login + password = Авторизация

Выбираем проект для дальнейшей реализации. Создаём общую структуру проекта. Добавляем регистрацию и авторизацию пользователей

[Определимся с проектом](#_c713xnuw3xem)

[Тема](#_7j3z8sn6q984)

[Оформление](#_bzjx7ug0ktvu)

[Цели](#_mxx36rpkikat)

[Создаем проект](#_40p2vcsf3y6j)

[Сгенерируем структуру проекта](#_ll9pa6fk0smv)

[Добавим главную страницу](#_dxobso96izq0)

[Авторизация](#_46mcu5nac44z)

[Теория](#_13vqlukjj5pi)

[Практическая часть урока](#_r4llbmk8bwmm)

[Создаем пользователя](#_kr8raleq3wta)

[Работа с формами](#_9hli94dlr1t4)

[GET и POST запросы (теория)](#_dknjupjfu8t2)

[Добавляем форму авторизации](#_1q7cwrj6jiu7)

[CSRF TOKEN](#_h92z8dmc6nml)

[Получаем и обрабатываем данные из запроса](#_l1x8a2xtfg9g)

[Домашнее задание](#_owpsqd9opgcf)

[Дополнительные материалы](#_y937sk8fclye)

[Используемая литература](#_jcvo0jn2tn60)

# Определимся с проектом

Начиная с текущего вебинара, вы будете создавать свой собственный проект по изученным материалам и дополнять его на протяжении следующих занятий. Это будет именно проект, сделанный вами с нуля, который в дальнейшем вы сможете добавить в свое резюме и показать работодателю.

## Тема

Тему проекта (сайта) можете выбрать самостоятельно, но с соблюдением следующих условий:

* Сайт должен содержать объекты с набором параметров, которые (объекты) можно разбить по различным категориям;
* Сайт должен предоставлять возможность авторизации;
* Должна быть предусмотрена возможность оставлять комментарии/отзывы зарегистрированным пользователям.

Пример, подходящие темы:

“Бар” - сайт с рецептами коктейлей. Коктейли - объекты с различными параметрами (список ингридиентов, их количество, описание процесса приготовления и т.д.). Коктейли можно разбить на разные категории: алкогольные, безалкогольные, горячие, прохладительные и т.д.

“Рецепты домашней кухни” - сайт с рецептами различных блюд. Здесь по аналогии с баром.

“Магазин сотовых телефонов” - сайт по продаже телефонов. Телефоны - объекты с параметрами/характеристиками (производитель, цена, параметры телефона). На категории легко разделить по производителю.

“Развлечени нашего города” - сайт с описанием мест вашего города, где можно хорошо отдохнуть. Места отдыха (кафе, ночные клубы, игровые площадки) - объекты с различными параметрами (тип места, адрес, описание, отзывы, …). Хорошо группируются по типам: кафе, кинотеатры и т.д.

На уроках я буду разрабатывать сайт “Gems” (драгоценные камни), который вы будете использовать в качестве примера, и мы будем рассматривать типичные ошибки и разбирать проблемы, с которыми вы столкнулись. Т.е. вы будете самостоятельно реализовывать аналогичный функционал в своём проекте.

## Оформление

В первую очередь, наша задача написать именно движок сайта, его информационную и функциональную составляющие. Но, конечно, хочется иметь красиво оформленный проект. Это будет полностью ваша задача, я смогу поправить HTML верстку, посоветовать с CSS оформлением, но именно разработка дизайна выходит за рамки текущего курса. Поэтому, если вам с трудом даётся дизайн, то не расстраивайтесь, просто постарайтесь аккуратно оформить имеющуюся информацию или используйте готовые макеты (к счастью, на просторах интернета их предостаточно).

Для оформления демо-сайта я буду пользоваться интерфейсным фрэймворком [bootstrap](http://bootstrap-3.ru/index.php). Его изучение, также выходит за рамки данной программы, но основные моменты я расскажу. Ознакомиться с его основами вы сможете самостоятельно, почитав в дополнительных материалах.

## Цели

Если ВЫ поставили перед собой задачу именно научиться работать с Django, а не просто посидеть и посмотреть видео, и будете выполнять все домашние задания, то, по окончанию курса, у вас будет относительно простой, но полноценный веб-сайт с реальной базой данных и вашей собственной админкой.

# Создаем проект

Приступим!

Всё, что мы будем делать, подробно рассматривалось на предыдущих уроках, если забыли - пересмотрите материалы.

