClientToken,
        RefreshToken = result.RefreshToken,
        ExpiresAt = DateTime.Now.AddSeconds(result.ExpiresIn)
    };

    var json = JsonConvert.SerializeObject(session);
    File.WriteAllText("session.json", json);
}

// 加载会话信息
public static AuthenticationResult LoadSession()
{
    if (!File.Exists("session.json"))
        return null;

    var json = File.ReadAllText("session.json");
    var session = JsonConvert.DeserializeObject<dynamic>(json);

    // 检查会话是否过期
    var expiresAt = DateTime.Parse(session.ExpiresAt.ToString());
    if (expiresAt < DateTime.Now)
    {
        File.Delete("session.json");
        return null;
    }

    return new AuthenticationResult
    {
        IsSuccess = true,
        PlayerName = session.PlayerName,
        UUID = session.UUID,
        AccessToken = session.AccessToken,
        ClientToken = session.ClientToken,
        RefreshToken = session.RefreshToken,
        ExpiresIn = (int)(expiresAt - DateTime.Now).TotalSeconds
    };
```

```
}

// 使用示例
var existingSession = LoadSession();
if (existingSession != null)
{
    Console.WriteLine($"已登录用户: {existingSession.PlayerName}");
}
else
{
    // 显示登录界面
    var result = await authenticator.AuthenticateAsync();
    SaveSession(result);
}
```

安装器

在 Qomicex.Core 中，我们已经为开发者实现了您在 Minecraft 开发过程中可能遇到的大部分模组基础设施。

支持列表

项目名称	支持状态	
Forge （旧版）	受支持	✅
Forge （新版）	受支持	✅
LiteLoader	受支持	✅
Fabric	受支持	✅
Optifine	受支持	✅
Quilt	受支持	✅
CurseForge 整合包	受支持	✅

[[toc]]

通用安装接口

所有的安装器都实现了相同的接口，这使得在代码中切换不同的安装器变得非常简单。

```
using Qomicex.Core.Modules.Helpers.Installers;

public interface IInstaller
{
    // 获取安装器信息
    InstallerInfo GetInstallerInfo();

    // 安装 Mod 加载器
    Task<InstallResult> InstallAsync(string versionId, string inheritsFromJson, string
para1, string para2, string para3, string para4);

    // 验证安装
    Task<ValidationResult> ValidateAsync(string versionId, string gameDirectory);

    // 获取支持的版本
```

```
    Task<IEnumerable<VersionInfo>> GetSupportedVersionsAsync(string gameVersion);
}
```

安装器基类

所有的安装器都继承自 InstallerBase 类，它提供了通用的安装功能。

```
using Qomicex.Core.Modules.Helpers.Installers;

public abstract class InstallerBase : IInstaller
{
    // 通用下载方法
    protected async Task<string> DownloadFileAsync(string url, string destinationPath);

    // 通用文件操作方法
    protected void ExtractArchive(string archivePath, string destinationPath);

    // 通用配置文件处理方法
    protected string MergeVersionJson(string mainVersionJson, string mergedVersionJson,
string? defaultVersionId);

    // 通用依赖检查方法
    protected async Task<bool> CheckDependenciesAsync(string versionId);
}
```

Forge 安装器

使用方法

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 Forge 安装器实例
var forgeInstaller = new ForgeInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft");

try
{
    // 安装 Forge
    var result = await forgeInstaller.InstallAsync(
        "1.20.1-forge-47.1.3", // 版本 ID
        null, // 继承版本 JSON（可选）
        "1.20.1", // 游戏版本
        "47.1.3", // Forge 版本
        null, // 额外参数 1
```

```
        null // 额外参数 2
    );

    if (result.IsSuccess)
    {
        Console.WriteLine("Forge 安装成功！");
        Console.WriteLine($"安装路径: {result.InstallPath}");
    }
    else
    {
        Console.WriteLine($"安装失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"安装过程中发生错误: {ex.Message}");
}
```

高级用法

```
using Qomicex.Core.Modules.Helpers.Installers;

var forgeInstaller = new ForgeInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft");

// 检查 Forge 版本是否与游戏版本兼容
var compatible = await forgeInstaller.CheckCompatibilityAsync("1.20.1", "47.1.3");
if (compatible)
    Console.WriteLine("版本兼容");

// 获取支持的 Forge 版本
var supportedVersions = await forgeInstaller.GetSupportedVersionsAsync("1.20.1");
foreach (var version in supportedVersions)
{
    Console.WriteLine($"版本: {version.Version}");
    Console.WriteLine($"类型: {version.Type}");
    Console.WriteLine($"下载链接: {version.DownloadUrl}");
}
```

Fabric 安装器

使用方法

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 Fabric 安装器实例
var fabricInstaller = new FabricInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft");

try
{
    // 安装 Fabric
    var result = await fabricInstaller.InstallAsync(
        "1.20.1-fabric-0.14.24", // 版本 ID
        null, // 继承版本 JSON（可选）
        "1.20.1", // 游戏版本
        "0.14.24", // Fabric 版本
        null, // 额外参数 1
        null // 额外参数 2
    );

    if (result.IsSuccess)
    {
        Console.WriteLine("Fabric 安装成功！");
        Console.WriteLine($"安装路径: {result.InstallPath}");
    }
    else
    {
        Console.WriteLine($"安装失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"安装过程中发生错误: {ex.Message}");
}
```

高级用法

```
using Qomicex.Core.Modules.Helpers.Installers;

var fabricInstaller = new FabricInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft");

// 获取支持的 Fabric 版本
var supportedVersions = await fabricInstaller.GetSupportedVersionsAsync("1.20.1");
foreach (var version in supportedVersions)
{
    Console.WriteLine($"版本: {version.Version}");
```

```
    Console.WriteLine($"类型: {version.Type}");
    Console.WriteLine($"下载链接: {version.DownloadUrl}");
}

// 验证 Fabric 安装
var validationResult = await fabricInstaller.ValidateAsync("1.20.1-fabric-
0.14.24", @"C:\Games\.minecraft");
if (validationResult.IsValid)
    Console.WriteLine("Fabric 安装验证成功！");
else
    Console.WriteLine($"验证失败: {validationResult.ErrorMessage}");
```

Quilt 安装器

使用方法

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 Quilt 安装器实例
var quiltInstaller = new QuiltInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft");

try
{
    // 安装 Quilt
    var result = await quiltInstaller.InstallAsync(
        "1.20.1-quilt-0.18.10", // 版本 ID
        null, // 继承版本 JSON（可选）
        "1.20.1", // 游戏版本
        "0.18.10", // Quilt 版本
        null, // 额外参数 1
        null // 额外参数 2
    );

    if (result.IsSuccess)
    {
        Console.WriteLine("Quilt 安装成功！");
        Console.WriteLine($"安装路径: {result.InstallPath}");
    }
    else
    {
        Console.WriteLine($"安装失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
```

```
{
    Console.WriteLine($"安装过程中发生错误: {ex.Message}");
}
```

