Guys, the objective of this new post is to create an ASP.NET MVC project from scratch with the recommendations that I am going to give you. For that I have made a video where we are going to explain step by step how to do it. In this video what is intended to teach is the following:

- **In the first place** , we are going to create a project from scratch and teach how to distribute the architecture correctly, our **Models** must be added as a different layer, since in the future we can reuse that layer for other projects.

- **Second** , we are going to explain why we must **encapsulate business rules in methods** , since in this way we avoid making logic in our controller and we can reuse our model from any controller. For example, if we have to list all the students in the database, with the Entity Framework it is very easy we can do something like this:

```
public ActionResult Index ()
{
     return View (new TestContext (). Student.ToList ())
}
```

But what is the problem with this? That our business logic is going to stay for that action, and we must copy the same code for each action if we want to list our Students again, there we see it easy since it is a simple list, but what if the list were more complex? Are we going to repeat all the code for the actions that were necessary? My solution is to encapsulate that logic in a method inside our Student class.

```
[Table ("Student")]
public partial class Student
{
     public Student ()
     {
         Course = new HashSet <Course> ();
     }

     public int id {get; set; }

     [Required]
     [StringLength (50)]
     public string Name {get; set; }

     [Required]
     [StringLength (100)]
     public string Last name {get; set; }
     public virtual ICollection <Course> Course {get; set; }

     public List <Student> List ()
     {
         var students = new List <Student> ();
         try
         {
```

```
            using (var context = new TestContext ())
            {
                students = context.Student.ToList ();
            }
        }
        catch (Exception)
        {
            throw;
        }

        return students;
    }
}
```

In this way, no matter how many times we have to list the Students, the logic will always do the same and we can reuse it from any controller.

- Finally, we are going to see a little what Layouts are and add Bootstrap to our project.

Without more to say, see the related article on site kodlogs.com and try to create your project from scratch, if you need help, post your question at the bottom where the comments are.