Why this tool?

Ever faced a situation like, have to call the WCF service with its functionality name and HTTP verb alone?

Ex:-

Functionality Name: PushNotification

HttpVerb: Get, Post, Delete, Put etc.

So, if there is a situation like, u have to use the functionality name and http verb alone to make a service call and should not use individual names to the WCF methods then?

Well **WebApi**, will address above mentioned problem for sure and it is built from scratch for the Rest based service.

But if there was a situation, to implement above solution in WCF then this may be a place to get started.

**Ex:-**

[**WebInvoke**(**Method = "\*"**, UriTemplate = "/NotificationPreference/", ResponseFormat = WebMessageFormat.Json)]

[OperationContract]

**[CustomOperationBehaviour]**

ResponseObj NotificationPreference(ReqObj value);

**Key things to remember:-**

1. Make the method as “\*” so it can accept all kind of Http verbs.
2. Make the universal input and output parameters instead of Http Method agnostic.

(Since WCF will serialize and de serialize on this type only!)

1. Annotate the method with WebInvoke instead of WebGet, so it can be invoked on any verb.
2. Only generic WCF method should be exposed to the internet, and make Http Method agnostic methods as public without adding the operation contract annotation.

So, once applied this idea it will affect entire WCF service?

Only the method which is annotated will only be affected and no other method will be affected. That’s the beauty of this idea.

**Sample WCF service will be like:-**

public class MultiHttpVerbServiceEx : IMultiHttpVerbServiceEx

{

**// Exposing method**

public ResponseObj NotificationPreference(ReqObj value)

{

return null;

}

**//Http Agnostic public method**

public ResponseObj GetNotificationPreference(ReqObj value)

{

return new ResponseObj() { name = "Rerouted GetNotificationPreference Call- " };

}

public ResponseObj PostNotificationPreference(ReqObj value)

{

return new ResponseObj() { name = "Rerouted PostNotificationPreference Call- " + value.Details };

}

}

How I implemented this idea?

I will track the Http verb of the incoming request, and append appropriate string to the functionality name to it to make a call internally.

Where I made this entire functionality to work?

I made use of the “**IOperationInvoker**” to do this entire method trap. There will be method which you have to implement with the signature, “**public object Invoke(object instance, object[] inputs, out object[] outputs)**”. Here I made the trick. There is a input parameter named “instance” and thanks to reflection which helped me to invoke a public method living inside the class.

How I will decide, which public method to call?

Inside this “Invoke” function, I will get the invoking Http verb, and I will append some strings according to incoming Http Method, and I will call the public method inside that service.

if (WebOperationContext.Current.IncomingRequest.Method == "GET")

{

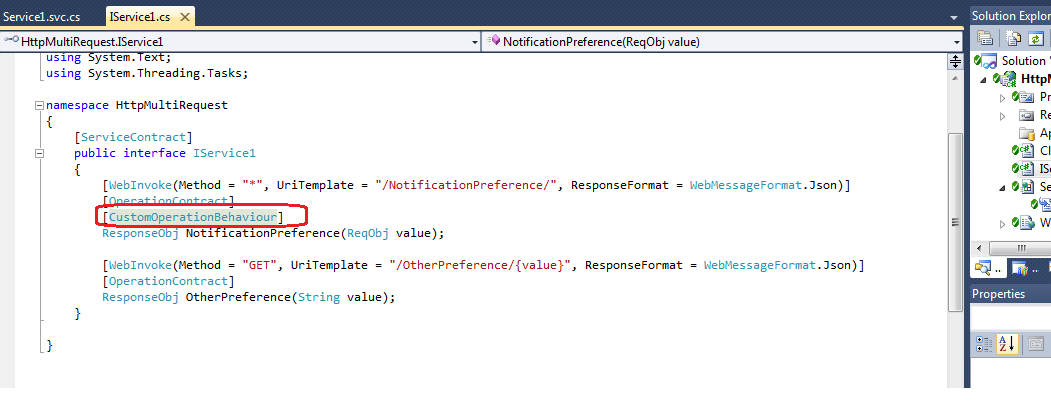
exeMethod = targetType.GetMethod("**Get**" + opsDesc.SyncMethod.Name, opsDesc.SyncMethod.GetParameters().Select(mi => mi.ParameterType).ToArray());

}

Want to see some working examples?

