Веб-розробка з Python та Django для Початківців

Виключно практичний посібник по веб-розробці з допомогою мови Python та фреймворку Django. Для початківців.

Віталій Подоба

Веб-розробка з Python та Django для Початківців

Виключно практичний посібник по веб-розробці з допомогою мови Python та фреймворку Django. Для початківців.

Віталій Подоба

This book is for sale at http://leanpub.com/djangofornewbie

This version was published on 2016-10-16

Tweet This Book!

Please help Віталій Подоба by spreading the word about this book on Twitter!

The suggested hashtag for this book is ##django #djangoproject #python.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=##django #djangoproject #python

Зміст

1. Вступ

Для кого і для чого дана книга?

Вітаю вас!

Якщо ви читаєте даний текст, то це, швидше за все, означає одне з двох:

- ви вирішили опанувати веб-розробку і обрали першою мовою для себе - мову програмування Python;
- ви уже розбираєтесь у веб-розробці, але хочете освоїти ази створення веб-аплікацій саме з допомогою веб-фреймворка Django.

Завданням даної секції книги якраз і є розібратись у тому, для кого було написано дану книгу, чим вона відрізняється від інших книг, які ви читали до цього часу та взагалі чого вам варто від неї очікувати.

Завдання книги

Ви мабуть подумали: "А для чого ще одна книга з веб-програмування? Та ще й такого популярного фреймворка як Django. Адже уже існує маса книг як англійською так і російською мовами." Тобто кожен бажаючий з легкістю може знайти необхідну інформацію щодо веб-розробки на даному фреймворку.

Я також так думав спочатку. Але згодом, працюючи із початківцями, зрозумів, що формат та підхід більшості існуючих книг є далеким від ідеального саме для людей, які ще не мають практики програмування. Навіть якщо книга базована на практичному проекті, то зазвичай формат подачі є швидше інформаційним та підходить людині, яка вже має хорошу практику програмування.

Із власного досвіду знаю, що початківцям недостатньо просто дати інформацію. Натомість, важливими чинниками успішності навчального матеріалу є:

- його практичне застосування;
- побудова усього матеріалу на прикладах;
- домашні завдання та заохочення до власних дій та покращення прикладів власними ідеями (тобто основний акцент на ініціативу студента та бажання вчитись розв'язувати проблеми використовуючи інформацію навчального матеріалу);
- наглядний матеріал, що безпосередньо показує процес вирішення проблем у живому непідготовленому режимі професійним програмістом (це як парне програмування);
- робота в команді собі подібних, де можна отримати як допомогу так і допомагати іншим;
- підтримка та мотивація, адже значно простіше долати перешкоди знаючи, що інші, такі як ти, також мають подібні проблеми на своєму шляху.

Якраз виходячи із вищенаведених критеріїв успішного навчального матеріалу, саме для початківців я і сформував не просто практичний посібник з Django, а цілий пакет додаткових матеріалів та сервісів для підтримки та мотивації.

Таким чином можна виділити два основні завдання даної книги:

1. **для тих, хто ще немає достатньої практики програмування**, - показати як можна почати програмувати реальний проект. І не лише повторювати код, приклади та проект із книги, а ускладнювати їх власними ідеями та завданнями і самостійно їх реалізовувати. Таким чином набувати практики розв'язування наближених до реальних проблем під час створення веб-аплікації на Django.
2. **для тих, хто вже є веб-розробником**, - надати лише необхідний мінімум інформації для освоєння азів веб-розробки з Python та фреймворком Django.

Підсумовуючи усе вищезгадане: перша і найважливіша порада вам, якщо ви початківець і ще не маєте достатньо практики програмування:

Якщо ви не маєте достатньо часу, бажання, чи планів на те, щоб активно працювати із вашим редактором коду протягом даної книги, ускладнювати

проект власними ідеями, вчитись самостійно розв'язувати проблеми, що виникатимуть у вас при цьому, тоді прошу вас зупинитись відразу і даремно не гаяти власного часу. Ця книга винятково для тих, хто вже починаючи з 4-ї глави книги відкриє свій улюблений редактор і почне регулярно виділяти час для кодування, читання чужого коду та розв'язування виникаючих проблем!

Якщо ж ви ще не визначились до кінця, що саме вас цікавить - чи вам потрібна веб-розробка, чи хочете мати справу з серверною стороною веб-сайтів, яку мову програмування обрати, що взагалі таке веб-розробка та програмування... - тоді пропоную спочатку ознайомитись із наступними матеріалами:

- Безкоштовний курс: "Програміст Початківець"[1]
- Як знайти себе в IT?[2]
- 5 причин, які заважають вам стати програмістом уже завтра[3]
- Що таке веб-розробка та ваша можлива роль у ній?[4]
- Як обрати першу мову програмування та чому Python є ідеальним для початківців?[5]

Я впевнений, що дані матеріали дадуть відповіді на більшість ваших запитань. Якщо все ж маєте будь-які сумніви, тоді продовжуйте освоєння даної книги лише після ознайомлення з цими матеріалами.

...

Ну як? Переконались, що дана книга - це саме те, що вам потрібно?...

Тоді можемо рухатись далі. Ця і наступні 2 безкоштовні глави даної книги дадуть вам ще більше розуміння того, чим ми з вами займатимемось та куди нас це все приведе.

---

[1] http://www.vitaliypodoba.com/email-course-dummy-programmer/
[2] http://www.vitaliypodoba.com/it-webinar/
[3] http://www.vitaliypodoba.com/newbie-prog-webinar/
[4] http://www.vitaliypodoba.com/webdev-webinar/
[5] http://www.vitaliypodoba.com/webinar-python/

Як працювати із даною книгою?

Перш за все хочу звернути вашу увагу на те, що дана книга йде у двох варіантах:

1. книга + код;
2. книга + код + закрита група підтримки + відео уроки + інтерв'ю з професійними програмістами + шпаргалки з технологій.

Якщо ви початківець, тоді вам потрібен саме другий варіант, який іде із всеможливими супорт матеріалами. Адже вам, як початківцю, не так важливо просто отримати інформацію, як отримати підтрику, мотивацію та детально показати (ідеально вживу) процес кодування та дебагу коду.

Щоб ефективно працювати із матеріалом даної книги, необхідний мінімум знань складає базове знання мови програмування Python включно з основами об'єктно-орієнтованого програмування (ООП). Якщо ви вважаєте, що даних знань вам бракує, тоді варто відкласти дану книгу; спершу підтягнути мову Python, а тоді вже повернутись і продовжити освоєння матеріалу даного підручника.

Ось кілька корисних джерел, де ви зможете швидко та якісно отримати базу мови програмування Python:

- англомовні Python курси на CodeCademy[6], Coursera[7], Udemy[8] та Udacity[9];
- російськомовний відео курс "Ввід в Python"[10] на Youtube;
- "Python 2: Курс Молодого Бійця"[11] у мене на блозі;
- набір посилань та рекомендацій із освоєння мови Python на Habrahabr[12].

[6] http://www.codecademy.com/tracks/python
[7] https://www.coursera.org/course/pythonlearn
[8] https://www.udemy.com/python-for-beginners/
[9] https://www.udacity.com/course/ud036
[10] https://www.youtube.com/playlist?list=PLEl2mW_X5hhkgW_e7B_ukSwPm3Q28eSyt
[11] http://www.vitaliypodoba.com/tutorials/python2-beginners-course/
[12] http://habrahabr.ru/post/150302/

> На сторінках цієї книги я даватиму посилання на зовнішні курси та навчальні матеріали у випадку, якщо та чи інша глава вимагатиме певних азів технології або мови (як от HTML, CSS чи Javascript).

Завдання книги - надати вам інформацію та приклади коду із поясненням азів веб-програмування з Django. Кожна глава закінчується домашнім завданням. Завдання починаються від найпростіших і закінчуються великими ідеями з покращення коду та функціоналу побудованого у даній главі. Таким чином маєте ідеї для подальшої практики.

Зазвичай, особливо на перших порах, у вас виникатиме маса проблем та запитань в процесі виконання домашніх завдань. І чим більше ви відходитимете від книжкових прикладів, тим більше виникатиме труднощів.

Саме для того, щоб не здаватись і доводити домашні завдання до кінця, до книги (Рекомендований пакет матеріалів) додається закрита група. Там можна обговорити те чи інше домашнє завдання, поставити конкретне технічне запитання та отримати на нього відповідь. У групі є як професійні програмісти, які добре розбираються у всіх аспектах веб-програмування, так і такі ж початківці, як ви. Така закрита тусовка значно збільшить ваші шанси для набуття практики програмування.

Із книгою також йде відео курс із сесіями програмування. Курс містить відео запис розробки того ж проекту, який описано в книзі, але форма подачі кардинально інша. Курс спеціально створено, як то кажуть, з першої спроби. Без спеціальної підготовки. Для чого? А для того, щоб ви, переглядаючи відео, могли бачити справжню живу сесію кодування професійного програміста, у якого в процесі роботи виникають справжні проблеми, які він розв'язує. Саме так. Відео містить усі проблемні місця включно із дебагом коду, брейкпоінтами, гуглінням, ковирянням коду та читанням коду фреймворку Django. Завдання відео курсу - навчити вас не просто робити все згідно з інструкцією, але й дати інструментарій та підходи для самостійного розв'язання ваших власних проблем під час програмування.

Адже програміст - це не той, хто знає мову програмування, а вміє скористатись нею для розв'язання реальних проблем та задач. Таким чином, відео курс, який

іде із повним пакетом книги, дасть вам відчуття парного програмування з професійним програмістом.

Даний відео курс рекомендую опрацьовувати паралельно з книгою, саме під час роботи над домашніми завданнями.

Також із книгою йде набір шпаргалок із Python, HTML, CSS та Git. Усі вони допоможуть вам сфокусуватись на програмуванні і не гаяти часу в пошуках тієї чи іншої функції в мові Python, тегу в HTML чи команди в репозиторії коду Git. З часом ви запам'ятаєте найбільш популярні функції, теги, правила, команди, але на початках дані підказки допоможуть вам добряче зекономити час і натомість проводити його над розв'язуванням безпосередніх завдань.

І як додатковий бонус до повного пакету книги додається кілька відео інтерв'ю із програмістами: як професіоналами, так і з джуніками (тими, що вже працюють). Рекомендую переглянути їх із самого початку, а також звертатись до них щоразу, коли рівень мотивації знижуватиметься, а рівень розчарування підвищуватиметься. В даних інтерв'ю ви знайдете історії людей, які вже вміють те, чого ви хочете навчитись. Ви отримаєте їхні поради щодо навчання програмуванню, пошуку роботи, ну і звичайно мотивацію та розуміння того, що не так вже й все складно.

