Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисципліни
"WEB-ПРОГРАМУВАННЯ"
для студентів напряму підготовки
6.050101 “Комп’ютерні науки”
всіх форм навчання

2017

Методичні вказівки до лабораторних робіт з дисципліни "Web-програмування" для студентів напряму підготовки 6.050101 “Комп’ютерні науки” всіх форм навчання / Укл. Т.В. Федорончак. – Запоріжжя: ЗНТУ, 2017. – 59 с.

Автори: Т.В. Федорончак, к.т.н., доцент

Рецензент: Т.О. Колпакова, к.т.н., ст. викладач

Відповідальний
за випуск: С.О. Субботін, д.т.н., професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол №11
від "10" липня 2017 р.

ЗМІСТ

Вступ.....4
Лабораторна робота № 1 Розмітка макету веб-сторінки за допомогою HTML5 та CSS3 5
Лабораторна робота № 2 Розробка респонсивного макету веб-сторінки 11
Лабораторна робота № 3 Знайомство з мовою PHP та MVC фреймворком CodeIgniter.....26
Лабораторна робота № 4 Створення веб-застосунку за допомогою фреймворку CodeIgniter 42
Лабораторна робота № 5 Створення складних інтерфейсів веб-застосунків за допомогою JavaScript та AJAX запитів 51
Література.....59

ВСТУП

Практична частина курсу «Web-програмування» призначена для отримання практичних навичок, необхідних для вирішення питань, пов'язаних із розробкою клієнт-серверних застосунків у глобальній мережі Інтернет з використання сучасних інструментальних засобів, технологій, мов програмування та загальноприйнятих стандартів.

Для виконання практичних завдань з курсу використовується наступне програмне забезпечення:

– XAMPP - багатоплатформова збірка веб-сервера, що містить Apache, MySQL, інтерпретатор скриптів PHP, мову програмування Perl та велику кількість додаткових бібліотек, що дозволяють запустити повноцінний веб-сервер;

– браузери Mozilla Firefox, Google Chrome, Opera, Internet Explorer зі встановленими інструментами для розробників (наприклад, FireBug);

– Komodo Edit 10 (чи інша IDE для веб-розробки, в тому числі демонстраційні версії платних застосунків, наприклад, phpStorm);

– phpMyAdmin - веб-застосунок з відкритим кодом на мові PHP із графічним веб-інтерфейсом для адміністрування СКБД MySQL.

В роботах також використовується безкоштовний MVC фреймворк CodeIgniter.

Лабораторна робота № 1
Розмітка макету веб-сторінки за допомогою HTML5 та CSS3

1.1 Мета роботи

Ознайомитися з підходами та технологіями, що використовуються для розмітки сучасних веб-сторінок, та розмітити веб-сторінку за допомогою мови розмітки HTML5 та каскадних таблиць стилів CSS3.

1.2 Основні теоретичні відомості

Розмітка (верстка) веб-сторінки – це процес створення веб-сторінки засобами мови HTML та CSS із попередньо створеного макету дизайну сайту, заздалегідь намальованого за допомогою графічних редакторів.

Таблична розмітка раніше була основним методом створення веб-сторінки (HTML3). Однак сьогодні вона використовується в основному лише для відображення табличних даних.

Розмітка за допомогою блоків (тег) та каскадних таблиць стилів (CSS), що описують їхнє оформлення, реалізує концепцію семантичної розмітки. При цьому розділяється зміст сторінки та її оформлення. Концепція семантичної розмітки найшла найбільше втілення в мові HTML5.

Шари (або ж блоки) представляють собою структурні елементи, які можна розміщувати на веб-сторінці шляхом накладання їх один на одного з точністю до піксела.

Скрипти (наприклад, на мові JavaScript) дозволяють динамічно змінювати параметри шарів. Шари можна переміщувати, ховати та відображати без перезавантаження всієї сторінки. Це надає можливість створювати на сторінці різні ефекти, такі як анімація, спадаюче меню, спливаючі підказки, плаваючі вікна тощо.

Кожна веб-сторінка, як правило, містить наступні стандартні блоки (рис. 1.1):

- header;
- logo;
- navigation/menu;
- content area;
- footer.

Крім того, основа макету сторінки зазвичай складається з однією, двох, трьох або чотирьох колонок.

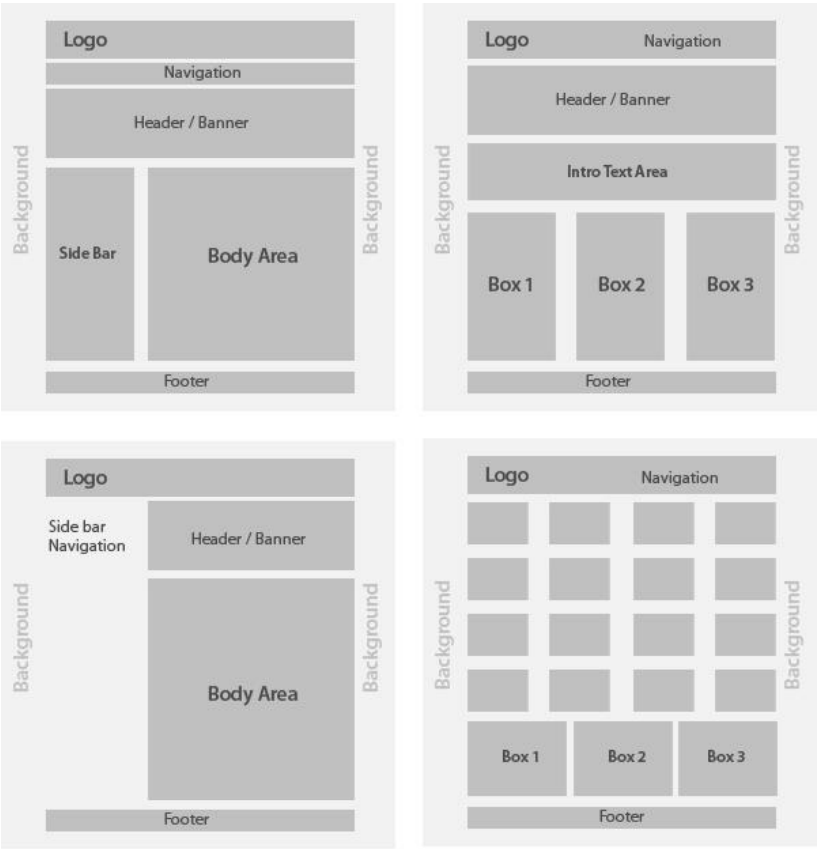

Рисунок 1.1 – Приклади макетів веб-сторінок

Всі макети сайтів, тобто типи розмітки сторінок можна розділити на такі принципові групи:

– фіксовані (fixed),
– гумові (fluid),
– респонсивні (responsive),
– адаптивні (adaptive).