LiteLoader 安装器

使用方法

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 LiteLoader 安装器实例
var liteLoaderInstaller = new LiteloaderInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft");

try
{
    // 安装 LiteLoader
    var result = await liteLoaderInstaller.InstallAsync(
        "1.12.2-liteloader-1.12.2-SNAPSHOT", // 版本 ID
        null, // 继承版本 JSON（可选）
        "1.12.2", // 游戏版本
        null, // LiteLoader 版本（自动获取）
        null, // 额外参数 1
        null // 额外参数 2
    );

    if (result.IsSuccess)
    {
        Console.WriteLine("LiteLoader 安装成功！");
        Console.WriteLine($"安装路径: {result.InstallPath}");
    }
    else
    {
        Console.WriteLine($"安装失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"安装过程中发生错误: {ex.Message}");
}
```

OptiFine 安装器

使用方法

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 OptiFine 安装器实例
var optifineInstaller = new OptiFineInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft");

try
{
    // 安装 OptiFine
    var result = await optifineInstaller.InstallAsync(
        "1.20.1-optifine-HD_U_I5", // 版本 ID
        null, // 继承版本 JSON（可选）
        "1.20.1", // 游戏版本
        "HD_U_I5", // OptiFine 版本
        null, // 额外参数 1
        null // 额外参数 2
    );

    if (result.IsSuccess)
    {
        Console.WriteLine("OptiFine 安装成功！");
        Console.WriteLine($"安装路径: {result.InstallPath}");
    }
    else
    {
        Console.WriteLine($"安装失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"安装过程中发生错误: {ex.Message}");
}
```

CurseForge 整合包安装器

使用方法

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 CurseForge 整合包安装器实例
var modpackInstaller = new CurseForgeModPackInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft");

try
{
```

```
    // 安装 CurseForge 整合包
    var result = await modpackInstaller.InstallAsync(
        "my-modpack", // 版本 ID
        null, // 继承版本 JSON（可选）
        "curseforge-modpack-id", // CurseForge 整合包 ID
        "1.0.0", // 整合包版本
        null, // 额外参数 1
        null // 额外参数 2
    );

    if (result.IsSuccess)
    {
        Console.WriteLine("CurseForge 整合包安装成功！");
        Console.WriteLine($"安装路径: {result.InstallPath}");
    }
    else
    {
        Console.WriteLine($"安装失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"安装过程中发生错误: {ex.Message}");
}
```

安装结果

所有的安装器都会返回一个统一的安装结果对象：

```
public class InstallResult
{
    public bool IsSuccess { get; set; }
    public string InstallPath { get; set; }
    public string ErrorMessage { get; set; }
    public string Log { get; set; }
}
```

验证安装

在安装完成后，您可以使用 ValidateAsync 方法来验证安装是否成功：

```
using Qomicex.Core.Modules.Helpers.Installers;

var installer = new FabricInstaller((int)DownloadSource.Official, @"C:\Games\.minecraft");
```

```
var validationResult = await installer.ValidateAsync("1.20.1-fabric-
0.14.24", @"C:\Games\.minecraft");

if (validationResult.IsValid)
{
    Console.WriteLine("安装验证成功！");
}
else
{
    Console.WriteLine($"验证失败: {validationResult.ErrorMessage}");

    // 列出所有验证问题
    foreach (var issue in validationResult.Issues)
    {
        Console.WriteLine($"问题: {issue.Description} ({issue.Severity})");
    }
}
```

安装过程监控

您可以通过安装事件来监控安装过程：

```
using Qomicex.Core.Modules.Helpers.Installers;

var installer = new ForgeInstaller((int)DownloadSource.Official, @"C:\Games\.minecraft");

// 安装进度事件
installer.ProgressChanged += (sender, args) =>
{
    Console.WriteLine($"安装进度: {args.ProgressPercentage}%");
    Console.WriteLine($"当前任务: {args.CurrentTask}");
    Console.WriteLine($"已下载: {args.DownloadedSize} / {args.TotalSize}");
};

// 下载完成事件
installer.DownloadCompleted += (sender, args) =>
{
    Console.WriteLine("下载完成！");
};

// 提取完成事件
installer.ExtractionCompleted += (sender, args) =>
{
    Console.WriteLine("文件提取完成！");
```

```
};

// 配置完成事件
installer.ConfigurationCompleted += (sender, args) =>
{
    Console.WriteLine("配置完成！");
};

try
{
    var result = await installer.InstallAsync(...);
}
catch (Exception ex)
{
    Console.WriteLine($"安装过程中发生错误: {ex.Message}");
}
```

安装优化

多线程下载

```
using Qomicex.Core.Modules.Helpers.Installers;

// 配置下载选项
var downloadOptions = new DownloadOptions
{
    MaxConcurrentDownloads = 8, // 最大并发下载数
    ChunkSize = 4 * 1024 * 1024, // 分块大小
    RetryCount = 3, // 重试次数
    Timeout = TimeSpan.FromSeconds(30) // 超时时间
};

var installer = new ForgeInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft", downloadOptions);
```

缓存配置

```
using Qomicex.Core.Modules.Helpers.Installers;

// 配置缓存选项
var cacheOptions = new CacheOptions
{
    EnableCache = true, // 启用缓存
    CacheLocation = @"C:\Games\.minecraft\cache", // 缓存位置
```

```
    CacheDuration = TimeSpan.FromDays(7), // 缓存有效期
    MaxCacheSize = 10 * 1024 * 1024 * 1024 // 最大缓存大小（10GB）
};

var installer = new FabricInstaller((int)DownloadSource.Official,
@"C:\Games\.minecraft", cacheOptions);
```

常见问题

安装失败

如果安装失败，可以检查以下几个方面：

1. 网络连接是否正常
2. 游戏目录是否有写入权限
3. 下载源是否可用
4. 磁盘空间是否充足
5. 安装文件是否完整

依赖检查失败

如果依赖检查失败，可以检查：

1. 是否缺少必要的依赖库
2. 依赖库的版本是否匹配
3. 依赖库是否已正确安装

配置文件错误

如果配置文件错误，可以检查：

1. 配置文件是否已正确生成
2. 配置文件中的路径是否正确
3. 配置文件是否有语法错误

资源补全器

在 Qomicex.Core 中，我们为开发者提供了用于补全 Minecraft 核心资源文件的补全器。这些资源包括：游戏音频、材质、贴图、语言文件、启动所必须的库文件等。

[[toc]]

资源补全器概述

资源补全器是 Qomicex.Core 提供的一个核心功能，它负责下载和管理 Minecraft 游戏所需的各种资源文件。资源补全器会根据游戏版本和 Mod 加载器类型自动判断需要下载哪些资源文件。

使用方法

基本用法

```
using Qomicex.Core.Modules.Helpers.Resources;

// 创建资源补全器实例
var completer = new ResourceCompleter(@"C:\Games\.minecraft");

try
{
    // 补全资源
    var result = await completer.CompleteResourcesAsync("1.20.1");

    if (result.IsSuccess)
    {
        Console.WriteLine("资源补全成功！");
        Console.WriteLine($"下载资源数: {result.DownloadedResources.Count}");
        Console.WriteLine($"总大小: {result.TotalSize}");
    }
    else
    {
        Console.WriteLine($"资源补全失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"资源补全过程中发生错误: {ex.Message}");
}
```

高级用法

