**Lecture 7. Авторизация и аутентификация Django**

**Расширение возможностей для администрирования сайта «мир книг» и создание пользовательских форм**

В предыдущих главах мы создали сайт «Мир КНИГ», который позволяет пользова­телям получать из каталога списки книг и авторов. И сейчас каждый посетитель сайта имеет доступ к одним и тем же страницам и типам информации из базы дан­ных.

Однако хотелось бы предоставить пользователю индивидуальные услуги, которые зависят от его предпочтений и предыдущего опыта использования сайта и его на­строек. Например, возможность добавлять на страницы сайта свою информацию, что актуально для интернет-магазинов, социальных сетей, 6логов и прочих подоб­ных ресурсов.

В этой главе мы расширим возможности нашего сайта «Мир КНИГ», добавив в него счетчик посещений домашней страницы, организованный на основе сессий. Этот относительно простой пример покажет, как с помощью сессий реализовать анализ поведения на сайте анонимных пользователей.

Мы также организуем на сайте «Мир КНИГ» поддержку пользовательских аккаунтов и таким образом продемонстрируем аутентификацию, разграничение уровней дос­тупа, сессии и пользовательские формы.

В этой главе будут рассмотрены следующие вопросы:

* что такое сессии в Django;
* создание страниц для авторизации пользователей в Django;
* организация проверки подлинности входа пользователя в систему;
* добавление к сайту страницы для создания заказов на книги;
* что такое формы и управление формами в Django;
* как создать форму для ввода и обновления информации на сайте на основе клас­са Form();
* как создать форму для ввода и обновления информации на сайте на основе клас­са ModelForm().

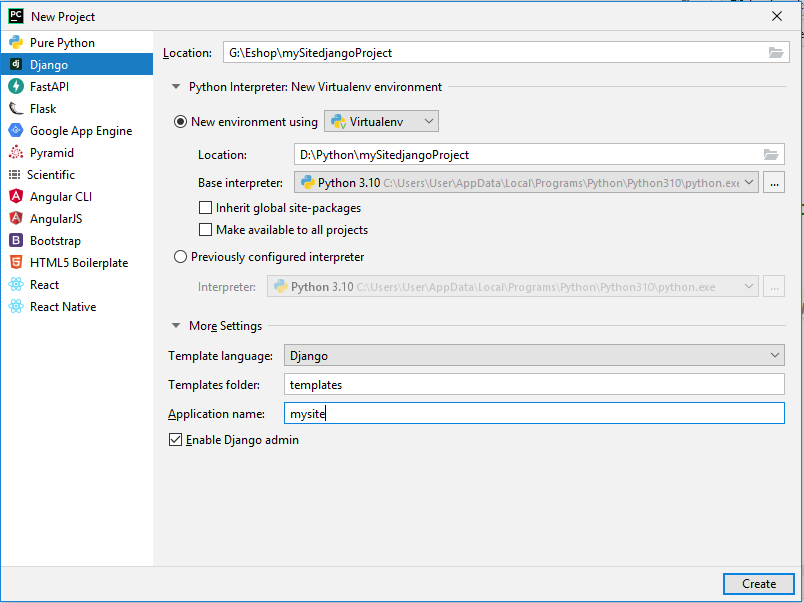
**Сессии в Django**

Сессии позволяют хранить и получать произвольные данные, запрошенные на ос­нове индивидуального поведения пользователя на сайте. Разберемся, что же такое сессии.

Все взаимодействия между браузерами пользователей и серверами осуществляются при помощи протокола НТТР, который не сохраняет состояние таких взаимодейст­вий (stateless). Это означает, что сообщения между клиентом и сервером являются полностью независимыми друг от друга — т. е. не существует какого-либо пред­ставления «последовательности» или поведения системы в зависимости от преды­дущих сеансов взаимодействия. Так что если вы хотите создать сайт, который будет отслеживать взаимодействия с клиентом, то вам нужно реализовать это само­стоятельно.

Сессии представляют собой механизм, который использует Django (как и весь остальной Интернет) для отслеживания состояния взаимодействий между сайтом и каким-либо браузером. Сессии позволяют хранить произвольные данные браузера и извлекать их в тот момент, когда между браузером и сайтом устанавливается соединение. Данные получаются и сохраняются в сессии при помощи соответст­вующего «ключа».

Django для этого использует файлы cookie. Они содержат специальный идентифи­катор сессии, который выделяет каждый браузер в соответствующую сессию. Ре­альные данные сессии по умолчанию хранятся в базе данных сайта. Впрочем, у вас есть возможность настроить Django так, чтобы сохранять данные сессий в других местах (кэше, файлах, «безопасных» cookie). Но все же хранение по умолчанию является хорошей и безопасной возможностью.



Сессии становятся доступны автоматически в тот момент, когда вы создаете «ске­лет» сайта. Необходимые настройки конфигурации выполняются в разделах installed\_apps и middleware файла проекта WebBooks\WebBooks\settings.py (рис. 10.1).

