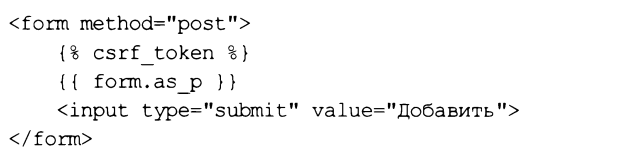
**Тема занятия № 26: Модуль 13. Формы связанные с моделями**

1. Вывод форм на экран

**Быстрый вывод форм**

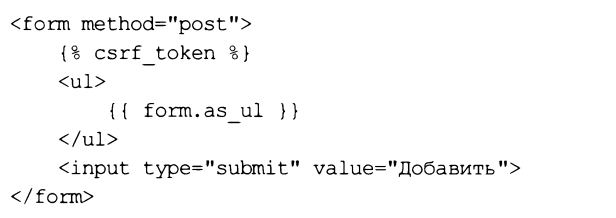
Быстрый вывод форм осуществляется вызовом одного метода из трех, поддерживаемых Django:

□ as\_p() — вывод по абзацам. Надпись для элемента управления и сам элемент управления, представляющий какое-либо поле формы, выводятся в отдельном абзаце и отделяются друг от друга пробелами. Пример использования:

****

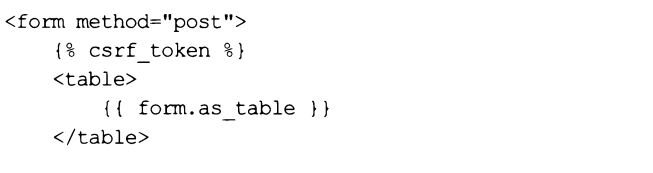
□ as ui () — вывод в виде маркированного списка. Надпись и элемент управления выводятся в отдельном пункте списка и отделяются друг от друга пробелами.

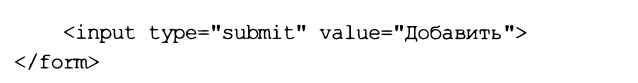
Теги <ui> и </ui> не формируются. Пример:

****

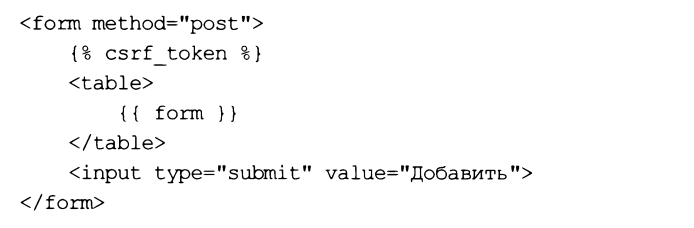
□ as table () — вывод в виде таблицы из двух столбцов: в левом выводятся надписи, в правом — элементы управления. Каждая пара “надпись—элемент управления” занимает отдельную строку таблицы. Теги <tabie> и </tabie> не выводятся.

Пример:





Можно просто указать переменную, хранящую форму, — метод as tableo будет вызван автоматически:

****

Обязательно уясним следующие моменты:

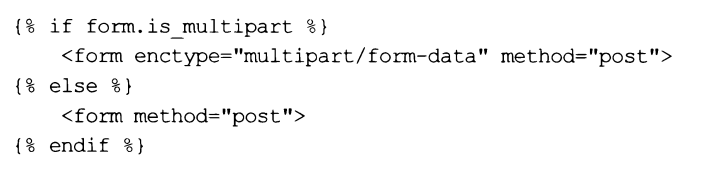
□ тег <form>, создающий саму форму, не выводится в любом случае. Его придется вставить в код шаблона самостоятельно;

□ кнопка отправки данных также не выводится, и ее тоже следует поместить в форму самостоятельно. Такая кнопка формируется одинарным тегом <input> с атрибутом type, значение которого равно "submit";

□ не забываем поместить в форму тег шаблонизатора csrf token. Он создаст в форме скрытое поле с электронным жетоном, по которому Django проверит, пришел ли запрос с того же самого сайта. Это сделано ради безопасности.

По умолчанию в веб-формах применяется метод кодирования данных Appiication/x-www-form-uriencoded. Но если форма отправляет файлы, то в ней нужно указать метод multipart/form-data.

