Top 5 New Features in ASP.Net Web API 2

Updated date Feb 01, 2021

**1. Attribute Routing**  
  
Along with convention-based routing, Web API 2 now supports attribute routing as well.  
  
In case of convention-based routing, we can define multiple route templates. When a request comes, it will be matched against already defined route templates, and forwarded to specific controller action depending on matched template.  
  
You can see the following default route template in the routing table for Web API:

Config.Routes.MapHttpRoute(

     name: "DefaultApi",

     routeTemplate: "api/{Controller}/{id}",

     defaults: **new** { id = RouteParameter.Optional }

);

This routing approach has benefits that all routing templates are defined at one common location but for certain URI patterns, it really becomes difficult to support (like nested routing on the same controller).  
  
With ASP.NET Web API 2, we can easily support the the URI pattern described above and others as well. The following shows an example of a URI pattern with attribute routing.  
  
URI Pattern --> books/1/authors

[Route("books/{bookId}/authors")]

**public** IEnumerable<Author> GetAuthorByBook(**int** bookId) { ..... }

**2. Cross Origin Resource Sharing (CORS)**  
  
Normally, browsers don't allow making cross-domain calls due to the same-origin policy and we know that. So, what exactly is Cross Origin Resource Sharing (CORS)?  
  
CORS is a mechanism that allows a web page to make an AJAX call to a domain other than the domain that actually rendered that specific web page. CORS is compliant with W3C standards and now the ASP.NET Web API supports it in version 2.

**3. Open Web Interface for .NET (OWIN) self-hosting**  
  
ASP.NET Web API 2 comes with a new self-hosting package, in other words Microsoft.AspNet.WebApi. OwinSelfHost.

According to <http://owin.org/>  
  
"OWIN defines a standard interface between .NET web servers and web applications. The goal of the OWIN interface is to decouple server and application, encourage the development of simple modules for .NET web development, and, by being an open standard, stimulate the open source ecosystem of .NET web development tools."  
  
So, according to the preceding description, OWIN is an ideal option for self-hosting a web application in a process other than IIS process.  
  
There are a number of OWIN implementations, like Giacomo, Kayak, Firefly and so on available (some may be partial or outdated) but Katana is the recommended one for Microsoft servers and Web API frameworks.

**4. IHttpActionResult**  
  
Along with the existing two approaches of creating a response from the controller action, ASP.NET Web API 2 now supports another way of doing it.  
  
IHttpResponseMessage is basically an interface that acts as a factory for HttpResponseMessage. It's very powerful because it extends the Web API. Using this approach we can compose any specific type of response.  
  
Please see the following link to learn [how to serve HTML with IHttpActionResult.](http://www.strathweb.com/2013/06/ihttpactionresult-new-way-of-creating-responses-in-asp-net-web-api-2/)

**5. Web API OData**  
  
The Open Data Protocol (OData) is a web protocol for querying and updating data. ASP.NET Web API 2 has added support for $expand, $select, and $value options for OData. By using these options, we can control the representation that is returned from the server.

* **$expand:** Normally, a response doesn't include related entities if we query an an OData collection. By using $expand, we can get related entities inline in response.
* **$select:** It's used if we want to include a subset of properites in response instead of all.
* **$value**: It allows returning a raw value of the property instead returning in OData format.