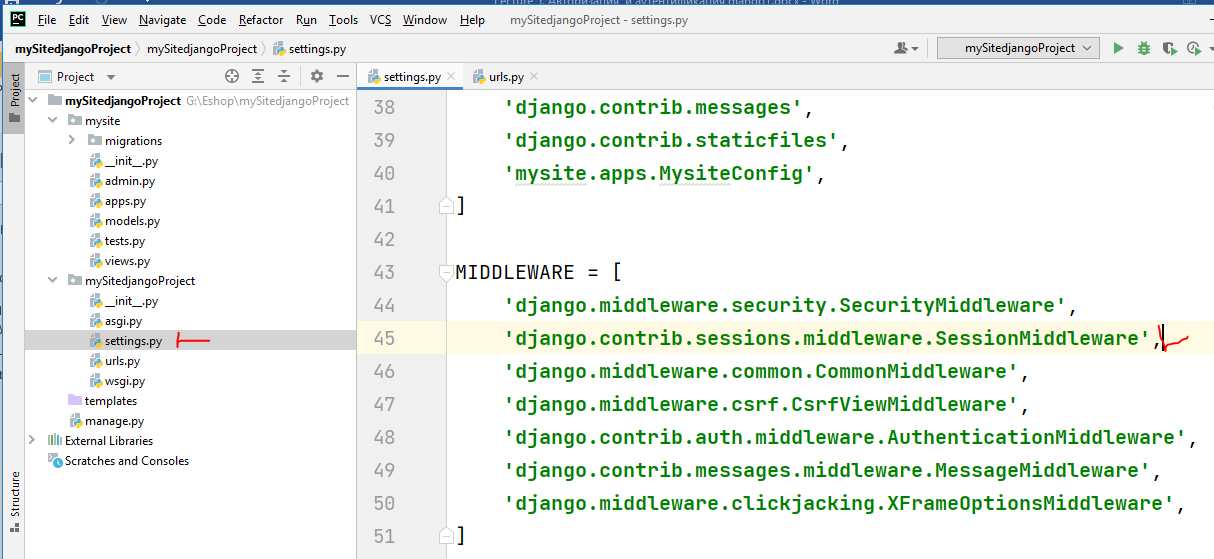


Рис. 10.1. Параметры настройки авторизации пользователей в файле settings.py

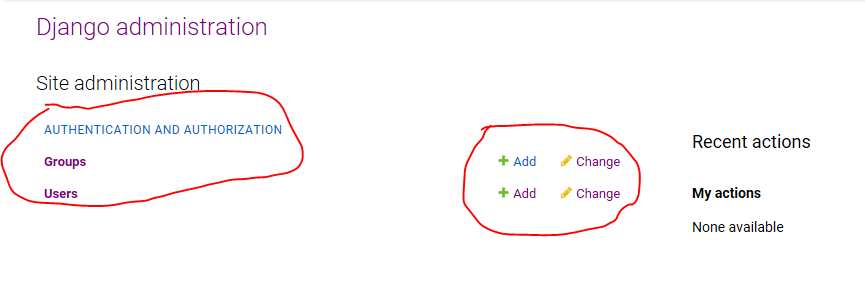
Доступ к переменной session можно получить в соответствующем представлении через параметр request (HttpRequest передается как первый аргумент в каждое пред­ставление). Переменная сессии представляет собой связь с определенным пользо­вателем (если быть более точным — с определенным браузером пользователя), который определяется при помощи идентификатора (id) сессии, получаемого из файла cookie браузера.

Переменная (или поле) session является объектом-словарем, в который можно де­лать записи неограниченное число раз. С ним вы можете выполнять любые стан­дартные операции: чтение, очистку всех данных, проверку наличия ключа, циклы по данным и т. п. Далее представлены фрагменты кода, которые показывают, как получать, задавать и удалять некоторые данные при помощи ключа my\_car, связан­ного с текущей сессией (браузером):

|  |
| --- |
| **from** django.contrib.auth.models  **import** User *# Создать пользователя и сохранить его в базе данных* user = User.objects.create\_user(**'myusername'**, **'myemail@crazymail.com'**, **'mypassword'**)  *# Обновить поля и сохранить их снова* user.firstnаmе = **Alfiya** user.lastname = **Mingazova** user.save () |

Тем не менее мы создадим группу, а затем пользователя в этой группе с помощью административной панели сайта. Несмотря на то что у нас пока нет никаких разрешений на добавление к нашей библиотеке каких-либо читателей, если мы захотим это сделать в будущем, то будет намного проще добавлять их к уже созданной группе <;: заданной аутентификацией. И административная панель сайта предоставляет наиболее быстрый способ создания соответствующих групп и аккаунтов.

Итак, запускаем сервер разработки и переходим к административной панели сайта (bttp://127.0.0.1:8000/admin/). Заходим на сайт при помощи параметров аккаунта суперпользователя (имени пользователя и пароля). Самая верхняя страница панели администратора показывает все наши модели. Для того чтобы увидеть записи в разделах Authentication и Autborisation, вы можете нажать на ссылку Группы или Пользователи в разделе ПОЛЬЗОВАТЕЛИ И ГРУППЫ (рис. 10.4).



**Рис. 10.4. Разделы административной панели сайта для создания групп и пользователей**

В первую очередь давайте создадим новую группу -нажмите для этого на ссылку +Добавить в строке Группы. Введите для нее имя: Пользователи сайта (рис. 10.5). Для этой группы не нужны какие-либо разрешения, поэтому просто нажмите кнопку СОХРАНИТЬ, и вы перейдете к списку групп.

Теперь давайте создадим пользователя. Для этого возвращаемся на главную страницу административной панели и нажимаем ссылку + Добавить в строке Пользователи, после чего откроется окно со списком пользователей (рис. 10.6).

