Использование RestEase в ASP.NET Core

Игорь Чакрыгин | OZON

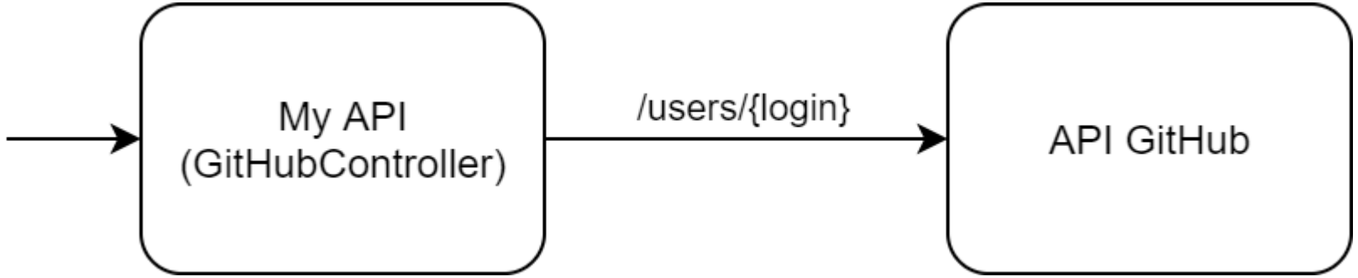
О чём доклад

- Различные способы использования HttpClient.
- Библиотека RestEase.
- Наши расширения для RestEase:
  - Регистрация RestClient в одну строку.
  - Обработка и пробрасывание исключений.

Использование HttpClient

Простой пример:

Необходимо сделать сервис на ASP.NET Core, который будет использовать API GitHub для получения информации о пользователе по логину.



https://api.github.com/users/chakrygin

https://api.github.com/users/cha

api.github.com/users/chakrygin


```
// 20191006174350
// https://api.github.com/users/chakrygin

{
  "login": "Chakrygin",
  "id": 6216874,
  "node_id": "MDQ6VXNlcjYyMTY4NzQ=",
  "avatar_url": "https://avatars2.githubusercontent.com/u/6216874?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/Chakrygin",
  "html_url": "https://github.com/Chakrygin",
  "followers_url": "https://api.github.com/users/Chakrygin/followers",
  "following_url": "https://api.github.com/users/Chakrygin/following{/other_user}",
  "gists_url": "https://api.github.com/users/Chakrygin/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/Chakrygin/starred{/owner}{/repo}",
  "subscriptions_url": "https://api.github.com/users/Chakrygin/subscriptions",
  "organizations_url": "https://api.github.com/users/Chakrygin/orgs",
  "repos_url": "https://api.github.com/users/Chakrygin/repos",
  "events_url": "https://api.github.com/users/Chakrygin/events{/privacy}",
  "received_events_url": "https://api.github.com/users/Chakrygin/received_events",
  "type": "User",
  "site_admin": false,
  "name": "Igor Chakrygin",
```

```
    "company": null,
    "blog": "chakrygin.ru",
    "location": "Russia, Moscow",
```

Решение «в лоб»

Просто создаём и используем HttpClient

```
1  public sealed class GitHubController : Controller
2  {
3      [HttpGet("users")]
4      public async Task<User> GetUser([FromQuery] string login)
5      {
6          var httpClient = new HttpClient();
7          httpClient.BaseAddress = new Uri("https://api.github.com");
8          httpClient.DefaultRequestHeaders.Add("User-Agent", "HttpClient");
9
10         using (var response = await httpClient.GetAsync($"users/{login}"))
11         {
12             response.EnsureSuccessStatusCode();
13
14             return await response.Content.ReadAsAsync<User>();
15         }
16     }
17 }
```

HttpClientFactory

Регистрируем именованый HttpClient

```
1  public sealed class Startup
2  {
3      public void ConfigureServices(IServiceCollection services)
4      {
5          services.AddHttpClient("GitHub", httpClient =>
6          {
7              httpClient.BaseAddress = new Uri("https://api.github.com");
8              httpClient.DefaultRequestHeaders.Add("User-Agent", "GitHubClient");
9          });
10
11         // ...
12     }
13
14     // ...
15 }
```

Используем именованый HttpClient

```
1  public sealed class GitHubController : Controller
2  {
3      private readonly HttpClient _httpClient;
4
5      public GitHubController(IHttpClientFactory httpClientFactory)
6      {
7          _httpClient = _httpClientFactory.CreateClient("GitHub");;
8      }
9
10     [HttpGet("users")]
11     public async Task<User> GetUser([FromQuery] string login)
12     {
13         using (var response = await _httpClient.GetAsync($"users/{login}"))
14         {
15             response.EnsureSuccessStatusCode();
16
17             return await response.Content.ReadAsAsync<User>();
18         }
19     }
20 }
```

Типизированный HttpClient

Объявляем типизированный HttpClient

```
1 public sealed class GitHubClient
2 {
3     // ...
4 }
```

