
# Формы

Лёзин Арсений, arseny.lezin@gmail.com

# Валидация данных

Процесс проверки входных данных на соответствие заданным условиям и ограничениям

# Механизмы валидации данных в Django

- Формы
- Валидаторы

# Формы

Задачи которые выполняют формы

- Подготовить данные для рендеринга в шаблоны
- Создать HTML-форму с данными
- Получить, провалидировать данные и модифицировать данные для дальнейшего использования

# Виды Форм в Django

- Обычные формы `forms.Form`
- Формы для моделей `forms.ModelForm`

# Пример формы

`forms.py`

```
from django imports forms

class UserForm(forms.Form):
    name = forms.CharField(
        label='User name',
        max_length=100
    )
```

Преобразуется в

```
<label for="name">User name:</label>
<input id="name" type="text" name="name" maxlength="100"
```

# Использование во views

views.py

```
from .forms import UserForm

def get_name(request):
    if request.method == 'POST':
        form = UserForm(request.POST)
        if form.is_valid():
                name = form.cleaned_data['name']
            ...
    else:
        form = UserForm()
    return render(request, 'name.html', {'form': form})
```

# Использование в шаблонах

`name.html`

```
<form action='/get-name/' method='POST'>
    {% csrf_token %}
        {{ form }}
    <input type="submit" value="Получить" />
</form>
```

# Важные моменты

- form.is_valid() проверяет валидность входных данных формы
- request.method возвращает тип запроса (GET, POST, PUT, DELETE, OPTIONS, PATCH ...)
- request.POST содержит в себе 'десериализованное тело запроса'
- Для преобразования формы в html достаточно вывести ее в шаблоне
- form.cleaned_data - содержит в себе объект из которого мы можем получить по ключу - значение
- {% csrf_token %} - шаблонный тег который добавляет `<input type='hidden' name='csrf_token' value='...' />`

# Динамические методы clean_* в формах

## Нужны для:

- Валидации
- Изменения

# Пример:

forms.py

```
from django imports forms

BAD_NAMES = [...]

class UserForm(forms.Form):
    name = forms.CharField(label='User name', max_length=

    def clean_name(self):
        name = self.cleaned_data['name']
        if name in BAD_NAMES:
            raise forms.ValidationError(
                'Выберите другое имя'
            )
        return name
```

# ValidationError

- После того как в clean_* методе произошло исключение ValidationError, текст ошибки помещается в словарь `form.errors` под ключиком field-а
- В случае если произошла ошибка и form.errors не пустой то `form.is_valid()` возвращает False

# Виды field-ов формы

- BooleanField
- CharField
- ChoiceField
- DateField
- DateTimeField
- DecimalField

- EmailField
- FileField
- FloatField
- ImageField
- IntegerField
- GenericIPAddressField
- NullBooleanField
- URLField

# Валидаторы

Пример:

```
from django.core.exceptions import ValidationError
from django.db import models

def validate_even(value):
    if value % 2 != 0:
        raise ValidationError('Oooops')

class MyForm(form.Form):
    even_field = forms.IntegerField(validators=[validate_

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

# Виджеты

Виджеты это объекты которые отвечают за то как поля формы будут отображаться в HTML

```
from django import forms

class CommentForm(forms.Form):
    name = forms.CharField()
    url = forms.URLField()
    comment = forms.CharField(widget=forms.Textarea)
```

# Полезные ссылки

- https://docs.djangoproject.com/en/2.0/topics/forms/
- https://docs.djangoproject.com/en/2.0/ref/forms/widgets/
- https://docs.djangoproject.com/en/2.0/ref/validators/

# ModelForm

Классы форм которые конструируются на базе класса модели

# Пример ModelForm

`models.py`

```
class Author(models.Model):
        name = models.CharField(max_length=100)
    birth_date = models.DateField(blank=True, null=True)
```

`forms.py`

```
class AuthorForm(forms.ModelForm):
    class Meta:
        model = Author
        fields = ['name']
        ...
```

# Основные свойства класса Meta

- model(Обязательное свойство) - класс модели на основе которой конструируется форма
- fields - Список полей которые будут включены в форму из модели или `__all__`
- exclude - Список полей которые нужно исключить, удобно использовать если не указали fields, но !опасно! если добавятся еще поля
- widgets - словарь имя_поля: виджет, замена стандартных виджетов
- error_messages - перегрузка стандартных сообщений

# Пример с exclude

```
class PartialAuthorForm(ModelForm):
    class Meta:
        model = Author
        exclude = ['title']
```

# Пример с widgets

```
class AuthorForm(ModelForm):
    class Meta:
        model = Author
        fields = ('name', 'birth_date')
        widgets = {
            'name': Textarea(
                attrs={'cols': 80, 'rows': 20}
            ),
        }
```

# Пример с перегруженными ошибками

```
from django.core.exceptions import NON_FIELD_ERRORS
from django.forms import ModelForm

class ArticleForm(ModelForm):
    class Meta:
        error_messages = {
            NON_FIELD_ERRORS: {
                'unique_together': "%(model_name)s's %(fi
            },
            'title': {
                'max_length': 'Слишком длинный заголовок
            }
        }
```

# Дополнительные поля

- labels

```
labels = {
     'name': 'Имя',
}
```

- help_texts

```
help_texts = {
    'name': 'Это подсказка така информативная',
}
```

# Метод save() у формы

Метод `save()` создает и сохраняет объект в базу. Если объект был передан в форму через `instance=` в конструкторе, то форма конструируется на основе переданого объекта и обновляется в случае сохранения формы

# Полезные ссылки

- https://docs.djangoproject.com/en/2.0/topics/forms/modelforms/