Фіксований тип макету - дизайн, в якому ширину елементів сторінки (стовпців, рисунків) задано точно в пікселях.

Переваги:

– легко розробляти;

– заздалегідь відомо як буде виглядати сайт.

Недоліки:

– для сайту існує лише один ідеальний розмір екрану;

– в деяких випадках може з'явитися скролінг.

Гумовий тип макету – дизайн, в якому ширину елементів сторінки задано у відсотках від поточного розміру екрану чи батьківського елементу.

Переваги:

– сайт буде заповнювати весь простір браузера, що значно поліпшує його вигляд;

– сайт буде однаково виглядати на різних розмірах екрану.

Недоліки:

– розробка такого дизайну дуже складна;

– на великих екранах можлива поява дуже довгих строк.

Для підлаштування макету під розмір екрану пристрою використовують респонсивний та адаптивний типи макетів.

Адаптивний дизайн сегментує користувачів на категорії за типом пристрою (desktop, tablet, mobile). На сервері в залежності від цього готується відповідна версія дизайну.

Респонсивний тип макету - дизайн, якій динамічно масштабує вміст сторінки під розмір екрану, в том числі може відбуватися переміщення блоків з одного місця на інше, або їхня заміна іншими блоками, що відображаються лише при визначеній роздільній здатності. Респонсивна верстка прийшла на зміну ідеї створення спеціальних мобільних версій сайтів, розташованих на окремих піддоменах (наприклад, m.wikipedia.org).

Переваги:

– сайт буде відображатися на різних розмірах екрану так, як це найбільш зручно користувачеві.

Недоліки:

– потребує старанного опрацювання декількох макетів для різних розмірів екранів.

Респонсивний дизайн будується за допомогою медіазапитів (media queries), флексбокс (flexbox) елементів та на так званих решітках (grid-layouts).

Також може використовуватися змішаний тип дизайну.

Валідність HTML-верстки – це її відповідність стандартам організації The World Wide Web Consortium (W3C). Відсутність помилок в розмітці документу - це один з головних показників якості розмітки. Перевірка розмітки на помилки та відповідність стандарту може бути проведена автоматично за допомогою он-лайн сервісу W3C чи за допомогою різних програм «валідаторів».

1.3 Завдання на лабораторну роботу

1.3.1 Навчитися позиціювати блочні елементи сторінок. Навчитися формувати макети веб-сторінок фіксованої ширини та такі, що розтягуються; макети з 1, 2 чи 3 колонок.

1.3.2 Обрати безкоштовно розповсюджуваний макет сайту в форматі PSD та розмітити за допомогою блоків веб-сторінку.

Приклад макету сторінки, що може бути використаний в роботі.

Рисунок 1.2 – Приклад макету для розмітки

1.3.3 Перевірити відображення макету в різних браузерах та виправити за необхідності помилки. При роботі для налагодження макету скористатися вбудованими в браузери інспекторами коду.

1.3.4 Провести валідацію макету.

1.4 Зміст звіту

1.4.1 Тема та мета роботи.

1.4.2 Результати роботи: загальний вигляд макету, схематичне подання макету, в якому відобразити виділені змістовні блоки, фрагменти коду HTML та CSS.

1.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

1.5 Контрольні запитання

1.5.1 Що таке HTML та CSS? В чому полягає підхід розмежування змісту та подання веб-сторінки?

1.5.2 З чого складається HTML документ? Що таке тег та елемент документу? Наведіть приклади тегів.

1.5.3 Призначення тегу метаданих веб-сторінки. Наведіть приклади.

1.5.4 В чому різниця між block та inline елементами HTML сторінки? Наведіть приклади. Поясніть поняття потоку документу.

1.5.5 Що таке спеціальні символи (сутності) HTML? Навіщо вони? Наведіть приклади.

1.5.6 Які недоліки та переваги розмітки сторінок за допомогою таблиць та блоків?

1.5.7 Які є способи додання стилів до документу?

1.5.8 Для чого застосовуються атрибути class та id елементів сторінок?

1.5.9 Що таке CSS селектори? Які вони бувають? Як обрати елемент з певним ідентифікатором, певним класом, певний тип елементів?

1.5.10 Що таке контекстні селектори?

1.5.11 Що мається на увазі під каскадним застосування стилів до елементів документу?

1.5.12 Що таке наслідування стилів?

1.5.13 Що таке псевдокласи CSS? Наведіть приклади.

1.5.14 Які є типи макетів сторінок? В чому їхні переваги та недоліки?

1.5.15 Що таке крос-браузерна верстка?

1.5.16 Які стандартні змістові блоки як правило містять веб-сторінки?

1.5.17 Що таке боксова модель? Які бувають типи боксових моделей? Як її змінити?

1.5.18 Що таке потік документу? Яким чином можна їм керувати?

1.5.19 Поясніть позиціювання елементів за допомогою властивості position.

1.5.20 Для чого необхідна CSS властивість float? Для чого необхідна властивість clear?

Лабораторна робота № 2
Розробка респонсивного макету веб-сторінки

2.1 Мета роботи

Ознайомитися з поняттям респонсивного макету і технологіями, що його реалізують, та навчитися розробляти HTML-сторінки, вигляд яких підлаштовуються під розмір екрану користувача.