## Сгенерируем структуру проекта

1. Для создания нового проекта: django-admin.py startproject <имя проекта>
2. Создаем новое приложение: python manage.py startapp <название приложения>
3. Проверяем, запуская сервер разработки: python manage.py runserver

## Добавим главную страницу

Содержимое файла **urls.py** заменим на:

|  |
| --- |
| from django.conf.urls import url  from mainapp.views import \*  urlpatterns = [  url(r'^$', main),  ] |

Создадим папку для хранения шаблонов templates, не забываем дописать путь к папке в settings.py

|  |
| --- |
| TEMPLATES = [  {  'BACKEND': 'django.template.backends.django.DjangoTemplates',  'DIRS': ['**templates**'],  'APP\_DIRS': True,  'OPTIONS': {  'context\_processors': [  'django.template.context\_processors.debug',  'django.template.context\_processors.request',  'django.contrib.auth.context\_processors.auth',  'django.contrib.messages.context\_processors.messages',  ],  },  },  ] |

Создавая html верстку главной страницы, вспомним про наследование шаблонов и сразу выделим основные элементы для всех страниц в   
**base.py**

|  |
| --- |
| {#Любая страница проекта должна наследоваться от данной страницы, или от её потомков#}  <!DOCTYPE html>  <html>  <head>  <title>  {% block title %}  BASE TEMPLATE  {% endblock %}  </title>  {% block css %}  {#Подключаем общие для всего сайта таблицы стилей #}  <link rel="stylesheet" href="/static/css/base.css"/>  {% endblock %}  {% block js %}  {% endblock %}  <meta charset="utf-8">  </head>  <body>  <div class="container">  <div class="page-header">  <div class="row">  <div class="ico col-md-2 text-center">  <img src="/static/images/icon\_gem.gif">  </div>  <div class="col-md-4 col-md-offset-6 text-center">  <h1>Драгоценные камни</h1>  </div>  </div>  </div>  <div class="row">  {% block menu %}  // Здесь код меню  {% endblock %}  </div>  <div class="row">  <div class="col-md-offset-2 col-md-8">  {% block content %}  Текст базового шаблона(просто для демонстрации)  {% endblock %}  </div>  </div>  <div class="modal-footer">  {% block footer %}  &copy; 2016 Booblegum  {% endblock %}  </div>  </div>  </body>  </html> |

Некоторые примеры будут даны не полностью, полный текст в приложении “исходные коды”

Наследуясь от базового шаблона, набросаем заготовку главной страницы  
**index.html**

|  |
| --- |
| {% extends "base.html" %}  {% block content %}  <div class="home-img row">  <img src="/static/images/HomePageImages-img958-1.jpg">  </div>  <div class="row">  // Здесь текст для главной страницы  </div>  {% endblock %} |

Создадим папку static в корне проекта для хранения всех статических файлов.

В конец settings.py добавим код:

|  |
| --- |
| STATICFILES\_DIRS = (  os.path.join(BASE\_DIR, "static"),  ) |

Добавим оформление главной страницы, создав и прикрепив base.css

# Авторизация

## Теория

Несмотря на то, что Django предоставляет готовую систему авторизации, которая скрывает много нюансов и берёт на себя практически всю рутину, принципы знать необходимо.

После регистрации пользователь может, используя логин и пароль, авторизоваться на сайте. Когда вы выполняете какие-либо действия, браузер отправляет запросы, обработав которые, сервер возвращает ответ (в нашем случае ответами являются готовые html страницы).

Как же сервер понимает, что запрос пришёл именно от вас (авторизованного пользователя)?

Http протокол построен так, что не предоставляет возможности поддерживать непрерывную связь с сервером, поэтому, обработав очередной запрос, сервер не хранит никакой информации о предыдущих запросах. Вследствие этого, каждый запрос должен содержать полную информацию, включая идентификатор пользователя, по которому сервер проверит, авторизован ли пользователь.

Если кратко, то это работает таким образом. Вы (пользователь) авторизуетесь, отправляя логин и пароль. После проверки django генерирует токен авторизации (уникальный идентификатор) и отправляет его в ответ браузеру. Браузер сохраняет токен в куках (cookies) и автоматически добавляет этот токен в заголовок к каждому запросу. Django извлекает токен и проверяет авторизацию.

Как правило, токены авторизации имеют время жизни, когда это время истечёт, вам потребуется повторно авторизоваться, введя логин и пароль. Т.е., пока в куках вашего браузера есть актуальный токен, сервер будет считать вас авторизованным пользователем.

Подробнее про работу cookie и http-за головки [здесь](http://borpost.ru/programmrovanie/php/http-zagolovki-v-zaprosax-k-serveru/).

