Difference Between Response.Redirect() and Server.Transfer() Methods in ASP.NET

Introduction

Both `Response.Redirect` and `Server.Transfer` methods are used to transfer a user from one web page to another web page. Both methods are used for the same purpose, but still there are some differences as follows.

The `Response.Redirect` method redirects a request to a new URL and specifies the new URL while the `Server.Transfer` method for the current request, terminates execution of the current page and starts execution of a new page using the specified URL path of the page.

Both `Response.Redirect` and `Server.Transfer` have the same syntax like:

Collapse | Copy Code

```
Response.Redirect("UserDetail.aspx");
Server.Transfer("UserDetail.aspx");
```

HTTP Status Codes

Before touching on more points, I want to explain some HTTP status codes, these are important for the understanding of the basic differences between these two. The HTTP status codes are the codes that the Web server uses to communicate with the Web browser or user agent.

1. **HTTP 200 OK**

   This is the most common HTTP status message. It indicates that the request was successful and the server was able to deliver on the request. The information returned with the response is dependent on the method used in the request. In a `GET` request, the response will contain an entity corresponding to the requested resource. In a `POST` request, the response will contain an entity describing or containing the result of the action.

2. **HTTP 302 Found**

   The HTTP response status code 302 Found is a common way of performing a redirection. The 302 status code indicates that the resource you are requesting has redirected to another resource.

Response.Redirect and Server.Transfer Request Handling

`Response.Redirect` sends an HTTP request to the browser, then the browser sends that request to the web server, then the web server delivers a response to the web browser. For example, suppose you are on the web page "*UserRegister.aspx*" page and it has a button that redirects you to the "*UserDetail.aspx*" web page.

11	200	HTTP	localhost:2531	/UserRegister.aspx	601	private	text/html; c...	iexplore:6516
13	302	HTTP	localhost:2531	/UserRegister.aspx	828	private	text/html; c...	iexplore:6516
14	200	HTTP	localhost:2531	/UserDetail.aspx	431	private	text/html; c...	iexplore:6516

Figure 1.1 Request and Response using Response.Redirect method

You can notice in Figure 1.1 that when you run the application, then you get a web page successfully so the HTTP status code is "HTTP 200 OK". Then you click on the button that redirects to another page using the `Response.Redirect` method. The `Response.Redirect` method first sends a request to the web browser so it is the "HTTP 302 Found" status, then the browser sends a request to the server and the server delivers a response so it is "HTTP 200 OK". It is called a round trip. Let's see that in Figure 1.2.

Figure 1.2 Round Trip by Response.Redirect method.

`Server.Transfer` sends a request directly to the web server and the web server delivers the response to the browser.

4	200	HTTP	localhost:2531	/UserRegister.aspx	601	private	text/html; c...	iexplore:5176
6	200	HTTP	localhost:2531	/UserRegister.aspx	431	private	text/html; c...	iexplore:5176

Figure 1.3 Request and Response using Server.Transfer method

Now Figure 1.3 explains that the first request goes to the web server and then gets a response so the round-trip is not made by the `Server.Transfer` method. You get the same page response but different content. That means your web page URL in the address bar will not be changed. For example, see Figure 1.3 showing you get the same page as the previous request page so the previous page still remains while in `Response.Redirect`, you get a different page in the response (see Figure 1.1). That means the address bar URL will be changed in the `Response.Redirect` method and also updates the browser history so you can move back from the browser back button.

