boolean

CORSO JUNIOR
.NET DEVELOPER

HTTP POST

Come inviare dati dalla View al Controller?

I metodi POST

I metodi **POST** ci permettono di **inviare dati** dalle View ai Controller.

Per creare un metodo POST dobbiamo:

1. Dichiarare che il metodo è di tipo POST con l'apposito attributo **[HttpPost]**
2. Dichiarare il **modello** ricevuto dalla vista e che viene passato nel body della richiesta HTTP!
3. Inserire l'attributo **[ValidateAntiForgeryToken]** per evitare attacchi WEB (CSRF)

Creare un metodo POST

In questo esempio, il metodo Create() viene invocato con il metodo POST, riceve un modello di tipo Profile e si occupa di creare il profilo con i dati ricevuti.

Con il metodo **IsValid()** della classe ModelState controlliamo che il modello sia valido.

```
1  public class ProfileController : Controller
2  {
3      [HttpPost]
4      [ValidateAntiForgeryToken]
5      public IActionResult Create(Profile data)
6      {
7          if (!ModelState.IsValid)
8          {
9              return View("Create", data);
10         }
11
12         using (ProfileContext context = new ProfileContext())
13         {
14             Profile profileToCreate = new Profile();
15             profileToCreate.Name = data.Name;
16             profileToCreate.Lastname = data.Lastname;
17             profileToCreate.Age = data.Age;
18
19             context.Profiles.Add(profileToCreate);
20
21             context.SaveChanges();
22
23             return RedirectToAction("Index");
24         }
25     }
26 }
```

Quando si passa un modello ad un metodo HTTP POST,
il modello deve avere almeno il **costruttore** vuoto dichiarato!

```
1  public class Profile
2  {
3    public string Name { get; set; }
4    public string Lastname { get; set; }
5    public int Age { get; set; }
6
7    public Profile()
8    {
9
10   }
11
12   public Profile(string name, string lastname, string age)
13   {
14     this.Name = name;
15     this.Lastname = lastname;
16     this.Age = Age;
17   }
18 }
```

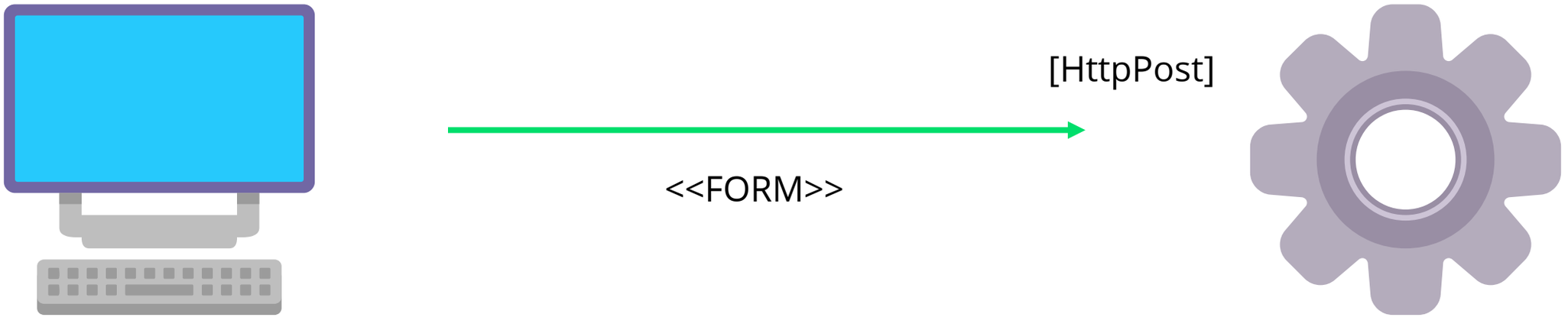
Creare un metodo POST

Con il metodo **RedirectToAction()** possiamo indicare a quale metodo reindirizzare l'utente una volta che l'operazione di inserimento ad esempio è andata a buon fine!

```
1  public class ProfileController : Controller
2  {
3      [HttpPost]
4      [ValidateAntiForgeryToken]
5      public IActionResult Create(Profile data)
6      {
7          if (!ModelState.IsValid)
8          {
9              return View("Create", data);
10         }
11
12         using (ProfileContext context = new ProfileContext())
13         {
14             Profile profileToCreate = new Profile();
15             profileToCreate.Name = data.Name;
16             profileToCreate.Lastname = data.Lastname;
17             profileToCreate.Age = data.Age;
18
19             context.Profiles.Add(profileToCreate);
20
21             context.SaveChanges();
22
23             return RedirectToAction("Index");
24         }
25     }
26 }
```

POST + Form = ❤️

I metodi POST si usano spessissimo nei FORM dei siti web!
Vengono infatti utilizzati per ricevere e validare i dati che l'utente vuole inviare al server!