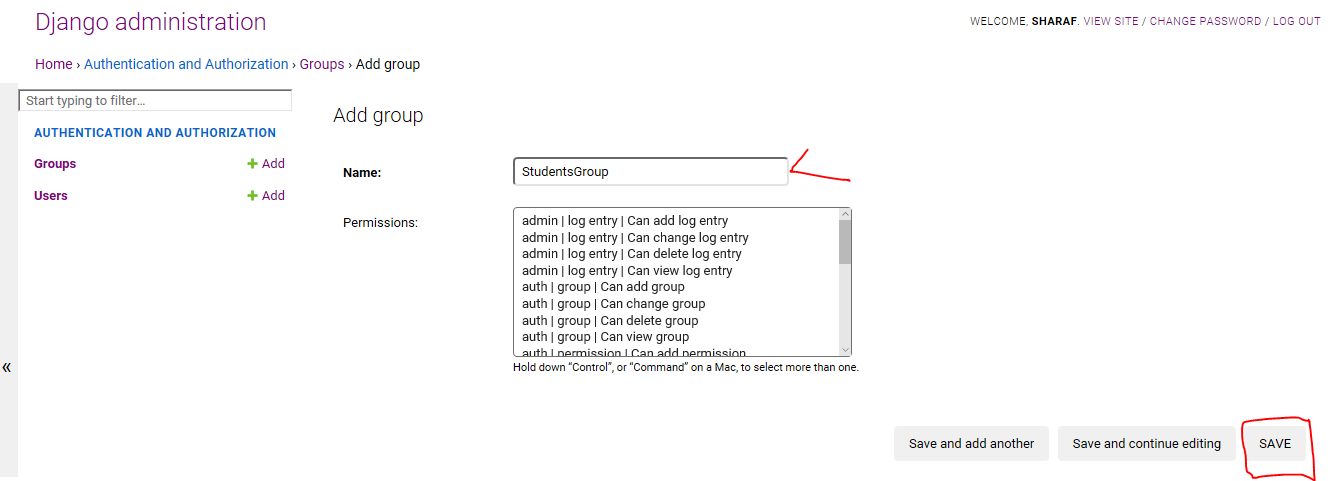
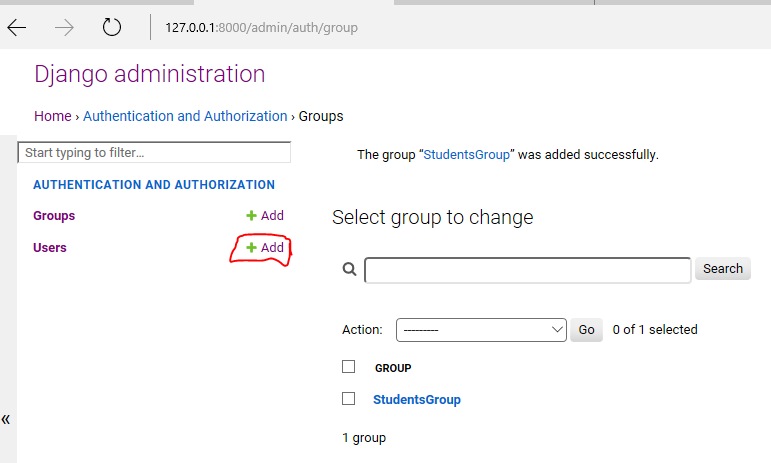
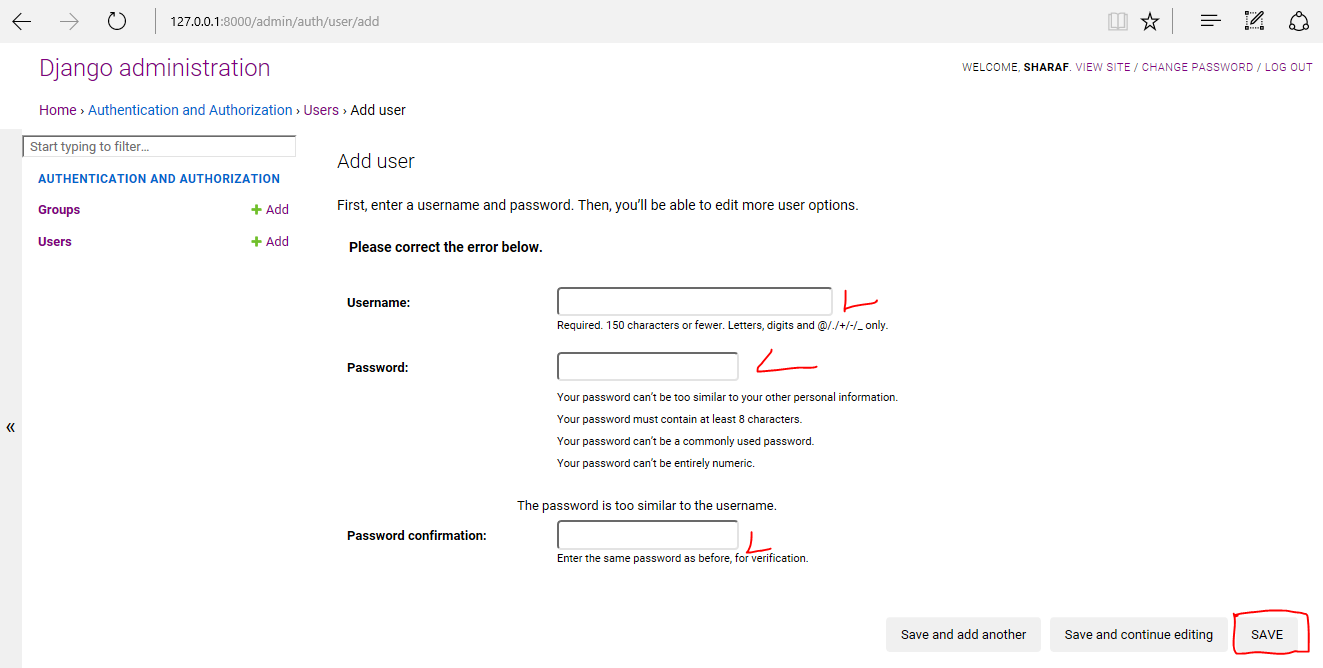


Рис. 10.5. Создание группы пользователей в административной панели сайта



**Рис. 10.6. Окно со списком пользователей в административной панели сайта**

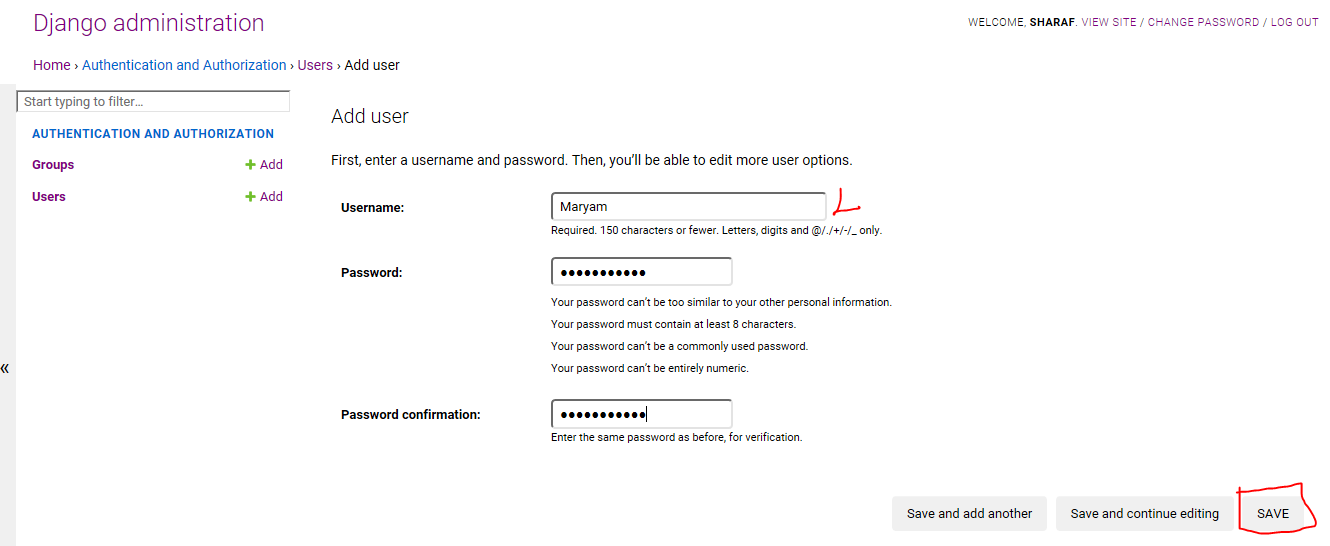
В этом окно нажмите на кнопку ДОБАВИТЬ ПОЛЬЗОВАТЕЛЯ+. Будет выполнен переход к диалоговому окну ввода сведений о новом пользователей (Рис 10.7). Введите здесь в полях Имя пользователя, Пароль и Подтверждение пароля соответствующие данные для нашего тестового пользователя и нажмите кнопку СОХРАНИТЬ для завершения процесса создания пользователя.



