
django

crispy forms

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy forms

Per quanto django offra strumenti veloci ed efficienti per realizzare form con pochissimo sforzo, non permette però di curarne l'aspetto e i form generati risultano funzionali ma graficamente poveri.

Per sopperire a questo è stato creato django-crispy-form, un'applicazione che permette di interagire con proprietà dei form direttamente dal backend controllando così il comportamento di rendering dei django form in modo elegante e DRY (Don't Repeat Yourself).

Come installare django-crispy-form
Per installare la versione più recente di django-crispy-form è sufficiente eseguire all'interno del pipenv:
`pip install --upgrade django-crispy-forms`

Aggiungi `crispy_forms` a `INSTALLED_APPS` in `settings.py`:

```
INSTALLED_APPS = (
    ...
    'crispy_forms',
)
```

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy forms

django-crispy-forms offre un supporto integrato per diversi framework CSS, chiamati template packs all'interno di django-crispy-forms:

- `bootstrap` → Bootstrap versione 2 è il template pack utilizzato di default da crispy-forms;
- `bootstrap3` → Twitter Bootstrap versione 3;
- `bootstrap4` → Twitter Bootstrap versione 4;
- `uni-form` → Uni-form è un framework che standardizza il markup e lo stile dei form e lo modella con CSS;
- `foundation` → Foundation viene definito dai creatori come "Il responsive framework front-end più avanzato al mondo". Questo template pack è accessibile esternamente attraverso `crispy-forms-foundation`

Per indicare a django il corretto template pack da utilizzare è sufficiente inserire nel file settings.py:

```
CRISPY_TEMPLATE_PACK = '<nome template pack>'
```

crispy filter

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy filter

Il metodo più semplice per usare crispy-forms è quello di utilizzare il filtro |crispy:

```
{% load crispy_forms_tags %}

<form action="home/" method="POST">
  {{ form|crispy }}
</form>
```

1. Aggiungi la tag `{% load crispy_form_tags %}` al template;

2. Aggiungi il filtro `|crispy` alla variabile form del context.

Per quanto il filtro `|crispy` sia molto utile, lo si può concepire come i metodi integrati `as_p`, `as_table` e `as_ul`, quindi non permette di customizzare l'output in modo consistente.
Il miglior metodo per avere un controllo totale dell'output dei template è quello di utilizzare il tag `{% crispy %}`

{% crispy %}

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy tag

django-crispy-forms implementa la classe `FormHelper` che va a definire il comportamento di rendering del form. Questa classe permette di controllare gli attributi del form e il suo layout. In questo modo si può scrivere un quantitativo minimo di HTML e tutte le specifiche circa la renderizzazione vengono indicate nei file `forms` e `views`.

```
class PersonaCrispyForm(forms.ModelForm):

    class Meta:
        model = Persona
        fields = ('nome', 'cognome', 'ruolo')
```

Per gli esempi che seguono utilizzeremo questo `ModelForm` relativo al model `Persona`, presente nell'app `soci` del progetto `tutorial_project`

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy tag

Per prima cosa è necessario importare la classe `FormHelper`:

```
from crispy_forms.helper import FormHelper
```

Come regola di massima, se è necessario apportare modifiche dopo l'istanziazione del form, è opportuno assegnare l'helper ad un attributo di istanza. In caso contrario si assegna `FormHelper` ad un attributo di classe.

La tag `{% crispy %}` accetta due parametri:
1. il nome assegnato all'oggetto `form` nel context (`form` di default);
2. l'attributo `helper` di `form`.

