Video 3.2

ImplementingGET

|  |
| --- |
| **Metadata**: Spot the problem, highlight it, and design the solution in 3 core steps  (To be covered in the video) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem / Solution (Not more than 50 words)** | **Step 1 (Not more than 10 words)** | **Step 2 (Not more than 10 words)** | **Step 3 (Not more than 10 words)** |
| This video explains GET verb implementation | Implementation walk though of GET. | Execute the Get commands and explain related items |  |

|  |
| --- |
| **Script** the Video – Plan your narration (viewers will see and hear this) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Introduction** | | | |
| **No.** | **Action on Screen  (Code / Screenshots / One line about the action occurring on screen)** | **Narration**  **(The corresponding explanation to the Action on Screen)** | | |
| 1 | Replace the slide with a screenshot of the actual slide  C:\Data\Work\ForPackt\Deliverables\Draft-Scripts\Section 3\Video 3.2\images\321.png | | **Video Introduction**  In the previous video we understood HTTP Verbs and now we will jump to the coding part and see implementation of Get verb. |
| 2 |  | |  |

|  |  |  |
| --- | --- | --- |
| **Steps or Tasks**  (Refer to the Writing Guidelines- Script Best Practices) | | |
| **No.** | **Action on Screen  (Code / Screenshots / One line about the action occurring on screen)** | **Narration**  **(The corresponding explanation to the Action on Screen)** |
| 3 |  | For any web api, we start with GET verb. In our previous section, we have implemented our first controller and exposed an API which returns a collection of resources or any specific resource based on the passed identifier. There we used GET. Let’s discuss it in bit detail. |
| 4 | [Produces("text/xml")]  public IEnumerable<Book> Get()  {  return this.repository.GetAllBooks();  } | Let’s first remove the Produces keyword from the Get method as it was used for explaining a new feature of ASP.NET 5. JSON is accepted everywhere and used so we will use it in our sample. |
| 5 |  | Let’s take a look at our first method Get which is returning list of Object. One key point, I would want to reiterate is that in Web API, we normally use the method name same as HTTP Verb so we don’t need to provide any more details for mapping of http verb to method. It looks clean and we save some extra code. Therefore for all the verbs that we discussed in previous video, we can create the method name same as the verb name. So we used here as Get. |
| 6 |  | Although it is not mandatory, we can have different names as well. Let’s first run the application  Now let’s go to Postman and put the resource URL.  http://localhost:<port>/api/books  Here we are specifying the GET method and lets send the request.  We get JSON response as expected which is list of books |
| 7 |  | Now let’s change the name of the method to FetchData and run it Send the request. We can see it is not able to call it as we get Not found status code. To make it working we need HTTPGet verb as..  Let’s run it and send the request again.  Awesome! we got it working |
| 8 |  | Let’s move to another method where we passed an Id and includeAuthor parameter while second one is optional. Here we had put HTTPGet keyword and this is required because here we are defining the route so that whenever the Id and other parameter is passed in URL it gets available as a method parameter. Let’s also initialize another property of this attribute name whose route name is *GetBooks* |
| 9 | mvcBuilder.AddJsonOptions(options =>  {  options.SerializerSettings.ReferenceLoopHandling = ReferenceLoopHandling.Ignore; }  ); | Let’s run it and send the request with id as *4* and includeAuthor as true  Oh.. it is showing InternalServerError. Now let’s try with false  It is working as expected!  This means that it is not able to serialize when Author is initialized. Here there is an issue in schema as Book contains Auhtor and Author contains books which makes a circular reference so it is not getting serialized.  To resolve this issue we need to configure the serializer so as to set it to ignore the loop reference. It can be done as  Now let’s run it again.  Awesome Now we are able to see the Book with Author details |
| 10 | options.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver(); | If we see that response in the body then we find the properties like Title etc are capitalized which creates issue sometime while processing it further.  We can make it camelcase just adding one more configuration as  options.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();  Now let’s run the application again and send the request again.  Now we are getting the response in camel case. |
| 12 |  | One more thing that we discussed in the GET request we need to pass all the data via URL only so here during Get request we are not allowed to put any data in Body as it is not valid |

|  |  |
| --- | --- |
| **Summary** | |
|  | In this video, we saw an example of GET request and that how the methods are mapped with HTTP GET and parameter are passed. |
| [Mandatory slide] – Next Video  Slide8.PNG | In the next video, We will move further and implement Post method. |