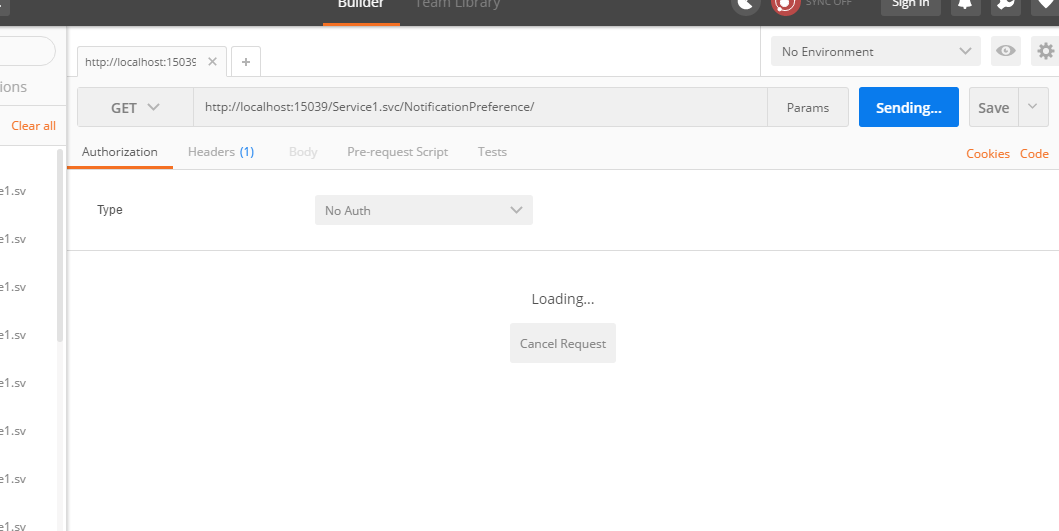
Ahhhh, Sure.

