Topic/Title:

Keywords/Questions:

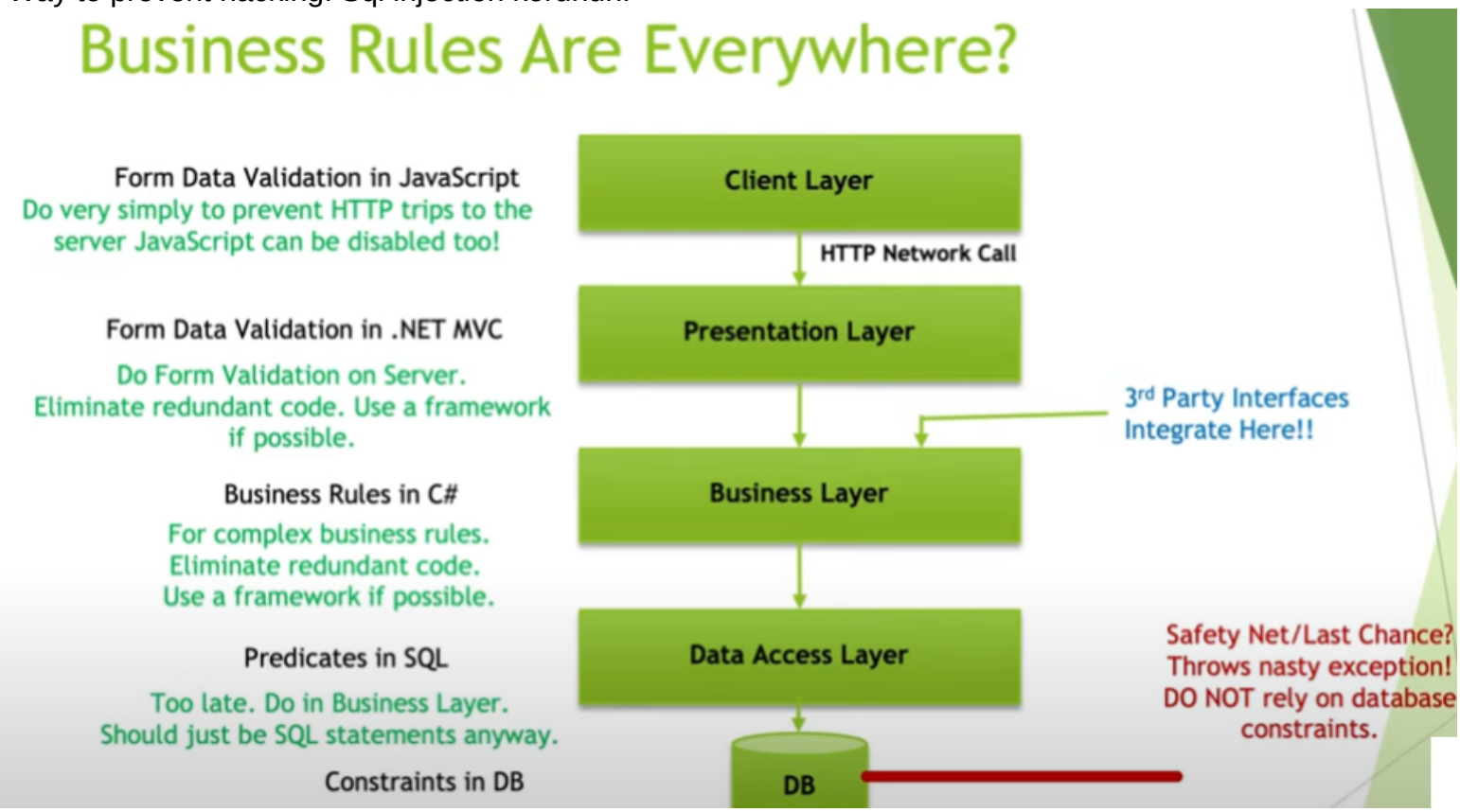
Notes:

Summary:

Why Data Validation?

Prevent corrupted or invalid data.

Way to prevent hacking. Sql injection korunun.



Client Layer

Uygulamanızın en üst katmanınız javascript'inizdir, bunu herhangi bir sunucu eylemi olmadan web sayfasında çalıştırabilirsiniz, ekstra trafiği önler ve bu nedenle javascript doğrulaması koymak kesinlikle iyi bir fikirdir, ancak yalnızca buna güvenmemelisiniz çünkü kullanıcı kesinlikle Javascript'i kapatabilir, bu web tarayıcınızdaki bir seçenektir ve ayrıca bir web sayfasındaki form gönderme seçeneklerini ortadan kaldırabilecek başka araçlar da vardır.