Come creare un FORM che usi il metodo POST

Dobbiamo creare una semplice View e scrivere il codice HTML come abbiamo sempre fatto, con qualche piccola aggiunta!

```
@model Profile

<div class="container">
  <form asp-controller="Profile" asp-action="Create" method="post">
    @Html.AntiForgeryToken()
    <div class="mb-3">
      <strong>Nome:</strong>
      <input asp-for="Name" class="d-block w-100" />
    </div>
    <div class="mb-3">
      <strong>Cognome:</strong>
      <input asp-for="Lastname" class="d-block w-100" />
    </div>
    <div class="mb-3">
      <strong>Età:</strong>
      <input asp-for="Age" class="d-block w-100" />
    </div>
    <div class="text-end">
      <input type="submit" class="btn btn-small btn-info" value="Salva">
    </div>
  </form>
</div>
```

Come creare un FORM che usi il metodo POST

Dopodiché dobbiamo ovviamente creare un metodo del Controller per visualizzare la View con il form! Questo metodo verrà invocato con una richiesta HTTP GET.

```
1  public class ProfileController : Controller
2      {
3          // ...
4
5          [HttpGet]
6          public IActionResult Create()
7          {
8              return View("Create");
9          }
10
11         // ...
12     }
```

LIVE CODING

Creiamo il form

Validare un FORM

.NET dispone di una validazione dei campi "automatica".
Per applicare le regole di validazione ci basta aggiungere le apposite direttive agli attributi del Model.

```
1  public class Profile
2  {
3    [Required(ErrorMessage = "Il campo è obbligatorio")]
4    [StringLength(25, ErrorMessage = "Il nome non può avere più di 25 caratteri")]
5    public string Name { get; set; }
6
7    [Required(ErrorMessage = "Il campo è obbligatorio")]
8    [StringLength(50, ErrorMessage = "Il nome non può avere più di 50 caratteri")]
9    public string Lastname { get; set; }
10
11   [Required(ErrorMessage = "Il campo è obbligatorio")]
12   [Range(18,80, ErrorMessage = "L'età deve essere compresa tra 18 e 80 anni")]
13   public int Age { get; set; }
14
15   // ...
16
17 }
```

Attributi di validazione

Gli attributi di validazione che possiamo configurare per ogni attributo della classe modello sono i seguenti:

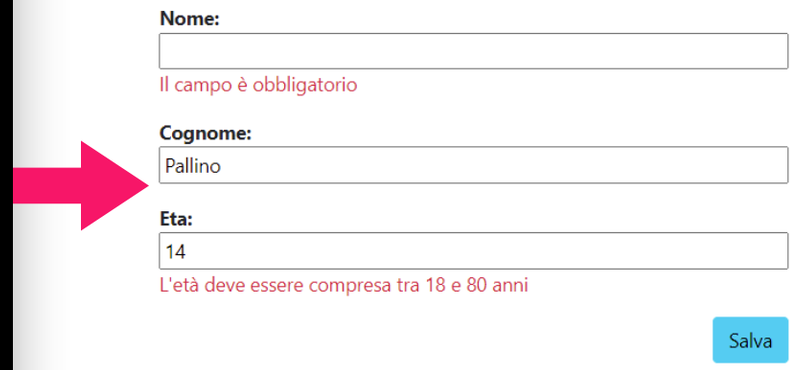
- **[EmailAddress]:** convalida che la proprietà abbia un formato di posta elettronica.
- **[Phone]:** convalida che la proprietà sia un numero di telefono
- **[Range(min, max)]:** convalida che la proprietà sia un numero nel range specificato
- **[Required]:** convalida che il valore non sia nullo
- **[StringLength(max)]:** convalida che non vengano usati più di tot caratteri
- **[Url]:** convalida che il testo inserito sia un URL valido

Model validation in ASP.NET Core MVC | Microsoft Docs

Aggiungiamo i messaggi d'errore alla View del FORM

```
@model Profile

<div class="container">
  <form asp-controller="Profile" asp-action="Create" method="post">
    @Html.AntiForgeryToken()
    <div class="mb-3">
      <strong>Nome:</strong>
      <input asp-for="Name" class="d-block w-100" />
      <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="mb-3">
      <strong>Cognome:</strong>
      <input asp-for="Lastname" class="d-block w-100" />
      <span asp-validation-for="Lastname" class="text-danger"></span>
    </div>
    <div class="mb-3">
      <strong>Età:</strong>
      <input asp-for="Age" class="d-block w-100" />
      <span asp-validation-for="Age" class="text-danger"></span>
    </div>
    <div class="text-end">
      <input type="submit" class="btn btn-small btn-info" value="Salva">
    </div>
  </form>
</div>
```



Definire delle regole di validazione personalizzate

Creare nuovi attributi di validazione

Se abbiamo delle esigenze particolari, possiamo creare un nuovo attributo creando una **nuova classe** che estende **ValidationAttribute** e facendo quindi l'**override** del metodo **IsValid().**

In questo modo stiamo definendo delle regole di validazione personalizzare.

```
1  public class MoreThanOneWordValidationAttribute : ValidationAttribute
2  {
3    // ...
4    protected override ValidationResult IsValid(object value, ValidationContext validationContext)
5    {
6      string fieldValue = (string) value;
7
8      if (fieldValue == null || fieldValue.Trim().IndexOf(" ") == -1)
9      {
10         return new ValidationResult("Il campo deve contenere almeno due parole");
11     }
12
13     return ValidationResult.Success;
14   }
15 }
```

Definire delle regole di validazione personalizzate

Usare gli attributi di validazione personalizzati

A questo punto possiamo usare il nuovo attributo che abbiamo appena creato su un campo del modello.

```
1  public class Profile
2  {
3    [Required(ErrorMessage = "Il campo è obbligatorio")]
4    [MoreThanOneWordValidationAttribute]
5    public string NameAndLastname { get; set; }
6
7    [Required(ErrorMessage = "Il campo è obbligatorio")]
8    [Range(18,80, ErrorMessage = "L'età deve essere compresa tra 18 e 80 anni")]
9    public int Age { get; set; }
10
11   public Profile()
12   {
13
14   }
15 }
```

LIVE CODING

Validiamo il nostro FORM

ESERCITAZIONE