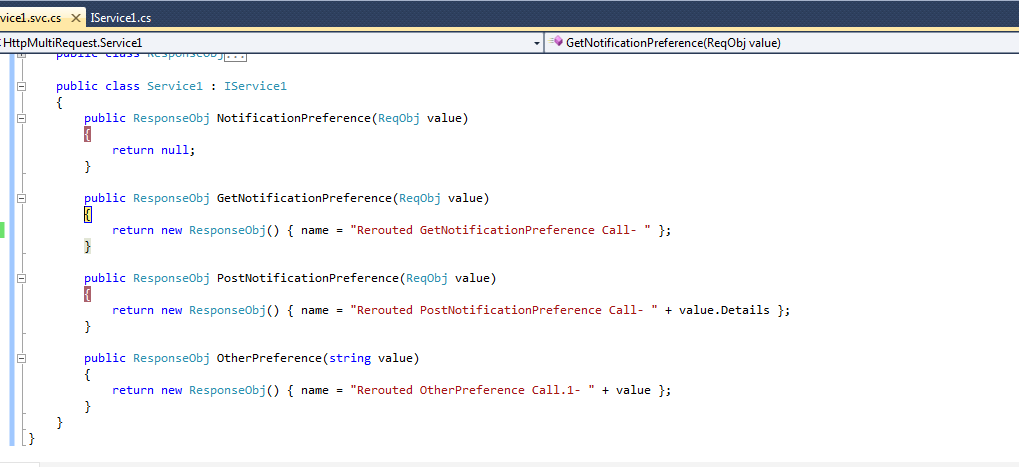
**Annotated Method:-**

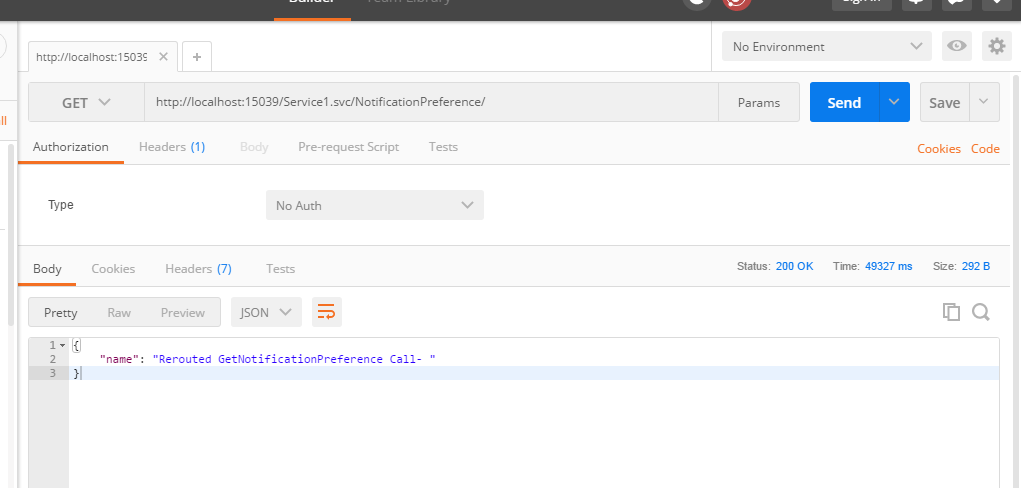


**Sample Service Calls For the Annotated Method:-**

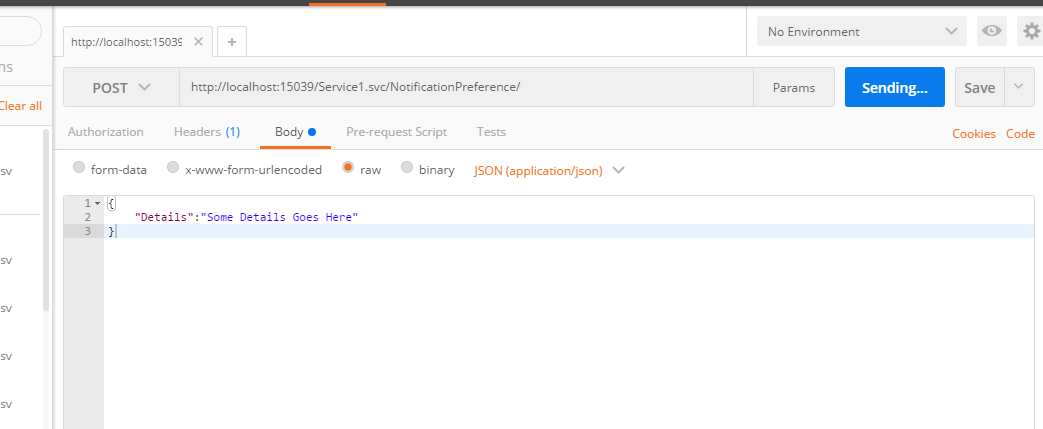
**Invoked By Get Http Verb:-**

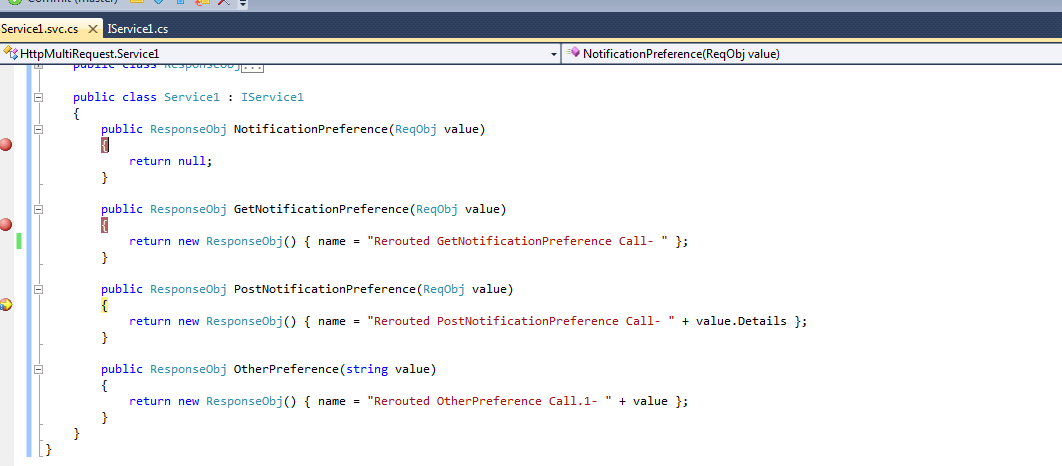