Регистрируем типизированный HttpClient

```
1  public sealed class Startup
2  {
3      public void ConfigureServices(IServiceCollection services)
4      {
5          services.AddHttpClient<GitHubClient>(httpClient =>
6          {
7              httpClient.BaseAddress = new Uri("https://api.github.com");
8              httpClient.DefaultRequestHeaders.Add("User-Agent", "GitHubClient");
9          });
10
11         // ...
12     }
13
14     // ...
15 }
```

Регистрируем типизированный HttpClient

```
1  public sealed class Startup
2  {
3      public void ConfigureServices(IServiceCollection services)
4      {
5          services
6              .AddHttpClient(nameof(GitHubClient), httpClient =>
7              {
8                  httpClient.BaseAddress = new Uri("https://api.github.com");
9                  httpClient.DefaultRequestHeaders.Add("User-Agent", "GitHubClient");
10             })
11             .AddTypedClient<GitHubClient>();
12
13         // ...
14     }
15
16     // ...
17 }
```

Используем типизированный HttpClient

```
1  public sealed class GitHubController : Controller
2  {
3      private readonly GitHubClient _gitHubClient;
4
5      public GitHubController(GitHubClient gitHubClient)
6      {
7          _gitHubClient = gitHubClient;
8      }
9
10     [HttpGet("users")]
11     public async Task<User> GetUser([FromQuery] string login)
12     {
13         return await _gitHubClient.GetUser(login);
14     }
15 }
```

Реализация типизированного HttpClient

```
1  public sealed class GitHubClient
2  {
3      private readonly HttpClient _httpClient;
4
5      public GitHubClient(HttpClient httpClient)
6      {
7          _httpClient = httpClient;
8      }
9
10     public async Task<User> GetUser(string login)
11     {
12         using (var response = await _httpClient.GetAsync($"users/{login}"))
13         {
14             response.EnsureSuccessStatusCode();
15
16             return await response.Content.ReadAsAsync<User>();
17         }
18     }
19 }
```

RestEase

RestEase

- Библиотека для реализации REST-клиентов.
- https://github.com/canton7/RestEase
- Особенности:
  - Простота
  - Строгая типизация
  - Кодогенерация в runtime

Объявляем интерфейс RestClient

```
1 [Header("User-Agent", "GitHubClient")]
2 public interface IGitHubClient
3 {
4     [Get("users/{login}")]
5     Task<User> GetUser([Path] string login);
6 }
```

Регистрируем RestClient

```
1  public sealed class Startup
2  {
3      public void ConfigureServices(IServiceCollection services)
4      {
5          services
6              .AddHttpClient(nameof(IGitHubClient), httpClient =>
7              {
8                  httpClient.BaseAddress = new Uri("https://api.github.com");
9              })
10             .AddTypedClient<IGitHubClient>(httpClient =>
11             {
12                 return RestClient.For<IGitHubClient>(httpClient);
13             });
14
15         // ...
16     }
17
18     // ...
19 }
```

Используем RestClient

```
1  public sealed class GitHubController : Controller
2  {
3      private readonly IGitHubClient _gitHubClient;
4
5      public GitHubController(IGitHubClient gitHubClient)
6      {
7          _gitHubClient = gitHubClient;
8      }
9
10     [HttpGet("users")]
11     public async Task<User> GetUser([FromQuery] string login)
12     {
13         return await _gitHubClient.GetUser(login);
14     }
15 }
```

Возможности RestEase

Атрибуты HTTP-методов и параметров

```
1  public interface IUserApi
2  {
3      [Get("users")]
4      Task<List<User>> GetUsers([Query] int page = 1);
5
6      [Get("users/{userId}")]
7      Task<User> GetUser([Path] long userId);
8
9      [Post("users")]
10     Task CreateUser([Body] User user);
11
12     [Put("users/{userId}")]
13     Task UpdateUser([Path] long userId, [Body] User user);
14
15     [Delete("users/{userId}")]
16     Task DeleteUser([Path] long userId);
17 }
```

Типы возвращаемых значений

- `Task`
- `Task<T>` (Используется Json.NET)
- `Task<string>`
- `Task<HttpResponseMessage>`
- `Task<Response<T>>`
- `Task<Stream>`

Поддержка CancellationToken

```
1 public interface IUserApi
2 {
3     [Post("users")]
4     Task CreateUser([Body] User user, CancellationToken cancellationToken);
5 }
```

Поддержка RawQueryString

```
1 public interface IUserApi
2 {
3     [Get("users/search")]
4     Task<List<User>> SearchUsers(
5         [RawQueryString] SearchUsersRequest request);
6 }
```

Обработка ошибок (ApiException)

```
1 try
2 {
3     return _userApi.GetUser(userId);
4 }
5 catch (ApiException ex)
6     when(ex.StatusCode == HttpStatusCode.NotFound)
7 {
8     return null;
9 }
```