```
using Qomicex.Core.Modules.Helpers.Resources;

// 创建资源补全器实例
var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 配置补全选项
var options = new ResourceCompleteOptions
{
    SourceType = DownloadSource.Official, // 下载源类型
    DownloadProgressChanged = (sender, args) =>
    {
        Console.WriteLine($"下载进度: {args.ProgressPercentage}%");
        Console.WriteLine($"已下载: {args.DownloadedSize} / {args.TotalSize}");
    },
    ExtractionProgressChanged = (sender, args) =>
    {
        Console.WriteLine($"提取进度: {args.ProgressPercentage}%");
        Console.WriteLine($"当前文件: {args.CurrentFile}");
    }
};

try
{
    // 补全资源
    var result = await completer.CompleteResourcesAsync("1.20.1", options);

    if (result.IsSuccess)
    {
        Console.WriteLine("资源补全成功！");
        Console.WriteLine($"下载资源数: {result.DownloadedResources.Count}");
        Console.WriteLine($"总大小: {result.TotalSize}");
    }
    else
    {
        Console.WriteLine($"资源补全失败: {result.ErrorMessage}");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"资源补全过程中发生错误: {ex.Message}");
}
```

资源补全结果

资源补全器会返回一个详细的补全结果对象：

```
public class ResourceCompleteResult
{
    public bool IsSuccess { get; set; }
    public List<string> DownloadedResources { get; set; }
    public List<string> ExtractedFiles { get; set; }
    public long TotalSize { get; set; }
    public string ErrorMessage { get; set; }
}
```

资源类型

游戏核心资源

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 补全游戏核心资源
var result = await completer.CompleteCoreResourcesAsync("1.20.1");
```

材质资源

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 补全材质资源
var result = await completer.CompleteAssetResourcesAsync("1.20.1");
```

库文件资源

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 补全库文件资源
var result = await completer.CompleteLibraryResourcesAsync("1.20.1");
```

语言文件资源

```
using Qomicex.Core.Modules.Helpers.Resources;
```

```
var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 补全语言文件资源
var result = await completer.CompleteLanguageResourcesAsync("1.20.1", "zh_cn");
```

资源验证

您可以使用资源验证功能来检查已下载的资源是否完整：

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 验证资源
var validationResult = await completer.ValidateResourcesAsync("1.20.1");

if (validationResult.IsValid)
{
    Console.WriteLine("资源验证成功！");
}
else
{
    Console.WriteLine($"资源验证失败：{validationResult.ErrorMessage}");

    // 列出所有验证问题
    foreach (var issue in validationResult.Issues)
    {
        Console.WriteLine($"问题：{issue.Description} ({issue.Severity})");
    }
}
```

资源下载选项

下载源配置

```
using Qomicex.Core.Modules.Helpers.Resources;

// 使用 BMCLAPI 下载源
var options = new ResourceCompleteOptions
{
    SourceType = DownloadSource.Bmclapi
};
```

```
var completer = new ResourceCompleter(@"C:\Games\.minecraft");
var result = await completer.CompleteResourcesAsync("1.20.1", options);
```

下载进度监控

```
using Qomicex.Core.Modules.Helpers.Resources;

var options = new ResourceCompleteOptions
{
    DownloadProgressChanged = (sender, args) =>
    {
        Console.WriteLine($"文件: {args.FileName}");
        Console.WriteLine($"进度: {args.ProgressPercentage}%");
        Console.WriteLine($"速度: {args.DownloadSpeed} KB/s");
        Console.WriteLine($"已下载: {args.DownloadedSize} / {args.TotalSize}");
    }
};

var completer = new ResourceCompleter(@"C:\Games\.minecraft");
var result = await completer.CompleteResourcesAsync("1.20.1", options);
```

文件提取进度监控

```
using Qomicex.Core.Modules.Helpers.Resources;

var options = new ResourceCompleteOptions
{
    ExtractionProgressChanged = (sender, args) =>
    {
        Console.WriteLine($"文件: {args.CurrentFile}");
        Console.WriteLine($"进度: {args.ProgressPercentage}%");
        Console.WriteLine($"提取文件数: {args.ExtractedFiles} / {args.TotalFiles}");
    }
};

var completer = new ResourceCompleter(@"C:\Games\.minecraft");
var result = await completer.CompleteResourcesAsync("1.20.1", options);
```

资源缓存

启用资源缓存

```
using Qomicex.Core.Modules.Helpers.Resources;

var options = new ResourceCompleteOptions
{
    EnableCache = true,
    CacheLocation = @"C:\Games\.minecraft\cache",
    CacheDuration = TimeSpan.FromDays(7)
};

var completer = new ResourceCompleter(@"C:\Games\.minecraft");
var result = await completer.CompleteResourcesAsync("1.20.1", options);
```

清除资源缓存

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 清除所有缓存
await completer.ClearCacheAsync();

// 清除特定版本的缓存
await completer.ClearCacheAsync("1.20.1");
```

资源修复

如果资源文件损坏或缺失，您可以使用资源修复功能：

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 修复资源
var repairResult = await completer.RepairResourcesAsync("1.20.1");

if (repairResult.IsSuccess)
{
    Console.WriteLine("资源修复成功！");
    Console.WriteLine($"修复资源数: {repairResult.RepairedResources.Count}");
}
else
{
```

```
    Console.WriteLine($"资源修复失败: {repairResult.ErrorMessage}");
}
```

资源检查

检查资源完整性

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 检查资源完整性
var checkResult = await completer.CheckResourceIntegrityAsync("1.20.1");

if (checkResult.IsComplete)
{
    Console.WriteLine("所有资源文件完整！");
}
else
{
    Console.WriteLine($"缺少以下资源文件: {string.Join(", ",
checkResult.MissingResources)}");
}
```

检查资源更新

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 检查资源更新
var updateResult = await completer.CheckResourceUpdatesAsync("1.20.1");

if (updateResult.HasUpdates)
{
    Console.WriteLine($"有 {updateResult.UpdateCount} 个资源文件需要更新");

    // 更新资源
    var result = await completer.UpdateResourcesAsync("1.20.1");

    if (result.IsSuccess)
        Console.WriteLine("资源更新成功！");
}
else
```

```
{
    Console.WriteLine("资源已经是最新版本！");
}
```

资源管理

获取资源信息

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 获取资源信息
var resourceInfo = await completer.GetResourceInfoAsync("1.20.1");

Console.WriteLine($"资源版本: {resourceInfo.Version}");
Console.WriteLine($"资源类型: {resourceInfo.Type}");
Console.WriteLine($"资源大小: {resourceInfo.Size}");
Console.WriteLine($"资源数量: {resourceInfo.Count}");
```

导出资源

```
using Qomicex.Core.Modules.Helpers.Resources;

var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 导出资源
var exportResult = await completer.ExportResourcesAsync("1.20.1",
@"C:\Temp\MinecraftResources");

if (exportResult.IsSuccess)
{
    Console.WriteLine($"资源导出成功！导出路径: {exportResult.ExportPath}");
}
else
{
    Console.WriteLine($"资源导出失败: {exportResult.ErrorMessage}");
}
```