2.2 Основні теоретичні відомості

Число користувачів, що виходять в Інтернет з мобільних пристроїв, стрімко зростає, але, на жаль, більшість мережевих ресурсів не пристосоване для цього. Для кожного мобільного пристрою необхідний особливий підхід до розташування контенту на екрані через обмежений розмір дисплея.

В респонсивному веб-дизайні (рис. 2.1) макет підлаштовується під пристрій, виходячи з його можливостей і розміру. Наприклад, певний контент на телефоні розташовується в одній колонці, а на планшеті – в двох.

Респонсивний дизайн будується за допомогою медіазапитів (media queries), флексбокс (flexbox) елементів та на так званих решітках (grid-layouts).

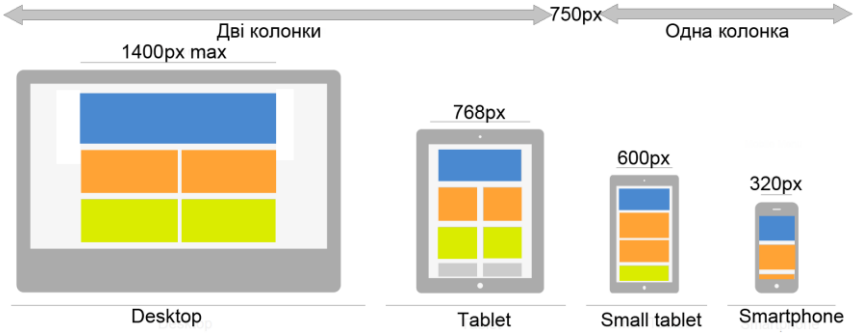

Рисунок 2.1 – Поняття респонсивного дизайну

2.2.1 Налаштування області перегляду

Страниці, що адаптуються до перегляду на різних пристроях, повинні мати в контейнері head метатег viewport. Він повідомляє браузеру, яким чином потрібно контролювати розміри і масштаб вікна

перегляду.

Додайте width=device-width, щоб адаптувати ширину вікна перегляду до екрану пристрою, та initial-scale=1, щоб забезпечити співвідношення 1:1 між пікселями CSS та незалежними пікселями пристрою, тобто:

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
```

2.2.2 Поліпшення чуйності макету за допомогою медіазапитів CSS

Медіазапити – це прості фільтри, які можна застосовувати до стилів CSS. Вони дозволяють змінювати стилі на підставі характеристик пристрою, повязаних з відображення контенту, включаючи тип, ширину , висоту, орієнтацію і навіть роздільну здатність екрану.

Наприклад, ви можете помістити до медіазапиту print всі стилі, необхідні для друку:

```
<link rel="stylesheet" href="print.css"
media="print">
```

Крім використання атрибута media в посиланні на таблицю стилів існує ще один спосіб застосувати медіазапити @media – їх можна вбудувати в файл CSS.

```
@media print {
  /* тут стилі для друку */
}
```

За допомогою медіазапитів можна створити чуйне середовище, в якому до кожного розміру екрану будуть застосовуватися відповідні стилі. Синтаксис медіазапитів допускає створення правил, які застосовуються на підставі характеристик пристрою.

```
@media (query) {
  /* CSS правила, які застосовуються коли є
співпадіння з запитом */
}
```

В респонсивному веб-дизайні найбільш часто використовуються функції min-width, max-width, min-height і max-height (хоча можливі й інші запити).

```
<link rel="stylesheet" media="(max-width: 640px)"
href="max-640px.css">
<link rel="stylesheet" media="(min-width: 640px)"
href="min-640px.css">
```

```
<link      rel="stylesheet"      media="(orientation:
portrait)" href="portrait.css">
<link      rel="stylesheet"      media="(orientation:
landscape)" href="landscape.css">
<style>
  @media (min-width: 500px) and (max-width: 600px) {
    h1 {
      color: fuchsia;
    }
    .desc:after {
      content:" In fact, it's between 500px and 600px
wide.";
    }
  }
</style>
```

В наведеному прикладі:

– при ширині браузера від 0 px до 640 px застосовуються стилі з файлу max-640px.css;

– при ширині браузера від 500 px до 600 px застосовуються стилі з @media;

– при ширині браузера від 640 px та вище застосовуються стилі з файлу min-640px.css;

– якщо в браузері ширина більша за висоту, застосовуються стилі з файлу landscape.css.

– якщо в браузері висота більша за ширину, застосовуються стилі з файлу portrait.css.

Основний принцип респонсивного веб-дизайну (і головна відмінність від макетів з фіксованою шириною) – рухливість і пропорційність. Використання відносних розмірів допомагає спростити макет і запобігти випадковому створенню компонентів, які не вміщаються в область перегляду.

Наприклад, установка параметра width рівним 100% для блоку div верхнього рівня призведе до того, що він буде заповнювати всю ширину області перегляду і ніколи не буде занадто малий або великий для неї.

Крім того, використання відносних одиниць дозволяє браузерам відображати контент, виходячи з призначеного для користувача масштабу. Це означає, що горизонтальна панель прокрутки на сторінці не знадобиться.

Контрольними точками респонсивного дизайну називаються

точки (ширина, висота), при яких змінюються стилі. Контрольні точки створюються на основі особливостей контенту, а не для окремих пристроїв.

Як правило, спочатку розробляється дизайн для самого маленького мобільного пристрою (mobile first), а потім здійснюється перехід до версій для великих екранів. При цьому необхідно уважно вибрати інформацію, яка буде показана або прихована в залежності від розміру екрана. Не забирайте контент тільки тому, що не можете розмістити його на екрані. Потреби користувачів не залежать від його розміру.

2.2.3 Прийоми респонсивного веб-дизайну за допомогою flexbox елементів

Прийоми респонсивного веб-дизайну швидко розвиваються, але є багато перевірених варіантів, які добре працюють при використанні як настільних комп'ютерів, так і мобільних пристроїв.

Модуль Flexbox-розмітки (flexible box - «гнучкий блок», на сьогодні W3C Candidate Recommendation) ставить завдання запропонувати більш ефективний спосіб розмітки, вирівнювання і розподілу вільного місця між елементами в контейнері, навіть коли їх розмір невідомий і/або динамічний (звідси слово «гнучкий»).

Головна задумка flex-розмітки в наділенні контейнера здатністю змінювати ширину, висоту і порядок своїх елементів для найкращого заповнення простору (в більшості випадків – для підтримки всіх видів дисплеїв і розмірів екранів). Flex-контейнер розтягує елементи для заповнення вільного місця або стискає їх, щоб запобігти виходу за межі.

Більшість макетів, що використовуються для створення респонсивних веб-сторінок, можна віднести до однієї з наступних пятьох категорій або їх комбінації. Ці шаблони, які спочатку були визначені Люком Вроблевськи (Luke Wroblewski), є надійною відправною точкою для створення будь-якої респонсивної сторінки.

Mostly Fluid (Здебільшого гумовий)

Шаблон Mostly fluid (рис. 2.2) складається головним чином з "гумової" сітки. На екранах великої або середньої ширини розміри блоків зазвичай залишаються незмінними, а регулюються тільки поля на більш широких екранах. На менших екранах виконується

перерахунок ширини блоків, зменшуються відступи, розміри зображень, шрифтів. На невеликих екранах (на телефонах) "гумова" сітка складається з однієї колонки, в якій блоки контенту розташовуються один під одним.

Однією з основних перевагцього шаблону є те, що необхідна лише одна ключова точка між невеликими екранами та екранами великого розміру.

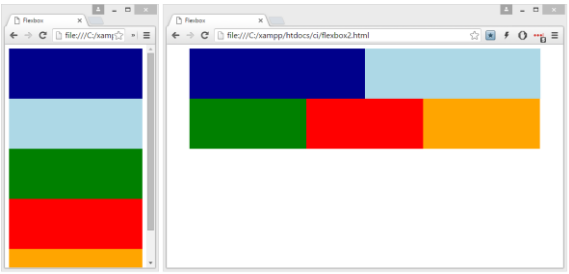

Рисунок 2.2 – Поняття найбільш гумового макету