Выяснить, какой метод следует указать в теге <form>, поможет метод Is muitipartо, поддерживаемый формой. Он возвращает True, если форма содержит поля, предназначенные для хранения файлов. И, соответственно, требует указания метода multipart/form-data, И False — в противном случае. Пример:



**Расширенный вывод форм**

Django предоставляет инструменты расширенного вывода форм, позволяющие располагать отдельные элементы управления произвольно.

Прежде всего экземпляр класса modeiform, представляющий связанную с моделью форму, поддерживает функциональность словаря. Ключи элементов этого словаря совпадают с именами полей формы, а значениями элементов являются экземпляры класса boundfieid, которые представляют отдельные поля формы.

Если указать в директиве шаблонизатора непосредственно экземпляр класса Boundfieid, то он будет выведен как HTML-код, создающий элемент управления для текущего поля. Вот так можно вывести область редактирования для ввода описания товара:



В результате мы получим такой HTML-код:



Еще класс boundfieid поддерживает следующие атрибуты:

□ label tag — HTML-код, создающий надпись для элемента управления, включая тег <1аье1>. Вот пример вывода кода, создающего надпись для области редактирования, в которую заносится описание товара:



Результирующий HTML-код:



□ label —только текст надписи;

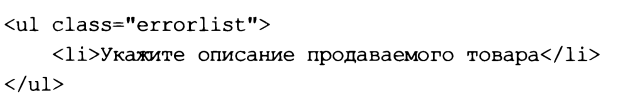
□ help text — дополнительный поясняющий текст;

□ errors —список сообщений об ошибках, относящихся к текущему полю.

Список ошибок можно вывести в шаблоне непосредственно:



В этом случае будет сформирован маркированный список с привязанным стилевым классом errorlist, а отдельные ошибки будут выведены как пункты этого списка, например:



Также можно перебрать список ошибок в цикле и вывести отдельные его элементы с применением любых других HTML-тегов.

Чтобы получить список сообщений об ошибках, относящихся ко всей форме, следует вызвать метод non field errors () формы:



□ is hidden — True, если это скрытое поле, False, если какой-либо иной элемент управления.

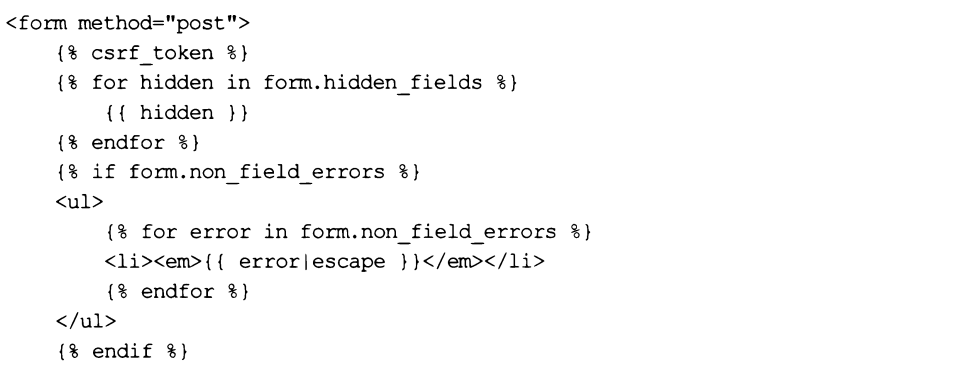
Методы класса modelform:

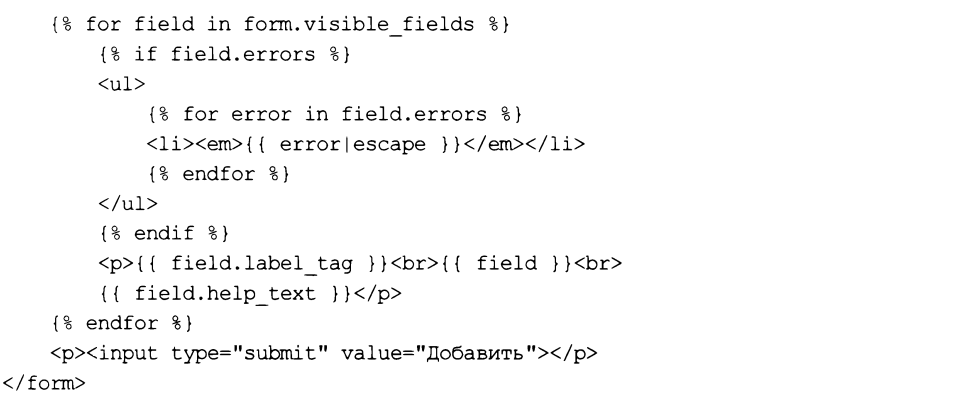
□ visibie fieids о — возвращает список видимых полей, которые представляются на экране обычными элементами управления;

□ hidden fieids о — возвращает список невидимых полей, представляющихся скрытыми полями HTML.