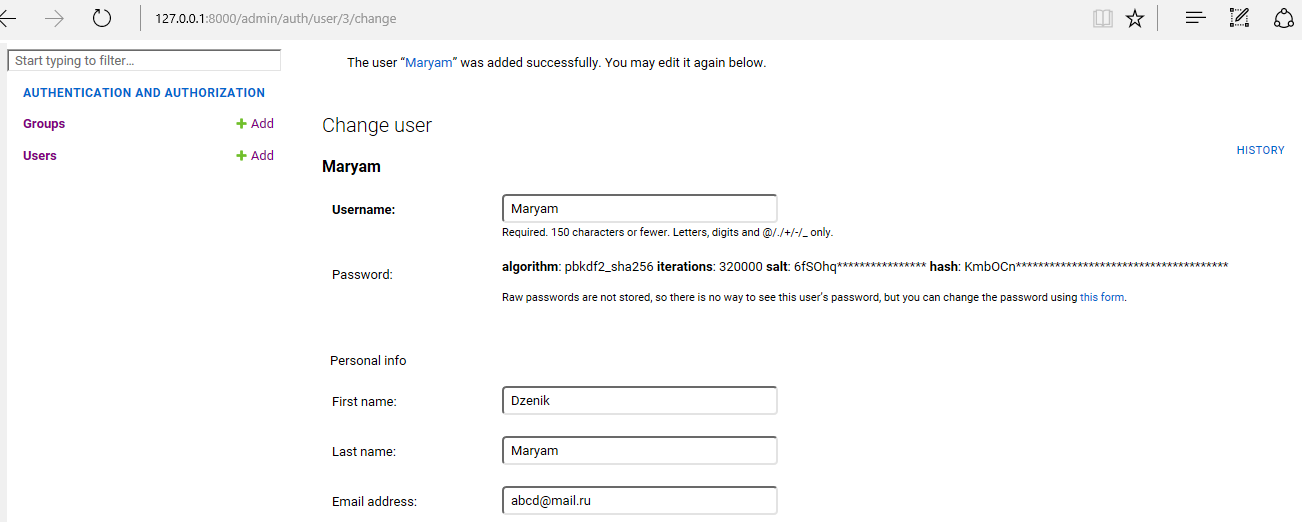
Административная часть сайта создаст нового пользователя и немедленно перенаправит вас на страницу изменения параметров пользователя (рис. 10.8), где вы сможете ввести для него некоторую персональную информацию:

* изменить имя пользователя (входное имя на сайт);
* добавить имя пользователя (по паспорту);
* добавить фамилию пользователя;
* ввести его электронный адрес.

Как можно видеть, мы задали пользователю имя: Maryam. Если прокрутить страницу с параметрами пользователя вниз, то можно увидеть блок, в котором можно задать для него права доступа (рис. 10.9).



После нажатие на кнопку Сохранить откроется следующее окно.



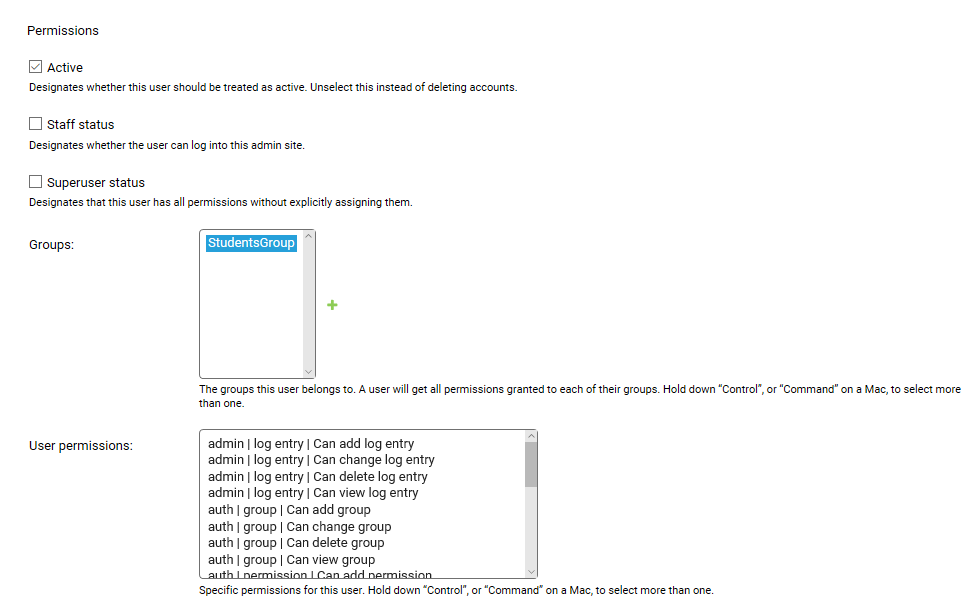


Рис. 10.9. Блок задания прав доступа пользователей в административной панели сайта

В этом блоке можно определить статус пользователя и признак активности (может быть установлен только соответствующий флаг). Кроме того, можно определить группу для пользователя и необходимые параметры доступа.

В самой нижней части страницы изменения параметров пользователя имеется блок Важные даты, в котором можно внести даты, относящиеся к этому пользователю (дату подключения к сайту и дату последнего входа), а также расположена кнопочная панель для сохранения введенных изменений и перехода на другие страницы (рис. 10.10).



Вот и все! Теперь у нас есть учетная запись «обычного пользователя» сайта, которую можно использовать для тестирования работы сайта (после того, как мы добавим страницы входа пользователя в систему).

**Создание страницы регистрации пользователя при входе на сайт Django предоставляет почти все, что нужно для**

Django предоставляет почти все, что нужно для создания страниц аутентификации входа, выхода из системы и управления паролями. Этот набор включает в себя URL-адpeca, представления (views) и формы, но не включает шаблоны (придется создать свой собственный шаблон самостоятельно).

**Примечание.** Мы можем поместить страницы аутентификации, включая URL-адреод и шаблоны, в папку catalog нашего единственного приложения. Однако при наличии нескольких приложений лучше отделить от него вход в систему и обеспечить доступность страни­цы авторизации пользователей для всего сайта. Именно так мы и поступим.

Начнем с создания ссылок на URL-aдpeca. Добавьте в нижнюю часть файла проек­та WebBooks\WebBooks\urls.py следующий код (листинг 10.3).

Листинг 10.3

|  |
| --- |
| **from** django.urls **import** path, include urlpatterns = [  path(**'accounts/'**, include(**'django.contrib.auth.urls'**)), ] |

Следующий шаг- создайте каталог (папку) registration в папке templates (рис. 10.11).

