

ClassName

static ANBTX

Method(API)

```
/// <summary>
/// Connection 객체를 생성합니다.
/// </summary>
private static void Connect()
{
    _client = new HttpClient();
    _client.BaseAddress = new Uri("http://api.anbtech.net:8080/");
    _client.DefaultRequestHeaders.Accept.Clear();

    _client.DefaultRequestHeaders.Accept.Add(
        new MediaTypeWithQualityHeaderValue("application/json"));

    IsConnect = true;
}
```

```
/// <summary>
/// Connection 객체를 닫습니다.
/// </summary>
private static void Close()
{
    if (_client != null) _client.Dispose();
    _client = null;
    IsConnect = false;
}
```

```
/// <summary>
/// 입력된 API 함수를 이용하여 데이터를 수집 합니다.
/// </summary>
/// <param name="strAPI"> </param>
/// <returns> </returns>
public static HttpResponseMessage Get(string strAPI)
{
    if (_client == null) Connect();
    return _client.GetAsync(strAPI).Result;
}
```

Sample Code : WinRestAPI 프로젝트의 /FrmRestAPITest.cs/btnRead_Click 함수 확인

Sample code:

```
var lstEmployee = new List<EmployeeVO>();
var response = ANBTX.Get("/api/employee");
if (response.IsSuccessStatusCode)
{
    lstEmployee = response.Content.ReadAsAsync<List<EmployeeVO>>().Result;
}
```

```
/// <summary>
/// 입력된 API 함수를 이용하여 CREATE 합니다.
/// </summary>
/// <param name="strAPI"> 호출 할 "/api/함수명" </param>
/// <param name="inputVo"> 함수에 입력 할 VO 객체 </param>
public static void Create(string strAPI, object inputVO)
{
    if (_client == null) Connect();
    HttpResponseMessage response = _client.PostAsJsonAsync(strAPI, inputVO).Result;
    response.EnsureSuccessStatusCode();
}
```

```
/// <summary>
/// 입력된 API 함수를 이용하여 UPDATE 합니다.
/// </summary>
/// <param name="strAPI"> 호출 할 "/api/함수명" </param>
/// <param name="inputVO"> 함수에 입력 할 VO 객체 </param>
public static void Update(string strAPI, object inputVO)
{
    // EmployeeVO, ProjectVO...etc..
    if (_client == null) Connect();
```

```
return PatchAsJsonAsync(_client, strAPI, inputVO);
```

```
}
```

comment : 현재 empId가 기존 값인 경우 update , empId가 기존에 없는 경우 create 액션이 일어남

```
public static HttpResponseMessage PatchAsJsonAsync<T>(this HttpClient client, string requestUri, T value)
{
    var content = new ObjectContent<T>(value, new JsonMediaTypeFormatter());
    var request = new HttpRequestMessage(new HttpMethod("PATCH"), requestUri) { Content = content };

    return client.SendAsync(request).Result;
}
```

```

/// <summary>
/// 입력된 API 함수를 이용하여 지정된 키값에 해당하는 값을 DELETE 합니다.
/// </summary>
/// <param name="strAPI">호출 할 "/api/함수명" </param>
/// <param name="id">제거하기위한 키값</param>
/// <returns></returns>
public HttpStatusCode Delete(string strAPI, string id)
{
    if (_client == null) Connect();

    var response = _client.DeleteAsync(string.Format("{0}/{1}",strAPI, id)).Result;

    return response.StatusCode;
}

```

comment : 호출 할 함수 URI의 마지막에 ID 정보를 추가하여 DeleteAsync를 호출하면 해당 ID가 제거됨.

1. c#에서 RestAPIClient 호출시

ANTBX_Update("/api/employee", vo객체)

: URI 전체 경로를 입력 안하는 이유는 RestAPIClient 의 Connect

메소드 호출시 앞부분에 해당하는 경로를 지정함

2. http://api.anbtech.net:8080/swagger-ui.html에서 Model Schema를 보고 VO 클래스 정의 하기

: json mapping을 위해 반드시 함수에서 전달된 모델스키마 구조를 그대로 구현해주어야 한다.

anbtech-rank-controller : Anbtech Rank Cor

GET

/api/rank

Response Class (Status 200)

Model

Model Schema

```

[
  {
    "rankCode": "string",
    "rankName": "string",
    "rankOrder": 0,
    "regEmpId": "string",
    "regEmpNm": "string",
    "registDate": "2017-03-14T14:20:39.548Z",
    "updateDate": "2017-03-14T14:20:39.548Z",
    "useYn": "string"
  }
]

```

"[" 는 List등의 배열을 의미 하고

"{" 는 하나의 Map 객체를 의미하며 보통 이부분을 VO Class 화 시킨다.

/api/rank를 C#으로 변환하면 List<Rank> 형태의 객체를 얻을 수 있다.

ps. 좌측형태로 만드는걸 직렬화 아래의 클래스오브젝트 형태로 만드는걸 역직렬화(Deserialization) 이라고 함.

C#의 VO 객체는 반드시 [Serializable]을 붙여서 생성한다.

```
[Serializable]
참조 2개
public class RankVO
{
    public string rankCode;
    public string rankName;
    public int rankOrder;
    public string regEmpId;
    public string regEmpNm;
    참조 1개
    public DateTime registDate { get; set; }
    참조 1개
    public DateTime updateDate { get; set; }
    public string useYn;
}
```