In this video we will discuss the differences between **updatemodel**and **tryupdatemodel** functions. [Please watch Part 15](http://csharp-video-tutorials.blogspot.com/2013/05/part-15-updatemodel-function-in-mvc.html), before proceeding.

Make changes to **"Create\_Post()"** controller action method as shown below.  
[HttpPost]  
[ActionName("Create")]  
public ActionResult Create\_Post()  
{  
    EmployeeBusinessLayer employeeBusinessLayer =  
        new EmployeeBusinessLayer();  
  
    Employee employee = new Employee();  
    UpdateModel(employee);  
    if (ModelState.IsValid)  
    {  
        employeeBusinessLayer.AddEmmployee(employee);  
        return RedirectToAction("Index");  
    }  
    else  
    {  
        return View();  
    }  
}  
  
Please note that, **"AddEmmployee()"** method is now inside the **"IF"** condition that checks the validity of ModelState using **ModelState.IsValid** boolean property. Run the application and navigate to the following URL.  
**http://localhost/MVCDemo/Employee/Create**  
  
Submit the page without entering any data. You will get an error stating - "The model of type 'BusinessLayer.Employee' could not be updated". This is because **"DateOfBirth"**property of **"Employee"** class is a **non-nullable** DateTime data type. DateTime is a value type, and needs to have value when we post the form. To make **"DateOfBirth"**optional change the data type to nullable DateTime as shown below.  
public class Employee  
{  
    public int ID { get; set; }  
    public string Name { get; set; }  
    public string Gender { get; set; }  
    public string City { get; set; }  
    public DateTime? DateOfBirth { get; set; }  
}  
  
Run the application and navigate to the following URL.  
**http://localhost/MVCDemo/Employee/Create**  
  
Submit the page without entering any data. You will now get a different error stating - Procedure or function 'spAddEmployee' expects parameter '@Name', which was not supplied.  
  
This is because, the following parameters of stored procedure **"spAddEmployee"** are all required.  
@Name  
@Gender  
@City  
@DateOfBirth  
  
To make all these parameters optional, modify the stored procedure as shown below.  
Alter procedure spAddEmployee      
@Name nvarchar(50) = null,      
@Gender nvarchar(10) = null,       
@City nvarchar (50) = null,       
@DateOfBirth DateTime  = null    
as      
Begin      
 Insert into tblEmployee (Name, Gender, City, DateOfBirth)      
 Values (@Name, @Gender, @City, @DateOfBirth)      
End  
  
**Run the application and navigate to the following URL.**  
http://localhost/MVCDemo/Employee/Create  
  
Submit the page without entering any data. You will now get a different error stating - Object cannot be cast from DBNull to other types.  
  
To fix this error, make changes to **"Employees"** property in **"EmployeeBusinessLayer.cs"** file as shown below. Notice that we are populating **"DateOfBirth"** property of **"Employee"** object only if **"DateOfBirth"** column value is not **"DBNull"**.  
public IEnumerable<Employee> Employees  
{  
    get  
    {  
        string connectionString =  
            ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;  
  
        List<Employee> employees = new List<Employee>();  
  
        using (SqlConnection con = new SqlConnection(connectionString))  
        {  
            SqlCommand cmd = new SqlCommand("spGetAllEmployees", con);  
            cmd.CommandType = CommandType.StoredProcedure;  
            con.Open();  
            SqlDataReader rdr = cmd.ExecuteReader();  
            while (rdr.Read())  
            {  
                Employee employee = new Employee();  
                employee.ID = Convert.ToInt32(rdr["Id"]);  
                employee.Name = rdr["Name"].ToString();  
                employee.Gender = rdr["Gender"].ToString();  
                employee.City = rdr["City"].ToString();  
                if (!(rdr["DateOfBirth"] is DBNull))  
                {  
                    employee.DateOfBirth = Convert.ToDateTime(rdr["DateOfBirth"]);  
                }  
  
                employees.Add(employee);  
            }  
        }  
  
        return employees;  
    }  
}  
  
**Run the application and navigate to the following URL**  
http://localhost/MVCDemo/Employee/Create  
  
Submit the page without entering any data. Notice that a blank employee row is inserted into tblEmployee table.  
  
Now let's make the following properties of "Employee" class required.  
Name  
City  
DateOfBirth  
  
To achieve this we can use **"Required"**attribute that is present in **System.ComponentModel.DataAnnotations** namespace. To use this namespace, BusinessLayer project need a reference to **"EntityFramework"** assembly. The changes to the "Employee" class are shown below.  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.ComponentModel.DataAnnotations;  
  
namespace BusinessLayer  
{  
    public class Employee  
    {  
        public int ID { get; set; }  
        [Required]  
        public string Name { get; set; }  
        public string Gender { get; set; }  
        [Required]  
        public string City { get; set; }  
        [Required]  
        public DateTime? DateOfBirth { get; set; }  
    }  
}  
  
**Run the application and navigate to the following URL.**  
http://localhost/MVCDemo/Employee/Create  
  
Submit the page without entering any data. We now get an error stating - The model of type 'BusinessLayer.Employee' could not be updated. Notice that this error is thrown when UpdateModel() function is invoked.  
  
Now let's use **TryUpdateModel**() instead of **UpdateModel**(). Make changes to **"Create\_Post()"** controller action method in **"EmployeeController"** as shown below.  
[HttpPost]  
[ActionName("Create")]  
public ActionResult Create\_Post()  
{  
    EmployeeBusinessLayer employeeBusinessLayer =  
        new EmployeeBusinessLayer();  
  
    Employee employee = new Employee();  
    TryUpdateModel(employee);  
    if (ModelState.IsValid)  
    {  
        employeeBusinessLayer.AddEmmployee(employee);  
        return RedirectToAction("Index");  
    }  
    else  
    {  
        return View();  
    }  
}  
  
**Run the application and navigate to the following URL**  
http://localhost/MVCDemo/Employee/Create  
  
Submit the page without entering any data. Notice that, we don't get an exception now and the user remains on **"Create"** view and the validation errors are displayed to the user.  
  
So, the difference is **UpdateModel**() throws an exception if validation fails, where as **TryUpdateModel**() will never throw an exception. The similarity is, both the functions are used to update the Model with the Form values and perform the validations.

Is it mandatory to use "UpdateModel()" or "Try"UpdateModel()" function to update the Model?  
**The answer is NO.**  
  
The above method can be re-written as shown below and we get the same behaviour.  
[HttpPost]  
[ActionName("Create")]  
public ActionResult Create\_Post(Employee employee)  
{  
    EmployeeBusinessLayer employeeBusinessLayer =  
        new EmployeeBusinessLayer();  
  
    if (ModelState.IsValid)  
    {  
        employeeBusinessLayer.AddEmmployee(employee);  
        return RedirectToAction("Index");  
    }  
    else  
    {  
        return View();  
    }  
}  
  
**So the next question is, Why do we need to explicitly invoke model binding?**  
If you want to limit on what can be bound, explicitly invoking model binding can be very useful. We will discuss more about this in a later video session.