```
<div class="container">
  <div class="box dark_blue"></div>
  <div class="box light_blue"></div>
  <div class="box green"></div>
  <div class="box red"></div>
  <div class="box orange"></div>
</div>
<style>
.container{
  display: flex;
  flex-wrap: wrap;
}
.box{
  width: 100%;
  height: 100px;
}
@media screen and (min-width: 550px){
  .dark_blue, .light_blue{
    width: 50%;
  }
  .green, .red, .orange{
    width: 33.3%;
  }
}
```

```
@media screen and (min-width: 700px){
  .container{
    width: 700px;
    margin: 0 auto;
  }
}
</style>
```

Column Drop (Стовпці один під одним)

В макетах, що складаються з декількох стовпців, які займають всю ширину екрану, шаблон Column Drop (рис. 2.3) переміщує стовпці униз один за одним по вертикалі, коли ширина вікна стає занадто малою для відображення відповідного стовпця. При цьому зазвичай загальний розмір елементів розмітки зберігається постійним.

На малих екранах всі стовпці будуть розташовані вертикально один під одним. Вибір контрольних точок для цього шаблону макета залежить від контенту і визначається для кожного варіанту дизайну окремо.

Рисунок 2.3 – Поняття макету стовпці один під одним

```
<div class="container">
  <div class="box dark_blue"></div>
  <div class="box light_blue"></div>
  <div class="box green"></div>
</div>
<style>
.container{
  display: flex;
  flex-wrap: wrap;
}
.box{
  width: 100%;
  height: 100px;
```

```
}
@media screen and (min-width: 450px){
  .dark_blue{
    width: 25%;
}
  .light_blue{
    width: 75%;
  }
}
@media screen and (min-width: 550px){
  .dark_blue, .green{
    width: 25%;
  }
  .light_blue{
    width: 50%;
  }
}
</style>
```

Layout shifter (Змінник макету)

Шаблон Layout shifter (рис. 2.4) є найбільш респонсивним, адже в ньому передбачається наявність декількох контрольних точок для екранів різної ширини і різних макетів для кожної з них.

Підтримка цього шаблону є більш складним завданням.

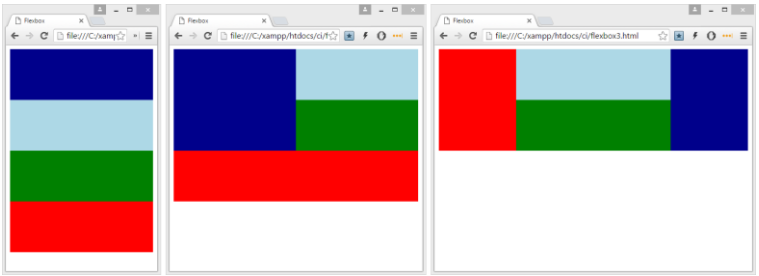

Рисунок 2.4 – Поняття змінника макету

```
<div class="container">
    <div class="box dark_blue"></div>
    <div class="container" id="container2">
        <div class="box light_blue"></div>
        <div class="box green"></div>
    </div>
    <div class="box red"></div>
```

```
</div>
<style>
.container{
  display: flex;
  flex-wrap: wrap;
  width: 100%;
}
.box{
  width: 100%;
  height: 100px;
}
@media screen and (min-width: 500px){
  .dark_blue{
    width: 50%;
    height: 200px;
  }
  #container2{
    width: 50%;
  }
}
@media screen and (min-width: 600px){
  .dark_blue{
    width: 25%;
    order: 1;
  }
  #container2{
    width: 50%;
  }
  .red{
    width: 25%;
    order: -1;
    height: 200px;
  }
}
@media screen and (min-width: 700px){
  .container{
    width: 700px;
    margin-right: auto;
    margin-left: auto;
  }
}
</style>
```

Tiny Tweaks (Маленькі зміни)

Шаблон Tiny Tweaks просто вносить невеличкі зміни в макет, наприклад, регулює розмір шрифта, змінює розмір зображень, або переміщує контент. Він застосовується до макетів, що складаються з одного стовпця, таких як односторінкові лінійні веб-сайти і статті з великою кількістю тексту.