Отже, підведемо підсумок щодо роботи із даною книгою:

- спочатку переглядаємо відео інтерв'ю із програмістами, мотивуємось, складаємо власний план досягнення цілей;
- далі налаштовуємось на довготривалу та плідну роботу, без халяви;
- далі читаємо главу книги, пробуємо приклади із прочитаної глави;
- якщо щось не виходить, як описано, - звертаємось за допомогою у закриту групу підтримки (спочатку перевірте чи вже немає відповіді на ваше запитання у групі, на більшість запитань початківця зазвичай існують відповіді від попередників, адже у всіх спільні проблеми);
- далі практика: працюєте над домашніми завданнями;
- під час роботи над домашніми завданнями виникатиме маса проблем і запитань, для вирішення яких знову звертаєтесь у закриту групу;
- також на етапі роботи із домашніми завданнями варто переглядати відео курс, з якого черпати практику вирішення реальних задач, користування інструментами для відлагодження коду;

- в процесі кодування користуєтесь шпаргалками.

Цю книгу я постійно вдосконалював і переписував під час роботи зі студентами, яких я менторив в напрямку веб-програмування на Django. Адже студенти давали мені живий фідбек в якості запитань та скарг стосовно тих речей, які вдавалися найважче. Таким чином я змінював пояснювальні тексти, а також додавав речі, які на перший погляд були для мене очевидними, але насправді викликали масу запитань. Саме завдяки студентам, я вважаю, мені вдалось дохідливо організувати книгу та додаткові матеріали, що врешті дозволить вам значно легше опановувати веб-програмування та набувати практику.

На завершення даної секції ще хочу повідомити вам, що дана книга не буде разовою роботою із моєї сторони, а натомість постійно вдосконалюватиметься новим та кращим матеріалом, відповідно до фідбеку читачів та нових версій Python та Django. Я планую постійно і на регулярній основі підтримувати матеріал даної книги в хорошій формі та в майбутньому релізити нові версії. Усі інші мої продукти будуть створюватись на базі даної книги таким чином, щоб надати допомогу початківцю на усіх його етапах від навчання азів програмування і аж до пошуку першої роботи.

> В останній главі також є пояснення скільки приблизно часу у вас може піти на освоєння матеріалу даної книги.

Форматування у книзі

Як ви уже мабуть зауважили, книгу я старався писати простою живою мовою без зайвого офіціозу. Приблизно такою мовою, якою я зазвичай пишу статті на своєму блозі. Адже вважаю, що таким чином можна краще надати практичний матеріал, передати власний досвід, організувати та змотивувати читача до дій, ніж пишучи сухою офіційною мовою книг. Час до часу можуть проскакувати сленгові слова, суржик, русизми, англіцизми і т.д. Я не намагався їх зменшити в книзі, натомість хочу, щоб книга звучала автентично та передавала настрій автора і давала вам відчуття реального персонального ментора з усіма його нюансами :-).

Окрім звичайного тексту, в даній книзі також використовується форматування блоків коду. Зазвичай, це спеціально підсвічений блок коду. Кожна мова матиме свою підсвітку. Ось приклад Python та HTML коду:

Приклад Python коду

```
class BaseExamFormView(object):
    def post(self, request, *args, **kwargs):
        # handle cancel button
        if request.POST.get('cancel_button'):
            return HttpResponseRedirect(reverse('exams_list') +
                '?status=Зміни скасовано.')
        else:
            return super(BaseExamFormView, self).post(
                request, *args, **kwargs)
```

Приклад HTML коду

```
<html>
  <head><title>Заголовок Сторінки</title>
  <body>
    <h1>Приклад Сторінки</h1>
    <p>Підсвітка Рулить!</p>
  </body>
</html>
```

Здебільшого книга містить лише практичний матеріал і лише той, що стосується проекту, який розроблятимемо на сторінках даного посібника.

> У тих місцях, де виникатиме потреба у невеликій додатковій теоретичній інформації, зауваженнях, практичних порадах чи нотатках від автора, ми використовуватимемо ось такі блоки тексту, огорнуті в рамку.

Які технології розглянемо у даній книзі?

На сторінках даної книги ми створимо наближений до реального веб-проект, в процесі якого працюватимемо із наступними технологіями:

- мова програмування Python[13];
- веб-фреймворк Django[14];
- бази даних sqlite[15] та MySQL[16];
- мова розмітки HTML[17];
- каскадні таблиці стилів CSS[18];
- мова програмування Javascript[19];
- Javascript бібліотека jQuery[20];
- HTML/CSS фреймворк Twitter Bootstrap[21];
- репозиторій коду Git[22].

У наступних главах ми детальніше поговоримо про кожну із вищенаведених технологій.

Тут лише зауважу, що основний акцент даної книги є на серверній розробці веб-сайту. Ми розроблятимемо також і клієнтську сторону веб-аплікації, але для цього використовуватимемо бібліотеку Twitter Bootstrap, що полегшить наше завдання та дозволить менше занурюватись у верстку, а більше фокусуватись на Django фреймворку.

[13] http://python.org/
[14] https://www.djangoproject.com/
[15] http://www.sqlite.org/
[16] http://www.mysql.com/
[17] http://www.w3.org/TR/html/
[18] http://www.w3.org/TR/CSS/
[19] http://uk.wikipedia.org/wiki/JavaScript
[20] http://jquery.com/
[21] http://getbootstrap.com/2.3.2/
[22] http://git-scm.com/

Організовуємось!

Як ви вже зрозуміли, дана книга - це не просто інформація з веб-програмування, а спроба надати справжню підтримку та підвищити шанси студента на успіх.

Початківець на шляху до успіху зустрічається з двома величезними проблемами, з якими варто навчитись працювати:

1. організація та самодисципліна;
2. мотивація та віра у власні сили.

Саме тому, окрім матеріалів по програмуванню, в дану книгу я також додав поради та рекомендації із самоорганізації. Вони допоможуть вам освоїти матеріал книги, отримати практику програмування, а не просто прочитати і забути.

Тому для початку давайте розберемо кілька важливих моментів щодо організації свого часу, щоб збільшити наші шанси на успіх.

Сила “Вже і Зараз”

Якщо ви ще цього не зрозуміли, то знайте: те, що ми робимо прямо тут і зараз безпосередньо впливає на наше майбутнє. Кожна наша думка, рішення і дія формують наше майбутнє, допомагають або заважають нам досягати цілей та формувати вдалі плани.

Саме тому так важливо перестати відкладати на завтра ті речі, які можна спробувати уже сьогодні, через годину, або ще краще прямо зараз!

Виходячи із даного принципу, рекомендую вам, особливо якщо ви ще початківець у програмуванні, виконувати приклади та домашні завдання з даної книги “не відходячи від каси”. Не відкладайте на завтра, адже завтра будуть нові ще цікавіші завдання.

Якщо хочете знати, чому ми усі відкладаємо найважливіші справи у нашому житті на завтра, а також те, як з цим бути, тоді почитайте мою статтю на дану тему. Вона дасть вам гарне розуміння ситуації, а з ним прийде і бажання щось

змінити: “Як початківцю не відкладати на завтра той код, що можна закодити уже сьогодні![23]”.

Правильний розпорядок дня

Наш мозок працює згідно 4-ох годинних фаз. Перші 4 години вашого дня він зазвичай є найактивнішим, має величезний потенціал та енергію до вирішення найважчих проблем дня. Потім настає 4 години обіднього часу, коли йому варто відпочити. І далі, після обіду, знову 4 години, коли він вже не такий свіжий, як на початку дня, але ще у гарному стані для продовження менш інтенсивної розумової праці. Як от, наприклад, навчання.

Звісно, у кожного з нас є свій розпорядок. Хтось любить вставати рано-вранці, а хтось довше повалятись в ліжку. Комусь програмується значно краще вночі, а комусь після обіду. Це все справа звички.

В будь-якому випадку рекомендую вам, по-перше, виділити фіксований проміжок часу у вашому дні, коли ви будете займатись практикою програмування. В ідеалі хоча б 2 години, ну або хоча б 1 годину. Але тоді мати таких проміжків хоча б два на один день.

По-друге, так само виділити час на навчання. Це може бути прочитання книги, проходження курсу, перегляд відео туторіалів і т.д.

По-третє, переконатись, що проміжок практики програмування завжди йде перед періодом навчання. Адже навчання не потребує такої кількості енергії та свіжості вашої думки, як програмування. Саме тому, першою у вашому денному графіку має бути практика програмування, а вже потім - навчання програмуванню та теорія.

І взагалі, значно краще щодня приділяти невелику кількість часу програмуванню, ніж один раз на тиждень по кілька годин за присіст. Наша підсвідомість працює так, що їй краще запам’ятовувати і обробляти інформацію регулярно невеликими порціями, таким чином перебуваючи постійно в атмосфері предмету, який ви освоюєте.

[23] http://www.vitaliypodoba.com/2014/09/start-coding-today/

У даній статті я описав, як виглядає мій робочий день, і пояснив чому найважчі справи я роблю із самого початку. Рекомендую взяти собі на озброєння кілька ідей: "Робочий день програміста або як не проживати все життя за компом?[24]".

Сила звички

Буде взагалі ідеально, якщо вищенаведені рекомендації із розпорядку вашого дня ви зможете ввести у своє життя на регулярній основі. На рівень звички.

Мотивація, бажання, мрії, зобов'язання - все це з часом проходить і перестає працювати. Але якщо до моменту, коли це все пройде, ви зможете виробити у себе звичку кожного дня хоча б годину програмувати, то це буде запорука успіху для вас як початківця у програмуванні.

Ще донедавна я і не уявляв, що звичка може бути таким потужним інструментом і ключем до всього! Хочете побудувати успішний бізнес? Випрацьовуйте правильні звички. Хочете поїхати у кругосвітню подорож? Випрацьовуйте правильні звички. Хочете навчитись професійно програмувати? Випрацьовуйте правильні звички.

Лише завдяки звичці я не зупиняюсь писати статті у своєму блозі. Лише завдяки звичці я написав дану книгу. Тому я впевнений, що, якщо ви виробите у собі правильні звички, вони гарантують вам успіх у вивченні програмування та освоєння фреймворку Django.

Як виробляти звички? Ось невеличкий огляд того, як я це робив: "Сила Звички або Як Досягати Результату[25]".

Тайм менеджмент

На завершення організаційної секції хочу згадати, що існує маса технік та підходів з управління своїм часом. Яка з них підійде саме вам? Не знаю. Потрібно самому багато різних речей спробувати і зрозуміти, що найкраще працює для вас.

[24] http://www.vitaliypodoba.com/2014/08/programmer-day/
[25] http://www.vitaliypodoba.com/2014/04/the-power-of-habit/

Коли я багато програмував протягом дня, то успішно використовував Техніку Помодоро[26]. Можливо вона допоможе і вам.

