

Forms

What are forms?

Forms are containers for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons...

Everybody here has already filled in some forms in their life. It is one way to gather information. The other ways are to upload files, or stream data (video, audio...)



Let's create a form

Create a new view and template to set up a form.

```
def forms(request):  
    return render(request, "movies/forms.html")
```



Template

```
{% extends "base.html" %}
```

```
{% block content %}
```

```
<form >
```

```
<label for="class_text">Grade:</label><br>
```

```
<input type="text" id="class_grade" name="class_grade"><br>
```

```
<label for="hschool_name">High School Name:</label><br>
```

```
<input type="text" id="high_school_name" name="high_school_name"><br>
```

```
</form>
```

```
{% endblock content %}
```

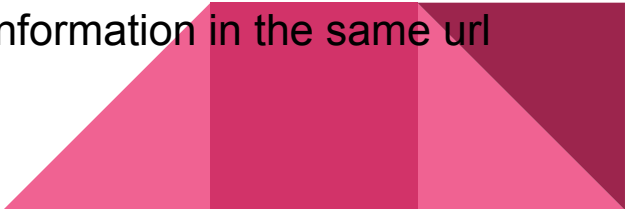


Let's add a submit button, and an URL action

We need the request with the data and where it is going:

```
{% block content %}  
  <form action="">  
    <label for="class_text">Grade:</label><br>  
    <input type="text" id="class_grade" name="class_grade"><br>  
    <label for="hschool_name">High School Name:</label><br>  
    <input type="text" id="high_school_name" name="high_school_name"><br>  
    <input type="submit" value="Submit">  
  </form>  
{% endblock content %}
```

We put the action link blank because we are going to send the information in the same url as the url that the form is in.



Questions

What type of request does the submit form is doing?

Where is the data we sent?

If you refresh the page, you are going to send the same data you fill it in the form.




GET and POST Requests

You can send a **GET** request when you are asking for data, filling the data you want on the form.

Warning: If you want the user to give you some data: password, names, any kind of information that you are going to save in the database or important information that you are going to retrieve in the database (like passwords). **DO NOT USE GET REQUESTS.** Instead, you must send a form as a **POST** query.

A **POST** request is typically sent via an HTML form and results in a change on the server.

As you see, in the **GET**, the information is in the url, so important information such as login should not appear in a url. And then information you want to save in the database should be in the post because this is the type of request in html when you want to send new, update or delete data.



FORM - POST REQUEST

```
<form action="" method="post">  
  <label for="class_text">Class grade:</label><br>  
  <input type="text" id="class_grade" name="class_grade"><br>  
  <label for="hschool_name">High School name:</label><br>  
  <input type="text" id="high_school_name" name="high_school_name"><br>  
  <input type="submit" value="Submit">  
</form>
```

What did it happen? Why is django saying that “CSRF verification failed”?

What is CSRF and how can we be protected?.

Let's watch: <https://www.youtube.com/watch?v=eHqbh0kyRYk>

SameSite cookie restrictions

SameSite is a browser security mechanism that determines when a website's cookies are included in requests originating from other websites.

SameSite cookie restrictions provide partial protection against a variety of cross-site attacks, including CSRF.

Browsers support the following SameSite restriction levels:

- Strict
- Lax
- None

Developer can add SameSite security when they set a cookie.



SameSite cookie restrictions: LAX

LAX: Browsers will send a cookie in cross-site requests, if both of the following conditions are met:

- The request uses the GET method.
- The request resulted from a top-level navigation by the user, such as clicking on a link.

The cookie is not included in cross-site POST requests.



SameSite cookie restrictions: Strict

Strict: Browsers will not send the cookie in any cross-site requests

Recommended when setting cookies that enable the privilege to modify data or perform other sensitive actions, such as accessing specific pages that are only available to authenticated users.



SameSite cookie restrictions: None

None: Disables SameSite restrictions altogether.

Browsers will send this cookie in all requests to the site that issued it, even those that were triggered by completely unrelated third-party sites.

