Views & Rotas 🗺️ 

Prof. Saulo Oliveira
Técnico em Informática para Internet
Instituto Federal do Ceará

Formulários

GET e POST

`GET` e `POST` são os únicos métodos HTTP a serem usados ao lidar com formulários.

- O formulário de login do Django é retornado usando o `POST` método, no qual o navegador agrupa os dados do formulário, codifica-os para transmissão, envia-os ao servidor e então recebe de volta sua resposta.
- `GET`, por outro lado, agrupa os dados enviados em uma string e usa isso para compor um URL. A URL contém o endereço para onde os dados devem ser enviados, bem como as chaves e valores dos dados.

> `GET` e `POST` normalmente são usados para finalidades diferentes. Qualquer solicitação que possa ser usada para alterar o estado do sistema – por exemplo, uma solicitação que faça alterações no banco de dados – deve usar `POST`. `GET` deve ser usado apenas para solicitações que não afetem o estado do sistema.

Construindo um formulário no Django

```python
# forms.py
from django import forms


class NameForm(forms.Form):
    your_name = forms.CharField(label="Your name", max_length=100)
```

Validadores (1)

Os validadores podem ser úteis para reutilizar a lógica de validação entre diferentes tipos de campos.

Um validador é uma chamada que recebe um valor e gera um `ValidationError` se não atender a alguns critérios.

```
from django.core.exceptions import ValidationError
from django.utils.translation import gettext_lazy as _


def validate_even(value):
    if value % 2 != 0:
        raise ValidationError(
            _("%(value)s is not an even number"),
            params={"value": value},
        )
```

Validadores (2)

Você pode adicionar isso a um campo de modelo através do atributo `validators`:

```
from django.db import models


class MyModel(models.Model):
    even_field = models.IntegerField(validators=[validate_even])
```

Validadores (3)

```
user = CharField(
    max_length=30,
    required=True,
    validators=[
        RegexValidator(
            regex='^[a-zA-Z0-9]*$',
            message='Username must be Alphanumeric',
            code='invalid_username'
        ),
    ]
)
```

Regex

Padrão	Significado
^	Nega a expressão OU marca 0 início de uma string
$	Marca 0 final de uma string
+	Prolonga 0 caractere anterior
*	Prolonga 0 caractere anterior, mas ele também pode não existir
?	Diz que há dúvida se 0 caractere anterior a ele existe
.	Substitui qualquer caractere (apenas um)
[A-Z]	Procura qualquer caractere maiúsculo de A a Z
[a-z]	Procura qualquer caractere minusculo de a a Z
[0-9]	Procura qualquer dígito de 0 a 9

Padrão	Significado
[125]	Procura os dígitos 1, 2 ou 5
[^0-9]	01 nega a expressão a seguir, logo, procura tudo que não for número.
[abc]	Procura os caracteres a, b ou c
[^A-Z]	Procura tudo que não for letra maiúscula
[a-zA-Z]	Procura tudo que for letra
\s	Procura espaços
\w	Procura caracteres, exceto os especiais
\d	Também procura qualquer dígito de 0 a 9

