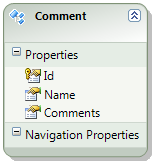
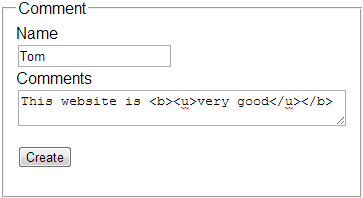
**Cross-site scripting attack** also called as **XSS attack**, is a security vulnerability found in Web applications. **XSS** allows hackers to inject client-side script into Web pages, and later, if that web page is viewed by others, the stored script gets executed. The consequences of XSS may range from a petty nuisance like displaying an alert() box to a significant security risk, like stealing session cookies.

**Let's understand cross site scripting attack with an example.** We will be using table tblComments for this demo.   
CREATE TABLE tblComments  
(  
   Id INT IDENTITY PRIMARY KEY,  
   Name NVARCHAR(50),  
   Comments NVARCHAR(500)  
)

Create an asp.net mvc 4.0 application. Right click on **"Models"** folder, and add **ADO.NET entity data model**, based on table **tblComments**. Change the name of the entity from **tblComment** to **Comment**.  
   
  
Build the solution, to make sure **"Comment"** model is compiled.  
  
**Now right click on Controllers folder, and add a "new controller". Set**  
**1.** Controller name = HomeController  
**2.** Template = MVC controller with read/write actions and views, using Entity Framework  
**3.** Model class = Comment   
**4.** Data context class = SampleDBContext  
**5.** Views = Razor (CSHTML)  
  
Click **"Add"**. This generates the **HomeController** and the required views.  
  
In **Create.cshtml** view, change  
@Html.EditorFor(model => model.Comments)  
TO  
@Html.TextAreaFor(model => model.Comments)  
  
**The above change is to display a multi-line textbox for entering comments.**  


**Now let's inject javascript using the "Create" view**. Navigate to **"localhost/MVCDemo/Home/Create"** view. Type the following client-side script, into "Comments" textbox, and click "Create". We get an error - A potentially dangerous Request.Form value was detected from the client (Comments="<script"). This is one of the security measures in place, to prevent script injecttion, to avoid XSS attack.   
<script type="text/javascript">  
alert('Your site is hacked');  
</script>  
  
Depending on the project requirement, there may be legitimate reasons, to submit html. For example, let's say we want to bold and underline a word in the comment that we type. To bold and underline the word **"very good"** in the following statement, we would type a statement as shown below and submit.  
**This website is <b><u>very good</u></b>**  
  
To allow this HTML to be submitted. We need to disable input validation. When we click the **"Create"** button on **"Create"** view, the **"Create()"**controller action method that is decorated with **[HttpPost]** attribute in the **"HomeController"**. To disable input validation, decorate Create() action method with "ValidateInput" attribute as shown below.  
[HttpPost]  
[ValidateInput(false)]  
public ActionResult Create(Comment comment)  
{  
    if (ModelState.IsValid)  
    {  
        db.Comments.AddObject(comment);  
        db.SaveChanges();  
        return RedirectToAction("Index");  
    }  
  
    return View(comment);  
}  
  
Build the solution. Navigate to **"Create"** view. Enter   
**Name**= Tom  
**Comments** = This website is <b><u>very good</u></b>  
   
  
Now click **"Create"**. Data gets saved as expected, and the user is redirected to Index action. On the **"Index"** view, instead of rendering the word **"very good"** with an under-line and in bold, the encoded html is displayed as shown below.  
encoded html   
  
By default asp.net mvc encodes all html. This is another security measure in place, to prevent XSS attack. To disable html encoding, make the fllowing change on Index.cshtml view.  
**CHANGE**  
@Html.DisplayFor(modelItem => item.Comments)  
**To**  
@Html.Raw(item.Comments)  
  
Navigate to Index view, and notice that, the word **"Very Good"** is rendered with an underline and in bold.  
Disable html encoding   
  
**The following 2 changes that we have done has opened the doors for XSS.**  
**1.** By allowing HTML to be submitted on Create.cshtml view  
**2.** By disabling HTML encoding on Index.cshtml  
  
**To initiate a cross site scripting attack**  
**1.** Navigate to "Create" view  
**2.** Type the following javascript code in "Comments" textbox  
<script type="text/javascript">  
alert('Your site is hacked');  
</script>  
**3.** Click "Create"  
  
Refresh Index view and notice that javascript alert() box is displayed, stating that the site is hacked. By injecting script, it is also very easy to steal session cookies. These stolen session cookies can then be used to log into the site and do destructive things.  
  
So, in short, by allowing HTML to be submitted and disabling HTML encoding we are opening doors for XSS attack.

In our next video, we will discuss preventing XSS while allowing only the HTML that we want to accept.