**TempData in ASP.NET MVC Application**

Now, we will see another approach to send the data from the controller action method to a view using the TempData.

The limitation of both ViewData and ViewBag is they are limited to one HTTP request only. So, if redirection occurs then their values become null means they will lose the data they hold. In many real-time scenarios, we may need to pass the data from one HTTP Request to the next subsequent HTTP Request. For example, we may need to pass the data from one controller to another controller or one action method to another action method within the same controller. Then in such situations like this, we need to use TempData.

##### **What is TempData in ASP.NET MVC?**

The TempData in ASP.NET MVC Framework is one of the mechanisms to pass a small amount of temporary data from a controller action method to a view as well as from a controller action method to another action method either within the same controller or to a different controller. TempData value will become null once the subsequent request is completed by default. But if you want then you can also change this default behavior. If you have a look at the definition Controller class, then you will find the following signature of the TempData property.

TempData in ASP.NET MVC

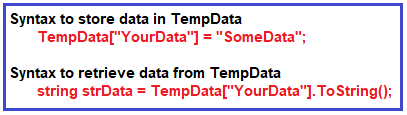
As you can see in the above image, the return type of the TempData is TempDataDictionary. Let us see the definition of the TempDataDictionary class.

TempDataDictionary in MVC

As you can see the TempDataDictionary class implements the IDictionary interface. So we can say that the TempData in ASP.NET MVC is a dictionary object. As it is a dictionary object, so it is going to store the data in the form of key-value pairs where each key must be a string and the value that we are passing to the dictionary is going to be stored in the form of an object type.

##### **How to Pass and Retrieve data From TempData in ASP.NET MVC:**

The most important point that you need to remember is, as it stores the data in the form of an object so while retrieving the data from TempData type casting is required. If you are accessing string value from the TempData, then it is not required to typecast. But it is mandatory to typecast explicitly to the actual type if you are accessing data other than the string type from the TempData.



##### **Let us understand TempData in ASP.NET MVC with one example.**

Modify the Employee Controller as shown below

**namespace** *FirstMVCDemo.Controllers*

**{**

**public** **class** EmployeeController : Controller

**{**

**public** ActionResult Method1**()**

**{**

TempData**[**"Name"**]** = "Pranaya";

TempData**[**"Age"**]** = 30;

**return** View**()**;

**}**

**public** ActionResult Method2**()**

**{**

string Name;

**int** Age;

**if** **(**TempData.ContainsKey**(**"Name"**))**

Name = TempData**[**"Name"**]**.ToString**()**;

**if** **(**TempData.ContainsKey**(**"Age"**))**

Age = **int**.Parse**(**TempData**[**"Age"**]**.ToString**())**;

// do something with user Name or Age here

**return** View**()**;

**}**

**public** ActionResult Method3**()**

**{**

string Name;

**int** Age;

**if** **(**TempData.ContainsKey**(**"Name"**))**

Name = TempData**[**"Name"**]**.ToString**()**;

**if** **(**TempData.ContainsKey**(**"Age"**))**

Age = **int**.Parse**(**TempData**[**"Age"**]**.ToString**())**;

// do something with userName or userAge here

**return** View**()**;

**}**

**}**

**}**

In the above example, we have added data into TempData and accessed the same data using a key inside another action method. Please notice that we have converted values into the appropriate type.

**Let’sunderstand TempData**  
1st Request: **http://localhost:xxxxx/Employee/Method1**  
2nd Request: **http://localhost:xxxxx/Employee/Method2**  
3rd Request: **http://localhost:xxxxx/Employee/Method3**

As you can see in the above example, we add **Name and Age** in TempData in the first request, and in the second subsequent request, we access the data from the TempData which we stored in the first request. However, we can’t get the same data in the third request because TempData will be cleared out after the second request.

##### **How to retain TempData values in the consecutive request?**

In order to retain the TempData value in the third consecutive request, we need to call **TempData.Keep()** method. Let’s see the use of **TempData.Keep()** method  with an example

**namespace** *FirstMVCDemo.Controllers*

**{**

**public** **class** EmployeeController : Controller

**{**

**public** ActionResult Method1**()**

**{**

TempData**[**"Name"**]** = "Pranaya";

TempData**[**"Age"**]** = 30;

**return** View**()**;

**}**

**public** ActionResult Method2**()**

**{**

string Name;

**int** Age;

**if** **(**TempData.ContainsKey**(**"Name"**))**

Name = TempData**[**"Name"**]**.ToString**()**;

**if** **(**TempData.ContainsKey**(**"Age"**))**

Age = **int**.Parse**(**TempData**[**"Age"**]**.ToString**())**;

TempData.Keep**()**;

// do something with userName or userAge here

**return** View**()**;

**}**

**public** ActionResult Method3**()**

**{**

string Name;

**int** Age;

**if** **(**TempData.ContainsKey**(**"Name"**))**

Name = TempData**[**"Name"**]**.ToString**()**;

**if** **(**TempData.ContainsKey**(**"Age"**))**

Age = **int**.Parse**(**TempData**[**"Age"**]**.ToString**())**;

// do something with userName or userAge here

**return** View**()**;

**}**

**}**

**}**

There are two overloaded versions of the Keep Method. They are as follows:

1. **public void Keep()**: It will mark all keys in the temp data dictionary for retention.
2. **public void Keep(string key)**: It will mark the specified key in the temp data dictionary for retention. Here, the key parameter is the key to retain in the dictionary