In this video we will discuss mapping asp.net request data to controller action simple parameter types. [Please watch Part 13](http://csharp-video-tutorials.blogspot.com/2013/05/part-13-formcollection-in-mvc.html), before proceeding with this video.

To save form data, to a database table, we used the following code in [Part 13](http://csharp-video-tutorials.blogspot.com/2013/05/part-13-formcollection-in-mvc.html). From [Part 13](http://csharp-video-tutorials.blogspot.com/2013/05/part-13-formcollection-in-mvc.html), it should be clear that, **FormCollection**class will automatically receive the posted form values in the controller action method.  
[HttpPost]  
public ActionResult Create(FormCollection formCollection)  
{  
    Employee employee = new Employee();  
    // Retrieve form data using form collection  
    employee.Name = formCollection["Name"];  
    employee.Gender = formCollection["Gender"];  
    employee.City = formCollection["City"];  
    employee.DateOfBirth =   
        Convert.ToDateTime(formCollection["DateOfBirth"]);  
  
    EmployeeBusinessLayer employeeBusinessLayer =   
        new EmployeeBusinessLayer();  
  
    employeeBusinessLayer.AddEmmployee(employee);  
    return RedirectToAction("Index");  
}  
  
The above controller **"Create"** action method can be re-written using simple types as shown below. Notice that, the create action method has got parameter names **that match** with the names of the form controls. To see the names of the form controls, right click on the browser and view page source. Model binder in mvc maps the values of these control, to the respective parameters.  
[HttpPost]  
public ActionResult Create(string name, string gender, string city, DateTime dateOfBirth)  
{  
    Employee employee = new Employee();  
    employee.Name = name;  
    employee.Gender = gender;  
    employee.City = city;  
    employee.DateOfBirth = dateOfBirth;  
  
    EmployeeBusinessLayer employeeBusinessLayer =   
        new EmployeeBusinessLayer();  
  
    employeeBusinessLayer.AddEmmployee(employee);  
    return RedirectToAction("Index");  
}  
  
But do we really to do these mappings manually. The answer is no. In a later video session we will see how to automatically map the request data, without having to do it manually.  
  
Please note that, **the order of the parameters does not matter**. What matters is the name of the parameter. If the parameter name is different from the form control name, then the form data will not be mapped as expected.