Assegnando il nome `helper` all'istanza di `FormHelper` sarà sufficiente utilizzare `{% crispy form %}` anziché `{% crispy form form.helper %}`.

```python
from django import forms
from crispy_forms.helper import FormHelper
from crispy_forms.layout import Submit

class PersonaCrispyForm(forms.ModelForm):

    helper = FormHelper()
    helper.from_id = 'persona-crispy-form'
    helper.form_method = 'POST'
    helper.add_input(Submit('submit', 'Submit'))

    class Meta:
        model = Persona
        fields = ('nome', 'cognome', 'ruolo')
```

```
_form_method = 'post'
_form_action = ''
_form_style = 'default'
form = None
form_id = ''
form_class = ''
form_group_wrapper_class = ''
layout = None
form_tag = True
form_error_title = None
formset_error_title = None
form_show_errors = True
render_unmentioned_fields = False
render_hidden_fields = False
render_required_fields = False
_help_text_inline = False
_error_text_inline = True
html5_required = False
form_show_labels = True
template = None
field_template = None
disable_csrf = False
use_custom_control = True
label_class = ''
field_class = ''
include_media = True
```

crispy tag

```
helper = FormHelper()
helper.from_id = 'persona-crispy-form'
helper.form_method = 'POST'
helper.add_input(Submit('submit', 'Submit'))
```

`FormHelper` offre moltissimi attributi che permettono di customizzare consistentemente il form.
Nel nostro esempio utilizziamo:
- `istanza.form_id` → ID che viene assegnato al tag `<form>`;
- `istanza.form_method` → metodo assegnato all'attributo method di `<form>` (`POST` di default);

A fianco la lista completa consultabile in crispy_forms/helper.py alla definizione di FormHelper.

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy tag

```
helper = FormHelper()
helper.from_id = 'persona-crispy-form'
helper.form_method = 'POST'
helper.add_input(Submit('submit', 'Submit'))
```

`instanza.add_input()` è un metodo della classe FormHelper che permette di aggiungere degli input_object al form, nel nostro caso un'istanza della classe `Submit`, che genera un pulsante "submit" per inviare i dati del form.

`Submit` è una classe presente in `crispy_forms.layout` che genera un elemento `<input type="submit"/>`.

Accetta molteplici argument al momento dell'istanza, dei quali alcuni posizionali:

- '`submit`' → il valore che verrà assegnato all'attributo name dell'input;

- '`Submit`' → il valore che verrà assegnato all'attributo value dell'input (il contenuto mostrato sul pulsante).

`Submit` provvede inoltre a popolare gli attributi dell'input con altri valori, non ultimi dei valori all'attributo `class`:
nel caso in cui il valore di `CRISPY_TEMPLATE_PACK` non sia `uni_form`, assegnarà le classi `btn btn-primary`

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy tag

Questo sarà il codice generato da crispy-forms:

```
<form id="persona-crispy-form" method="post">
  <input type="hidden" name="csrfmiddlewaretoken" value="XWImVyAgoUTvfcmmz5uFtVzMgz3JiFlniLhcKxB9zBTVIXkyBaimFOv63IlNIpZe">
  <div id="div_id_nome" class="form-group">
    <label for="id_nome" class=" requiredField">Nome<span class="asteriskField">*</span></label>
    <div class=""><input type="text" name="nome" maxlength="50" class="textinput textInput form-control" required id="id_nome"></div>
  </div>
  <div id="div_id_cognome" class="form-group">
    <label for="id_cognome" class=" requiredField">Cognome<span class="asteriskField">*</span></label>
    <div class=""> <input type="text" name="cognome" maxlength="50" class="textinput textInput form-control" required id="id_cognome"> </div>
  </div>
  <div id="div_id_ruolo" class="form-group">
    <label for="id_ruolo" class=" requiredField">Ruolo<span class="asteriskField">*</span></label>
    <div class="">
      <select name="ruolo" class="select form-control" required id="id_ruolo">
        <option value="" selected>---------</option>
        <option value="1">Impiegato</option>
        <option value="2">Responsabile</option>
        <option value="3">Direttore</option>
      </select>
    </div>
  </div>
  <div class="form-group">
    <div class="">
      <input type="submit" name="submit" value="Submit" class="btn btn-primary" id="submit-id-submit" />
    </div>
  </div>
</form>
```