Із власних спостережень я зрозумів, що розподіл свого часу на менші частини допомагає мені краще його використовувати і менше гаяти на непотрібні речі. До прикладу, якщо кожні 30 хвилин роблю перерву на 5-ти хвилинний відпочинок, тоді встигаю більше, ніж коли працюю, не встаючи 2 години. За 30 хвилин я просто не встигаю відволікатись на непотрібні речі.

А загалом, треба навчитись домовлятись із собою. Кожен раз, коли займаєтесь не надто важливою справою, запитайте себе: "А чи це приведе мене через тиждень (місяць, рік, 10 років) до моєї цілі?". Мені допомагає.

Експериментуйте і шукайте свої підходи для самоорганізації та самодисципліни, адже без них далеко не заїдеш.

Мотивуємось!

У мене вже неодноразово були ситуації, коли студент на персональному менторстві опускав руки, розчаровувався і готовий був здатись буквально у перший тиждень співпраці.

Я, як ніхто інший, тепер знаю, що мотивація, віра в себе, порада з боку - є одними з найпотужніших інструментів у допомозі початківцю освоїти програмування.

Що ж стоїть на шляху початківця, що може швидко обламати його крила у перші тижні практики?

Несвідомий код

Мабуть найбільшою проблемою людини, яка починає розбирати багато технологій, мов і напрямків одразу (а так воно у вебі часто буває), є розчарування через те, що багато речей на практиці якщо і вдаються, то вдаються несвідомо. Без повного розуміння усієї картини.

[26] http://www.vitaliypodoba.com/2014/04/pomodoro-accomplish-more-in-25-minutes/

Дядько Google, “copy-paste” (скопіював-вставив), “метод наукового тику”, випадкові спроби та експерименти - ось це все з чого я сам починав програмувати. І це незважаючи на те, що я перед тим прочитав з 5-7 книг по усіх потрібних мовах та технологіях.

Якщо ви лише починаєте набиратись практики програмування, тоді, впевнений, у вас буде подібна ситуація. Адже ніхто не починає ходити в дитинстві допоки 1000 разів не впаде.

Тому не бійтесь несвідомого коду. Просто сідайте і пишіть. Чим більше напишете, тим швидше перейдете на рівень свідомого коду.

Ось рівні коду початківця:

1. написати будь-який код;
2. зробити цей код таким, щоб запускався, тобто без синтаксичних помилок;
3. оновити код так, щоб він працював;
4. оновити код так, щоб він працював правильно;
5. заглянути ще раз в середину коду і тепер зробити, щоб він ще й виглядав гарно згідно з кращими практиками мови та технологій;
6. дописати тести та документацію (якщо звичайно має зміст у даній ситуації);
7. вернутись до свого коду за деякий час і виявити у собі бажання переписати код ще кращим чином. Якщо бажання з’явилось, значить за цей час людина прогреснула.

Тому, аби перейти на рівень 7, треба якомога швидше почати рівень 1. Крапка. І без розчарувань та ідеалів щодо результатів від прочитання теоретичних книг.

Переломний момент

З часом рівень “несвідомого коду” знижується і чим далі, тим з більшою швидкістю. Варта лише не здатись на початку при перших проблемах, а далі кожен наступний виклик буде долатись вами простіше і простіше.

Кажуть, що для того, щоб стати експертом у будь-якій справі, потрібно провести як мінімум 10.000 годин свідомої практики.

Так от, я думаю, що для програміста-початківця, щоб дійти до переломного моменту після якого можна **самостійно** (хоч інколи і повільно) вирішувати більшість проблем у веб-розробці з Django, потрібно в середньому 6-8 тижнів регулярної щоденної практики. Мінімум 2-3 години програмування на день. Це рівень, коли вам вже не потрібен наставник для вирішення кожної проблеми, а більшість запитань ви з'ясовуєте та відпрацьовуєте самостійно.

Після 6-8 тижнів ви не станете професіоналом, але протягом них початківець зазвичай переходить із першого рівня написання коду до написання коду, що працює правильно. І, що дуже важливо, - самостійно, без допомоги ззовні.

Тому перші кілька тижнів є особливо важкими. На них варто відповідно налаштуватись, змотивуватись і зрозуміти, що дуже багато речей буде незрозуміло. Але тим не менше, доведеться просто сідати і писати, ковиряти та читати чужий код.

Просто скажіть собі: "це нормально, так має бути, це не я невдаха, це усі так починають!". Зрозумійте, що більшість професійних програмістів саме так і починали.

Вища мета

Кожного разу, коли:

- я хочу відкласти важливу некомфортну справу на пізніше,
- маю великі несподівані проблеми на своєму шляху,
- не впевнений у правильному виборі,
- просто лінуюсь,

я стараюсь зупинитись і згадати про свою вищу мету, про план та цілі, які я розробив раніше. Зазвичай, кожна поточна проблема вирішується легше, якщо ви згадуєте те, що вас чекає коли успішно подолаєте даний етап.

Вища мета поповнює вашу енергію та додає мотивації у кожній складній ситуації. Головне не забувати про неї і регулярно піднімати голову над буденними проблемами.

Але, щоб цей метод досягнення цілей працював, переконайтесь, що ваша мета та цілі є дійсно такими, від яких у вас захоплює дух. Такими, які по-справжньому мотивують вас та з легкістю піднімають вас кожного ранку з ліжка.

Ви мабуть зауважили, що на моєму блозі близько половини усіх статей мають мотиваційний та підбадьорюючий характер. Я дуже стараюсь, щоб дозволити людині глянути на велику гору проблем, як на маленький посильний горбик. Адже в протилежному випадку, більшість людей навіть не почне сходження на неї. Ось лише декілька із цих статей, читайте, мотивуйтесь і повертайтесь до них у важкі хвилини:

- Мої Спостереження про викладання та навчання програмуванню[27],
- Чотири Кроки до Гугла без Університетського Ступеня[28],
- 5 способів, щоб стати найкращим у своїй справі[29],
- Мої поради усім новим програмістам або як мотивуватись?[30].

Домашнє завдання

Ви можете здивуватись, чому вступна глава займає так багато сторінок і чому він стільки “лиє води”? Коли ми вже нарешті перейдемо до діла і почнемо програмувати?

На це я вам скажу, що дана глава насправді є однією із найважливіших у цій книзі. Питання підтримки, самоорганізації та мотивації є найважливішими для початківців у програмуванні. Саме тому я ще не один раз буду заряджати сторінки даної книги “тайм менеджмент” фішками, “мотивашками” та іншими речима, які не мають прямого відношення до програмування.

[27] http://www.vitaliypodoba.com/2013/08/my-observations-teaching-and-learning-programming/
[28] http://www.vitaliypodoba.com/2014/04/four-steps-to-google-without-degree/
[29] http://www.vitaliypodoba.com/2014/06/how-to-be-best/
[30] http://www.vitaliypodoba.com/2014/08/what-i-tell-all-new-programmers/

Але, якщо ви супер-організований, оптимістичний та вмотивований супермен, тоді надалі можете сміливо пропускати такі ліричні відступи.

Усім іншим щиро раджу:

- переглянути свій робочий день;
- переглянути своє питання: “Для чого мені усе це?”;
- поставити далекоглядні високі цілі, які вас будуть по-справжньому мотивувати;
- виділити собі години для роботи із книгою та години для самого програмування;
- в нашій закритій групі підтримки написати про себе і познайомитись із тусовкою.

Якщо все зробите правильно, тоді вважайте, що 70% успіху у вас в кишені!

В наступній главі ми з вами оглянемо ази веб-розробки. Розглянемо, що таке створення веб-сайту, які технології і мови задіяні у процесі його створення і гарно підготуємось до нашого веб-проекту.

2. Що таке веб-розробка?

У цій главі поговоримо про те, що таке інтернет для нас і як він працює. Які технології використовуються при створенні нового веб-сайту та те, як взаємодіють сервер, що обслуговує веб аплікацію та веб переглядач (або ще як його називають веб браузер), який дає можливість користуватись даною аплікацією.

Думаю ви уже можете дещо знати про веб-розробку та суміжні технології, але щоб усі читачі та студенти починали маючи один і той самий необхідний мінімум теоретичних знань, спільний старт, пропоную ще раз коротко оглянути основи веб-розробки та те, з чого складається *інтернет.*

Комунікація Клієнт - Сервер

Отже, що таке інтернет для нас як користувачів? Це набір різних порталів, до яких ми звикли, користуємось щоденно, шукаємо інформацію, переглядаємо відео, слухаємо музику, комунікуємо з друзями, читаємо новини і тому подібне.

Для того, щоб ми могли користуватись інтернетом нам потрібен браузер - програма, яка дозволяє переглядати та користуватись веб-сайтами. У наш час маємо кілька найпопулярніших браузерів, серед яких є Chrome, Firefox, Internet Explorer, Safari та Opera.

Програма браузер дозволяє нам взаємодіяти із веб-сайтом, дані якого (всеможливі файли та сторінки) лежать на віддаленому сервері.

Для того, щоб взаємодіяти із сервером існує спеціальний формат адрес під назвою URL (Universal Resource Locator - Універсальний Локатор Ресурсів). Саме ця адреса дозволяє визначити, на якому віддаленому сервері знаходиться сторінка, яку нам потрібно. Також адреса містить шлях до цієї сторінки чи файлу в межах того ж таки віддаленого сервера.

Ось приклад URL адреси сторінки продажу даної книги:

http://www.vitaliypodoba.com/books/django-for-beginners

Частина *www.vitaliypodoba.com* вказує на віддалений сервер, де знаходиться мій блог. А частина адреси "/books/django-for-beginners" вказує, де саме знаходиться сторінка з книгою в межах того віддаленого сервера.

Також адреса може містити додаткові параметри, які йдуть одразу за знаком запитання наприкінці:

http://www.vitaliypodoba.com/books/django-for-beginners/buy/?package=recommended

Але детальніше про формат адрес ми ще погоримо під час подальшої розробки нашого з вами практичного проекту.

Зазвичай по дорозі між веб браузером та віддаленим сервером знаходиться велика кількість інших серверів, що працюють в інтернеті. І кожен наш запит на сервер проходить усі ці віддалені сервера. Саме завдяки їм ми маємо доступ до кінцевого сервера, де знаходиться потрібний нам сайт.

Інформація, якою обмінюються веб переглядач та віддалений сервер зазвичай представлена у форматі HTML[31]. Про нього ми трохи детальніше поговоримо у наступних секціях даної глави. Формат HTML описує вміст сторінки, яку показує веб браузер.

Оскільки ми маємо кілька серверів у ланцюжку, що з'єднує наш веб переглядач із кінцевим віддаленим сервером, усі вони повинні спілкуватись між собою у певному наперед узгодженому стандарті. Ці стандарти комунікації серверів в інтернеті описуються такими словами як:

- TCP/IP[32] - протокол, що пояснює серверам як вони повинні передавати інформацію між собою на низькому рівні;
- HTTP[33] - протокол, який пояснює браузеру як саме робити запит за документом із сервера, а серверу, у свою чергу, пояснює як готувати відповідь браузеру. Його ви мали бачити, як мінімум, у більшості адрес веб-сайтів, адже вони починаються із "http://" стрічки.