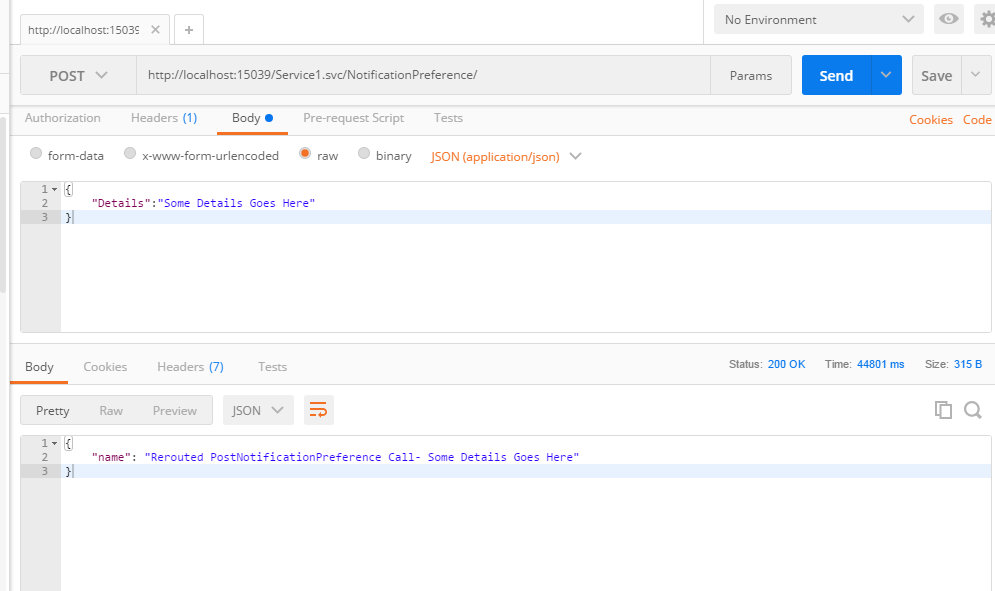




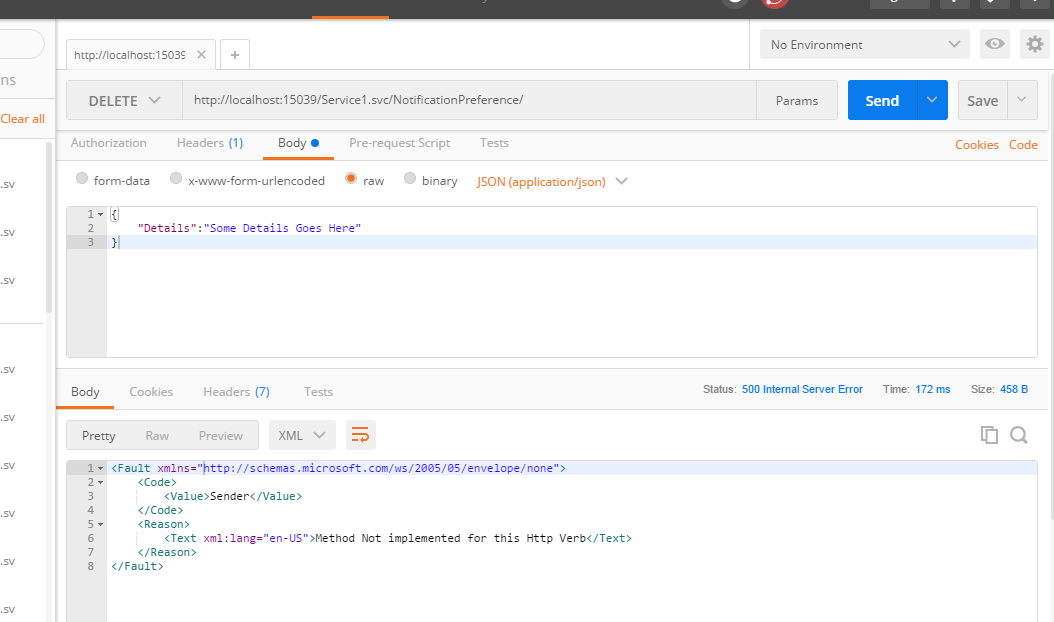
**Invoked by Post Http Verb:-**



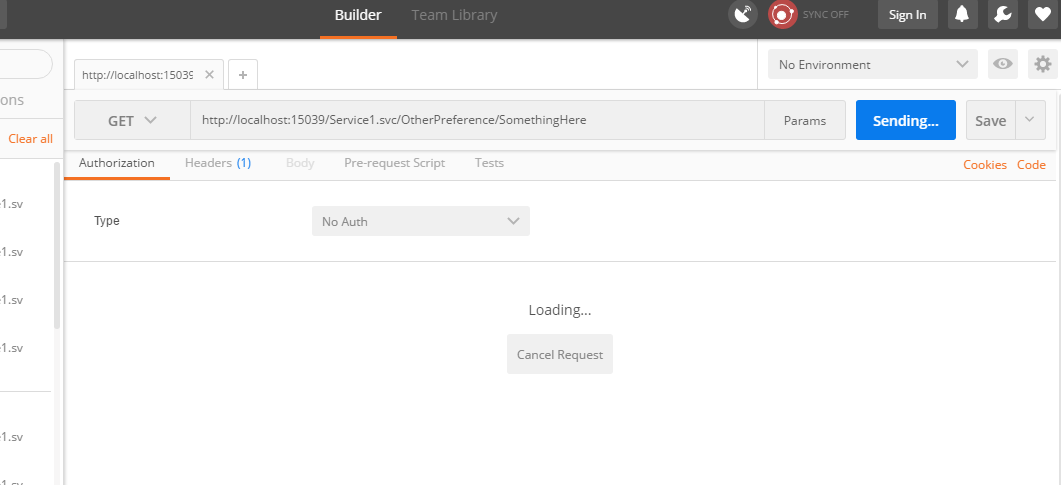




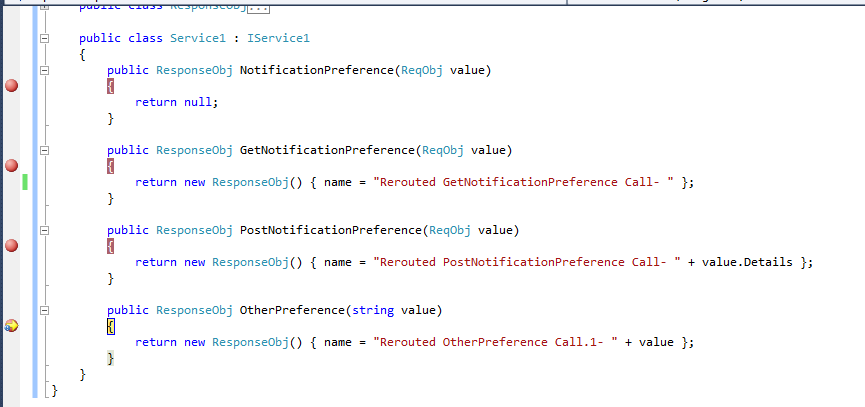