Presentation Layer

Kendinize devam eden form doğrulaması var ve sunucuda çalışıyor, böylece kullanıcılar oraya giremez ve web sitesini hackleyemez ve kodu göndermeden önce değiştiremez, .Net MVC gibi bir çerçeve kullanın.

Business Layer

uygulamanızın denetleyicisinde, bir kullanıcı girdisini kabul edecek miyim veya veritabanına gidip bazı bilgiler mi alacağım gibi belirli şeylere sahip olacaksınız ve böylece iş katmanınız verilerin kontrolünü elinde bulundurduğundan emin olmak için orada kontrol edin.

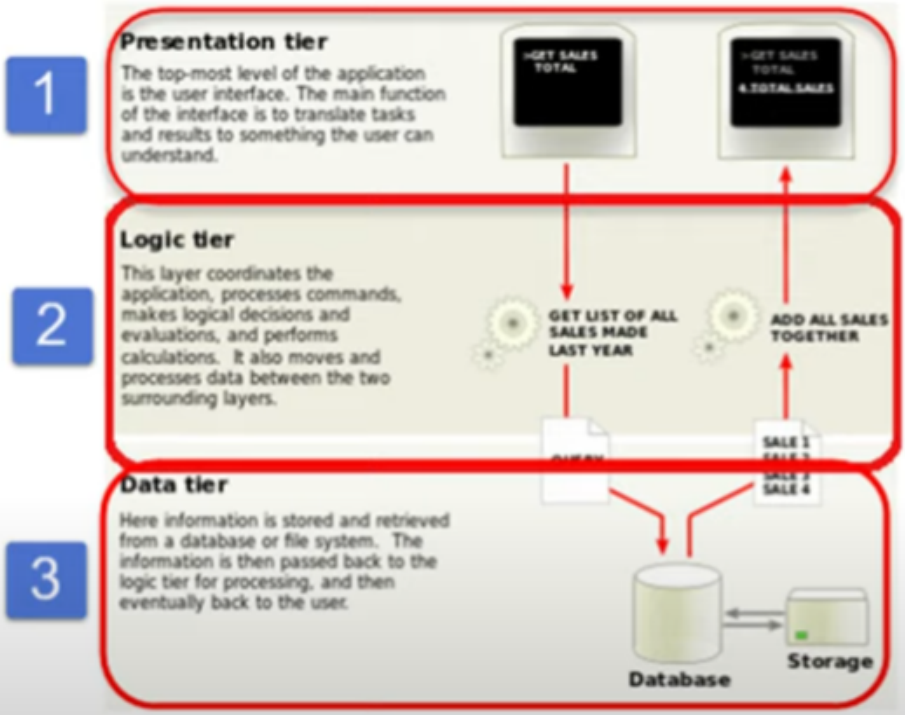
Data Access Layer

SQL kodu yazmayı düşünüyorsunuz ve bu nedenle seçme ifadeleri yazıp ifadeleri ekliyorsunuz, burada neredeyse çok geç çünkü veriler formdan geçersiz bir şekilde geldiyse, hazır olup olmadığını kontrol etmek için iyi bir yer değildir. veritabanına git, sadece SQL ekleme ve birleştirme ve silme işlemleriniz ile çalışmanız ve yukarıdan gelen verileri geçerli bir durumda kabul etmeniz muhtemelen daha iyi bir fikirdir.

DB

Gerçekte tasarlanmış tabloları veritabanına almanız gerekir ve uygulamalarınızın gideceği doğrulamayı yapmak için tablo tasarımına güveniyorsanız, karakter sınırı veya dize veya tamsayı veya beklediğiniz her şey gibi alan sınırlamalarını koyarsınız. kilitlenme ve kullanıcılarınız iyi bir deneyime sahip olmayacak

What are some Best Practices?

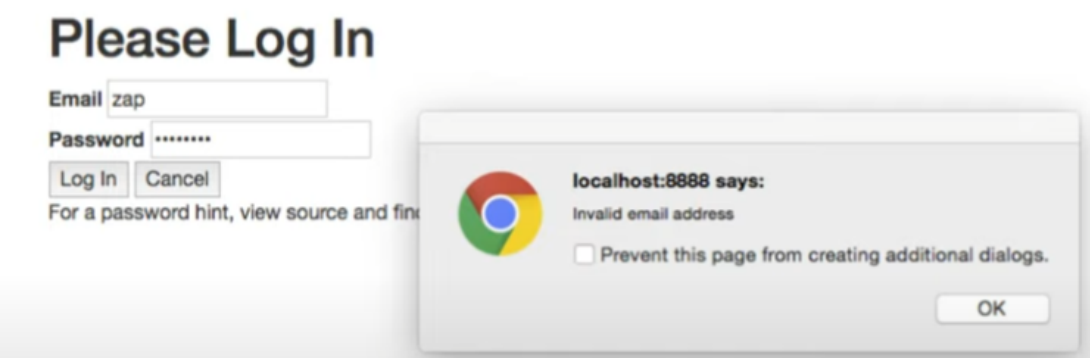


Build a layered architecture so you can put Data Validation and Business Rules in its proper place.