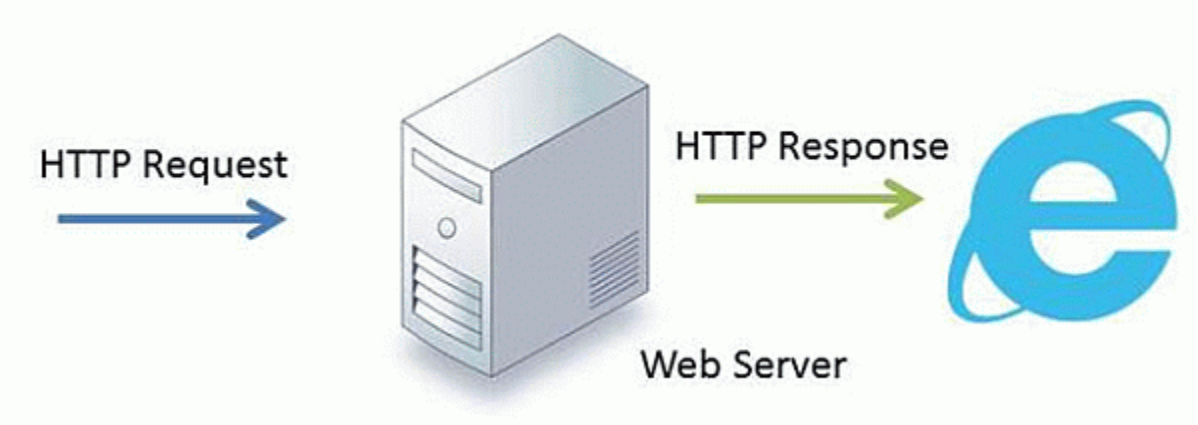

Figure 1.4 Server.Transfer method request and response

`Response.Redirect` can be used for both *.aspx* and HTML pages whereas `Server.Transfer` can be used only for *.aspx* pages and is specific to ASP and ASP.NET.

`Response.Redirect` can be used to redirect a user to an external website. `Server.Transfer` can be used only on sites running on the same server. You cannot use `Server.Transfer` to redirect the user to a page running on a different server.

Using the Code

When you use `Server.Transfer`, then the previous page also exists in server memory while in the `Response.Redirect` method, the previous page is removed from server memory and loads a new page in memory. Let's see an example.

We add some items in the context of the first page "*UserRegister.aspx*" and thereafter the user transfers to another page "*UserDetail.aspx*" using `Server.Transfer` on a button click. The user will then be able to request data that was in the context because the previous page also exists in memory. Let's see the code.

Code of the *UserRegister.aspx.cs* page:

⊟Collapse | Copy Code

```
using System;
public partial class UserRegister : System.Web.UI.Page
{
    protected void btn_Detail_Click(object sender, EventArgs e)
    {
        Context.Items.Add("Name", "Sandeep Singh Shekhawat");
        Context.Items.Add("Email", "sandeep.shekhawat88@gmail.com");
       Server.Transfer("UserDetail.aspx");
    }
}
```

Code of the *UserDetail.aspx.cs* page:

⊟Collapse | Copy Code

```
 using System;
public partial class UserDetail : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        try
        {
            Response.Write(string.Format("My name is {0} and email address is
{1}",
                                          Context.Items["Name"].ToString(),

Context.Items["Email"].ToString()));
        }
        catch (NullReferenceException ex)
        {
            Response.Write(ex.Message);
        }
    }
}
```

Using the code above, we can get data from the previous page to the new page by the `Context.Item` collection. But if you use the `Response.Redirect` method, then you can't get context data on another page and get an exception object as `null` because the previous page doesn't exist in server memory.

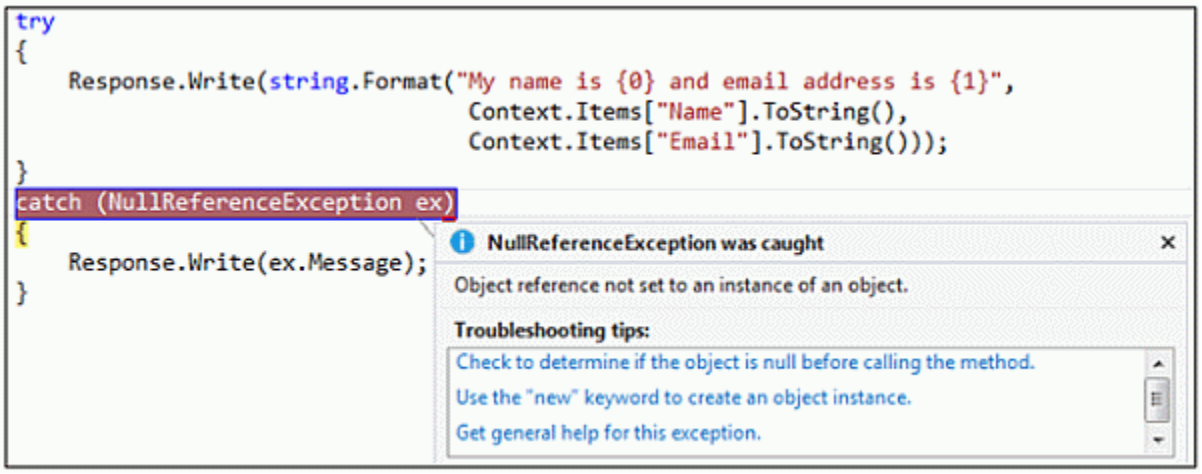

Figure 1.5 Null Exception by Context object

Mostly the `Server.Transfer` method is preferable to use because `Server.Transfer` is faster since there is one less roundtrip, but some people say that `Server.Transfer` is not recommended since the operations typically flow through several different pages causing a loss of the correct URL of the page, but again it all depends on your requirement.

By Sandeep Singh Shekhawat