导入资源

```
using Qomicex.Core.Modules.Helpers.Resources;
```

```
var completer = new ResourceCompleter(@"C:\Games\.minecraft");

// 导入资源
var importResult = await completer.ImportResourcesAsync("1.20.1",
@"C:\Temp\MinecraftResources");

if (importResult.IsSuccess)
{
    Console.WriteLine($"资源导入成功！导入资源数：{importResult.ImportedResources.Count}");
}
else
{
    Console.WriteLine($"资源导入失败：{importResult.ErrorMessage}");
}
```

常见问题

资源下载失败

如果资源下载失败，可以检查以下几个方面：

1. 网络连接是否正常
2. 下载源是否可用
3. 磁盘空间是否充足
4. 游戏目录是否有写入权限
5. 防火墙是否阻止了下载请求

资源验证失败

如果资源验证失败，可以检查：

1. 资源文件是否被病毒扫描程序损坏
2. 磁盘是否有坏道
3. 文件权限是否正确
4. 是否有其他程序正在使用该文件

资源提取失败

如果资源提取失败，可以检查：

1. 归档文件是否完整
2. 归档文件是否被加密
3. 磁盘空间是否充足
4. 磁盘是否有坏道

资源不完整

如果资源不完整，可以尝试：

1. 重新补全资源
2. 使用修复功能
3. 检查网络连接
4. 检查磁盘空间

系统信息获取

功能概述

SystemInfoHelper 提供了跨平台的系统信息获取功能，包括操作系统名称、架构、内存信息等。

使用示例

```
using Qomicex.Core.Modules.Helpers;

// 获取操作系统信息
string osName = SystemInfoHelper.OsName; // 例如："Windows"
string osArch = SystemInfoHelper.OsArch; // 例如："x64"
string separator = SystemInfoHelper.Separator; // 例如："\\"

// 获取系统内存信息
ulong totalMemory = SystemInfoHelper.GetTotalMemory(); // 总内存（字节）
ulong availableMemory = SystemInfoHelper.GetAvailableMemory(); // 可用内存（字节）

// 获取系统信息字符串
string systemInfo = SystemInfoHelper.GetSystemInfo();
// 输出示例："Windows 10.0.19045 64位 (X64) - 16.0GB总内存，8.5GB可用"

// 打印系统信息
Console.WriteLine($"操作系统: {osName} ({osArch})");
Console.WriteLine($"内存: {totalMemory / (1024*1024*1024)}GB 总，{availableMemory /
(1024*1024*1024)}GB 可用");
```

通用工具

功能概述

GeneralHelper 提供了各种通用工具方法，包括 Mod 加载器检测、文件操作、加密/解密等。

使用示例

```
using Qomicex.Core.Modules.Helpers;

// 检测版本文件中的 Mod 加载器类型
string gameDir = @"C:\Game\.minecraft";
string versionId = "1.20.1-forge-47.1.3";
string[] modLoaders = new GeneralHelper().GetModLoaderType(versionId, gameDir);
// 输出示例：["Forge 47.1.3"]

// 从 ZIP 文件中读取指定文件
string jarPath = @"C:\Game\.minecraft\versions\1.20.1\1.20.1.jar";
string manifest = GeneralHelper.ReadSpecifyFileFromZip(jarPath, "META-INF/MANIFEST.MF");

// 验证文件哈希
string filePath = @"C:\Game\.minecraft\versions\1.20.1\1.20.1.jar";
string expectedSha1 = "abc123...";
bool isValid = GeneralHelper.VerifyFileChecksum(filePath,
expectedSha1, HashAlgorithmName.SHA1);

// 清理临时文件
GeneralHelper.ClearTempFiles();

// 转换文件大小格式
string sizeStr = GeneralHelper.FormatFileSize(1024 * 1024 * 100); // "100 MB"
```

Java 管理

功能概述

JavaHelper 提供了 Java 运行时检测、搜索和验证功能，支持快速搜索、深度搜索和自定义搜索。

使用示例

```
using Qomicex.Core.Modules.Helpers;

// 快速搜索 Java 运行时
List<JavaHelper.JavaInfoExtended> quickResults =
JavaHelper.SearchForJavaAsync(JavaHelper.QuickOptions).Result;
foreach (var javaInfo in quickResults)
{
    Console.WriteLine($"Java 版本: {javaInfo.Version}");
    Console.WriteLine($"路径: {javaInfo.Path}");
    Console.WriteLine($"状态: {javaInfo.State}");
}

// 自定义搜索选项
var customOptions = new JavaHelper.JavaSearchOptions
{
    Mode = JavaHelper.JavaSearchMode.Deep,
    GameDir = @"C:\Game\.minecraft",
    IncludeJRE = true,
    IncludeJDK = false,
    MaxDepth = 3,
    MaxResults = 20,
    ScanHiddenFolders = false,
    IncludeNetworkDrives = false,
    CustomRootPath = @"C:\Program Files\Java"
};

List<JavaHelper.JavaInfoExtended> customResults =
JavaHelper.SearchForJavaAsync(customOptions).Result;

// 验证 Java 路径
var validationResult = JavaHelper.ValidateJavaPath(@"C:\Program Files\Java\jre1.8.0_311");
Console.WriteLine($"Java 验证状态: {validationResult.State}");
if (!validationResult.IsValid)
{
    Console.WriteLine($"错误信息: {validationResult.StateMessage}");
}
```

```
// 获取推荐的 Java 版本
var recommendedJava = JavaHelper.GetRecommendedJavaVersion("1.20.1");
Console.WriteLine($"推荐 Java 版本: {recommendedJava}");
```

游戏资源管理

功能概述

GameResourceHelper 提供了 Minecraft 游戏版本列表获取、版本信息解析等功能，支持官方源和 BMCLAPI 源。

使用示例

```
using Qomicex.Core.Modules.Helpers.Resources;

// 创建游戏资源助手实例
var gameHelper = new GameResourceHelper();

// 获取 Minecraft 版本列表（官方源）
List<GameResourceHelper.VersionInfo> officialVersions = await
gameHelper.GetMinecraftListAsync(1);
foreach (var version in officialVersions)
{
    Console.WriteLine($"版本 ID: {version.Id}");
    Console.WriteLine($"类型: {version.Type}");
    Console.WriteLine($"发布时间: {version.ReleaseTime}");
    Console.WriteLine($"信息 URL: {version.Url}");
    Console.WriteLine();
}

// 获取 Minecraft 版本列表（BMCLAPI 源）
List<GameResourceHelper.VersionInfo> bmclapiVersions = await
gameHelper.GetMinecraftListAsync(0);

// 获取版本详细信息
dynamic versionManifest = await gameHelper.GetMinecraftVersionManifest(1);
string latestRelease = versionManifest?.latest?.release;
string latestSnapshot = versionManifest?.latest?.snapshot;
Console.WriteLine($"最新正式版: {latestRelease}");
Console.WriteLine($"最新快照版: {latestSnapshot}");
```