Veri doğrulamanızı ne zaman ve nerede yapmalısınız, bu nedenle her şeyden önce uygulamanızı bir planla oluşturun, programınızın mimarisini oluşturan n katmanlı bir yaklaşım kullanın, veri doğrulamanızın gerçekleşebileceği iyi bir nokta bulmanın en iyi yoludur, böylece en üstte görebilirsiniz javascript doğrulaması gibi olan sunum katmanının ortadaki mantık katmanı, muhtemelen veri doğrulama kurallarını düşünmeniz gereken yerdir ve daha sonra tabii ki, güvenebiliyorsanız veritabanlarıyla etkileşime girdiğiniz alttaki veri katmanındadır. seviye 2'nin doğru olması o zaman seviye 3 bir kez daha daha iyi çalışır Javascript'e güvenmeyin sadece kullanmak için geçerli bir araçtır, sunucuya fazladan gezileri önler ancak bunu geri dönüşünüz olarak kullanmayın bu önleme içindir sunucunun her şeyi yapmasına gerek kalmamasına yardımcı olmak için, ancak kesinlikle verilerinizi doğrulamanın güvenli bir yolu değildir.

What are some Best Practices?

Do not use Client Side JavaScript except for enforcing very very simple rules.



What are some Best Practices?

- Use a data validation framework if possible.

```
3 references
public class AppointmentModel
{
    [Required]
    [StringLength(20, MinimumLength =4)]
    [DisplayName("Patient's full name")]
    1 reference
    public string patientName { get; set; }

    [Required]
    [DataType(DataType.Date)]
    [DisplayName("Choose the desired date for your next visit")]
    1 reference
    public DateTime appointmentDate { get; set; }

    [Required]
    [DataType(DataType.Currency)]
    [DisplayName("How much money do you have?")]
    1 reference
    public decimal patientNetWorth { get; set; }


    [DisplayName("What is the name of doctor you wish to see?")]
    [DefaultValue("Zhivago")]
    3 references
    public string doctorLastName { get; set; } = "Zhivago";
```

MVC projesi aç. Models klasörüne `AppointmentModel.cs classı ekle.`

```
public AppointmentModel(string patientName, DateTime appointmentDate, decimal patientNetWorth, string
doctorLastName, int currentPainLevel)
        {
            this.patientName = patientName;
            this.appointmentDate = appointmentDate;
            this.patientNetWorth = patientNetWorth;
            this.doctorLastName = doctorLastName;
            this.currentPainLevel = currentPainLevel;
        }

        public string patientName { get; set; }
        public DateTime appointmentDate { get; set; }
        public decimal patientNetWorth { get; set; }
        public string doctorLastName { get; set; }
        public int currentPainLevel { get; set; }
    }
```

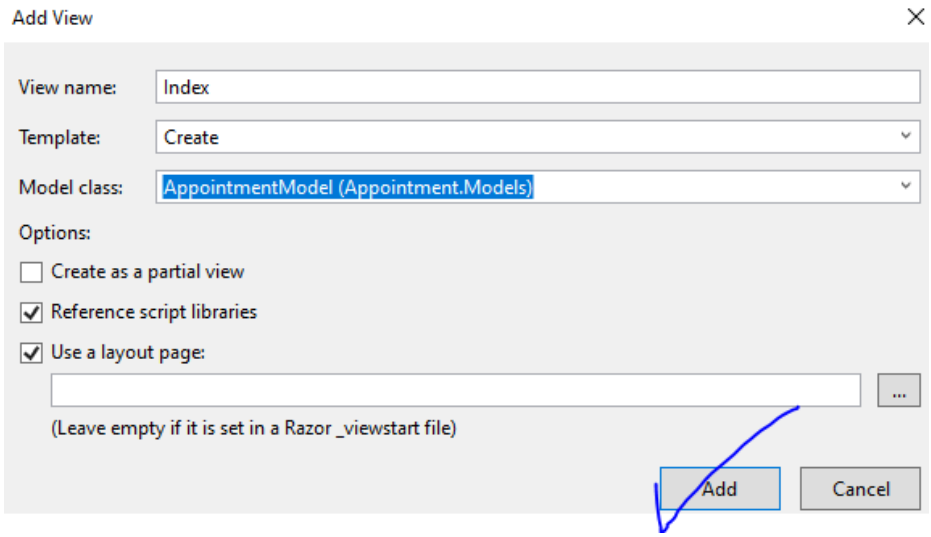
Usually an appointment would use the patient id number and the doctor's id number.