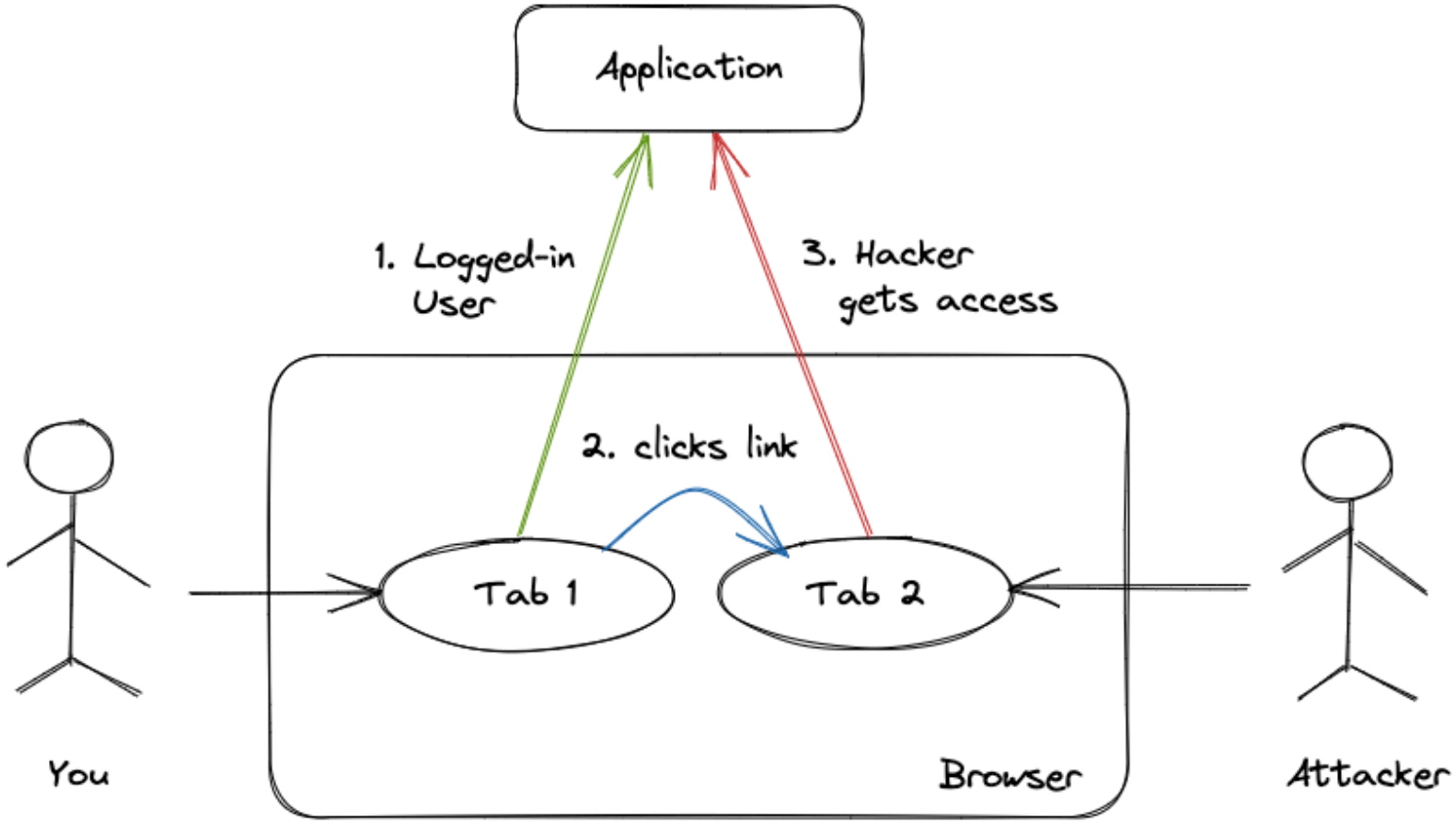
Cross-Site Request Forgery (CSRF)

Falsificações de solicitações de sites cruzados.

Esse tipo de ataque ocorre quando um site malicioso contém um link, um botão de formulário ou algum JavaScript que se destina a executar alguma ação em seu site, usando as credenciais de um usuário conectado que visita o site malicioso em seu navegador.

> ⚠️ Um tipo de ataque relacionado, `login CSRF`, onde um site de ataque engana o navegador de um usuário para fazer login em um site com as credenciais de outra pessoa, também é coberto.

Cross-Site Request Forgery (CSRF)



Fonte: https://www.writesoftwarewell.com/how-csrf-attack-works-cross-site-request-forgery/

Referências

- Django. **Working with forms**. Disponível em: https://docs.djangoproject.com/en/5.0/topics/forms/. Acessado em 20 de março de 2024.