[31] http://uk.wikipedia.org/wiki/HTML
[32] http://uk.wikipedia.org/wiki/TCP/IP
[33] https://uk.wikipedia.org/wiki/HTTP

Таким чином браузер може робити *запит* на сервер за певною URL адресою у потрібному форматі згідно HTTP протоколу, а сервер обробляє даний запит, готує *відповідь* також у наперед визначеному форматі (згідно HTTP протоколу).

Оскільки браузер і сервер знають протокол HTTP і знають як спілкуватись між собою, вони можуть обмінюватись документами, а ми, відповідно, бачити результат цього обміну у вигляді веб-сайтів та веб-сторінок.

Відповідно далі постає питання, а як працюють самі сторінки?

HTML - Мова розмітки гіпертекстових документів

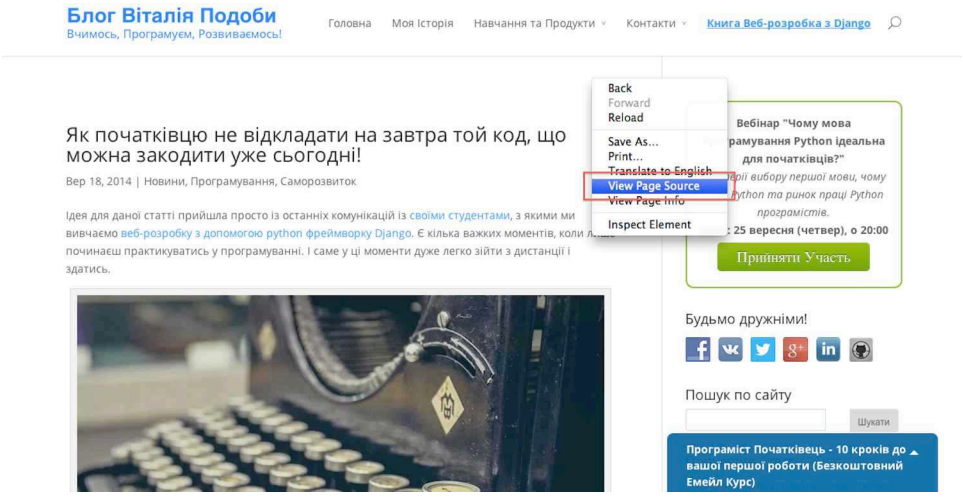
Більшість сторінок в інтернеті побудована саме з допомогою мови HTML (HyperText Markup Language - Мова Розмітки Гіпертекстових Документів).

> З Вікіпедії: Гіпертекст (англ. Hypertext) — текст для перегляду на комп'ютері, який містить зв'язки з іншими документами ("гіперзв'язки" чи "гіперпосилання"); читач має змогу перейти до пов'язаних документів безпосередньо з вихідного (первинного) тексту, активізувавши посилання. Найпопулярнішим зразком гіпертексту є World Wide Web, у якому веб-оглядач переміщує користувача з одного документу на інший, щойно той «натисне» на гіперпосилання.

Тобто наші старі добрі посилання, або ще як ми їх просто називаємо "лінки", і є тою причиною, чому ми веб сторінки також називаємо "Гіпертекстом". Саме це слово є складовою абревіатури HTML.

Таким чином мова HTML є основою основ усіх веб сайтів, а значить і веб-розробки. З допомогою цієї мови ми можемо описати структуру сторінки, її ієрархію (так, HTML документ є ірархічною структурою вкладених HTML тегів), вставити параграфи тексту та заголовки, списки і таблиці, зображення, відео та звук, реалізувати форми.

Якщо зайти на будь-який сайт і клацнути там правою клавішею мишки на вільному від тексту і зображень місці на сторінці, отримаємо контекстне меню, у якому знайдемо елемент “View Page Source” або “Перегляд HTML Коду”, або щось подібне (взалежності від вашої встановленої мови та веб переглядача).



Як переглянути HTML код сторінки

Вибравши його вам відкриється нове вікно переглядача із HTML кодом даної сторінки. Приблизно так всередині виглядають більшість веб сторінок в інтернеті:

```
<html lang="uk">
<!--<![endif]-->
<head>
        <meta charset="UTF-8"/>
        <title></title>
        <meta name="description" content=""/>   <meta name="keywords" content=""/>      <script type='text/javascript'>window.mod_pagespeed_start =
Number(new Date());</script><link rel="canonical" href="http://www.vitaliypodoba.com"/>

        <link rel="pingback" href="http://www.vitaliypodoba.com/xmlrpc.php"/>

                <!--[if lt IE 9]>
        <script src="http://www.vitaliypodoba.com/wp-content/themes/Divi/js/html5.js" type="text/javascript"></script>
        <![endif]-->

        <script type="text/javascript">
                document.documentElement.className = 'js';
        </script>

        <link rel="alternate" type="application/rss+xml" title="Віталій Подоба &raquo; стрічка" href="http://www.vitaliypodoba.com/feed/"/>
<link rel="alternate" type="application/rss+xml" title="Віталій Подоба &raquo; Канал коментарів" href="http://www.vitaliypodoba.com/comments/feed/"/>
<meta content="VP Divi v.1.0.0" name="generator"/><link rel='stylesheet' id='mailchimpSF_main_css-css' href='http://1-
ps.googleusercontent.com/h/www.vitaliypodoba.com/,qmcsf_action=main_css,aver=3.9.2.pagespeed.ce.4HTwIVql2T.css' type='text/css' media='all'/>
<!--[if IE]>
<link rel='stylesheet' id='mailchimpSF_ie_css-css'  href='http://www.vitaliypodoba.com/wp-content/plugins/mailchimp/css/ie.css?ver=3.9.2' type='text/css'
media='all' />
<![endif]-->
<link rel='stylesheet' id='crayon-css' href='http://1-ps.googleusercontent.com/h/www.vitaliypodoba.com/wp-content/plugins/crayon-syntax-
highlighter/css/min/crayon.min.css,qver=2.6.5.pagespeed.ce.yk5ohkhOXR.css' type='text/css' media='all'/>
```

Кусок HTML коду сторінки на моєму блозі vitaliypodoba.com

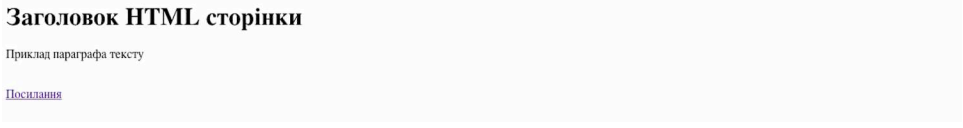
Якщо у спрощеному варіанті, то структура HTML документу виглядає при-

близно отак:

Приклад HTML документу

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Приклад HTML Сторінки</title>
    <meta name="description" value="Мета опис сторінки" />
  </head>
  <body>
    <h1>Заголовок HTML сторінки</h1>
    <p id="paragraph">Приклад параграфа тексту</p>
    <br />
    <a class="my-link" href="http://google.com">Посилання</a>
  </body>
</html>
```

Якщо збережете код вище наведеного прикладу у ф̣айл у себе на комп'ютері та відкриєте у себе в браузері, тоді отримаєте щось подібне на оце:



Приклад HTML сторінки у браузері

Як бачите основу HTML складають *теги*. Це елементи огорнуті в кутові дужки. Якщо глянете на 4-ий рядочок прикладу вище, то побачите тег "title". Він складається із відкриваючого елементу "<title>", вмісту "Приклад HTML Сторінки" та закриваючого елементу "</title>". Є теги, які не мають вмісту, а також є теги, які не мають закриваючого елемента. Приклад можете бачити у вище наведеному коді на 10-му рядочку: тег переводу рядка "
".

Також теги можуть мати властивості, так звані *атрибути*. 5-ий рядок коду у прикладі вище містить тег "meta" з двома атрибутами "name" (ім'я) та "value" (значення). Кожен із атрибутів має ім'я та значення, що іде одразу після ім'я та знаку "=". Кожен тег має власний набір атрибутів притаманний саме йому.

Структура документу зазвичай містить такі обов'язкові елементи як

- *html*: кореневий елемент будь-якого HTML документу;
- *head*: шапка документу; містить метадані сторінки та під'єднання інших ресурсів до сторінки (такі як javascript та css файли);
- *body*: тіло документу; містить теги, що є видимою частиною веб-сторінки.

Існує велика кількість HTML тегів, але найбільш поширених, які ми будемо з вами використовувати найчастіше, є лише в межах 20-ти.

> В шпаргалці по HTML та CSS, яка йде із рекомендованим набором до даної книги, ви можете ознайомитись із найбільш поширеними елементами та їх застосуванням.

Детальніше із синтаксисом мови HTML можете ознайомитись на сторінці вікіпедії[34]. А щоб отримати мінімум практики та розуміння як вона працює в браузері, рекомендую пройти один-два базових онлайн курси. Далі в книзі я буду пояснювати нюанси мови HTML рівно на стільки, на скільки це буде нам необхідно для реалізації нашого проекту.

На завершення даної секції хочу зауважити, що при освоюванні мови HTML варто також звернути увагу на такі поняття як семантичний HTML[35], доступність у вебі[36] та мікро-формати[37]. Таким чином в подальшій своїй практиці ви верстатимете сторінку не лише так, щоб вона виглядала як в оригінальному дизайні, але й правильно з точки зору внутрішнього коду.

HTML чи HTML5?

Це є уже традиційним запитанням початківців щодо того, яку версію мови HTML вивчати: HTML 4.01 чи HTML 5?

[34] http://uk.wikipedia.org/wiki/HTML
[35] http://webgyry.info/semantics-html
[36] http://bit.ly/vpwebaccessibility
[37] http://bit.ly/vphtmlmicroformats

HTML 5 - це наступна версія мови HTML із значними покращеннями в сторону семантичного (змістовного) вебу та новими мультимедійними можливостями. Під семантикою розуміємо надання тегам не лише функції для представлення даних, але й і для пояснення цілі цих даних.

HTML 5 є свого роду уніфікацією XML, HTML 4.01, XHTML. Робота над даним стандартном почалась ще у 2004 році і триває і досі і ще буде тривати далеко не один рік. Різні веб браузери підтримують уже цілий ряд HTML 5 функціоналу, але є відмінності між їхніми реалізаціями.

Тому, якщо давати коротку відповідь з якої версії HTML мови починати, то звичайно починайте із поточної HTML 4.01. Куди б HTML 5 в майбутньому не повернула, HTML 4.01 все одно буде хорошою базою для неї ще довгий час.

Детальніше про різноманітні варіанти стандарту HTML, такі як XHTML[38], HTML5[39], і т.д. можете ознайомитись знову ж таки на сторінці вікіпедії[40].

CSS - Каскадні таблиці стилів

З часом прості веб-сторінки із одноманітним нецікавим текстом усім набридли, тому почали придумувати та додавати стилі до HTML елементів. Спочатку це було у вигляді нових атрибутів напряму в HTML теги, а також через табличні стилі для створення потрібного лейауту сторінки.

Згодом винайшли так звані каскадні таблиці стилів - нову мову розмітки для описування стилів для елементів на сторінці - CSS. Каскадні тому, що вони дозволяють визначати стилі елементів на різних рівнях і унаслідувати стилі від вищих в ієрархії елементів.

CSS давав надзвичайні переваги, адже його можна було описувати в окремому файлику, а тоді під'єднувати до документа веб-сторінки. Таким чином змінивши кілька стрічок в CSS файлі можна було одним махом оновити верстку на усьому сайті.

Давайте продовжимо із нашого прикладу із секції HTML:

[38] http://en.wikipedia.org/wiki/XHTML
[39] http://en.wikipedia.org/wiki/HTML5
[40] http://en.wikipedia.org/wiki/HTML#WhatWG_HTML_versus_HTML5

Приклад HTML документу

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Приклад HTML Сторінки</title>
    <meta name="description" value="Мета опис сторінки" />
  </head>
  <body>
    <h1>Заголовок HTML сторінки</h1>
    <p id="paragraph">Приклад параграфа тексту</p>
    <br />
    <a class="my-link" href="http://google.com">Посилання</a>
  </body>
</html>
```