Controllers klasörüne
`AppointmentController controller ekle`

RouteConfig ekle

```
routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Appointment", action = "Index", id = UrlParameter.Optional
}
            );
```

AppointmentController ActionResult Index() sağ tık add View




```
[Required]
        [DisplayName("Patient's Full Name")]
        public string patientName { get; set; }
        [Required]
        [DisplayName("Choose the desired date for your next visit")]
        public DateTime appointmentDate { get; set; }
        [Required]
        [DisplayName("How much money do you have?")]
        public decimal patientNetWorth { get; set; }
        [Required]
        [DisplayName("What is the name of doctor you wish to see?")]
        public string doctorLastName { get; set; }
        [Required]
        [DisplayName("What is your cuurent pain level (0 to 10)")]
        public int currentPainLevel { get; set; }
```

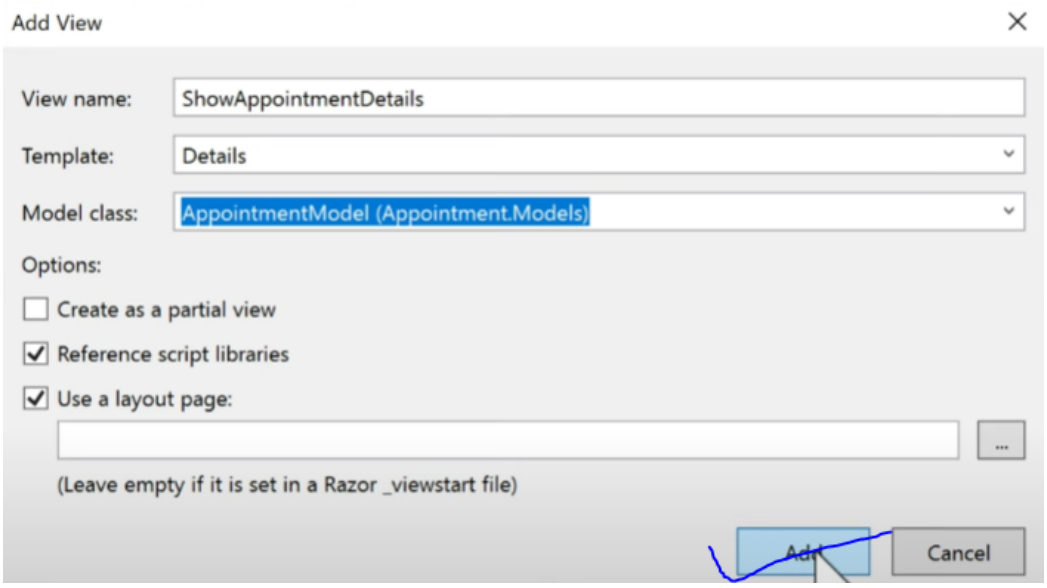
AppointmentModel.cs

```
  [Required]
        [StringLength(20,MinimumLength =2)]
        [DisplayName("Patient's Full Name")]
        public string patientName { get; set; }
        [Required]
        [DataType(DataType.Date)]
        [DisplayName("Choose the desired date for your next visit")]
        public DateTime appointmentDate { get; set; }
        [Required]
        [DataType(DataType.Currency)]
        [DisplayName("How much money do you have?")]
        public decimal patientNetWorth { get; set; }
        [DisplayName("What is the name of doctor you wish to see?")]
          public string doctorLastName { get; set; } = "Tarik Onur Tiryaki" ;
        [Required]
        [Range(0,10)]
        [DisplayName("What is your cuurent pain level (0 to 10)")]
        public int currentPainLevel { get; set; }
```

Next: Display a response to the form input.

AppointmentController.cs şu metodu ekle ve ardından View ekle

```
public ActionResult ShowAppointmentDetails()
        {
            return View();
        }
```




```
public ActionResult ShowAppointmentDetails(Models.AppointmentModel appointment)
        {
            return View("ShowAppointmentDetails",appointment);
        }
```

AppointmentModel.cs ekle

```
public AppointmentModel()
        {

        }
```

Index.cshtml ekle

```
@using (Html.BeginForm("ShowAppointmentDetails","Appointment"))     //ActionName , Controller Name
```

AppointmentController.cs ekle

```
 public ActionResult ShowAppointmentDetails(Models.AppointmentModel appointment)
        {
            if (appointment.doctorLastName == null)
            {
                appointment.doctorLastName = "Tarik Onur Tiryaki";
            }
            if (ModelState.IsValid)
            {
                return View("ShowAppointmentDetails", appointment);
            }
            else
            {
                return View("Index");
            }

        }
```