MVC Form Process

Grand Circus

Step by Step Process

The only way we can call methods in our controllers is via URLs.

URLs look like regular web addresses, but have a “GET” or “POST” added on to them, along with a hidden “body” area for data (which might be empty).

GET and POST calls are identical except that word “GET” or “POST” added on.

Step by Step Process

When the user enter a URL into the address bar in the browser, the browser sends that URL (and “GET”) to our web server.

Our web server figures out which controller and method to call based on the “path” of the URL.

That method runs and returns *something* back to the browser. That something is usually a bunch of HTML.

Step by Step Process

In order to create that HTML, we (the programmer) provide the web server with a “view.”

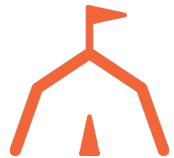
The view consists of HTML and some C# code that gets called. The C# code is embedded in the view, and gets replaced with the results of that C# code.

Step by Step Process

We can pass data from our action to the view through several means:

- We can put it in ViewData
- We can put it in ViewBag
- We can put it in the model

With the model we specify a type. With ViewData, we might need to cast to our type.

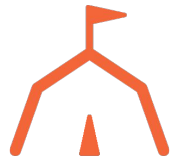


Step by Step Process: Forms

For a form to work, we need **two** URLs (and two actions/functions).

One URL (function) returns the form itself.

The other URL is the URL called when the user clicks the Submit button in the form. We code this in the action attribute of the form element.



Step by Step Process: Forms

We usually (99.999% of the time) include method="post" in our form element.

That means when the user clicks submit, the URL will get called with a POST word, and the data from the form will go through the hidden “body” part, not in the URL address itself.

Step by Step Process: Forms

When the data from the form gets sent to our application, the data comes in as a parameter in our action.

That data can either be several parameters (usually strings, but you can do numbers), or a single object.

ASP.NET takes care of creating that object for us with the specified type. It fills in the fields with the data from the form.

Step by Step Process: Forms

We can then do validation on that data, and if the validation fails send back a different view to the user (usually the original form with a message added).

Or if the data is fine, we send it on to the view via the Model.

The view can then use that data in any way it wants to, usually some kind of thank you message.