**Sample Custom Error For Not Implemented Http Verbs:-**

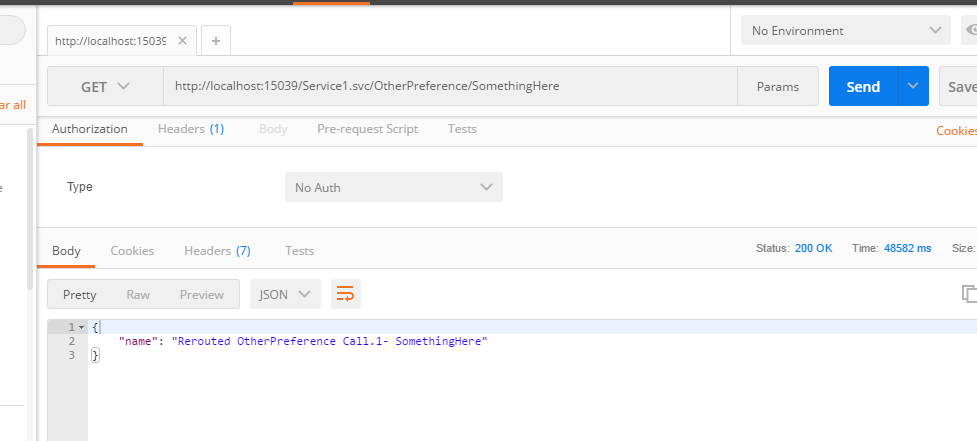


**For Non Annotated Methods (as usual flow goes here):-**



**BreakPoint Hit:-**





**Error Message for the non implemented methods:-**