Приведен код, создающий форму для добавления объявления, в которой сообщения об ошибках выводятся курсивом; надписи, элементы управления и поясняющие тексты разделяются разрывом строки (HTML-тегом <br>); а невидимые поля (если таковые есть) выводятся отдельно от видимых. Сама форма показана на рис. 13.1.

Рис. 13.1. Веб-форма:

****

****

1. Валидация в формах

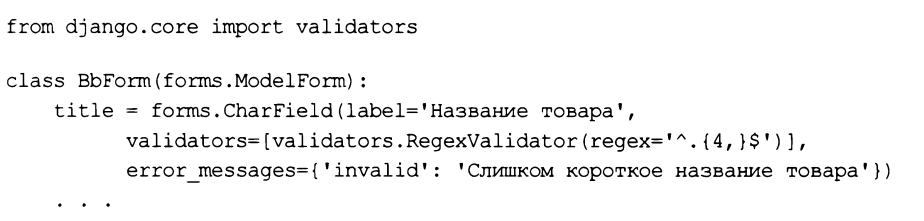
Валидацию можно выполнять не только в модели, но и в формах, связанных с моделями, применяя аналогичные инструменты.

**Валидация полей формы**

Валидацию отдельных полей формы можно реализовать двумя способами: с применением валидаторов или путем переопределения методов формы.

**Валидация с применением валидаторов**

Валидация с помощью валидаторов в полях формы выполняется так же, как и в полях модели. Вот пример проверки, содержит ли название товара больше четырех символов, с выводом собственного сообщения об ошибке:

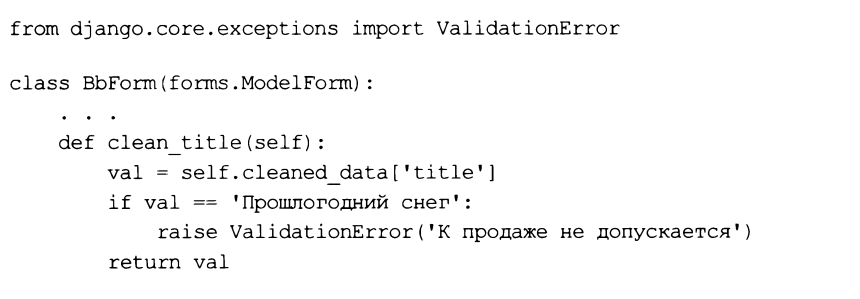


Разумеется, мы можем использовать не только стандартные валидаторы, объявленные в модуле django.core.validators, НО И СВОИ собственные.

**Валидация путем переопределения методов формы**

Более сложная валидация значения какого-либо поля реализуется в классе формы, в переопределенном методе с именем вида с1еап\_<имя поля>(self). Этот метод должен получать значение поля из словаря, хранящегося в атрибуте cieaned data, не должен принимать параметров и всегда обязан возвращать значение проверяемого поля. Если значение не проходит валидацию, то в методе следует возбудить исключение validationerror.

Вот пример проверки, не собирается ли посетитель выставить на продажу прошлогодний снег:

** Валидация формы**

Выполнить более сложную проверку или проверить значения сразу нескольких полей формы, т. Е. Провести валидацию формы, можно в переопределенном методе Clean (self) класса формы.

Метод не должен принимать параметров, возвращать результата, а обязан при неудачной валидации возбудить исключение validationerror, чтобы указать на возникшую ошибку. Первым же действием он должен вызвать одноименный метод базового класса, чтобы он заполнил словарь, хранящийся в атрибуте cieaned data (если мы этого не сделаем, то не сможем получить данные, занесенные в форму).

Далее приведен пример реализации такой же проверки, описание товара должно быть занесено, а значение цены должно быть неотрицательным.