本地资源管理

功能概述

LocalResourceHelper 负责 Minecraft 本地资源的管理，包括库文件检查、资源下载、版本合并等。

使用示例

```
using Qomicex.Core.Modules.Helpers.Resources;

// 检查版本文件的合规性
string versionJsonPath = @"C:\Game\.minecraft\versions\1.20.1\1.20.1.json";
string versionJson = File.ReadAllText(versionJsonPath);
bool isCompliant = LocalResourceHelper.CheckCompliance(versionJson);
Console.WriteLine($"版本文件合规性: {isCompliant}");

// 获取缺失的库文件
string gameDir = @"C:\Game\.minecraft";
List<string> missingLibraries = await LocalResourceHelper.GetMissingLibraries(gameDir,
"1.20.1");
Console.WriteLine($"缺失的库文件数量: {missingLibraries.Count}");
foreach (var lib in missingLibraries)
{
    Console.WriteLine(lib);
}

// 下载缺失的资源
bool downloadSuccess = await LocalResourceHelper.DownloadMissingResources(gameDir,
"1.20.1");
Console.WriteLine($"资源下载状态: {downloadSuccess}");

// 验证游戏资源完整性
bool resourcesComplete = LocalResourceHelper.VerifyGameResources(gameDir, "1.20.1");
Console.WriteLine($"游戏资源完整性: {resourcesComplete}");
```

模组加载器管理

功能概述

ModLoaderResourceHelper 提供了各种 Minecraft 模组加载器的信息获取和管理功能，支持 Forge、Fabric、Quilt、LiteLoader 等。

使用示例

```
using Qomicex.Core.Modules.Helpers.Resources;

// 创建模组加载器资源助手实例
var modLoaderHelper = new ModLoaderResourceHelper();

// 获取 Fabric 版本列表
var fabricVersions = await modLoaderHelper.GetFabricVersionsAsync();
Console.WriteLine("可用的 Fabric 版本:");
foreach (var version in fabricVersions)
{
    Console.WriteLine($"  - {version}");
}

// 获取 Forge 版本列表
var forgeVersions = await modLoaderHelper.GetForgeVersionsAsync("1.20.1");
Console.WriteLine("可用的 Forge 版本 (1.20.1):");
foreach (var version in forgeVersions)
{
    Console.WriteLine($"  - {version}");
}

// 获取 Quilt 版本列表
var quiltVersions = await modLoaderHelper.GetQuiltVersionsAsync();
Console.WriteLine("可用的 Quilt 版本:");
foreach (var version in quiltVersions)
{
    Console.WriteLine($"  - {version}");
}

// 获取 LiteLoader 版本列表
var liteLoaderVersions = await modLoaderHelper.GetLiteLoaderVersionsAsync("1.20.1");
Console.WriteLine("可用的 LiteLoader 版本 (1.20.1):");
foreach (var version in liteLoaderVersions)
{
    Console.WriteLine($"  - {version}");
}
```

```
// 获取 OptiFine 版本列表
var optifineVersions = await modLoaderHelper.GetOptiFineVersionsAsync("1.20.1");
Console.WriteLine("可用的 OptiFine 版本 (1.20.1):");
foreach (var version in optifineVersions)
{
    Console.WriteLine($"  - {version}");
}
```

安装器

功能概述

Qomicex.Core 提供了多种 Minecraft 模组加载器的安装器，包括 Forge、Fabric、Quilt、LiteLoader 和 OptiFine。所有安装器都实现了 IInstaller 接口。

使用示例

安装 Fabric

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 Fabric 安装器实例
var fabricInstaller = new FabricInstaller((int)InstallerBase.DownloadSource.Official,
@"C:\Game\.minecraft");

// 安装 Fabric 到指定版本
string versionId = "1.20.1-fabric-0.14.24";
string fabricVersion = "0.14.24";
string gameVersion = "1.20.1";

try
{
    fabricInstaller.InstallAsync(versionId, null, fabricVersion, gameVersion, null, null);
    Console.WriteLine("Fabric 安装成功！");
}
catch (Exception ex)
{
    Console.WriteLine($"Fabric 安装失败: {ex.Message}");
}
```

安装 Forge

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 Forge 安装器实例
var forgeInstaller = new ForgeInstaller((int)InstallerBase.DownloadSource.Official,
@"C:\Game\.minecraft");

// 安装 Forge 到指定版本
string versionId = "1.20.1-forge-47.1.3";
string forgeVersion = "47.1.3";
string gameVersion = "1.20.1";
```

```
string installerPath = @"C:\Downloads\forge-1.20.1-47.1.3-installer.jar";

try
{
    forgeInstaller.InstallAsync(versionId, null, forgeVersion, gameVersion,
installerPath, null);
    Console.WriteLine("Forge 安装成功! ");
}
catch (Exception ex)
{
    Console.WriteLine($"Forge 安装失败: {ex.Message}");
}
```

安装 Quilt

```
using Qomicex.Core.Modules.Helpers.Installers;

// 创建 Quilt 安装器实例
var quiltInstaller = new QuiltInstaller((int)InstallerBase.DownloadSource.Official,
@"C:\Game\.minecraft");

// 安装 Quilt 到指定版本
string versionId = "1.20.1-quilt-0.18.10";
string quiltVersion = "0.18.10";
string gameVersion = "1.20.1";

try
{
    quiltInstaller.InstallAsync(versionId, null, quiltVersion, gameVersion, null, null);
    Console.WriteLine("Quilt 安装成功! ");
}
catch (Exception ex)
{
    Console.WriteLine($"Quilt 安装失败: {ex.Message}");
}
```

启动器

功能概述

Launcher 模块提供了 Minecraft 游戏启动功能，包括启动参数配置、游戏启动过程管理等。

使用示例

```
using Qomicex.Core.Modules.Launcher;
using Qomicex.Core.Modules.Launcher.Launcher;

// 创建启动器实例
var launcher = new Launcher();

// 配置启动参数
var launchParams = new LauncherParam
{
    GameDir = @"C:\Game\.minecraft",
    Version = "1.20.1-fabric-0.14.24",
    JavaPath = @"C:\Program Files\Java\jre1.8.0_311\bin\javaw.exe",
    MaxMemory = 4096, // 4GB
    MinMemory = 1024, // 1GB
    WindowWidth = 854,
    WindowHeight = 480,
    Fullscreen = false,
    Server = null, // 单机游戏
    Username = "PlayerName",
    AuthenticationToken = "your-auth-token"
};

// 启动 Minecraft
var launchResult = launcher.StartGameAsync(launchParams);

// 监听启动事件
launcher.OnGameStarted += () => Console.WriteLine("游戏启动成功！");
launcher.OnGameError += (error) => Console.WriteLine($"游戏启动失败：{error}");

// 停止游戏
// launcher.StopGame();
```

账户管理

功能概述

Account 模块提供了 Minecraft 账户管理功能，包括 Microsoft 账户登录、Yggdrasil 认证和 Tongyi 账户支持。

使用示例

Microsoft 账户登录

```
using Qomicex.Core.Modules.Helpers.Account.Microsoft;

// 创建 Microsoft 账户实例
var microsoftAccount = new Microsoft();

// 获取授权 URL
string authUrl = microsoftAccount.GetAuthorizationUrl();
Console.WriteLine("请在浏览器中打开以下 URL 进行授权:");
Console.WriteLine(authUrl);
Console.WriteLine("授权完成后，请输入返回的代码:");
string code = Console.ReadLine();

// 完成授权
try
{
    var oauthResponse = await microsoftAccount.AuthorizeAsync(code);
    Console.WriteLine($"访问令牌: {oauthResponse.AccessToken}");
    Console.WriteLine($"刷新令牌: {oauthResponse.RefreshToken}");
    Console.WriteLine($"过期时间: {oauthResponse.ExpiresIn} 秒");

    // 获取用户信息
    var userInfo = await microsoftAccount.GetUserInfoAsync(oauthResponse.AccessToken);
    Console.WriteLine($"用户 ID: {userInfo.Id}");
    Console.WriteLine($"用户名: {userInfo.Gamertag}");
}
catch (Exception ex)
{
    Console.WriteLine($"授权失败: {ex.Message}");
}
```

