**ViewBag and ViewData in ASP.NET Core**

[**Layouts**](https://www.tektutorialshub.com/layouts-and-sections-in-asp-net-mvc-core/)

[**Model & ViewModel**](https://www.tektutorialshub.com/model-and-viewmodel-in-asp-net-core/)

The Views needs to get the data from the controller. One of the ways in which to pass the data to the view is using the ViewData or ViewBag object. In this tutorial, we look at how to use ViewBag and ViewData to pass the data to the View.

**What is Viewdata**

The ViewData is a property of the Controller Base class, which returns a **[ViewDataDictionary](https://docs.microsoft.com/en-us/aspnet/core/api/microsoft.aspnetcore.mvc.viewfeatures.viewdatadictionary)** object.

The ViewDataDictionary as the name suggests is a dictionary object which allows us to store key-value pairs. The key must be a case-insensitive string. To pass data to the view, you can assign values to the dictionary using the Key. You can store any number of items as needed in the ViewData.

The ViewData is passed to the view from the Controller. When you invoke the Viewmethod in the Controller Action method, the ViewData is automatically passed to the View.

In the View, you can access the value stored in the Viewdata using the key.

The data stored in the ViewData object exists only during the current request. As soon as the view is generated and sent to the client, the Viewdata object is cleared.

**How to Use ViewData**

The following example shows how to use the ViewData in a Controller Action

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public IActionResult SomeAction()  {      ViewData["Greeting"] = "Hello World!";      return View();  } |

In the above example, we have added the “Hello World” string to the ViewData using the key “Greeting”.

You can simply call the following method in the View to retrieve the value stored in the “Greeting” Key.

**[Model Binding : Passing Data from View to Controller](https://www.tektutorialshub.com/model-binding-passing-data-from-view-to-controller/" \t "_self)**

C#



|  |  |
| --- | --- |
| 1  2  3 | @ViewData["Greeting"] |

**Passing the Object using View Data**

In the above example, we stored the string data in the ViewData. The string data can be used directly without the need for a cast.

You can store any type of data such as integer, Boolean or object in Viewdata.

To use any other object, you must cast the ViewData values to specific types when you extract them. The following example illustrates how to use Viewdata to pass an object from the Controller to View.

C#



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public IActionResult Index()  {      ViewData["Greeting"] = "Hello World";      ViewData["Product"] = new ProductModel()      {          ProductID=1,          Name = "Samsung galaxy Note",          Brand = "Samsung",          Price = 19000      };        return View();  } |

In the View , we extract the product from the Viewdata and cast it to the ProductModel class before using it.

C#



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | @{      // Since Product isn't a string, it requires a cast.      var product = ViewData["Product"] as ProductModel;  }    @ViewData["Greeting"]!    @product.ProductID<br>  @product.Name<br>  @product.Brand<br>  @product.Price<br> |

You can use ViewData to pass data from Controller to View and within Views, including Partial Views and Layouts.

**[Application Startup in ASP.NET Core](https://www.tektutorialshub.com/application-startup-in-asp-net-core/" \t "_self)**

**What is Viewbag**

As we see from the previous example you can store anything in the ViewDataDictionay, but, to access anything from the viewdata values, we need to perform type casts.

The ViewBag uses the dynamic feature that was added in C# 4.0. It is a wrapper around the ViewData and provides the dynamic properties for the underlying ViewDatacollection.

The ViewBag can be more convenient to work with since it doesn’t require casting.

Using the ViewBag the above code becomes

C#



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public IActionResult SomeAction()  {      ViewBag.Greeting = "Hello";      ViewBag.Product = new ProductModel()      {          ProductID = 1,          Name = "Samsung galaxy Note",          Brand = "Samsung",          Price = 19000      };        return View();  } |

And you can refer it in the View as shown below

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | @ViewBag.Greeting World <br>    @ViewBag.product.ProductID<br>  @ViewBag.product.Name<br>  @ViewBag.product.Brand<br>  @ViewBag.product.Price<br> |

**ViewData Vs ViewBag**

Both ViewBag and ViewData uses the same ViewDataDictionary underneath. Hence, you can use both ViewData and ViewBag and mix and match between them when reading from or write to them.

For Example

C#



|  |  |
| --- | --- |
| 1  2  3 | ViewData["Greeting"] = "Hello World"; |

Can be retrieved using the ViewBag syntax

**[Passing data from Controller to View in ASP.NET Core MVC](https://www.tektutorialshub.com/passing-data-from-controller-to-view-in-asp-net-core-mvc/" \t "_self)**

C#



|  |  |
| --- | --- |
| 1  2  3 | @ViewBag.Greeting |

**The difference between ViewData and ViewBag**

The ViewData uses the dictionary syntax to access the values, while ViewBag uses the property syntax.

The ViewData derives from ViewDataDictionary, so it has dictionary properties that can be useful, such as ContainsKey, Add, Remove, and Clear.

Viewbag derives from DynamicViewData, so it allows the creation of dynamic properties using dot notation (@ViewBag.SomeKey = <value or object>), and no casting is required. The syntax of ViewBag makes it quicker to add to controllers and views.

The ViewData allows use whitespaces in Keys since they are strings. For Example ViewData[“Some Key With Whitespace”]. This is not possible with ViewBag.

The ViewData requires typecasting for data type other than string values. It also needs to check for null values to avoid error. It is simpler to check for Null Values using ViewBag. Example @ViewBag.Person?.Name

**When to use ViewData or ViewBag**

The ViewData and ViewBag are equally good options when you want to pass the small amount of data from the controller to the View. The choice usually depends on the personal preference

The Drawback of ViewData/ViewBag is that they are resolved dynamically at the runtime. They do not offer any compile-time type checking, Hence more error prone

The ViewData and ViewBag can pass data from the Controller to View. It cannot be used to pass data from One Controller to another controller.

**Summary**

In this tutorial, we looked at how to use ViewBag & ViewData to pass data from the Controller to View.