```
.c1 {
  padding: 10px;
  width: 100%;
}
@media (min-width: 500px) {
  .c1 {
    padding: 20px;
    font-size: 1.5em;
  }
}
@media (min-width: 800px) {
  .c1 {
    padding: 40px;
    font-size: 2em;
  }
}
```

Off Canvas (Ззовні екрана)

Замість того щоб розміщувати елементи контенту вертикально один під одним, шаблон розміщує контент, який використовується рідко, скажімо, елементи навігації або меню програми за межами екрану, показуючи його тільки тоді, коли це дозволяє зробити розмір екрану (рис. 2.5). На невеликих же екранах контент можна відкрити одним клацанням.

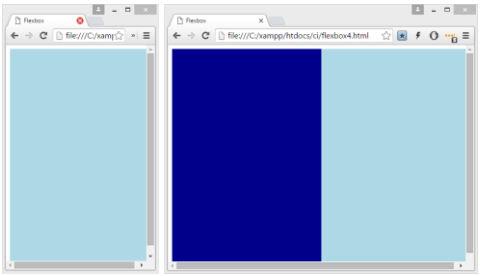

Рисунок 2.5 – Поняття зміннику макету

```
<nav id="drawer" class="dark_blue">
</nav>
<main class="light_blue">
</main>
html, body, main{
  height: 100%;
  width: 100%;
}
.dark_blue{
  background-color: darkblue;
}
.light_blue{
  background-color: lightblue;
}
nav{
  width: 300px;
  height: 100%;
  position: absolute;
  transform: translate(-310px,0);
  transition: transform 0.3s ease;
}
nav.open{
  transform: translate(0,0);
}
@media screen and (min-width: 600px){
  nav{
    position: relative;
    transform: translate(0,0);
  }
  body{
    display: flex;
    flex-flow: row nowrap;
  }
  main{
    width: auto;
    flex-grow: 1;
  }
}
```

2.2.4 Bootstrap

Bootstrap - вільно поширюваний фреймворк для розробки front-end респонсівних сайтів і веб-застосунків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток,

блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

Даний фреймоврк відрізняється високою швидкістю роботи, масштабованістю, легкістю налаштування, великою кількістю готових шаблонів оформлення і величезним співтовариством розробників.

Bootstrap використовує сучасні напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів.

Bootstrap дотримується концепції mobile first.

Основні інструменти Bootstrap:

– сітки для побудови каркасу сторінки;

– типографіка - опис шрифтів, визначення деяких класів для шрифтів, таких як код, цитати тощо;

– медіа – управління зображеннями та відео;

– таблиці - кошти оформлення таблиць, аж до додавання функціональності сортування.

– форми – засоби для оформлення форм і деяких подій, що відбуваються з ними;

– навігація - класи оформлення для табів, вкладок, пагінації, меню і панелі інструментів;

– алерт - оформлення діалогових вікон, підказок і спливаючих вікон.

Детально ознайомитися з усіма можливостями фреймворка можна на офіційному сайті.

Основною перевагою фреймворка є хороша реалізація 12-колоночної grid-сітки та підтримка адаптивності.

Приклад найпростішої сторінки з підключеним фрейморком:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta  name="viewport"  content="width=device-width,
initial-scale=1">
  <link                                rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7
/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1
.1/jquery.min.js"></script>
  <script        src="https://maxcdn.bootstrapcdn.com/
```

```
bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container-fluid">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>
</body>
</html>
```

Основна сітка макета Bootstrap складається з 12 колонок, які автоматично масштабуються при зміні розміру екрана. Кожен батьківський блок (рядок) розбивається на 12 рівних частин, а дочірні блоки (колонки) можуть займати від 1 до 12 частин батька і будуть розташовуватися горизонтально один за одним, поки не вийдуть за межі 12 частин (рис. 2.6).

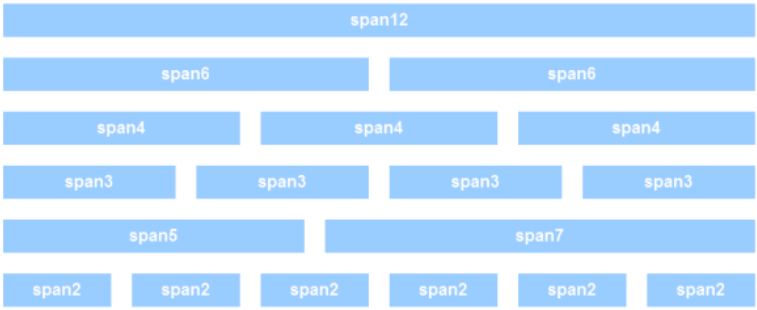

Рисунок 2.6 – Сітка Bootstrap

Для побудови макета існують такі правила:

– вміст сайту має бути обгорнутий в блок-контейнер, який може мати один з класів: «container» - контейнер фіксованої ширини або «container-fluid» - контейнер, що займає всю ширину вікна.

– для створення горизонтальних груп колонок використовуються рядки (class = "row");

– контент повинен розміщуватися в колонках (columns) і тільки колонки повинні бути безпосередніми нащадками рядків;

– якщо не задавати колонці жоден з класів, то вона розтягнеться по всій ширині батька;

– кожна колонка може мати один з визначених класів, який вказує клас сітки і необхідне число з дванадцяти доступних колонок

(class = "col-K-S", де K - клас сітки, S - розмір, наприклад .col-xs-4);

– класи сіток застосовуються до пристроїв, в яких ширина більше або дорівнює контрольним розмірами, і скасовують класи сіток, призначених для менших пристроїв;

– якщо в одному блоці з класом .row знаходиться більше 12 колонок, кожна група додаткових колонок буде переходити на нову рядок як єдине ціле;

– між колонками існують відступи;

– рядки можна вкладати в колонки.

Bootstrap містить чотири класи сіток:

– xs (для телефонів, екран <768px);

– sm (для планшетів, >=768px);

– md (для компютерів, >=992px);

– lg (для великих компютерів, >=1200px).

Даний клас показує мінімальний розмір екрану, на якому почне працювати правило. Для створення респонсівного дизайну ці класи комбінуються.