Yggdrasil 认证

```
using Qomicex.Core.Modules.Helpers.Account.Yggdrasil;

// 创建 Yggdrasil 账户实例
var yggdrasilAccount = new Yggdrasil();

// 登录账户
string username = "your-username";
string password = "your-password";

try
{
    var loginResponse = await yggdrasilAccount.LoginAsync(username, password);
    Console.WriteLine($"访问令牌: {loginResponse.AccessToken}");
    Console.WriteLine($"客户端令牌: {loginResponse.ClientToken}");
    Console.WriteLine($"用户 ID: {loginResponse.User.Id}");

    // 验证访问令牌
    bool isValid = await yggdrasilAccount.ValidateAsync(loginResponse.AccessToken);
    Console.WriteLine($"访问令牌有效: {isValid}");
}
catch (Exception ex)
{
    Console.WriteLine($"登录失败: {ex.Message}");
}
```

日志分析

功能概述

LogAnalysis 模块提供了 Minecraft 游戏日志和崩溃报告的分析功能，帮助开发者快速定位和诊断问题。

使用示例

```
using Qomicex.Core.Modules.Helpers.LogAnalysis;
using Qomicex.Core.Modules.Helpers.LogAnalysis.Models;

// 创建日志分析器实例
var analyzer = new MinecraftLogAnalyzer();

// 分析崩溃报告
string crashReportPath = @"C:\Game\.minecraft\crash-reports\crash-2026-02-08.txt";
LogAnalysisResult result = await analyzer.AnalyzeAsync(crashReportPath);

if (result.IsSuccess && result.IsGameCrashed)
{
    Console.WriteLine($"Minecraft 版本: {result.MinecraftVersion}");
    Console.WriteLine($"Mod 加载器: {result.ModLoader}");
    Console.WriteLine($"检测到问题: {result.Issues.Count} 个");

    foreach (var issue in result.Issues)
    {
        Console.WriteLine($"[{issue.Severity}] {issue.Name}");
        Console.WriteLine($"  位置: 第{issue.LineNumber}行");
        Console.WriteLine($"  匹配: {issue.MatchedText}");

        foreach (var solution in issue.Solutions.OrderBy(s => s.Priority))
        {
            Console.WriteLine($"  → {solution.Description}");
        }
    }
}
else
{
    Console.WriteLine("分析失败或未检测到游戏崩溃");
    if (!string.IsNullOrEmpty(result.ErrorMessage))
    {
        Console.WriteLine($"错误信息: {result.ErrorMessage}");
    }
}
```

数据模块

功能概述

DataModules 提供了 Qomicex 项目的数据结构定义，包括游戏配置、系统信息、版本信息等。

使用示例

```
using Qomicex.Core.DataModules;

// 游戏配置
var launcherConfig = new DataModules.DataDetails.Launcher
{
    MemoryLimit = 4096, // MB
    WindowSize = new DataModules.DataDetails.Launcher.WindowSize
    {
        Width = 854,
        Height = 480
    },
    JavaPath = @"C:\Program Files\Java\jre1.8.0_311\bin\javaw.exe",
    GameDir = @"C:\Game\.minecraft"
};

// 系统信息
var systemInfo = new DataModules.SystemInfo
{
    OsName = SystemInfoHelper.OsName,
    OsArch = SystemInfoHelper.OsArch,
    TotalMemory = SystemInfoHelper.GetTotalMemory(),
    AvailableMemory = SystemInfoHelper.GetAvailableMemory()
};

// 账户信息
var account = new DataModules.DataDetails.Account
{
    Username = "PlayerName",
    AccessToken = "your-access-token",
    ClientToken = "your-client-token",
    Selected = true
};
```

扩展资源管理

功能概述

Expansion 模块提供了对第三方资源平台的集成支持，目前包括两个主要平台：

1. **CurseForge** - 最大的 Minecraft 资源分享平台，提供 Mod、纹理包、地图等资源
2. **Modrinth** - 开源的 Minecraft 资源平台，专注于现代 Mod 生态系统

这些模块允许应用程序直接从这些平台搜索、获取和管理 Minecraft 相关资源。

平台支持

CurseForge

- **资源类型**：Mod、Modpack、纹理包、数据包、光影包、地图
- **API支持**：完整的搜索、获取和下载功能
- **认证方式**：API Key 认证
- **使用场景**：需要访问大量资源的启动器或管理工具

Modrinth

- **资源类型**：Mod、Modpack、纹理包、数据包、光影包、地图
- **API支持**：搜索、获取和下载功能
- **认证方式**：无密钥认证（公开API）
- **使用场景**：需要开源和现代资源的应用程序

基本架构

Expansion 模块采用抽象设计，允许轻松添加新的资源平台支持。每个平台实现都继承自统一的接口，提供一致的访问方式。

使用示例

初始化 CurseForge 平台

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

string apiKey = "your-curseforge-api-key";
var curseForge = new Mods(apiKey);

// 或者直接使用特定资源类型
var modpacks = new Modpacks(apiKey);
var resourcePacks = new ResourcePacks(apiKey);
```

初始化 Modrinth 平台

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinth = new Mods();

// 或者直接使用特定资源类型
var modpacks = new ModPacks();
var resourcePacks = new ResourcePacks();
```

资源类型说明

所有资源平台支持以下资源类型：

1. **Mods** - 游戏模组，增强或修改 Minecraft 的功能
2. **Modpacks** - 预配置的模组组合，提供完整的游戏体验
3. **ResourcePacks** - 纹理包，改变游戏的视觉效果
4. **DataPacks** - 数据包，改变游戏规则和行为
5. **Shaders** - 光影包，增强游戏的图形效果
6. **Worlds** - 存档地图，提供自定义游戏世界

通用功能

每个资源平台都提供以下通用功能：

1. **搜索** - 根据关键词、分类、版本等条件搜索资源
2. **详情获取** - 获取资源的详细信息、作者、版本等
3. **下载** - 下载资源文件
4. **版本管理** - 获取资源的不同版本信息
5. **依赖处理** - 解析和获取资源的依赖项

最佳实践

1. **API密钥管理**：对于 CurseForge，确保妥善管理 API 密钥，避免硬编码
2. **错误处理**：实现适当的错误处理和重试机制
3. **缓存策略**：对搜索结果和资源信息实现缓存，减少 API 调用
4. **并发控制**：对大量下载和搜索操作实现适当的并发控制