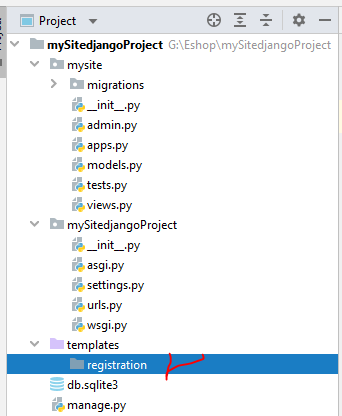


Рис.10.11. Создание каталога registration в папке templates

Затем в эту папку добавьте шаблон — файл login.html (листинг 10.4).

Листинг 0.4

|  |
| --- |
| {% **extends "base.html"** %} {% **block content** %}  {% **if form**.**errors** %}  <**p**>Ваш имя пользователя и пароль не совпали. Пожалуйста,  попробуйте еще раз.</**p**>  {% **endif** %}  {% **if next** %}  {% **if user**.**is\_authenticated** %}  <**р**> Вы не имеете доступа к этой странице. Войдите в систему  с другими параметрами  </**р**>  {% **else** %}  <**p**> Войдите в систему, чтобы увидеть эту страницу.</**p**>  {% **endif** %}  {% **endif** %}   <**form method="post" action="**{% **url 'login'** %}**"**>   {% **csrf\_token** %}  <**table**>  <**tr**>  <**td**>{{ **form**.**username**.**label\_tag** }}</**td**>  <**td**>{{ **form**.**username** }}</**td**>  </**tr**>  <**tr**>  <**td**>{{ **form**.**password**.**label\_tag** }}</**td**>  <**td**>{{ **form**.**password** }}</**td**>  </**tr**>  </**table**>  <**input type="submit" value="Bxoд"**/>  <**input type="hidden" name="next" value="**{{ **next** }}**"**/>   </**form**>   *{# Предполагается, что вы настроили представление password\_reset в своем URLconf#}* <**p**><**a href="**{% **url 'password\_reset'** %}**"**> Утерян пароль? </**a**></**p**> {% **endblock** %} |

Этот шаблон имеет сходство с теми шаблонами, что мы создавали раньше. Он рас­ширяет наш базовый шаблон и переопределяет блок контента. Остальная часть кода — это довольно-таки стандартный код обработки формы, о котором мы пого­ворим в следующем разделе. Все, что вам нужно сделать, — это показать форму, в которую можно ввести имя пользователя и пароль, а если будут введены недопус­тимые значения, предложить (когда страница обновится) ввести правильные.

Чтобы сделать наш шаблон доступным, откроем настройки проекта (файл settings.py) и обновим в секции TEMPLATES (рис. 10.12) строку 'dirs':

**'DIRS'**: [BASE\_DIR / **'templates'**]

Если теперь перейти по адресу: <http://127.0.0.1:8000/accounts/login/>, то вы должны увидеть страницу входа с полями идентификации пользователя (рис. 10.13).

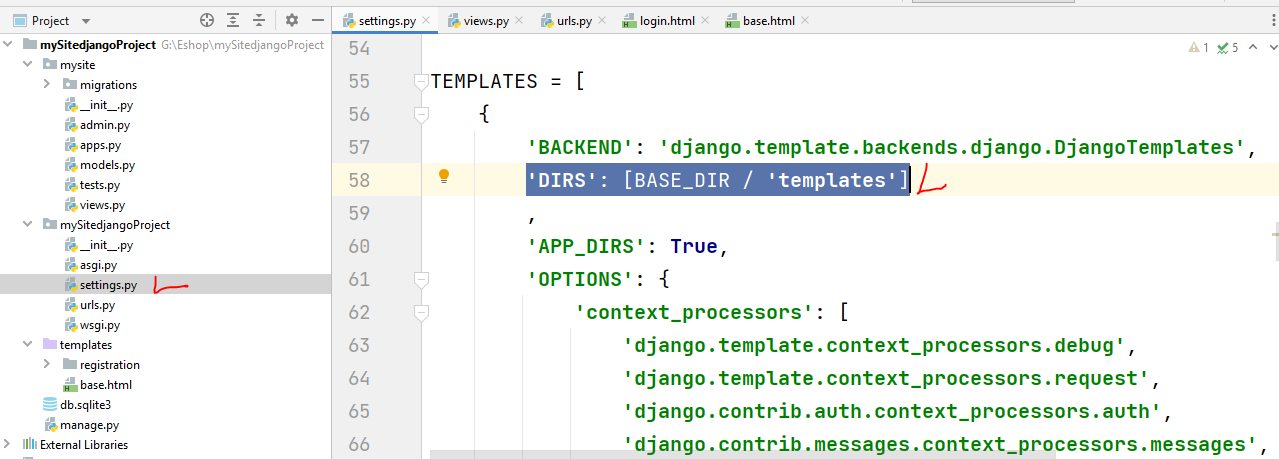


Рис. 10.12. Обновление в секции TEМPLATES сrроки ' DIRS'

Введите здесь входные данные пользователя, и вы будете перенаправлены на дру­гую страницу сайта — по умолчанию это страница профиля пользователя по адре­су: <http://127.0.0.1:8000/accounts/profile/>. Но поскольку страницу с профилем мы еще не сделали, то вам будет выдано сообщение об ошибке с информацией, что страница profile отсутствует (рис. 10.14).

Ликвидируем этот недочет. Откройте настройки проекта (файл settings.py) и добавь­те в конец файла следующий код (листинг 10.5).

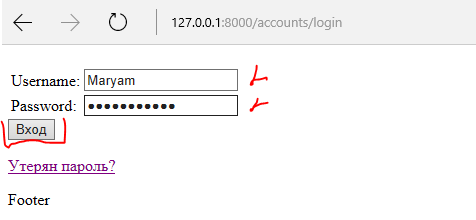


Рис. 10.13. Страница входа с идентификацией пользователя

Листинг 10.5

|  |
| --- |
| *# Переадресация на главную страницу сайта после входа в систему* LOGIN\_REDIRECT\_URL=**"/"** |

Теперь, если повторить вход в систему, то ошибка больше выдаваться не будет, и пользователь будет успешно перенаправлен на домашнюю страницу сайта. Если же вы перейдете по URL-aдpecy выхода: http://127.0.0.1:8000/accounts/logout/, то увидите, что пользователь выведен из системы, но это по умолчанию окно панели администратора сайта (рис. 10.15).

