عنوان: **تهیه‌ی گزارش از آخرین ارائه‌های یک پروژه در GitHub**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۰۳ ۱۱:۲۰
آدرس: www.dotnettips.info
گروه‌ها: JSON, Software deployment, json.net, GitHub

چندی قبل مطلب « اطلاع از بروز رسانی نرم افزار ساخته شده » را در سایت جاری مطالعه کردید. در این روش بسیار متداول، شماره نگارش‌های جدید برنامه در یک فایل XML و مانند آن قرار می‌گیرند و برنامه هربار این فایل را جهت یافتن شماره‌های مندرج در آن اسکن می‌کند. اگر پروژه‌ی شما سورس باز است و در GitHub هاست شده، روش دیگری نیز برای یافتن این اطلاعات وجود دارد. در GitHub می‌توان از طریق آدرسی به شکل https://api.github.com/repos/user_name/project_name/releases به اطلاعات آخرین ارائه‌های یک پروژه (قرار گرفته در برگه‌ی releases آن) با فرمت JSON دسترسی یافت (یک مثال). در ادامه قصد داریم روش استفاده‌ی از آن‌را بررسی کنیم.

ساختار JSON ارائه‌های یک پروژه در GitHub

ساختار کلی اطلاعات ارائه‌های یک پروژه در GitHub چنین شکلی را دارد:

```
[
  {
    release_info,
    "assets": [
      {
      }
    ]
  }
]
```

در اینجا آرایه‌ای از اطلاعات ارائه‌ی یک پروژه ارسال می‌شود. هر ارائه نیز دارای دو قسمت است: لینکی به صفحه‌ی اصلی release در GitHub و سپس آرایه‌ای به نام assets که در آن اطلاعات فایل‌های پیوستی مانند نام فایل، آدرس، اندازه و امثال آن قرار گرفته‌اند.

تهیه‌ی کلاس‌های معادل فرمت JSON ارائه‌های برنامه در GitHub

اگر بخواهیم قسمت‌های مهم خروجی JSON فوق را تبدیل به کلاس‌های معادل دات نتی کنیم، به دو کلاس ذیل خواهیم رسید:

```
using Newtonsoft.Json;
using System;

namespace ApplicationAnnouncements
{
    public class GitHubProjectRelease
    {
        [JsonProperty(PropertyName = "url")]
        public string Url { get; set; }

        [JsonProperty(PropertyName = "assets_url")]
        public string AssetsUrl { get; set; }

        [JsonProperty(PropertyName = "upload_url")]
        public string UploadUrl { get; set; }

        [JsonProperty(PropertyName = "html_url")]
        public string HtmlUrl { get; set; }

        [JsonProperty(PropertyName = "id")]
        public int Id { get; set; }

        [JsonProperty(PropertyName = "tag_name")]
        public string TagName { get; set; }

        [JsonProperty(PropertyName = "target_commitish")]
        public string TargetCommitish { get; set; }

        [JsonProperty(PropertyName = "name")]
```

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

        [JsonProperty(PropertyName = "body")]
        public string Body { get; set; }

        [JsonProperty(PropertyName = "draft")]
        public bool Draft { get; set; }

        [JsonProperty(PropertyName = "prerelease")]
        public bool PreRelease { get; set; }

        [JsonProperty(PropertyName = "created_at")]
        public DateTime CreatedAt { get; set; }

        [JsonProperty(PropertyName = "published_at")]
        public DateTime PublishedAt { get; set; }

        [JsonProperty(PropertyName = "assets")]
        public Asset[] Assets { get; set; }
    }

    public class Asset
    {
        [JsonProperty(PropertyName = "url")]
        public string Url { get; set; }

        [JsonProperty(PropertyName = "id")]
        public int Id { get; set; }

        [JsonProperty(PropertyName = "name")]
        public string Name { get; set; }

        [JsonProperty(PropertyName = "label")]
        public string Label { get; set; }

        [JsonProperty(PropertyName = "content_type")]
        public string ContentType { get; set; }

        [JsonProperty(PropertyName = "state")]
        public string State { get; set; }

        [JsonProperty(PropertyName = "size")]
        public int Size { get; set; }

        [JsonProperty(PropertyName = "download_count")]
        public int DownloadCount { get; set; }

        [JsonProperty(PropertyName = "created_at")]
        public DateTime CreatedAt { get; set; }

        [JsonProperty(PropertyName = "updated_at")]
        public DateTime UpdatedAt { get; set; }
    }
}
```

در اینجا از ویژگی JsonProperty جهت معرفی نام‌های واقعی خواص ارائه شده‌ی توسط GitHub استفاده کرده‌ایم.
پس از تشکیل این کلاس‌ها، مرحله‌ی بعد، دریافت اطلاعات JSON از آدرس API ارائه‌های پروژه در GitHub و سپس نگاشت آن‌ها می‌باشد:

```
using (var webClient = new WebClient())
{
    webClient.Headers.Add("user-agent", "DNTProfiler");
    var jsonData = webClient.DownloadString(url);
    var gitHubProjectReleases = JsonConvert.DeserializeObject<GitHubProjectRelease[]>(jsonData);

    foreach (var release in gitHubProjectReleases)
    {
       foreach (var asset in release.Assets)
       {
           // …
       }
    }
}
```

حین کار با WebClient یا هر روش دیگری که برای دریافت اطلاعات از وب استفاده می‌کنید، حتما نیاز است user-agent را ذکر

کرد. در غیر اینصورت GitHub درخواست شما را برگشت خواهد زد. پس از دریافت اطلاعات JSON، با استفاده از متد JsonConvert.DeserializeObject کتابخانه‌ی JSON.NET ، می‌توان آن‌ها را تبدیل به آرایه‌ای از GitHubProjectRelease کرد.

یک نکته: اگر به صفحه‌ی اصلی ارائه‌های یک پروژه در GitHub دقت کنید، شماره‌ی تعداد بار دریافت یک ارائه مشخص نشده‌است. در این API، عدد DownloadCount، بیانگر تعداد بار دریافت پروژه‌ی شما است.