In ASP.NET MVC, ViewBag, ViewData, and TempData are used to pass data from a controller to a view. Each of these serves different purposes and has unique behaviors. Below are detailed explanations, including examples:

1. ViewBag:

Type: dynamic type object.

Purpose: Used to pass temporary data from the controller to the view. It is not meant to hold data for the entire life cycle of a request, and it is typically used for small amounts of data or dynamic properties that don't require typecasting or null checks.

Scope: Exists only for the current request.

Typecasting/Null checks: No typecasting required, but it may cause runtime errors if data is accessed incorrectly.

Example of using ViewBag:

```
public ActionResult Index()
{
    Record rec = new Record
    {
        Id = 101,
        RecordName = "Bouchers",
        RecordDetail = "The basic stocks"
    };
    // Passing data to the view using ViewBag
    ViewBag.Message = rec;
    return View();
}
```

In the corresponding view:

```
<p>@ViewBag.Message.RecordName</p>      <!-- Displays: Bouchers -->
```

@ViewBag.Message.RecordDetail <!-- Displays: The basic stocks -->

2. ViewData:

Type: Dictionary<string, object>.

Purpose: ViewData is used to store key-value pairs (where the key is always a string). It is more structured than ViewBag and allows access to data through string keys.

Scope: Exists only for the current request.

Typecasting/Null checks: Requires typecasting and null checks. If the data is not cast to the correct type or if you access an empty key, it could lead to runtime errors.

Example of using ViewData:

```
public ActionResult Index()
{
    Record rec = new Record
    {
        Id = 101,
        RecordName = "Bouchers",
        RecordDetail = "The basic stocks"
    };
    // Passing the Record object to the view using ViewData
    ViewData["Message"] = rec;

    // Returning the view
    return View();
```

In the corresponding view:

```
<p>@((Record)ViewData["Message"]).RecordName</p>  <!-- Displays: Bouchers -->
```

@((Record)ViewData["Message"]).RecordDetail <!-- Displays: The basic stocks -->

@((Record)ViewData["Message"]).RecordName

ViewData["Message"] retrieves the value stored in the ViewData dictionary under the key "Message". Since ViewData stores data as an object, you need to explicitly cast it back to the Record type using (Record).

After casting, you access the **RecordName** property of the Record object (rec), which holds the value "Bouchers".

The result of this expression is "Bouchers", which gets rendered inside the tag.

3. TempData:

Type: TempDataDictionary.

Purpose: TempData is used to store data that needs to persist across requests, such as after a redirect. It is often used for scenarios like displaying a message after a redirect (e.g., a confirmation message or error message).

Scope: Data stays for one more HTTP request after the current one, making it useful for scenarios where you want to pass data during a redirect.

Typecasting/Null checks: Like ViewData, TempData requires typecasting and null checks to avoid errors.

Automatically Cleared: After the data is read, TempData automatically clears the data. However, you can preserve data for another request by calling TempData.Keep().

Example of using TempData:

```
public ActionResult TemporaryEmployee()
{
    Employee employee = new Employee
```

```
    {
        EmpID = "121",
        EmpFirstName = "John",
        EmpLastName = "Nguyen"
    };

    // Storing the employee object in TempData
    TempData["Employee"] = employee;

    // Redirecting to another action
    return RedirectToAction("PermanentEmployee");
}
```

In the PermanentEmployee action, you can access the data stored in TempData:

```
public ActionResult PermanentEmployee()
{
```

```
    Employee employee = TempData["Employee"] as Employee;
    if (employee != null)
    {
        // Do something with the employee data
    }
    return View();
}
```

In this case, the data in TempData["Employee"] will be available to the PermanentEmployee action, but only for that request. If you want to keep it available for another request, you can call TempData.Keep().

When to Use Each:

Use ViewBag when you need to pass small, dynamic data to a view without worrying about typecasting.

Use ViewData when you need to pass structured data (key-value pairs) that requires typecasting.

Use TempData when you need to persist data across a redirect or between controller actions (for instance, to show confirmation messages or error messages after a redirect).