Please watch [Part 89](http://csharp-video-tutorials.blogspot.com/2013/09/part-89-remote-validation-in-aspnet-mvc.html) from [asp.net mvc tutorial](http://www.youtube.com/user/kudvenkat/videos?view=1&flow=grid), before proceeding.  
  
Out of the box, **Remote**attribute only works when JavaScript is enabled. If the end user, disables JavaScript on his/her machine then the validation does not work. This is because RemoteAttribute requires JavaScript to make an asynchronous AJAX call to the server side validation method. As a result, the user will be able to submit the form, bypassing the validation in place. This why it is always important to have server side validation.  
  
   
  
**To make server side validation work, when JavaScript is disabled, there are 2 ways**  
**1.** Add model validation error dynamically in the controller action method  
**2.** Create a custom remote attribute and override IsValid() method  
  
In this video, we will discuss, **adding model validation error dynamically in the controller action method.**We will continue with the example, that we worked with in Part 89. Modify the Create action method that is decorated with [HttpPost] attribute as shown below.  
[HttpPost]  
public ActionResult Create(User user)  
{  
    // Check if the UserName already exists, and if it does, add Model validation error  
    if (db.Users.Any(x => x.UserName == user.UserName))  
    {  
        ModelState.AddModelError("UserName", "UserName already in use");  
    }  
    if (ModelState.IsValid)  
    {  
        db.Users.AddObject(user);  
        db.SaveChanges();  
        return RedirectToAction("Index");  
    }  
  
    return View(user);  
}  
  
**At this point, disable JavaScript in the browser, and test your application.** Notice that, we don't get client side validation, but when you submit the form, server side validation still prevents the user from submitting the form, if there are validation errors.  
  
However, delegating the responsibility of performing validation, to a controller action method violates separation of concerns within MVC. Ideally all validation logic should be in the Model. Using validation attributes in mvc models, should be the preferred method for validation. In our next video, we will discuss, creating a custom remote attribute and overriding **IsValid()** method. 