Так застосування класу .col-sm- до елементу вплине на стиль не тільки для маленьких пристроїв, наприклад планшетів, але також для середніх і великих пристроїв з шириною екрану, що більше або дорівнює 768px (тобто ≥768px), якщо не вживаються класи. col-md- і .col-lg-.

Розглянемо приклад.

```
<!-- Stack the columns on mobile by making one full-
width and the other half-width -->
<div class="row">
    <div class="col-xs-12 col-md-8">.col-xs-12 .col-
md-8</div>
    <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-
4</div>
</div>

<!-- Columns start at 50% wide on mobile and bump up
to 33.3% wide on desktop -->
<div class="row">
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-
4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-
4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-
```

```
4</div>
</div>

<!-- Columns are always 50% wide, on mobile and
desktop -->
<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>
<div class="row">
  <div class="col-xs-9">.col-xs-9</div>
  <div class="col-xs-4">.col-xs-4<br>Since 9 + 4 = 13
&gt; 12, this 4-column-wide div gets wrapped onto a
new line as one contiguous unit.</div>
  <div       class="col-xs-6">.col-xs-6<br>Subsequent
columns continue along the new line.</div>
</div>
```

Колонки можна зрушувати вправо за допомогою класів «.col-K-offset-S», де K - клас сітки, S - розмір зсуву.

Крім того в залежності від класу сітки можна робити елементи видимими (.visible-K-D, де D = {block, inline, inline-block}) і невидимими (.hidden-K).

2.3 Завдання на лабораторну роботу

2.3.1 Реалізувати наведені в роботі приклади прийомів респонсивного дизайну з використанням flexbox елементів.

2.3.2 Адаптувати розроблений в попередній роботі дизайн для перегляду на різних пристроях. Передбачити не менше двох контрольних точок (наприклад, 768px та 1024px).

2.4 Зміст звіту

2.4.1 Тема та мета роботи.

2.4.2 Результати роботи: вигляд макету на різних пристроях, фрагменти коду.

2.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

2.5 Контрольні запитання

2.5.1 Які є типи макетів сторінок? В чому їхні переваги та недоліки?

2.5.2 Як можна організувати різний вигляд веб-сторінок для різних типів пристроїв?

2.5.3 Що таке респонсивний дизайн?

2.5.4 Для чого використовується мета тег viewport?

2.5.5 Навіщо використовується атрибут media при підключенні зовнішньої таблиці стилів?

2.5.6 Що таке медіазапити? Наведіть синтаксис та приклади використання.

2.5.7 Що таке flexbox елемент? Поясніть його призначення та поведінку.

2.5.8 Що таке дизайн, побудований на решітках? Як він будується? Чому зазвичай решітки будуються на 12 колонках?

2.5.9 Що таке контрольна точка (breakpoint) в респонсивному дизайні?

Лабораторна робота № 3
Знайомство з мовою PHP та MVC фреймворком CodeIgniter

3.1 Мета роботи

Отримати практичні навички веб-програмування за допомогою мови PHP та ознайомитися з MVC фреймворком CodeIgniter.

3.2 Основні теоретичні відомості

3.2.1 Клієнт-серверна архітектура веб-застосунків

Сучасні веб-застосунки використовують клієнт-серверну архітектуру, яка за останні десятиліття стала одним із основних способів організації розподілених програмних систем.

При такій організації клієнтом виступає веб-браузер (Mozilla Firefox, Google Chrome, Opera, Internet Explorer, Apple Safari тощо), а у якості серверного застосунку виступає веб-сервер (Apache HTTP Server, Microsoft Internet Information Services, nginx тощо).

Задачею веб-браузера є візуалізація HTML-сторінок та мультимедійних об‘єктів, з яких вони складаються. В свою чергу веб-сервер виконує роль оброблювача запитів від віддалених веб-клієнтів. Обробкою може бути отримання статичної веб-сторінки безпосередньо із HTML-файлу, генерація динамічного вмісту із використанням спеціалізованих та універсальних мов програмування (PHP, Python, Java та інші), передача даних від користувача на веб-сервер, зокрема, файлів, а також оновлення визначеного фрагменту вже відображеної веб-сторінки (так звана Ajax-технологія).

В подібній ситуації протоколом обміну або способом формування запиту та генерації відповідей виступає HTTP (HyperText Transfer Protocol) – протокол обміну гіпертексту. Під гіпертекстом розуміють зв‘язані між собою за допомогою спеціальної адресації (URL) документи в мережі Internet. Основними рисами протоколу HTTP є наступне.

1. Обмін даними відбувається за схемою «запит»-«відповідь», отже кожну транзакцію обміну даними ініціює веб-клієнт, а веб-сервер генерує відповідь.

2. HTTP відноситься до верхнього (прикладного) рівня моделі відкритих систем OSI, тому його синтаксис найбільш наближений до природної мови.

3. HTTP не зберігає історію попередніх транзакцій, тому збереження стану реалізують інші засоби (зокрема, на основі cookie-змінних).

4. Кожний ресурс, доступ до якого виконується завдяки HTTP, ідентифікується через URL (unified resource locator).

5. Запит HTTP складається із рядка запиту («метод» «адреса ресурсу» «версія HTTP»), заголовків клієнта («назва»: «значення») і, опціонально, тіла запиту.

6. Відповідь HTTP складається із статусного рядка («версія HTTP», «код статусу», «опис коду статусу»), заголовків сервера («назва»: «значення») і, опціонально, тіла відповіді.

Обмін інформацією відбувається із використанням методів HTTP: GET, POST, PUT, HEAD, TRACE, OPTIONS, CONNECT, DELETE.

Протокол HTTP за своєю природою не орієнтований на збереження передісторії попередніх станів обміну повідомленнями між клієнтом і сервером. Тому в цей протокол уведено два спеціальні заголовки (Set-Cookie та Cookie), які надають можливості серверу встановлювати на клієнті невеликі обсяги даних (до 4096 байтів), а клієнту після цього з кожним запитом повертати збережені дані. Таким чином, сервер може однозначно і унікально ідентифікувати кожного клієнта. Час актуальності цих даних встановлює сервер за допомогою спеціального синтаксису заголовка Set-cookie.