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy tag

```
<form id="persona-crispy-form" method="post">
  ...
</form>
```

La tag <form> con id e method specificati

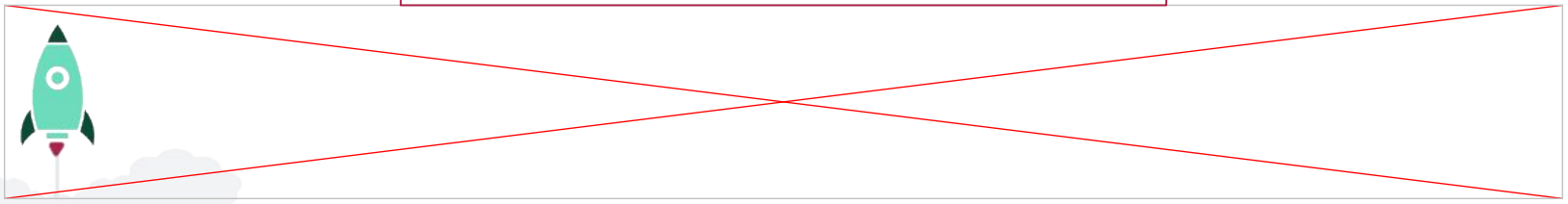
```
<input type="hidden" name="csrfmiddlewaretoken" value="XWImVyAg
```

django {% csrf_token %}

Ogni campo dotato di elementi, classi e attributi specifici

```
<div id="div_id_nome" class="form-group">
  <label for="id_nome" class=" requiredField">Nome<span class="asteriskField">*</span></label>
  <div class=""><input type="text" name="nome" maxlength="50" class="textinput textInput form-control" required id="id_nome"></div>
</div>
```

Input “submit” generato dalla classe Submit



UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy tag

django-crispy-forms offre lo strumento `Layout` che permette di modificare radicalmente il modo in cui i campi vengono renderizzati. Permette di stabilire l'ordine dei campi, inserirli in elementi `<div>` o altre strutture, aggiungere `HTML`, aggiungere id, classi o attributi a qualsiasi cosa.

Per utilizzare la classe `Layout` è sufficiente assegnarla all'attributo `layout` di un'istanza di `FormHelper`:

```
helper = FormHelper()
helper.from_id = 'persona-crispy-form'
helper.form_method = 'POST'
helper.layout = Layout()
```

L'oggetto `Layout` viene popolato da *layout objects* che si possono identificare come componenti del form.

L'oggetto `Layout` accetta un numero arbitrario di oggetti che diventeranno parte integrante del form.

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

crispy tag

Tutti le classi utilizzate nell'esempio vengono importate da `crispy_forms.layout`

```
helper.layout = Layout(
    Div(
        HTML("<p>Inserisci i dati del socio:</p>"),
        Field('nome', style="color: red;", css_class="bg-dark", title="Nome"),
        Field('cognome', style="color: orange;", css_class="bg-dark", title="Cognome"),
        Field('ruolo', style="color: green;", css_class="bg-dark", title="Ruolo"),
        css_class="d-flex justify-content-between"
    ),
    Submit('submit', 'Submit')
)
```

Nell'esempio riportato sopra, vengono passati all'oggetto `Layout` un oggetto `Div` e un oggetto `Submit`:
- `Div` → Contiene un oggetto `HTML` (che accetta stringhe contenenti `HTML` puro) e tre oggetti `Field` contenenti il nome del campo e altri attributi che verranno assegnati all'elemento `<input>` corrispondente;
- `Submit` → si comporta come l'oggetto submit passato al metodo `add_input` di `FormHelper`

Per approfondire le potenzialità di Layout, visita
https://django-crispy-forms.readthedocs.io/en/latest/layouts.html

francesco.faenza@unimore.it