After the model binder is finished updating the model properties with new values, the model binder uses the current model metadata and ultimately obtains all the validators for the model. The MVC runtime provides a validator to work with data annotations (the DataAnnotationsModelValidator). This model validator can find all the validation attributes and execute the validation logic inside. The model binder catches all the failed validation rules and places them into model state
All the validation annotations (such as Required and Range) ultimately derive from the ValidationAttribute base class

[Required]
ako e prazno ili null
[StringLength(160)]
do 160
[Range(35,44)]
The values are inclusive. The Range attribute can work with integers and doubles, and another overloaded version of the constructor takes a Type parameter and two strings (which can allow you to add a range to date and decimal properties,
[StringLength(160, MinimumLength=3)]
od 3 do 160
[Compare("Email")]
Compare ensures two properties on a model object have the same value.
[Remote("CheckUserName", "Account")]
The Remote attribute enables you to perform client-side validation with a server callback. Take, for example, the UserName property of the RegisterModel class in the MVC Music Store. No two users should have the same UserName value, but validating the value on the client to ensure the value is unique is difficult (to do so you would have to send every single username from the database to the client). With the Remote attribute you can send the UserName value to the server, and compare the value against the values in the database.

```
public JsonResult CheckUserName(string username)
{
var result = Membership.FindUsersByName(username).Count == 0;
return Json(result, JsonRequestBehavior.AllowGet);
}
```

The controller action will take a parameter with the name of the property to validate and return a true or false wrapped in JavaScript Object Notation (JSON).
[Display(Name="First Name")]
[ReadOnly(true)]
[DataType(DataType.Password)]
[HiddenInput(DisplayValue = false)]
The HiddenInput attribute lives in the System.Web.Mvc namespace and tells the runtime to render an input element with a type of hidden.

[MaxWords(10)]

Custom Validation Attribute

```
public class MaxWordsAttribute : ValidationAttribute
{
private readonly int _maxWords;
public MaxWordsAttribute(int maxWords)
{
_maxWords = maxWords;
}
protected override ValidationResult IsValid(
object value, ValidationContext validationContext)
{
if (value != null)
{
var valueAsString = value.ToString();
if (valueAsString.Split(' ').Length > _maxWords)
{
return new ValidationResult("Too many words!");
}
}
return ValidationResult.Success;
}
}
```