Спробуємо зробити розмір заголовка трохи меншим, розмір тексту параграфа трохи меншим і курсивом, а посилання зробимо білим кольором на чорному фоні та ще й поставимо його в правому верхньому куті сторінки.

Ось CSS код, що для нас все це зробить:

Приклад CSS стилів

```
h1 {
  font-size: 16px;
}
p#paragraph {
  font-size: 14px;
}
a.my-link {
  position: absolute;
  top: 10px;
  right: 20px;
  display: block;
  width: 200px;
  height: 20px;
```

```
14    color: #ffffff;
15    background-color: #000000;
16  }
```

Якщо скопіювати даний кусок CSS коду в окремий файлик і під'єднати його наступним чином у наш HTML документ:

Під'єднуємо CSS файл у HTML документ

```
1  <html>
2    <head>
3      <meta charset="UTF-8" />
4      <title>Приклад HTML Сторінки</title>
5      <meta name="description" value="Мета опис сторінки" />
6      <link rel="stylesheet" href="main.css" />
7    </head>
8    <body>
9      <h1>Заголовок HTML сторінки</h1>
10     <p id="paragraph">Приклад параграфа тексту</p>
11     <br />
12     <a class="my-link" href="http://google.com">Посилання</a>
13   </body>
14 </html>
```

(зауважте на 6-ій стрічці як ми під'єднали наші стилі із файла під назвою *main.css* з допомогою тегу *link*) і знову відкрити наш HTML файл у веб переглядачі, тоді отримаєте щось подібне до цього:

Наша HTML сторінка із стилями

Синтаксис мови CSS є доволі простий і складається із блоків правил. Кожен блок правил має:

- *селектор* - ідентифікує, якому саме елементу на сторінці застосовуються правила даного блоку; у нашому вищенаведеному прикладі ми маємо 3 селектора: 1) *h1* (усі теги h1 на сторінці), 2) *p#paragraph* (тег параграфа з ідентифікатором *paragraph*), 3) *a.my-link* (усі посилання на сторінці, що мають клас *my-link*); назва тегу, клас та ідентифікатор є одними із найпоширенішими елементами селектора, проте їх існує величезна кількість і детальніше можна ознайомитись із списком елементів, які можна використовувати у селекторі в довіднику по CSS[41];
- *властивості* - блок правил, які йдуть одразу після селектора і є огорнуті у фігурні дужки; вони описують які саме властивості змінити (напр. положення елементів на сторінці, кольори, розміри, шрифти, декорації, фон та межі, і т.д.) в елементі на сторінці обраного селектором, що передує даному блоку властивостей; властивостей існує величезна кількість і їх усіх можна переглянути знову ж таки в довіднику по CSS[42]; а протягом книги ми будемо розбирати саме ті із них, які пригодяться нам у нашому проекті.

Крім того селектори працюють таким чином, що можуть перебивати один одного дозволяючи писати специфічніші стилі для одних елементів і залишаючи інші елементи незмінними. Це дуже потужна властивість мови CSS, яка дозволяє з легкістю унаслідувати стилі та уникати дубляжу в коді.

Детальніше про конкретні селектори і властивості HTML елементів будемо обговорювати в процесі роботи над проектом.

Мова браузерів - Javascript

Для покращення користувацького досвіду під час користування веб-сторінками було вирішено вбудувати мову програмування у веб браузери. Після багатьох спроб і різноманітних підходів стандартом у світі веб стала мова *Javascript*[43]. На початках використовувалась для вузького набору завдань із клієнтської

[41] http://css.manual.ru

[42] http://css.manual.ru

[43] *http://uk.wikipedia.org/wiki/JavaScript*

валідації форм та динамізації деяких елементів. Поява технології *AJAX*, яка дозволила з допомогою мови Javascript комунікувати із сервером, цілком змінила відношення до мови і тепер ця мова переросла в одну із найпопулярніших та використовується не лише на стороні браузера, а і для програмування серверної сторони веб-аплікацій.

Ось приклад мови Javascript:

Приклад Javascript кодylang=javascript

```
window.onload = function(){
    var mylink = document.getElementsByClassName('my-link')[0],
        paragraph = document.getElementById('paragraph');
    mylink.onclick = function() {
        if (paragraph.style.visibility == '') {
            paragraph.style.visibility = 'hidden';
        } else {
            paragraph.style.visibility = '';
        };
        return false;
    };
};
```

У ньому ми, одразу після завантаження сторінки, чіпляємо подію кліку мишки на наш лінк з попереднього прикладу із HTML документом із класом *mylink*. Після приєднання даного скрипта до нашої HTML сторінки параграф із текстом буде то ховатись, то показуватись.

Щоб самим потестувати даний код збережіть вище наведений кусок Javascript коду в окремий файл і під'єднайте його до нашої HTML сторінки наступним чином:

Під'єднуємо Javascript файл до HTML сторінки

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Приклад HTML Сторінки</title>
    <meta name="description" value="Мета опис сторінки" />
    <link rel="stylesheet" href="main.css" />
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>Заголовок HTML сторінки</h1>
    <p id="paragraph">Приклад параграфа тексту</p>
    <br />
    <a class="my-link" href="http://google.com">Посилання</a>
  </body>
</html>
```

На 7-ій стрічці HTML документу бачимо тег *script*, який посилається на файл *main.js*. Браузер, знайшовши даний тег, спробує піти на сервер за скриптом *main.js*, отримати його та запустити код, що у ньому знаходиться.

Спробуйте тепер, після підключення нашого Javascript коду, відкрити HTML документ і поклікати по посиланню. Якщо зробили все правильно, параграф тексту повинен ховатись та показуватись почергово при кожному кліку.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманний "в спадок" від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

Детальніше про мову Javascript можете почитати на сторінці вікіпедії[44]. Протягом книги ми з вами будемо працювати зазвичай із додатковими бібліотеками, що значно полегшать нашу роботу на стороні браузера, а не використовувати *голий* Javascript.

[44] http://uk.wikipedia.org/wiki/JavaScript

Специфіка фронтенду

Усі вище перечислені технології (HTML, CSS, Javascript) складають основу так званої фронтенд (front-end, ще називають її клієнтською) веб-розробки. Іншими словами тієї частина веб-аплікації, яка безпосередньо взаємодіє із користувачем у веб-браузері.

Кожна із цих мов має затверджений стандарт і є, відповідно, реалізованою на стороні веб переглядача. Але вже так історично склалось, що веб-браузерів на даний момент маємо далеко не один, і що ще більше погіршує ситуацію - кожен із них по різному і в різній мірі підтримує ці стандарти.

На даний момент найпопулярнішими веб-переглядачами є:

- Chrome[45],
- Firefox[46],
- Internet Explorer[47],
- Safari[48] десктопний та мобільний,
- Opera[49],
- мобільний Android Browser[50].

Таким чином те, що ви запрограмуєте на стороні клієнта і заставите чудово працювати у веб-браузері Chrome, може не зовсім коректно виглядати чи працювати у веб переглядачі Internet Explorer. І чим багатша на функціонал клієнтська сторона вашого веб-сайту, тим більша ймовірність, що вам прийдеться більше часу затратити на підпасовку вашої програми під решту веб-переглядачів.

[45] http://www.google.com.ua/intl/uk/chrome/
[46] https://www.mozilla.org/uk/firefox/new/
[47] http://windows.microsoft.com/uk-ua/internet-explorer/download-ie
[48] https://www.apple.com/safari/
[49] http://www.opera.com/uk
[50] http://www.android.com/about/

> Зазвичай власник проекту сам визначає, які переглядачі та користувачі його цікавлять. Відповідно до цих вимог вам, як розробнику, прийдеться адаптувати вашу веб-аплікацію для різних веб-переглядачів та платформ. Так, під різні платформи ваш сайт може виглядати по різному, навіть на тих самих версіях браузерів.

Лише з практикою та досвідом приходить вміння швидко вирішувати проблеми із різними версіями операційних систем та веб-переглядачів.

В той же час вже давно почали придумувати і реалізовувати масу різноманітних бібліотек та фреймворків, таких собі "надбудов" над основними мовами, які абстрагують вас від більшості відмінностей між платформами, операційними системами та різними веб-браузерами. І на даний час уже існує велика кількість бібліотек з допомогою, яких ваш HTML, CSS та Javascript код буде працювати в більшості випадків з першого разу для усіх браузерів.

У проекті даної книги ми використовуватимемо два таких додаткових інструменти:

- jQuery: Javascript бібліотека, яка дозволяє швидко реалізовувати багату на функціонал сторінку;
- Twitter Bootstrap: HTML/CSS/Javascript фреймворк, який дасть нам доступ до наперед визначених популярних на веб-сторінках віджетів; як от навігація, закладки, форми, кнопки, випадайки, лейаут сторінки і т.д.

jQuery - покращений Javascript

Якщо ми хочемо, щоб наш Javascript код працював одразу у більшості популярних веб-переглядачах, тоді нам прийдеться писати багато умовних операторів у нашому коді. Адже існує багато відмінностей в реалізації цієї мови у різних веб-браузерах, інші назви фукцій, інша робота з подіями та об'єктами на веб-сторінці (тегами та атрибутами).

Щоб уникнути перевитрат часу і не винаходити ровер заново, ми будемо користуватись однією із найпопулярніших Javascript бібліотек - jQuery[51].

Вона не лише дозволить нам з легкістю забути про різні версії браузерів, але й дасть величезну кількість додаткових функцій, що збережуть нам масу часу.

Ось лише порівняйте, як виглядатиме наш Javascript код із попереднього прикладу (де ми заставляли параграф тексту зникати на сторінці при кліку на посилання):

Javascript код з використанням jQuery