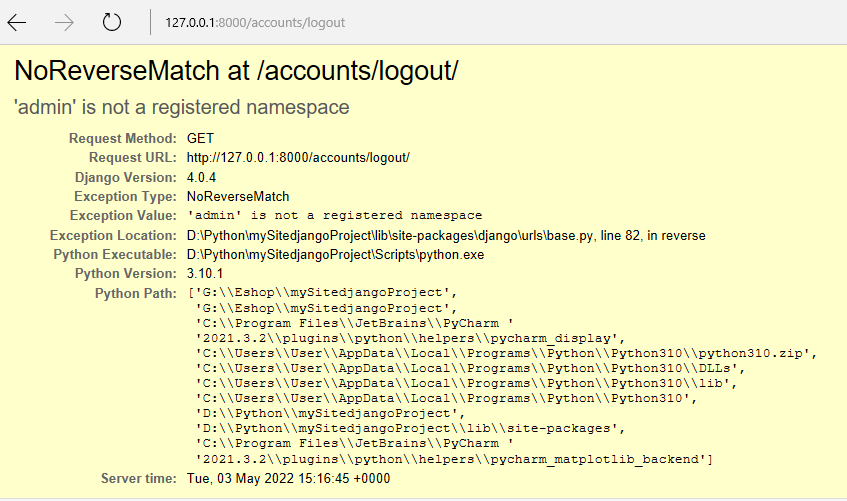


Рис. 10.14. Сообщение об отсутствии страницы profile

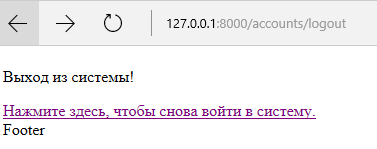


Рис. 10.15. Окно с сообщением об успешном выходе из системы панели администратора сайта

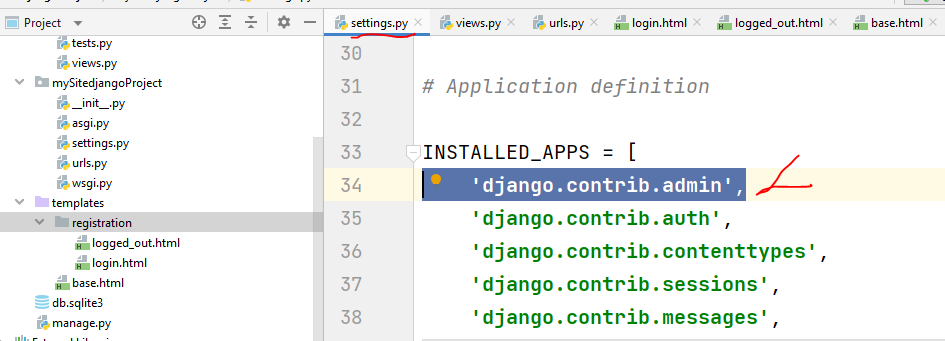
Тем не менее это не то, что нам нужно (такое окно должны получать только пользователи, у которых есть статус is\_staff, - т. е. персонал, обслуживающий сайт). Чтобы этого не происходило, создадим шаблон страницы выхода из системы (файл templates\registration\logged\_out.html) и внесем в этот файл следующий код (листинг 10.6).

Листинг 10.6

|  |
| --- |
| {% **extends "base.html"** %} {% **block content** %}  <**p**>Выход из системы!</**p**>  <**a href="**{% **url 'login'** %}**"**> Нажмите здесь, чтобы снова войти в систему.</**a**> {% **endblock** %} |

Этот шаблон очень прост. Он показывает сообщение о том, что вы вышли из сис­темы, и предоставляет ссьшку, которую вы можете нажать, чтобы вернуться на экран входа в систему.

Попробуем снова перейти на страницу выхода из системы. Если вы опять видите страницу выхода из системы панели администратора сайта (см. рис. 10.15) вместо своей собственной страницы выхода, то пугаться не стоит. Проверьте настройку INSTALLED\_APPS вашего проекта в файле settings.py и убедитесь, что строка: **'django.contrib.admin'** стоит на первом строке (рис. 10.16).



Если все было сделано правильно, то при запросе по адресу: http:// 127.0.0.1:8000/accounts/logout/ будет загружен страница на основе вашего шаблона logged\_out.html (рис. 10.17).

Рис. 10.17. Страница успешного выхода из системы на основе шаблона logged\_out.html

**Создание страницы для сброса пароля пользователя**

Довольно часто интернет-пользователи забывают свой пароль входа на тот или иной сайт. Для таких забывчивых пользователей необходимо предусмотреть воз­можность восстановления доступа к своему аккаунту путем смены пароля. Система сброса пароля по умолчанию использует электронную почту, чтобы отправить пользователю ссылку на сброс. Для этого на сайте необходимо создать форму, что­бы получить адрес электронной почты пользователя, отправить ему электронное письмо, разрешить ввести новый пароль и сделать отметку о завершении процесса.

Для этих целей создадим в каталоге \templates\registration\ новую форму (шаблон) password\_reset\_form.html (рис. 10.18).

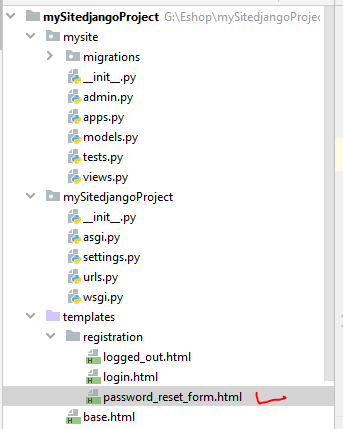


Рис. 10.18. Создание файла шаблона (формы) для сброса пароля пользователя

В созданный файл внесем следующий код (листинг 10.7).

|  |
| --- |
| {% **extends "base.html"** %} {% **block content** %}  <**form action="" method="post"**>  {% **csrf\_token** %}  {% **if form**.**email**.**errors** %}  {{ **fom**.**email**.**errors** }}  {% **endif** %}  <**label**>Baш email</**label**><**br**>  <**p**>{{ **form**.**email** }}</**p**>  <**input type="submit" class="btn btn-default btn-lg" value="Cбpoc пароля"**/>  </**form**> {% **endblock** %} |

Затем создадим в каталоге \templates\registration\ форму password\_reset\_done.html (рис. 10.19), которая будет показана пользователь после того, как он введет адрес электронной почты, и внесем в нее следующий код (листинг 10.8).

