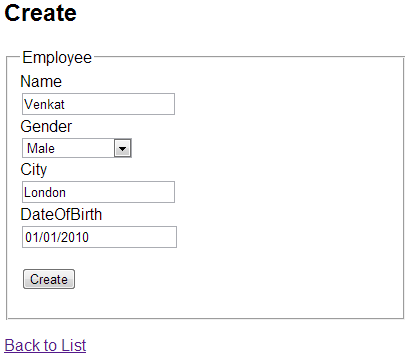
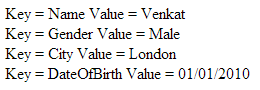
I**n this video we will discuss using FormCollection object in mvc and it's purpose.**Please [watch Part 12](http://csharp-video-tutorials.blogspot.com/2013/05/part-12-creating-view-to-insert-data.html), before proceeding with this video.  
  
**FormCollection**class will automatically receive the posted form values in the controller action method, in **key/value** pairs. **Keys** & **values** can be accessed using key names or index.

**We implemented the following "Create" view in**[**Part 12**](http://csharp-video-tutorials.blogspot.com/2013/05/part-12-creating-view-to-insert-data.html)  
   
  
We can use the **FormCollection**to loop thru each key and it's value that is posted to the server.  
[HttpPost]  
public ActionResult Create(FormCollection formCollection)  
{  
    if (ModelState.IsValid)  
    {  
        foreach (string key in formCollection.AllKeys)  
        {  
            Response.Write("Key = " + key + "  ");  
            Response.Write("Value = " + formCollection[key]);  
            Response.Write("<br/>");  
        }  
    }  
    return View();  
}  
  
**The output is as shown below**  
   
  
**Create the following stored procedure** to insert employee data into tblEmployee table  
Create procedure spAddEmployee    
@Name nvarchar(50),    
@Gender nvarchar (10),    
@City nvarchar (50),    
@DateOfBirth DateTime    
as    
Begin    
 Insert into tblEmployee (Name, Gender, City, DateOfBirth)    
 Values (@Name, @Gender, @City, @DateOfBirth)    
End  
  
**Add the following method to EmployeeBusinessLayer.cs file**.  
public void AddEmmployee(Employee employee)  
{  
    string connectionString =  
            ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;  
  
    using (SqlConnection con = new SqlConnection(connectionString))  
    {  
        SqlCommand cmd = new SqlCommand("spAddEmployee", con);  
        cmd.CommandType = CommandType.StoredProcedure;  
  
        SqlParameter paramName = new SqlParameter();  
        paramName.ParameterName = "@Name";  
        paramName.Value = employee.Name;  
        cmd.Parameters.Add(paramName);  
  
        SqlParameter paramGender = new SqlParameter();  
        paramGender.ParameterName = "@Gender";  
        paramGender.Value = employee.Gender;  
        cmd.Parameters.Add(paramGender);  
  
        SqlParameter paramCity = new SqlParameter();  
        paramCity.ParameterName = "@City";  
        paramCity.Value = employee.City;  
        cmd.Parameters.Add(paramCity);  
  
        SqlParameter paramDateOfBirth = new SqlParameter();  
        paramDateOfBirth.ParameterName = "@DateOfBirth";  
        paramDateOfBirth.Value = employee.DateOfBirth;  
        cmd.Parameters.Add(paramDateOfBirth);  
  
        con.Open();  
        cmd.ExecuteNonQuery();  
    }  
}  
  
**To save form data**, to a database table, copy and paste the following code in **EmployeeController.cs** file.  
[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");  
}  
  
Do we really have to write all the dirty code of retrieving data from **FormCollection**and assign it to the properties of **"employee"** object. The answer is no. This is the job of the **modelbinder** in MVC. We will discuss modelbinders in our next video.