Збереження стану характерне для таких веб-застосунків як клієнти електронної пошти, Internet-магазини, інформаційні портали тощо.

Важливим для розробника веб-застосунків аспектом протоколу HTTP є коди статусу обробки запиту клієнта (наприклад, 200 OK, 301 Moved permanently, 403 Forbidden, 404 Not found тощо).

Таким чином, використовуючи специфічні властивості протоколу HTTP (правила адресації, формат пакету, заголовки, методи та коди статусу), реалізуються різноманітні способи взаємодії веб-клієнта та веб-браузера із використанням спеціалізованих мов програмування та розмітки (PHP, Python, HTML, XML тощо).

3.2.2 Мова програмування PHP

Мова PHP (Personal Home Pages або Preprocessor Hypertext Preprocessor) набула статусу найбільш вживаної для написання

програмного коду з боку веб-сервера. Причиною тому є такі її переваги:

– простий синтаксис, в основному запозичений із традиційних мов програмування C та C++, об'єктно-орієнтовані можливості, а також слабка типізація;

– наявність механізму автоматичного збирання сміття, що спрощує виконання операції вивільнення пам'яті;

– наявність великої кількості бібліотечних модулів, інтегрованих у інтерпретатор (для роботи із файлами та мережею, обробки масивів та рядків; модулі шифрування, обробки графічних файлів, взаємодії з Internet-службами, веб-сервісами тощо);

– наявність засобів взаємодії з найбільш популярними СУБД (MySQL, Oracle, PostrgeSQL та іншими);

– наявність дистрибутивів для більшості сучасних операційних систем (Unix, Linux, Windows, MacOS) та спрощена інтеграція із популярним веб-сервером Apache.

Код мовою PHP створюється у довільному текстовому редакторі, а файл програми зберігається із розширенням php (наприклад, index.php). Крім того, у код PHP може бути вбудований і HTML-код сторінки. В такому випадку HTML-інструкції будуть ігноруватись інтерпретатором і передаватимуться клієнту без змін. Для виділення PHP-коду, його необхідно огорнути у спеціальні теги:

<?php програмний код ?>.

3.2.3 Поняття PHP фреймворку

Останнім часом основною тенденцією при розробці веб-застосунків є використання різноманітних програмних бібліотек, які прискорюють процес створення кінцевого продукту, а також надають можливості систематизувати програмний код за рахунок застосування шаблонів проектування, зокрема, «Модель-Подання-Контролер» (MVC).

Існує велика кількість подібних бібліотек, найбільш універсальні та потужні часто називають каркасами розробки або фреймворками (frameworks).

Фреймворк – структура програмної системи; програмне забезпечення, що полегшує розробку і поєднання різних компонентів великого програмного проекту. Фреймворк містить в собі велику

кількість різних за призначенням бібліотек. При побудові програм за допомогою фреймворків використовується підхід, при якому будь-яка конфігурація програми будується з двох частин: перша, постійна частина – каркас, що не змінюється від конфігурації до конфігурації і несе в собі гнізда, в яких розміщується друга, змінна частина – модулі (або точки розширення).

PHP фреймворки стали базовою платформою для розробки веб-застосунків. Вони забезпечують основну структуру програми. Використання PHP-фреймворків, дозволяє економити велику кількість часу, зменшити навантаження на процес розробки, позбутися проблеми повторюваного коду, і швидко створювати застосунки.

Серед фреймворків для мови PHP слід відзначити такі як Laravel, Zend Framework, CakePHP, Code Igniter, Symfony та Yii.

Узагальнена характеристика фреймворків наведена нижче.

1. Орієнтація на сучасні об‘єктно-орієнтовані підходи організації веб-застосунків, що надає змогу будувати масштабовані, контрольовані та надійні проекти.

2. Використання шаблону MVC, який надає можливості розділити програмний код взаємодії з даними (model), візуальне подання у вигляді шаблонів сторінок (view) та узагальнений алгоритм (бізнес-логіку) обробки запиту користувача (controller), який безпосередньо використовує моделі та подання проекту.

3. Наявність уніфікованого програмного інтерфейсу до реляційних баз даних, що надає можливість використовувати різні СКБД єдиним способом.

4. Наявність мови шаблонів, що надає можливості максимально розділити HTML-шаблон сторінки від даних, які наповнюють її інформаційним змістом.

5. Використання вбудованого механізму кешування, що забезпечує прискорення завантаження вже переглянутих сторінок.

6. Використання засобів валідації даних, переданих від користувача з HTML-форм.

7. Застосування механізму зручної для користувача адресації сторінок веб-застосунку з можливістю відсікання небажаних або хибних адресів.

8. Автоматичне шифрування Cookie-змінних, які надсилаються з сервера, а також управління сесіями.

9. Наявність великої кількості бібліотечних функцій, які

автоматизують найбільш вживані операції з даними веб-застосунку.

10. Фільтрація вхідних даних від користувача для уникнення зловмисних дій.

3.2.4 Шаблон MVC

На сьогодні більшість PHP проектів побудовані за допомогою архітектури MVC. MVC - це архітектурний шаблон проектування, який дозволяє відокремити бізнес-логіку від користувальницького інтерфейсу, а так само виділити область логіки яка виробляє обмін інформацією між базою даних і призначеним для користувача інтерфейсом. Таким чином можна змінити логіку програми, не зачіпаючи інтерфейсної частини, або навпаки.

В шаблоні MVC (рис. 3.1) модель (model) – це та частина архітектури, яка взаємодіє з базою даних, подання (view) - представляє ту частину, яку безпосередньо бачить користувач, тобто користувацький інтерфейс, і контролер (controller) - це область логіки, яка контролює і керує всіма її складовими і даними.