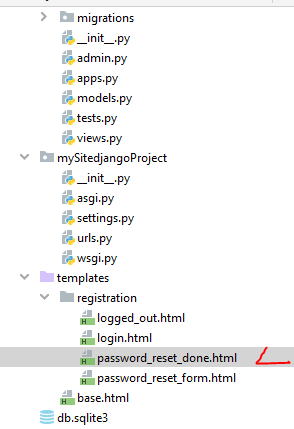


Рис. 10.19. Создание файла шаблона (формы) для смены пароля пользователя

Листинг 10.8

|  |
| --- |
| {% **extends "base.html"** %} {% **block content** %}  <**p**>Мы отправили вам по электронной почте инструкции по установке пароля. Если они не получены в течение нескольких  минут, то проверьте папку со спамом.  </**p**> {% **endblock** %} |

Следующий шаблон предоставляет собой текст письма, которое мы отправим пользователю по электронной почте, - в нем содержится ссылка на сброс пароля. Для него в том же каталоге \templates\registration\ создадим файл password\_reset\_email.html и внесем в него следующий код (листинг 10.9).

**Листинг 10.9**

|  |
| --- |
| Для смены пароля пользователя с электронной почтой {{ **email** }} перейдите по ссылке ниже: {{ **protocol**}}://{{ **domain** }}{% **url 'password\_reset\_confirm' uidb64**=**uid token**=**token** %} |

И это еще не все - теперь необходимо дать пользователю возможность ввести новый пароль. По нажатию на ссылку в полученном электронном письме пользователь должен попасть на страницу сайта, .где сможет это сделать. Чтобы предоставить ему такую возможность, в том же каталоге \templates\registration\ создадим шаблон password\_reset\_confirm.html и внесем в него следующий код (листинг 10.10).

|  |
| --- |
| {% **extends "base.html"** %} {% **block content** %}  {% **if validlink** %}  <**p**>Пожалуйста, введите (и подтвердите) свой новый пароль.</**p**>  <**form action="" method="post"**>  {% **csrf\_token** %}  <**table**>  <**tr**>  <**td**>{{ **form**.**password1**.**errors** }}  <**label for="id\_new\_password1"**> Новый пapoль:</**label**></**td**>  <**td**>{{ **form**.**new\_password1** }}</**td**>  </**tr**>  <**tr**>  <**td**>{{ **form**.**new\_password2**.**errors** }}  <**label for="id\_new\_password2"**> Подтвердите пapoль:</**label**></**td**>  <**td**>{{ **form**.**new\_password2** }}</**td**>  </**tr**>  <**tr**>  <**td**></**td**>  <**td**><**input type="submit" value=" Сменить пароль "**/></**td**>  </**tr**>  </**table**>  </**form**>  {% **else** %}  <**hl**> Ошибка при смене пароля!!!</**hl**>  <**p**> Ссылка на сброс пароля была недействительной, возможно, потому, что она уже использовалась. Пожалуйста,  запросите новый сброс пароля.</**p**>  {% **endif** %} {% **endblock** %} |

Ну и, наконец, создадим в том же каталоге \templates\registration\ последний шаблон password\_reset\_complete.html, с помощью которого уведомим пользователя об успеш­ном завершении смены пароля, и внесем в него следующий код (листинг 10.11).

|  |
| --- |
| {% **extends "base.html"** %} {% **block content** %}  <**h1**>Пароль был успешно иэменен!</**h1**>  <**p**>  <**a href="**{% **url 'login'** %}**"**>Повторить вход в систему?</**a**>  </**p**> {% **endblock** %} |

Теперь вы сможете проверить функцию смены пароля по ссылке на странице входа в систему.

Примечание

Имейте в виду, что Django отправляет электронные письма для смены пароля только на те адреса пользователей, которые хранятся в базе данных в его аккаунте!

Следует также отметить, что система смены пароля требует, чтобы ваш сайт под­держивал электронную почту, описание организации которой выходит за рамки книги. Так что эта часть сайта пока еще не будет работать полноценно, однако раз­работчик имеет возможность протестировать работу всех созданных шаблонов страниц.

Чтобы разрешить их тестирование, поместите в конец файла settings.py следующую строку (листинг 10.12).

Листинг 10.12

|  |
| --- |
| EMAIL\_BACKEND = **'django.core.mail.backends.console.EmailBackend'** |

Это обеспечит просмотр любых писем, отправлеЮ1ых на консоль (чтобы вы могли скопировать ссылку на сброс пароля с консоли). Получить дополнительную ин­формацию об отправке с сайта писем пользователям по электронной почте можно в оригинальной документации на Django.

Проверим работу созданных нами шаблонов и форм. Для этого запустим наш сайт и войдем на страницу входа пользователя: http://127.0.0.1:8000/accounts/login/ (рис. 10.20).

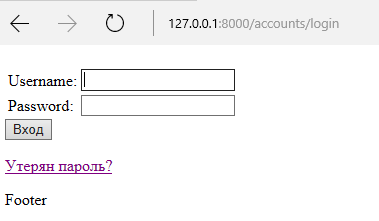


Рис. 10.20. Страница входа в аккаунт пользователя

Щелкнем на этой странице на ссылке Утерян пароль? - вам будет выдана страница с предложением ввести свой электронный адрес для смены пароля (рис. 10.21).

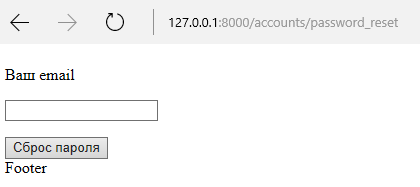


Рис. 10.21. Страница ввода электронного адресе пользователь, для смены пароль

Введем электронный адрес пользователя и нажмем на кнопку Сброс пароли появится страница с сообщением, что пользователь на его электронный адрес отправлено письмо с инструкциями по смене пароля (рис. 10.22).

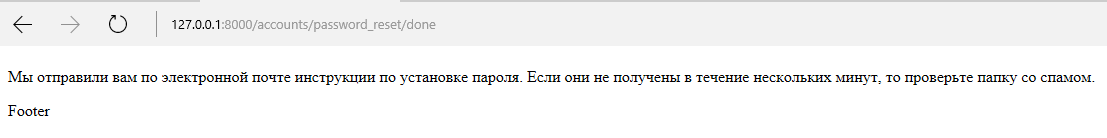


Рис. 10.22. Страница, сообщающая пользователю об отправке ему инструкции по смене пароля

Поскольку наш сайт еще не настроен на отправку электронной почты с внешнего сервера на реальный адрес, то для тестирования работы приложения инструкцию о смене пароля отправьте в окно терминала PyCharm (рис. 10.23).



Рис. 10.23. Инструкция для смены пароля, отправленная пользователю по электронной почте

Щелкните мышью на ссылке, представленной в полученной пользователем инструкции в окне терминала PyCharm, - вы вернетесь на наш сайт, где будет загружена страница для ввода нового пароля (рис. 10.24).