CurseForge 资源管理

功能概述

CurseForge 模块提供了对 CurseForge 资源平台的完整访问支持，包括资源搜索、获取详情、下载等功能。

支持的资源类型

Mods

Mods 是最常见的资源类型，增强或修改 Minecraft 的功能。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");
```

Modpacks

Modpacks 是预配置的模组组合，提供完整的游戏体验。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeModpacks = new Modpacks("your-api-key");
```

ResourcePacks

ResourcePacks 改变游戏的视觉效果，包括材质、纹理等。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeResourcePacks = new ResourcePacks("your-api-key");
```

DataPacks

DataPacks 改变游戏规则和行为，包括合成配方、进度系统等。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeDataPacks = new DataPacks("your-api-key");
```

Shaders

Shaders 增强游戏的图形效果，包括光照、阴影等。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeShaders = new Shaders("your-api-key");
```

Worlds

Worlds 提供自定义游戏世界，包括生存地图、冒险地图等。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeWorlds = new Worlds("your-api-key");
```

资源搜索

基础搜索

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");
var searchResults = await curseForgeMods.SearchAsync("sodium");

foreach (var result in searchResults)
{
    Console.WriteLine($"Mod 名称: {result.Name}");
    Console.WriteLine($"作者: {string.Join(", ", result.Authors.Select(a => a.Name))}");
    Console.WriteLine($"下载次数: {result.DownloadCount}");
    Console.WriteLine($"简介: {result.Summary}");
    Console.WriteLine();
}
```

高级搜索

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");
var searchResults = await curseForgeMods.SearchAsync(
    "sodium",
    gameVersion: "1.20.1",
    modLoaderType: ModLoaderType.Fabric,
    sortField: SortField.Popularity,
    page: 0,
```

```
    pageSize: 20
);
```

资源详情获取

获取资源详细信息

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");
var modDetails = await curseForgeMods.GetModDetailsAsync("sodium");

Console.WriteLine($"Mod 名称: {modDetails.Name}");
Console.WriteLine($"作者: {string.Join(", ", modDetails.Authors.Select(a => a.Name))}");
Console.WriteLine($"描述: {modDetails.Description}");
Console.WriteLine($"下载次数: {modDetails.DownloadCount}");
Console.WriteLine($"最新版本: {modDetails.LatestVersion}");

// 获取所有版本
var versions = await curseForgeMods.GetModVersionsAsync("sodium");
Console.WriteLine($"可用版本数量: {versions.Count}");
foreach (var version in versions.Take(5))
{
    Console.WriteLine($"  - 版本: {version.Version} ({version.GameVersion})");
    Console.WriteLine($"    文件大小: {version.FileSize} KB");
    Console.WriteLine($"    下载链接: {version.DownloadUrl}");
}
```

资源下载

下载最新版本

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");
string modId = "sodium";

// 获取最新版本信息
var modDetails = await curseForgeMods.GetModDetailsAsync(modId);
var latestVersion = modDetails.LatestVersion;

// 下载资源
string savePath = @"C:\Downloads\Sodium-" + latestVersion + ".jar";
bool downloadSuccess = await curseForgeMods.DownloadModAsync(modId,
```

```
latestVersion, savePath);

if (downloadSuccess)
{
    Console.WriteLine($"Mod 下载成功: {savePath}");
}
else
{
    Console.WriteLine("Mod 下载失败");
}
```

下载特定版本

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");
string modId = "sodium";
string version = "0.4.14";

string savePath = @"C:\Downloads\Sodium-" + version + ".jar";
bool downloadSuccess = await curseForgeMods.DownloadModAsync(modId, version, savePath);

if (downloadSuccess)
{
    Console.WriteLine($"Mod 下载成功: {savePath}");
}
else
{
    Console.WriteLine("Mod 下载失败");
}
```

依赖处理

解析和下载依赖

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");
string modId = "sodium";

// 获取 Mod 详情
var modDetails = await curseForgeMods.GetModDetailsAsync(modId);

// 获取依赖列表
```

```
var dependencies = await curseForgeMods.GetModDependenciesAsync(modId);

Console.WriteLine($"{modId} 的依赖项:");
foreach (var dependency in dependencies)
{
    Console.WriteLine($"  - {dependency.Name} (版本: {dependency.Version})");

    // 下载依赖项
    string savePath = $@"C:\Downloads\{dependency.Name}-{dependency.Version}.jar";
    bool downloadSuccess = await curseForgeMods.DownloadModAsync(dependency.Id,
dependency.Version, savePath);

    if (downloadSuccess)
    {
        Console.WriteLine($"    下载成功: {savePath}");
    }
    else
    {
        Console.WriteLine($"    下载失败: {dependency.Name}");
    }
}
```

配置选项

设置自定义下载源

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");

// 设置自定义下载源（可选）
curseForgeMods.SetDownloadSource("https://your-custom-mirror.com/");

// 下载资源
await curseForgeMods.DownloadModAsync("sodium", "0.4.14", @"C:\Downloads\Sodium.jar");
```

调整请求超时

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");

// 调整请求超时（默认10秒）
curseForgeMods.SetRequestTimeout(TimeSpan.FromSeconds(30));
```

```
// 搜索资源
var results = await curseForgeMods.SearchAsync("sodium");
```

错误处理

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.CurseForge;

var curseForgeMods = new Mods("your-api-key");

try
{
    var searchResults = await curseForgeMods.SearchAsync("sodium");
    Console.WriteLine($"找到 {searchResults.Count} 个结果");
}
catch (ApiException ex)
{
    Console.WriteLine($"API 错误: {ex.Message}");
    Console.WriteLine($"状态码: {ex.StatusCode}");
}
catch (RateLimitException ex)
{
    Console.WriteLine($"请求限流: {ex.Message}");
    Console.WriteLine($"重试时间: {ex.RetryAfter}");
}
catch (Exception ex)
{
    Console.WriteLine($"错误: {ex.Message}");
}
```

最佳实践

1. **API密钥管理**：确保 API 密钥的安全存储和使用
2. **错误处理**：实现适当的错误处理和重试机制
3. **缓存策略**：对搜索结果和资源信息实现缓存
4. **并发控制**：对大量下载和搜索操作实现适当的并发控制
5. **请求限流**：遵守平台的请求限制，避免被封禁

Modrinth 资源管理

功能概述

Modrinth 模块提供了对 Modrinth 资源平台的完整访问支持，包括资源搜索、获取详情、下载等功能。Modrinth 是一个开源的 Minecraft 资源平台，专注于现代 Mod 生态系统。

支持的资源类型

Mods

Mods 是最常见的资源类型，增强或修改 Minecraft 的功能。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();
```

Modpacks

Modpacks 是预配置的模组组合，提供完整的游戏体验。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthModpacks = new ModPacks();
```

ResourcePacks

ResourcePacks 改变游戏的视觉效果，包括材质、纹理等。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthResourcePacks = new ResourcePacks();
```

DataPacks

DataPacks 改变游戏规则和行为，包括合成配方、进度系统等。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthDataPacks = new DataPacks();
```

Shaders

Shaders 增强游戏的图形效果，包括光照、阴影等。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthShaders = new Shaders();
```