```
1  $(function(){
2    $('.my-link').click(function(){
3      var paragraph = $('#paragraph');
4      if (paragraph.is(':hidden')) {
5        paragraph.show();
6      } else {
7        paragraph.hide();
8      }
9      return false;
10   });
11 });
```

Так, рядочків коду зменшилось лише на один, проте їхня довжина значно коротша і, знаючи англійську, можна легко зрозуміти, що відбувається у кожному з них. В реальних проектах роль подібних бібліотек просто неоціненна!

Twitter Bootstrap

В нашому практичному проекті ми будемо створювати багато форм, елементів навігації, кнопок, полів і тому подібних елементів. В більшості випадків такі елементи є схожі між собою на різних веб-сайтах. Саме тому на даний момент існує багато фреймворків, які дозволяють швидко будувати найчастіше використовувані елементи на веб-сторінках без проведення великої кількості часу на їхній вигляд та функціонал.

[51] http://jquery.com

Дана книга зосереджена на веб-фреймворку Django та серверній розробці в першу чергу. В той час як ми з вами оглянемо повний цикл веб-розробки протягом практичного проекту, фронтенд розробка (зокрема верстка) не є основним акцентом даної книги. І саме тому ми скористаємось супер-популярним фреймворком Twitter Bootstrap[52] для створення більшості графічного інтерфейсу, який працюватиме в усіх популярних веб-переглядачах.

Таким чином використовуючи певні правила при написанні HTML тегів та атрибутів ми одразу матимемо готові веб елементи на наших сторінках. Це зекономить наш час при верстці сторінок.

Підключивши Twitter Bootstrap скрипти і стилі наступний доволі простий приклад HTML сторінки:

HTML сторінка з використанням Twitter Bootstrap

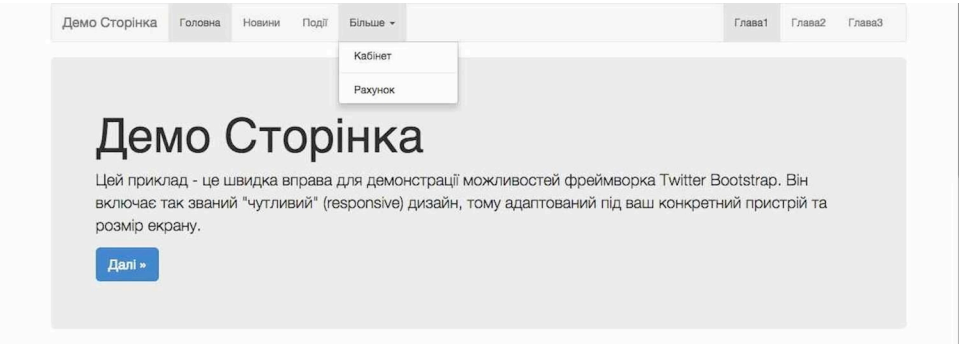
```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Приклад HTML Twitter Bootstrap Сторінки</title>
    <meta name="description" value="Мета опис сторінки" />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/boo\
tstrap/3.2.0/css/bootstrap.min.css">
    <script type="text/javascript" src="http://ajax.googleapis.com/a\
jax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/\
bootstrap.min.js"></script>
  </head>
  <body>

    <div class="container">

      <!-- Навігаційна панель -->
      <div class="navbar navbar-default" role="navigation">
        <div class="container-fluid">
          <div class="navbar-header">
```

[52] http://getbootstrap.com

```
21
22              <!-- Цей елемент стане у пригоді для малих екранів -->
23              <button type="button" class="navbar-toggle collapsed"
24                      data-toggle="collapse" data-target=".navbar-coll\
25  apse">
26                <span class="sr-only">Toggle navigation</span>
27                <span class="icon-bar"></span>
28                <span class="icon-bar"></span>
29                <span class="icon-bar"></span>
30              </button>
31              <!-- Лого -->
32              <a class="navbar-brand" href="#">Демо Сторінка</a>
33            </div>
34
35            <div class="navbar-collapse collapse">
36
37              <!-- Головна навігація -->
38              <ul class="nav navbar-nav">
39                <li class="active"><a href="#">Головна</a></li>
40                <li><a href="#">Новини</a></li>
41                <li><a href="#">Події</a></li>
42                <li class="dropdown">
43                  <a href="#" class="dropdown-toggle"
44                     data-toggle="dropdown">Більше
45                     <span class="caret"></span></a>
46                  <ul class="dropdown-menu" role="menu">
47                    <li><a href="#">Кабінет</a></li>
48                    <li class="divider"></li>
49                    <li><a href="#">Рахунок</a></li>
50                  </ul>
51                </li>
52              </ul>
53
54              <!-- Користувацькі лінки -->
55              <ul class="nav navbar-nav navbar-right">
```

```
56                <li class="active"><a href="#">Глава1</a></li>
57                <li><a href="#">Глава2</a></li>
58                <li><a href="#">Глава3</a></li>
59              </ul>
60
61            </div>
62          </div>
63        </div>
64
65        <!-- Main component for a primary marketing message or call to\
66  action -->
67        <div class="jumbotron">
68          <h1>Демо Сторінка</h1>
69          <p>Цей приклад - це швидка вправа для демонстрації можливост\
70 ей фреймворка Twitter Bootstrap. Він включає так званий "чутливий" (\
71 responsive) дизайн, тому адаптований під ваш конкретний пристрій та \
72 розмір екрану.</p>
73            <a class="btn btn-lg btn-primary" href="#" role="button">
74              Далі &raquo;</a>
75          </p>
76        </div>
77
78      </div>
79
80    </body>
81  </html>
```

набуде ось якого вигляду у вашому веб-браузері:



Приклад Twitter Bootstrap сторінки

Також спробуйте зменшити ширину вікна вашого веб-переглядача і побачите як класно верстка сторінки адаптується під нього.

Таким чином із допомогою Twitter Bootstrap ми з легкістю отримали:

- навігаційну панель із лого, основною навігацією та випадайкою, а також другорядною навігацією;
- тіло сторінки із гарно застиленим текстом та кнопкою “Далі”.

і при цьому лише написали HTML код із потрібними тегами та класами. Крім того ця сторінка гарно виглядає і на вузьких екранах. Лише уявіть скільки часу нам потрібно було б, щоб зверстати її з нуля, додати динаміки, заставити виглядати добре у різних браузерах та вужчих екранах.

Саме тому ми використовуватимемо даний фреймворк.

На сайті бібліотеки[53] можна детальніше ознайомитись із усіма доступними віджетами та функціоналом.

Бекенд

До цього моменту ми розглянули ту кухню, яка готує нам картинку у веб-переглядачі: HTML, CSS, Javascript та деякі надбудови над ними.

[53] http://getbootstrap.com

Але, звісно, часи, коли HTML код писали вручну в статичних файлах і клали їх на сервері у папочки давно минули. В даний час більшість сучасних веб-сайтів в інтернеті генерується на льоту з допомогою мов програмування, веб-фреймворків та систем менеджменту вмісту (CMS - Content Management System).

Саме тому наші з вами знання, як веб-девелоперів, не обмежуються мовами, що використовуються в браузері при верстці HTML сторінки. Однаково важливими є усі ті технології, що дають нам можливість мати динамічні веб-сайти із багатим функціоналом, отримувати та запам'ятовувати дані від користувача:

- *база даних*: уже із назви зрозуміло, що цей інструмент дозволяє зберігати та отримувати дані для веб-аплікації; використовувати ми будемо реляційну базу даних MySQL;
- *серверна мова програмування*: мова, якою ми отримуватимемо і оброблятимемо запити від користувачів, працюватимемо із базою даних та генеруватимемо HTML код для наших веб-сторінок; у нашому випадку це, звісно, буде мова програмування Python.

Бекенд (з англ. "back-end") ще називають серверною стороною. Тобто та частина веб-сайту, яка відповідає за обслуговування запитів користувача та генерацію коду для веб-сторінки.

Мова програмування Python

Більше, ніж половина нашого з вами часу буде присвячена написанню коду саме на мові Python.

З допомогою Python ми:

- організуємо структуру веб-адрес аплікації;
- налаштуємо проект;
- реалізуємо збереження та отримання даних із бази;
- запрограмуємо усі функціональні сторінки нашого веб-сайту;
- і ще багато іншого.

Ця мова є кросплатформеною та широкого призначення (звичайно ми її використовуватимемо суто для веб-розробки). Мова Python є динамічною інтерпритованою мовою з простим синтаксисом, що дає можливість програмісту надзвичайно швидко створювати програми та фокусуватись на вирішенні кінцевих проблем.

Ось приклад Python коду з реального Django проекту, який підтверджує, що дана мова є дійсно простою, лаконічною та зрозумілою:

Приклад Python коду із Django проекту

```
#stat_payments
class StatsPayments(TemplateView):
    template_name = 'aikido_school/stats_payments.html'

    @method_decorator(permission_required('aikido_school.stats_payme\
nts'))
    def dispatch(self, *args, **kwargs):
        return super(StatsPayments, self).dispatch(*args, **kwargs)

    def get_context_data(self,**kwargs):
        context = super(StatsPayments, self).get_context_data(**kwar\
gs)

        year = None
        month = None
        filter = None
        object_list = Group.objects.all()
        if self.request.GET.get('month') and self.request.GET.get('y\
ear'):
            year = self.request.GET.get('year')
            month = self.request.GET.get('month')
            filter = Q(student__fee__fee_date__year=year)&\
                     Q(student__fee__fee_date__month=month)
        if filter is not None:
            object_list = object_list.filter(filter)
        object_list = object_list.annotate(sum=Sum('student__fee__pr\
```