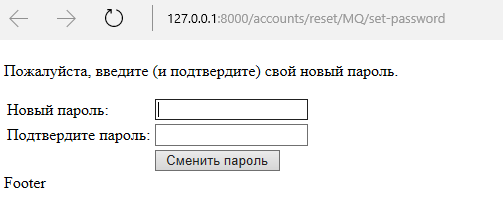


Рис. 10.24. Страница для смены пароля пользователя

На этой странице введем новый пароль, его подтверждение и нажмем ан кнопку сменить пароль. Если все было сделано правильно, то откроется окно с сообщением об успешной смене пароля (рис. 10.25). Нажмите в этом окне на ссылку повторить вход в систему? и вы вернетесь на главной странице

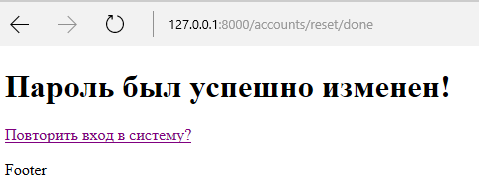


Рис. 10.25. Страница с сообщением об успешной смене пароля

В базе данных пароли пользователей хранятся в зашифрованном виде, и даже глав­ный администратор сайта не может их знать. В этом можно убедиться, открыв с помощью менеджера SQLite таблицу БД с аккаунтами пользователей.

**Поверка подлинности входа пользователя в систему**

При желании можно получить информацию о текущем статусе зарегистрированных в системе пользователей. Для этого в шаблонах присутствует переменная { {user}} (она добавляется в контекст шаблона по умолчанию при настройке проекта).

Обычно сначала проверяется переменная шаблона {{user.is\_authenticated}}, позво­ляющая определить, имеет ли пользователь право видеть конкретный контент. Что­бы продемонстрировать это, мы обновим нашу боковую панель и покажем в ней ссьшку Вход (если пользователь вышел из системы) и ссылку Выход (если он вошел в систему).

Откройте базовый шаблон \catalog\templates\base.html и добавьте следующий текст (листинг 10.13).

Листинr 10.13

|  |
| --- |
| <**ul class="sidebar-nav"**>  {% **if user**.**is\_authenticated** %}  <**li**>Пользователь: {{ **user**.**get\_usernarne** }}</**li**>  <**li**><**a href="**{% **url 'logout'** %}**?next=**{{ **request**.**path** }}**"**>Logout</**a**></**li**>  {% **else** %}  <**li**><**a href="**{% **url 'login'** %}**?next=Hrequest.path}}"**>Login</**a**></**li**>  {% **endif** %} </**ul**> |

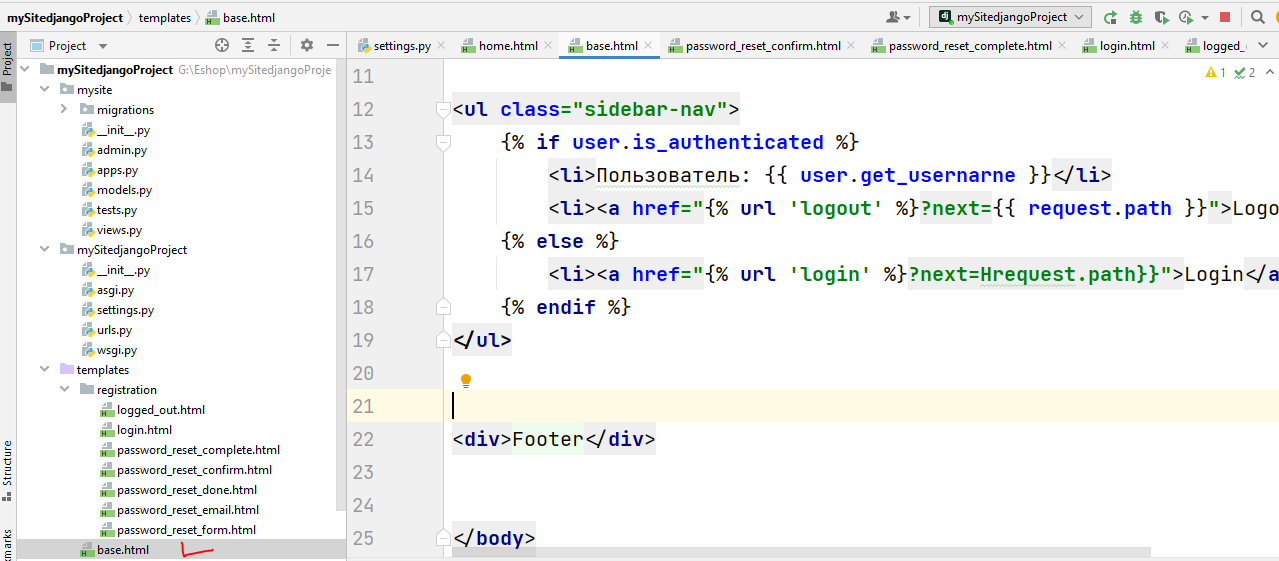


Рис. 10.27. Блок проверки входа/выхода пользователя на сайт в боковой панели базового шаблона

Как можно видеть, в этом коде используются теги шаблона if ... eise...endif для условного отображения текста в зависимости от того, является ли значение {{user.is\_authenticated}} истинным. Если пользователь аутентифицирован, то мы знаем, что это вошел зарегистрированный пользователь, поэтому мы вызываем { {user.get\_username}}, чтобы отобразить его имя.

Затем мы создаем URL адpec для входа и выхода из системы, используя тег шабло­на URL-адpeca и имена соответствующих конфигураций uri. Обратите также вни­мание на то, как мы добавили фрагмент: ?next={{request.path}} в конец блока uri. Это означает, что следующий URL-адpec содержит адрес (URL) текущей страницы в конце связанного URL-адpeca. После успешного входа пользователя в систему представления будут использовать значение next, чтобы перенаправить пользовате­ля обратно на страницу, где он изначально нажал ссылку входа в систему или вы­хода из нее.

Протестируем то, что мы сейчас сделали. Если теперь загрузить главную страницу сайта, то мы увидим, что на боковой панели появилась дополнительная ссылка с надписью Вход (рис. 10.28). Щелкните мышью на сылке Вход, и вы попадете на страницу идентификации пользователя (рис. 10.29).

Рис. 10.28. Ссылка Вход на главной странице сайта, ведущая на страницу идентификации пользователя

Введите на этой странице имя и пароль нашего тестового пользователя и нажмите на ссылку Вход -мы опять попадем на главную страницу сайта, но при этом в боковой панели появится имя вошедшего пользователя, а ссылку Вход будет заменена на ссылку Выход (рис. 10.30).