View-to-Controller

There are four ways to pass the data from View to Controller which are explained below:

1. Traditional Approach: In this approach, we can use the request object of the `HttpRequestBase` class. This object contains the input field name and values as name-value pairs in case of the form submit. So we can easily get the values of the controls by their names using as indexer from the request object in the controller.

   For example: Let's say you are having an input in the form with name '`txtName`', then its values can be retrieved in controller from request object like below:

```
string strName = Request["txtName"].ToString();
```

2. Through `FormCollection`: We can also get post requested data by the `FormCollection` object. This object also has requested data as the name/value collection as the `Request` object.

   For example:

```
[HttpPost]
public ActionResult Calculate(FormCollection form)
{
        string strName = form["txtName"].ToString();
        . . . . . .
}
```

3. Through Parameters: We can also pass the input field names as parameters to the post action method by keeping the names same as the input field names. These parameters will have the values for those fields and the parameter types should be `string`. Also, there is no need to define the parameters in any specific sequence.

   For example:

```
 [HttpPost]
public ActionResult Calculate(string txtName)
{
    string strName = Convert.ToString(txtName);
    . . . . . .
}
```

In all of the above approaches, we need to even convert the non-`string` type to `string` type due to which if any parsing fails, then the entire action may fail here. Here we have to convert each value

to avoid any exceptions but, in the below 4th approach of passing data from view to controller, it reduces the amount of code.

4. Strongly typed model binding to view: Here, we need to create a strongly typed view which will bind directly the model data to the various fields of the page.

   For example:

i. Create a model with the required member variables.

   Let's say we have a model named 'Person' with member variable named as 'Name'

ii. Now pass the empty model to the view as parameter in the controller action.

   For example:

```
public ActionResult GetName()
{
    Person person = new Person();
    return View(person);
}
```

iii. Prepare the strongly typed view to display the model property values through html elements as below:

   For example:

```
<div><%= Html.Encode(person.Name) %></div>
```

iv. Create the action method that handles the POST request & processes the data.

   For example:

```
 [HttpPost]
public ActionResult GetPersonName(Person person)
{
    return Content(person.Name.ToString());
}
```