```
27 ice'),
28             count=Count('student__fee__price'))
29
30         form = StatsPaymentsForm(self.request.GET or None, initial={
31             'month': datetime.today().month, 'year': datetime.today(\
32 ).year})
33
34         context['object_list'] = object_list
35         context['one_column'] = True
36         context['form'] = form
37         if year and month:
38             context['year'] = year
39             context['month'] = _(month_name[int(month)])
40
41         return context
```

Якщо ви взялись за дану книгу, то це означає, що у вас уже є мінімальна база з мови програмування Python. Якщо ж ні, тоді зверніться до Вступу та перегляньте наданий там список матеріалів з необхідним мінімумом інформації.

Як тест для ваших знань спробуйте переглянути та зрозуміти поверхнево (адже код працює у фреймворку Django, який нам ще треба буде освоїти) логіку та конструкції мови Python у класі наведеному вище.

Синтаксис мови, ООП та інші базові концепції мови ми не будемо розбирати в процесі проекту, а звертатимемо увагу лише на найважливіші аспекти веб-програмування, Django і комунікацію між сервером та веб-переглядачем.

База даних MySQL

Для зберігання даних нашої аплікації ми спочатку скористаємось простою базою даних *sqlite3*, а вже пізніше переключимось на реляційну базу даних MySQL[54]. Це одна із найпоширеніших баз даних, яку ми можемо безкоштовно використовувати у наших цілях.

[54] http://uk.wikipedia.org/wiki/MySQL

Ця база даних дасть нам можливість:

- зберігати дані;
- робити пошук серед даних;
- та, звісно, отримувати дані у потрібному форматі.

Веб-фреймворк Django дасть нам усі необхідні інструменти для роботи з даними таким чином, що нам практично не доведеться мати справу напряму із SQL запитами. Більшість завдань із даними ми зможемо реалізувати не відходячи від мови Python.

Тому в книзі ми лише поверхнево оглянемо мову запитів реліційної бази даних - SQL. Короткий огляд найбільш необхідних інструментів та команд при роботі з базою MySQL можете додатково знайти у конспекті-шпаргалці, що йде разом із *Рекомендованим* пакетом книги.

Веб-фреймворк Django

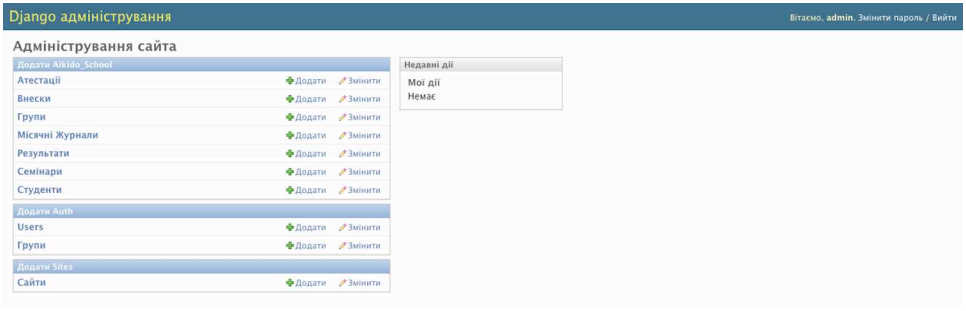
Згодом, коли використання мови програмування для генерації HTML сторінок уже стало буденною річчю, програмісти не зупинились і продовжили удосконалювати процес створення веб-сайтів. Після кожного завершеного веб-проекту набуті навички, підходи та код переносились та адаптувались під нові проекти таким чином перетворюючись на маленькі фреймворки для швидкого створення нових веб-аплікацій. Адже для чого придумувати колесо, якщо можна використати попередній код перед цим зробивши його універсальнішим.

Веб-фреймворк - це набір інструментів, бібліотек, аплікацій, заготовок, що забезпечують стандартний та найбільш затребуваний набір функціоналу для переважної більшості веб-аплікацій та сайтів.

Одним із найпоширеніших веб-фреймворків на мові Python є Django[55]. Він дозволяє із блискавичною швидкістю створювати прототипи веб-сайтів і надає необхідний мінімум функціоналу:

[55] https://www.djangoproject.com/

- *моделі*: місток між Python класами та базою даних, який робить усю складну роботу щодо збереження та пошуку даних для вас;
- *шаблони*: генерація HTML коду без використання Python мови, натомість із мінімальними вкрапленнями логіки з допомогою динамічних Django тегів напряму у HTML код; це дає можливість нам мати так званий MVC[56] підхід та розділяти логіку від даних та представлення;
- *диспетчер URL*: інструмент, з допомогою якого можемо легко будувати ієрархію URL адрес нашого веб-сайту;
- *адміністративна частина*: іде разом із Django та дозволяє керувати даними веб-сайту та його налаштуваннями без додаткової розробки;



Django адміністративна частина

- *користувачі*: якщо ваша веб-аплікація потребує логованих користувачів, тоді нічого додаткового практично не доведеться розробляти; Django дає весь функціонал, що дозволить вам мати дані користувачів, форми логування, реєстрації та усі налаштування з безпеки, ролі та дозволи для користувачів;
- *форми*: додаткові Python класи для роботи із веб-формами; вони економлять масу часу при розробці стандартних форм;
- *розробницькі інтерфейси (Development APIs)*: найпоширенішим є REST[57]; з Django можна легко навчити вашу веб-аплікацію "говорити" у форматі REST, який часто буває корисним для мобільних додатків та комунікації із іншими сервісами;

[56] http://bit.ly/vpmvc
[57] http://uk.wikipedia.org/wiki/REST

- *міграція*: такий інструмент як South дозволяє легко змінювати та мігрувати ваші дані в базі, їхній формат та вигляд, а також робити це повторно без будь-яких проблем;
- як і у будь-якому іншому веб-фреймворку такі речі як деплоймент на кінцевий сервер, документація, автоматичні тести, атомарна розробка (модульність), розробницькі інструменти, є добре продумані у фреймворку Django; вони допоможуть вам швидше та з легкістю закінчувати ваші веб-проекти.

Крім того, що можна користуватись усіма вбудованими функціями фреймворку, також існує маса бібліотек та аплікацій.

Ну і ще один не менш важливий аргумент за Django - це OpenSouce фреймворк. Його код можна вільно використовувати, читати та змінювати.

> В більшості випадків напряму змінювати код OpenSource бібліотек у власних проектах є поганою ідеєю. Це в майбутньому дасть вам багато проблем при спробі оновлення проекту до нової версії цієї бібліотеки. Натомість Python є настільки динамічною мовою, що дозволяє практично будь-які зміни у код робити ззовні, із власного коду. Так званий “monkey patching”.

Репозиторій коду Git

Ця секція немає напряму відношення до веб-розробки. Репозиторій коду - це швидше потреба для кожного проекту з програмним кодом. Особливо, якщо над ним працює більше, ніж одна людина.

Репозиторій коду - це місце збереження вихідного коду та інших матеріалів програмного продукту, яке забезпечує правильний формат зберігання та:

- одночасну роботу двох і більше людей над проектом;
- історію змін в коді;

- одночасну роботу над кількома різними релізами та функціоналом незалежно один від одного;
- резервну копію коду.

Наш проект ми будемо зберігати в репозиторії коду Git[58]. Протягом останніх років він набув неабиякої популярності як мінімум в OpenSource спільнотах та проектах. А публічний сервіс для зберігання проектів в репозиторіях Git - github.com[59] став передовим для OpenSource проектів.

Під час роботи над проектом ми регулярно зберігатимемо зміни в репозиторії притримуючись кращих практик. Тому по завершенню книги ви не лише володітимете азами Django фреймворку та веб-розробки, але й непогано працюватимете із репозиторієм.

Для загального ознайомлення із принципом роботи репозиторію коду рекомендую ознайомитись із статею про Git на мому блозі: Що таке репозиторій коду або ЛікБез по Git[60].

Також із Рекомендованим пакетом книги йде конспект-шпаргалка щодо використання репозиторію Git.

Домашнє завдання

Отже, в даній главі ми коротко пройшлись по основних аспектах веб-розробки. Оглянули принцип комунікації сервер-клієнт, протокол HTTP, фронтенд розробку (HTML, CSS, Javascript, jQuery, Twitter Bootstrap), серверну сторону (Python, MySQL, Django), репозиторій коду Git.

Протягом розробки нашого з вами практичного проекту ми будемо детальніше розбирати потрібні нам аспекти кожної із цих технологій. Але, якщо ви поки лише початківець у веб-розробці, тоді перед переходом до наступної глави рекомендую детальніше ознайомитись із ними. Це дасть вам можливість краще засвоїти матеріал книги.

Отже, на домашнє завдання затратьте кілька годин і:

[58] http://uk.wikipedia.org/wiki/Git
[59] https://github.com/
[60] http://www.vitaliypodoba.com/2014/06/git-basics/

- знайдіть та пройдіть невеличкий онлайн туторіал або курс по мові HTML на зручній для вас мові; в інтернеті є маса доступних матеріалів як на англійській так і на російській та українській мовах;
- те ж саме із CSS; достатньо навіть вище згаданих посилань на статті у вікіпедії;
- ознайомтесь із мовою Javascript;
- також із мовою запитів реляційних баз даних: SQL;
- перечитайте та поекспериментуйте із Лікбезом з Git репозиторію;
- і загалом, усі наведені посилання на зовнішні ресурси у даній главі є важливими для подальшого ознайомлення.

Python не згадав, адже це само собою розуміється. Без нього нікуди.

Усю теорію, що набудете в результаті виконання даного домашнього завдання ми з вами успішно застосуємо на практиці в подальших главах. Також не забувайте одразу експериментувати із прикладами при проходженні та засвоєнні теоретичних матеріалів.