## Практическая часть урока

### Создаем пользователя

Форму регистрации мы добавим немного позже. Пока создадим пользователя с помощью консоли, как мы делали на предыдущих уроках.

Создадим пустую базу данных:

|  |
| --- |
| python manage.py migrate |

Добавим супер пользователя:

|  |
| --- |
| python manage.py createsuperuser |

Именно под этим пользователем мы и будем авторизовываться.

Прежде чем продолжить, научимся отправлять данные на сервер используя формы.

# Работа с формами

## GET и POST запросы (теория)

Запросы бывают различных типов: GET, POST, PUT и т.д. Мы пока рассмотрим только два первых.

GET-запрос — это самый распространённый вид HTTP-запроса. При помощи его происходит запрос браузером. HTTP-запрос типа GET состоит только из HTTP-заголовков, тело у него отсутствует.

Пример GET-запроса с двумя параметрами:

|  |
| --- |
| GET /enter?login=admin&password=qwerty HTTP/1.1 |

POST-запрос - если необходимо передать на веб-сервер большой объем данных, например, текст сообщения или множество параметров, используют POST-запрос. В этом типе запроса параметры помещаются в тело HTTP-запроса, а размер передаваемых данных в байтах указывается в заголовке Content-Length. Таким образом в URL передаваемые параметры не видны.

Немного подробнее [здесь](http://w-wb.com/cgi/cgi_05_01.php).

## Добавляем форму авторизации

Добавим html-форму авторизации, т.к. форма должна отображаться на всех страницах, я добавлю её в базовый шаблон.  
дополним **base.html**

|  |
| --- |
| <form method="post" action="/user/login/"> {% csrf\_token %}  <input type="text" name="login" value="" placeholder="Логин">  <input type="password" name="password" value="" placeholder="Пароль">  <button type="submit" class="btn btn-default">Send</button>  </form> |

Данный пример (как и все последующие) очищен кода стилей, полный пример с применением bootstrap в приложении “исходные коды”.

method="post" - указываем метод отправки  
action="/user/login/" - url, на который буду отправлены данные  
{% csrf\_token %} - защитный токен

### CSRF TOKEN

Строка {% csrf\_token %} нужна, чтобы использовать встроенную защиту против межсайтовой подделки запроса.

Защиту можно отключить, если удалить строку (выделенную красным) в settings.py

|  |
| --- |
| MIDDLEWARE\_CLASSES = (  'django.contrib.sessions.middleware.SessionMiddleware',  'django.middleware.common.CommonMiddleware',  'django.middleware.csrf.CsrfViewMiddleware',  'django.contrib.auth.middleware.AuthenticationMiddleware',  'django.contrib.auth.middleware.SessionAuthenticationMiddleware',  'django.contrib.messages.middleware.MessageMiddleware',  'django.middleware.clickjacking.XFrameOptionsMiddleware',  'django.middleware.security.SecurityMiddleware',  ) |

Но это крайне не рекомендуется.

Подробнее про CSRF [здесь](http://djbook.ru/rel1.7/ref/contrib/csrf.html).

При нажатии на кнопку “отправить”, форма автоматически формирует POST запрос, добавляя логин и пароль в тело POST-запроса.

## Получаем и обрабатываем данные из запроса

Для обработки запроса, в **urls.py** добавляем:

|  |
| --- |
| url(r'^user/login/$', login), |

Во **views.py** пишем соответствующую функцию-обработчик:

|  |
| --- |
| from django.shortcuts import render, HttpResponseRedirect  from django.contrib import auth  from django.http import Http404  def login(request):  if request.method == 'POST':  print("POST data =", request.POST)  username = request.POST.get('login')  password = request.POST.get('password')  user = auth.authenticate(username=username, password=password)  if user is not None:  auth.login(request, user)  return HttpResponseRedirect("/")  else:  return render(request, 'index.html', {'username': username, 'errors': True})  raise Http404 |

Разберем подробно код функции login().

Атрибут объекта request.method принимает значение метода запороса. Функция-обработчик ожидает только POST-запросы:

|  |
| --- |
| if request.method == 'POST':  ...  raise Http404 |

Если запрос пришел не POST’ом, вызываем исключение 404 - страница не найдена.

request.POST - данные из запроса хранятся в атрибуте, который называется также как метод, т.е. из GET-запроса данные будут в request.GET. Сами данные хранятся в объекте QueryDict, но вы можете работать с ними, как с обычным словарем (это. по сути и есть словарь, только расширенный доп.методами).

auth.authenticate(username=username, password=password) - пробуем аутентифицировать пользователя. Если успешно - auth вернет объект User пользователя, если нет - None.

|  |
| --- |
| if user is not None:  auth.login(request, user)  return HttpResponseRedirect("/") |

Если аутентификация прошла успешно - залогиниваем пользователя и перенаправляем (HttpResponseRedirect) на главную страницу.

|  |
| --- |
| else:  return render(request, 'index.html', {'username': username, 'errors': True}) |

Если же были введены неверные логин или пароль, то отображаем главную страницу, и в качестве контекста передаем информацию об ошибке.