Worlds

Worlds 提供自定义游戏世界，包括生存地图、冒险地图等。

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthWorlds = new Worlds();
```

资源搜索

基础搜索

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();
var searchResults = await modrinthMods.SearchAsync("sodium");

foreach (var result in searchResults)
{
    Console.WriteLine($"Mod 名称: {result.Name}");
    Console.WriteLine($"作者: {string.Join(", ", result.Authors)}");
    Console.WriteLine($"下载次数: {result.DownloadCount}");
    Console.WriteLine($"简介: {result.Summary}");
    Console.WriteLine();
}
```

高级搜索

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();
var searchResults = await modrinthMods.SearchAsync(
    "sodium",
    gameVersion: "1.20.1",
    loaders: new[] { "fabric" },
    categories: new[] { "optimization" },
    index: Index.Popularity,
```

```
    page: 0,
    pageSize: 20
);
```

资源详情获取

获取资源详细信息

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();
var modDetails = await modrinthMods.GetModDetailsAsync("AANobbMI");

Console.WriteLine($"Mod 名称: {modDetails.Name}");
Console.WriteLine($"作者: {string.Join(", ", modDetails.Authors)}");
Console.WriteLine($"描述: {modDetails.Description}");
Console.WriteLine($"下载次数: {modDetails.DownloadCount}");
Console.WriteLine($"最新版本: {modDetails.LatestVersion}");

// 获取所有版本
var versions = await modrinthMods.GetModVersionsAsync("AANobbMI");
Console.WriteLine($"可用版本数量: {versions.Count}");
foreach (var version in versions.Take(5))
{
    Console.WriteLine($"  - 版本: {version.Version} ({version.GameVersion})");
    Console.WriteLine($"    文件大小: {version.FileSize} KB");
    Console.WriteLine($"    下载链接: {version.DownloadUrl}");
}
```

资源下载

下载最新版本

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();
string modId = "AANobbMI"; // Sodium 的 Modrinth ID

// 获取最新版本信息
var modDetails = await modrinthMods.GetModDetailsAsync(modId);
var latestVersion = modDetails.LatestVersion;

// 下载资源
string savePath = @"C:\Downloads\Sodium-" + latestVersion + ".jar";
```

```
bool downloadSuccess = await modrinthMods.DownloadModAsync(modId, latestVersion, savePath);

if (downloadSuccess)
{
    Console.WriteLine($"Mod 下载成功: {savePath}");
}
else
{
    Console.WriteLine("Mod 下载失败");
}
```

下载特定版本

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();
string modId = "AANobbMI"; // Sodium 的 Modrinth ID
string version = "0.4.14";

string savePath = @"C:\Downloads\Sodium-" + version + ".jar";
bool downloadSuccess = await modrinthMods.DownloadModAsync(modId, version, savePath);

if (downloadSuccess)
{
    Console.WriteLine($"Mod 下载成功: {savePath}");
}
else
{
    Console.WriteLine("Mod 下载失败");
}
```

依赖处理

解析和下载依赖

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();
string modId = "AANobbMI"; // Sodium 的 Modrinth ID

// 获取 Mod 详情
var modDetails = await modrinthMods.GetModDetailsAsync(modId);

// 获取依赖列表
```

```
var dependencies = await modrinthMods.GetModDependenciesAsync(modId);

Console.WriteLine($"{modId} 的依赖项:");
foreach (var dependency in dependencies)
{
    Console.WriteLine($"  - {dependency.Name} (版本: {dependency.Version})");

    // 下载依赖项
    string savePath = $@"C:\Downloads\{dependency.Name}-{dependency.Version}.jar";
    bool downloadSuccess = await modrinthMods.DownloadModAsync(dependency.Id,
dependency.Version, savePath);

    if (downloadSuccess)
    {
        Console.WriteLine($"    下载成功: {savePath}");
    }
    else
    {
        Console.WriteLine($"    下载失败: {dependency.Name}");
    }
}
```

配置选项

设置自定义下载源

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();

// 设置自定义下载源（可选）
modrinthMods.SetDownloadSource("https://your-custom-mirror.com/");

// 下载资源
await modrinthMods.DownloadModAsync("AANobbMI", "0.4.14", @"C:\Downloads\Sodium.jar");
```

调整请求超时

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();

// 调整请求超时（默认10秒）
modrinthMods.SetRequestTimeout(TimeSpan.FromSeconds(30));
```

```
// 搜索资源
var results = await modrinthMods.SearchAsync("sodium");
```

错误处理

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();

try
{
    var searchResults = await modrinthMods.SearchAsync("sodium");
    Console.WriteLine($"找到 {searchResults.Count} 个结果");
}
catch (ApiException ex)
{
    Console.WriteLine($"API 错误: {ex.Message}");
    Console.WriteLine($"状态码: {ex.StatusCode}");
}
catch (RateLimitException ex)
{
    Console.WriteLine($"请求限流: {ex.Message}");
    Console.WriteLine($"重试时间: {ex.RetryAfter}");
}
catch (Exception ex)
{
    Console.WriteLine($"错误: {ex.Message}");
}
```

最佳实践

1. **API 调用频率**：Modrinth API 有请求限制，建议实现适当的缓存
2. **错误处理**：实现适当的错误处理和重试机制
3. **缓存策略**：对搜索结果和资源信息实现缓存
4. **并发控制**：对大量下载和搜索操作实现适当的并发控制
5. **资源版本管理**：确保下载的资源与 Minecraft 版本兼容

资源元数据

Modrinth 资源包含丰富的元数据：

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;
```

```
var modrinthMods = new Mods();
var modDetails = await modrinthMods.GetModDetailsAsync("AANobbMI");

Console.WriteLine("资源元数据:");
Console.WriteLine($"ID: {modDetails.Id}");
Console.WriteLine($"Slug: {modDetails.Slug}");
Console.WriteLine($"许可证: {modDetails.License}");
Console.WriteLine($"项目主页: {modDetails.Homepage}");
Console.WriteLine($"源代码: {modDetails.SourceCode}");
Console.WriteLine($"问题追踪: {modDetails.IssueTracker}");
Console.WriteLine($"Discord: {modDetails.Discord}");
Console.WriteLine($"下载总数: {modDetails.DownloadCount}");
Console.WriteLine($"创建时间: {modDetails.CreatedAt}");
Console.WriteLine($"更新时间: {modDetails.UpdatedAt}");
Console.WriteLine($"标签: {string.Join(", ", modDetails.Tags)}");
```

批量操作

批量获取资源详情

```
using Qomicex.Core.Modules.Helpers.Resources.Expansion.Modrinth;

var modrinthMods = new Mods();
var modIds = new[] { "AANobbMI", "P7dR8mSH", "Mr4mRFp7" }; // Sodium, Lithium, Phosphor

var modDetailsList = new List<ModDetails>();

foreach (var modId in modIds)
{
    try
    {
        var details = await modrinthMods.GetModDetailsAsync(modId);
        modDetailsList.Add(details);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"获取资源 {modId} 详情失败: {ex.Message}");
    }
}

// 处理结果
foreach (var details in modDetailsList)
{
    Console.WriteLine($"资源: {details.Name}");
    Console.WriteLine($"作者: {string.Join(", ", details.Authors)}");
```

```
    Console.WriteLine($"下载次数: {details.DownloadCount}");
    Console.WriteLine();
}
```