...

У наступній главі ми дамо визначення нашому практичному проекту, сформулюємо основні завдання та специфікації, а також перелічимо усі ті навички, що отримаємо в результаті імплементації даного проекту.

3. Проект: база даних для обліку студентів

Коли я починав роботу над даною книгою, то перша ідея була взяти за проект розробку персонального блога програміста. І вже написавши код для близько половини проекту я зрозумів, що він не є показовим та бракує специфікацій для важливих аспектів веб-розробки, як от робота з формами, автентифіковані користувачі, інтенсивна робота з базою даних і ще кілька речей, які, я вважав, є необхідними навіть для початківців.

На той час я вже мав кількох студентів на персональному менторстві за напрямком веб-розробка з Python та Django, а також мав невеликий список типових задач, що дають Python джунікам.

Тому, щоб не видумувати велосипед, я взяв найпоширеніший тестовий проект, розширив його деяким додатковим функціоналом таким чином, щоб в кінцевому результаті надати необхідний мінімум знань та практики початківцю.

> Список тестових проектів у мене був ще від часів, коли проходив інтерв'ю на інших фірмах, а також дехто із студентів уже пробували подаватись на позиції джуніорів і ділились зі мною своїм досвідом співбесіди. Ці ж задачки я час до час використовую під час пошуку джуніора Python програміста у свою власну фірму. Зазвичай такий тестовий проект я даю кандидату на один-два тижні розробки, взалежності від складності проекту.

Специфікації проекту

У цій книзі ми займемось розробкою бази даних для обліку студентів. Основними сутностями цієї бази будуть студенти та групи.

Кожен студент матиме мінімальний набір необхідних полів (усі поля є обов’язковими):

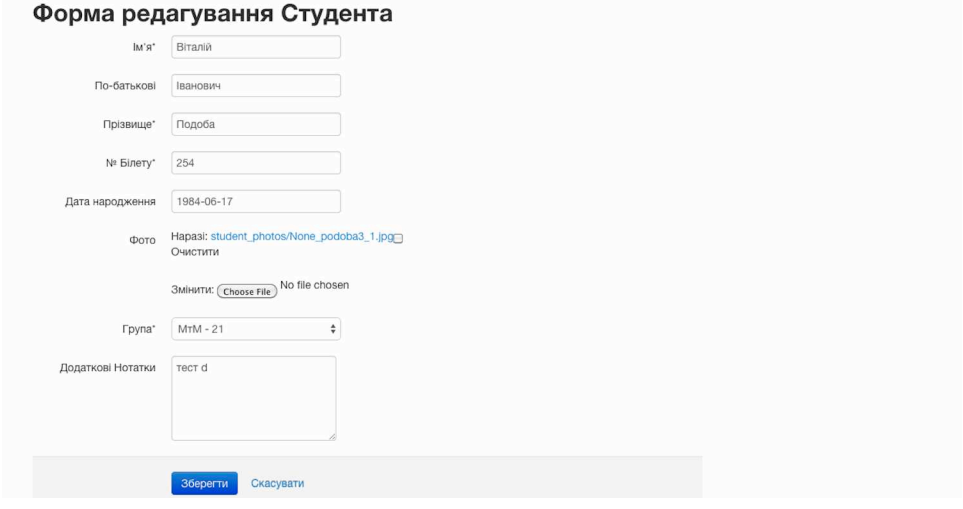
- повне ім’я: ім’я, прізвище та по-батькові;
- дата народження;
- фото студента;
- номер студентського білету;
- група;
- та додаткові нотатки.

Група в свою чергу матиме:

- назву (обов’язкове поле);
- старосту групи - одного із студентів (не обов’язкове поле);
- та додаткові нотатки.

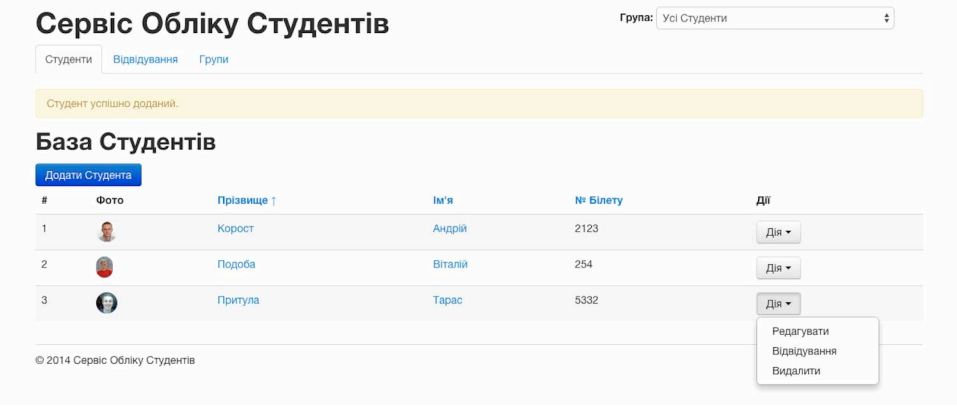
Наша аплцікація дозволятиме:

- додавати нових студентів та групи;
- редагувати існуючих студентів та групи;



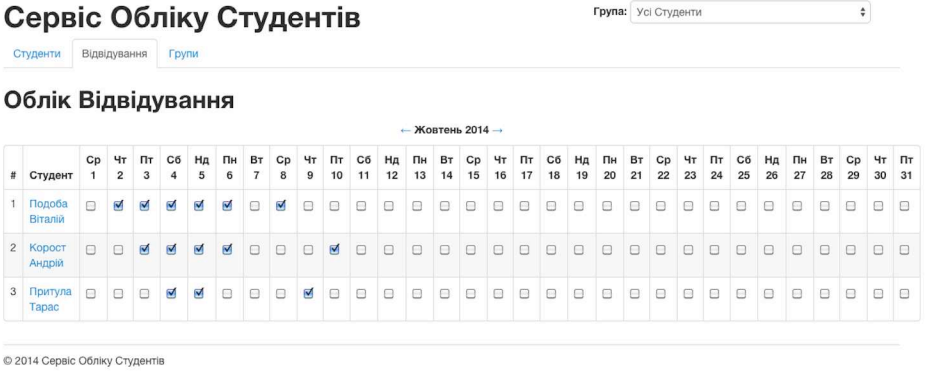
Форма редагування студента

- видаляти існуючих студентів та групи;
- переглядати списки існуючих студентів та груп, а також переглядати студентів в контексті обраної групи;



Сторінка із списком студентів

- вести журнал присутності студентів;



Журнал відвідуваності студентів

- працювати лише залогованим користувачам;
- запускати окремий скрипт в командній стрічці, щоб отримати список студентів у вказаній групі;

Крім того вимоги проекту включатимуть:

- адаптацію усього інтерфейсу нашої веб-аплікації під українську мову засобами інтернаціоналізації в Django;
- виділення частину коду проекту в окрему аплікацію з подальшою можливістю використання коду в інших проектах;
- автоматичний тест для одного користувацького сценарію;
- ведення журналу щодо створення, редагування та видалення студентів і груп.

Ну і зрозуміло, що наприкінці виконаної роботи потрібно буде перенести проект і налаштувати програму на кінцевому сервері.

Чого ви навчитесь протягом проекту?

Таким чином після реалізації проект згідно вище зазначених специфікацій ми з вами навчимось:

- налаштовувати своє робоче середовище, створювати проект та аплікації в Django;
- працювати з базою даних з допомогою Django моделей;
- верстати форми, елементи навігації та динамізувати сторінки з допомогою HTML, CSS, Twitter Bootstrap, Javascript бібліотеки jQuery та технології Ajax[61];
- будувати Django в'юшки (від англ. view - вид, бачити) та працювати із динамічними шаблонами;
- перекладати (інтернаціоналізувати) інтерфейс веб-аплікації;
- будувати структуру веб-адрес сайту в Django з допомогою диспетчера адрес;
- писати власну Django "мідлвару" (middleware[62]);
- створювати Django процесор контексту;

[61] http://uk.wikipedia.org/wiki/AJAX
[62] http://bit.ly/vpmiddleware

- писати Django команду;
- реалізувати автоматичний тест;
- працювати із Django сигналами (подіями) та Python модулем логування;
- створювати *кастомний* шаблонний тег;

> Слово "*кастомний*" (від англ. custom) я буду часто вживати у даній книзі разом із словами "код", "рішення", "проект". Воно може вживатись у одному із наступних значень: власний, окремий, під замовлення або "на ключ". Тобто *кастомний код* - це код, який спеціально написаний під даний проект. Також, "*кастомізувати*" означатиме змінювати дефолтну поведінку чи функціонал, зазвичай в Django.

- працювати із Django адмінкою та робити невеликі зміни до її інтерфейсу;
- обробляти веб-форми як з допомогою кастомного Python коду на сервері так і з допомогою заготованих в Django класів форм;
- використовувати вбудовану в Django систему авторизації та автентифікації, щоб дозволити лише залогованим користувачам використовувати нашу аплікацію;
- деплоїти проект на продакшин сервер.

> Продакшин сервер (з англ. production - виробництво) - це кінцевий комп'ютер, на якому встановлений програмний продукт для його використання кінцевими користувачами.

> Деплоймент (з англ. deployment - розгортання) - це процес, що робить програмний продукт доступним для його користувачів. Зазвичай він полягає у перенесенні коду програми на кінцевий сервер (так званий “продакшин”), звідки він обслуговує кінцевих користувачів.

Що не входить у даний проект?

Завдяки роботі над проектом у даній книзі ви набудете усіх необхідних знань, щоб претендувати на посаду джуніора веб-розробника на Django фреймворку.

Даний фреймворк покриває і більш обширні та складніші теми, які є важливими для освоєння при роботі над великими та високо-навантаженими проектами. Але початківцю вони зазвичай не потрібні та і є занадто складними для перших кроків. Відповідно їх зазвичай не вимагають від джуніорів.

Тому дані теми не входять у дану книгу та швидше за все будуть розкриті у наступних моїх навчальних матеріалах, уже для тих, хто осягнув основи веброзробки з допомогою Django:

- monkey patching[63] в контексті Django;
- реалізація кастомних полів для Django моделей;
- поглиблена кастомізація адмін частини Django;
- оптимізація швидкодії Django сайту;
- робота на низькому рівні із базою даних (SQL запити, транзакції, оптимізація швидкодії при роботі з базою);
- безпека при розробці на Django;
- кешування в Django;
- реалізація REST API;
- автоматичне тестування, юніт тести і т.д. лише поверхнево оглянемо на прикладі тесту одного користувацького сценарію поведінки.

[63] https://ru.wikipedia.org/wiki/Monkey_patch

Частина даних тем є розкритою в серії постів у мене на блозі: Кращі практики розробки з Django[64]. В майбутньому я планую присвятити окремі серії постів таким темам як безпека та швидкодія в Django.

Домашнє завдання

Тепер ви уже можете собі уявити, що ми з вами будуватимемо та який об'єм робіт на нас чекає. На перший погляд списки вимог досить довгі, особливо як для початківців. Але не варто засмучуватись, адже фреймворк Django для кожного із завдань підготував для нас цілий ряд інструментів, які дуже спростять наше з вами життя.

Професійний програміст, який знається на розробці під Django, може зазвичай виконати даний проект в межах 2-3х днів. Джуніор в межах 1-2х тижнів. А початківець, який з нуля вивчає веб-програмування та фреймворк Django, від 1-го до 6ти місяців (за умови, що вже володіє азами серверної мови програмування). Принаймні так показує мій досвід роботи із початківцями.

На домашнє завдання цієї глави пропоную вам детальніше переглянути зображення нашої аплікації, що наведені вище та подумати як можна їх покращити, у чому бачите недоліки.

А також відвідайте сторінку Django у вікіпедії, зокрема секцію "*Можливості*[65]" та ознайомтесь детальніше із набором інструментів та функціоналу, який даний фреймворк пропонує одразу по встановленні. Якщо щось не буде зрозуміло - нічого страшного, це завдання швидше для ознайомлення, щоб ви могли зорієнтуватись в можливостях Django.

...

В наступній главі ми вже нарешті переходимо до практики. Налаштуємо наше робоче середовище, підготуємо редактор коду, заінсталюємо необхідні інструменти, створимо свій перший Django проект і побачимо дефолтну сторінку аплікації. Тому пропоную зробити паузу, приділити час та увагу

[64] *http://www.vitaliypodoba.com/2014/07/django-dev-env-hints*
[65] *http://bit.ly/vpdjangofeatures*

домашньому завданню, запастись чаєм чи кавою, а вже тоді рухатись далі до наступної глави.