Чтобы ошибка о неверных данных отобразилась, дорабатываем шаблон формы регистрации модифицируем **base.html**

|  |
| --- |
| {% if errors %}  <p class="error">Сожалеем, вы неправильно ввели логин или пароль</p>  {% endif %}  <form method="post" action="/user/login/">  {% csrf\_token %}  <div class="form-group">  <input type="text" name="login" value="{{ username }}" placeholder="Логин">  <input type="password" name="password" value="" placeholder="Пароль">  </div>  <button type="submit" class="btn btn-default">Send</button>  </form> |

Шаблонный тег {% if %} мы уже разбирали

Далее доработаем форму так, чтобы форма отображалась только для неавторизованного пользователя.Для уже авторизованного добавим функцию logout (разлогиниться).

модифицируем **base.html**

|  |
| --- |
| {% if not user.is\_authenticated %}  {% if errors %}  <p class="error">Сожалеем, вы неправильно ввели логин или пароль</p>  {% endif %}  <form method="post" action="/user/login/">  {% csrf\_token %}  <div class="form-group">  <input type="text" name="login" value="{{ username }}" placeholder="Логин">  <input type="password" name="password" value="" placeholder="Пароль">  </div>  <button type="submit" class="btn btn-default">Send</button>  </form>  {% else %}  {{ user.username }}  <a href="/user/logout/">logout</a>  {% endif %} |

Функция-обработчик logout() выглядит очень просто

|  |
| --- |
| def logout(request):  auth.logout(request)  return HttpResponseRedirect("/") |

Не забываем связать url /user/logout/ с функцией logout()

Подробнее про систему аутентификации [здесь](http://djbook.ru/rel1.8/topics/auth/default.html).

# Домашнее задание

Пришло время вам самим заняться разработкой собственного проекта:

1. Выбрать тему проекта по своему желанию, опираясь на рекомендации
2. Используя данную методичку как руководство, создать всё то, что мы выполнили на текущем уроке, а именно:
   1. Структуру проекта с одним приложением.
   2. Заготовку главной страницы. Не ленитесь, потратьте время, набросайте будущий дизайн в любом графическом редакторе или на бумаге, так будет проще понимать, что нужно делать. Обдумайте тематику вашего сайта, о чём вы хотите рассказать.
   3. Создать пустую базу данных и добавить в неё супер пользователя.
   4. Добавить форму и функционал авторизации. Постарайтесь, чтобы форма находилась на видном месте и не выпадала из общего дизайна.
3. \* Сделать выдвижную форму авторизации, которая появляется при клике на определённую область.

Все проблемные или не до конца понятные элементы снабжайте комментариями. Что-то хотели сделать, но не получилось, не удаляйте код - закомментируйте, снабдив подробными комментариями. При проверке ДЗ, я постараюсь решить ваши проблемы и рассказать о наиболее частых проблемах в начале следующего занятия.

# Дополнительные материалы

Все то, о чем сказано в методичке, но подробнее:

1. Ссылки на бесплатные хранилища шаблонов сайтов: [раз](http://www.templatemonster.com/ru/), [два](http://startbootstrap.com/template-categories/all/), [три](http://www.html5webtemplates.co.uk/templates.html)
2. [Основы Bootstrap](http://bootstrap-3.ru/getting-started.php), [примеры](http://bootstrap-3.ru/bootstraptheme.php)
3. [Cookies and Headers](http://borpost.ru/programmrovanie/php/http-zagolovki-v-zaprosax-k-serveru/)(рус)
4. [GET и POST запросы](http://w-wb.com/cgi/cgi_05_01.php)
5. [CSRF Token](http://djbook.ru/rel1.7/ref/contrib/csrf.html)
6. [Система аутентификации пользователя](http://djbook.ru/rel1.8/topics/auth/default.html)

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [Django Book(rus)](http://djbook.ru/)
2. [Основы bootstrap](http://bootstrap-3.ru/getting-started.php)
3. [Кратко о GET и POST запросах](http://w-wb.com/cgi/cgi_05_01.php)