Обработка ошибок (AllowAnyStatusCode)

```
1 public interface IUserApi
2 {
3     [AllowAnyStatusCode]
4     [Get("users/{userId}")]
5     Task<User> GetUser([Path] long userId);
6 }
```

Другие возможности:

- Передача параметров через свойства,
  а не через параметры метода.
- Поддержка обобщённых интерфейсов и методов
- Поддержка наследования интерфейсов
- Возможности для кастомизации

Расширение AddRestClient для RestEase

Регистрируем RestClient

```
1  public sealed class Startup
2  {
3      public void ConfigureServices(IServiceCollection services)
4      {
5          services
6              .AddHttpClient(nameof(IGitHubClient), httpClient =>
7              {
8                  httpClient.BaseAddress = new Uri("https://api.github.com");
9              })
10             .AddTypedClient<IGitHubClient>(httpClient =>
11             {
12                 return RestClient.For<IGitHubClient>(httpClient);
13             });
14
15         // ...
16     }
17
18     // ...
19 }
```

Расширение AddRestClient

```
1  public static class RestClientExtensions
2  {
3      public static IHttpClientBuilder AddRestClient<T>(
4          this IServiceCollection services, Action<HttpClient> configure = null)
5          where T : class
6      {
7          return services
8              .AddHttpClient(typeof(T).Name, (serviceProvider, httpClient) =>
9              {
10                 var configuration = serviceProvider.GetRequiredService<IConfiguration>();
11                 var section = configuration.GetSection(typeof(T).Name.TrimStart('I'));
12                 httpClient.BaseAddress = section.GetValue<Uri>("BaseAddress");
13                 httpClient.Timeout = section.GetValue<TimeSpan>("Timeout");
14
15                 if (configure != null)
16                     configure(httpClient);
17             })
18             .AddTypedClient<T>(RestClient.For<T>);
19     }
20 }
```

Используем AddRestClient

```
1  public sealed class Startup
2  {
3      public void ConfigureServices(IServiceCollection services)
4      {
5          services.AddRestClient<IGitHubClient>();
6
7          // ...
8      }
9
10     // ...
11 }
```

appsettings.json

```
1 {
2     "GitHubClient": {
3         "BaseAddress": "https://api.github.com",
4         "Timeout": "00:00:10"
5     }
6 }
```

Обработка исключений с RestEase

Фильтр для обработки исключений

```
1  public sealed class ExceptionFilter : ExceptionFilterAttribute
2  {
3      private readonly IHostingEnvironment _hostingEnvironment;
4
5      public ExceptionFilter(IHostingEnvironment hostingEnvironment)
6      {
7          _hostingEnvironment = hostingEnvironment;
8      }
9
10     public override void OnException(ExceptionContext context)
11     {
12         context.Result = context.Exception is ApiException ex
13             ? GetApiExceptionResult(ex)
14             : GetExceptionResult(context.Exception);
15     }
16 }
```

Ответ сервиса для обычных ошибок

```
1  private IActionResult GetExceptionResult(Exception ex)
2  {
3      var response = new ExceptionResponse
4      {
5          ApplicationName = _hostingEnvironment.ApplicationName,
6          EnvironmentName = _hostingEnvironment.EnvironmentName,
7          Message = ex.Message,
8          StackTrace = ex.StackTrace.Split(Environment.NewLine),
9      };
10
11     var result = new JsonResult(response);
12     result.StatusCode = StatusCodes.Status500InternalServerError;
13
14     return result;
15 }
```

Ответ сервиса для HTTP-ошибок

```
1 private IActionResult GetApiExceptionResult(ApiException ex)
2 {
3     return new ContentResult
4     {
5         StatusCode = (int) ex.StatusCode,
6         Content = ex.Content,
7         ContentType = ex.ContentHeaders.ContentType.ToString(),
8     };
9 }
```

Пример исключения


```
4  {
5    "applicationName": "Ozon.Pvz.Api.User",
6    "environmentName": "Staging",
7    "message": "ArticleCheckInCurrentPlace:  Операция отменена. Не найден пользователь системы.\nТранзакция завершилась в триггере. Выполнение пакета
     прервано.",
8    "stackTrace": [
9      "   at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)",
10     "   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)",
11     "   at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)",
12     "   at System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet
       bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)",
13     "   at System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString)",
14     "   at System.Data.SqlClient.SqlCommand.CompleteAsyncExecuteReader()",
15     "   at System.Data.SqlClient.SqlCommand.EndExecuteNonQueryInternal(IAsyncResult asyncResult)",
16     "   at System.Data.SqlClient.SqlCommand.EndExecuteNonQuery(IAsyncResult asyncResult)",
17     "   at System.Threading.Tasks.TaskFactory`1.FromAsyncCoreLogic(IAsyncResult iar, Func`2 endFunction, Action`1 endAction, Task`1 promise, Boolean
       requiresSynchronization)",
18     "--- End of stack trace from previous location where exception was thrown ---",
```