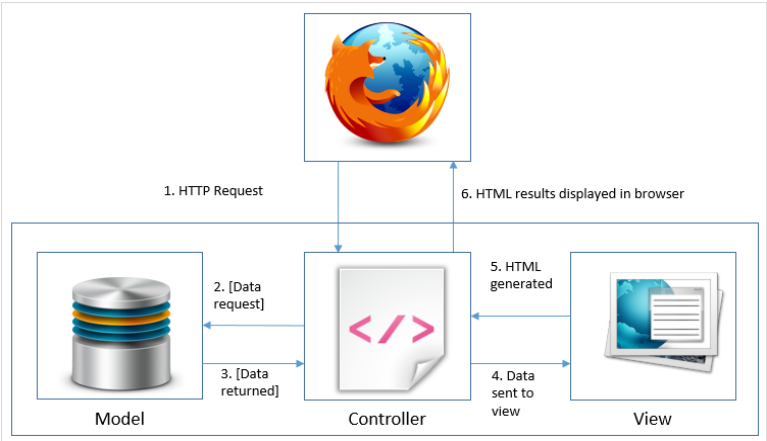

Рисунок 3.1 - Концепція PHP фреймворків

3.2.5 Особливості фреймворку CodeIgniter

CodeIgniter (CI) - популярний MVC PHP-фреймворк з відкритим вихідним кодом, призначений для розробки повноцінних веб-систем і застосунків.

Виконання програми засобами CI відбувається через єдиний

інтерфейс – програмний файл index.php, який виконує ініціалізацію та застосовує налаштування проекту з конфігураційних файлів, перетворює вхідні дані від користувача на внутрішні структури даних CI та передає управління засобам маршрутизації.

Модуль маршрутизації аналізує вхідні дані HTTP-запиту, а саме: метод HTTP, URL-адресу та заголовки. Якщо сторінка з такими ж вхідними даними вже існує у внутрішньому кеші CI, то її вміст повертається без подальшої обробки безпосередньо користувачеві (через веб-сервер). В іншому випадку наступним кроком є фільтрація вхідних даних з метою уникнення несанкціонованих дій з боку можливого зловмисника: екранування даних, вилучення небажаних тегів тощо. Після цього в залежності від URL запит надходить до визначеного контролера застосунку. Контролер, в свою чергу, обмінюється даними з моделями (виконує запити до бази даних, файлів тощо), за необхідності викликає бібліотечні та допоміжні функції і, наприкінці, передає дані у подання, де формується кінцева HTML-сторінка, яка передається користувачеві.

Графічно дану схему виконання застосунку з використанням фреймворку CI ілюструє рисунок 3.2, на якому напівжирним шрифтом виділено ті елементи структури проекту, які кодуються програмістом, при цьому решта – вбудовані модулі CI.

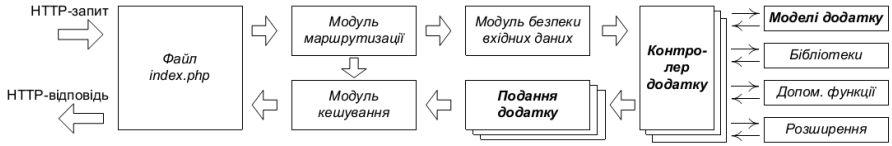

Рисунок 3.2 – Схема виконання програми CodeIgniter

Комплект файлів CI об‘єднується в групи ієрархічно поєднаних каталогів в залежності від функціонального призначення. Головним каталогом веб-застосунку є Application, а ядро системи розташоване у System. На рисунку 3.3 представлено файлову структуру проекту з використанням фреймворку CI.

З рисунку 3.3 видно, що різноманітні налаштування проекту групуються в каталозі config, зокрема, загальні параметри конфігурації застосунку у config.php, налаштування бази даних зберігаються у database.php, маршрутизація URL – у routes.php тощо.

Це дозволяє змінювати різноманітні налаштування безпосередньо в процесі функціонування веб-застосунку.

При створенні нового застосунку необхідно визначити базовий URL створюваного веб-сайту в загальних параметрах та параметри доступу до бази даних

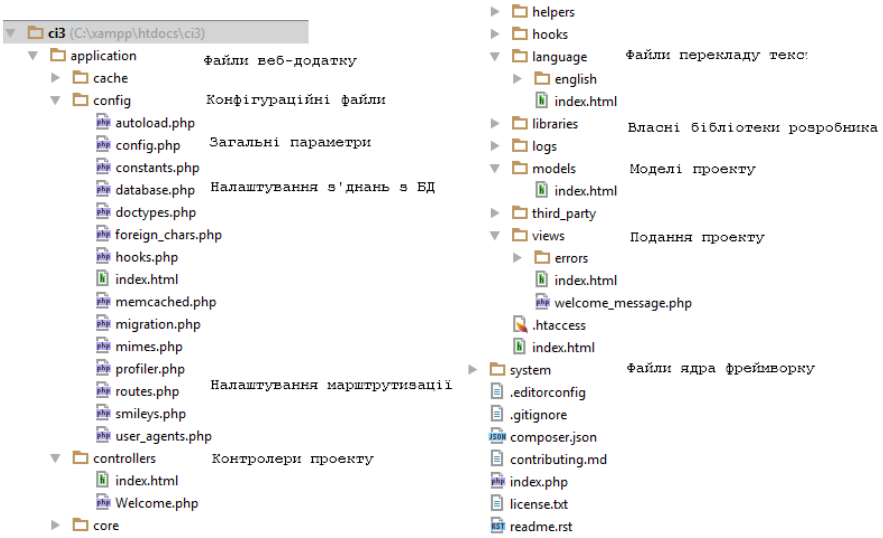

Рисунок 3.3 – Файлова структура проекту CodeIgniter

Найголовнішими блоками проекту, які реалізовує безпосередньо програміст є:

– контролери (controllers), які є точками входу при виконанні обробки відповідного запиту та виконують функцію управління ходом обробки запиту від користувача.;

– моделі (models), які уніфікують доступ до зовнішніх даних (баз даних, файлів, мережних служб тощо);

– подання (views), які зберігають шаблони веб-сторінок, відокремлюючи дизайн від інформаційного вмісту HTML-документу.

За замовчуванням каталоги controllers та views містять файли Welcome.php та welcome_message.php відповідно. Ці файли можна використовувати як зразок для створення власного проекту.

Розглянемо детальніше хід виконання програми на прикладі. http://[your-site-url]/index.php.

Алгоритм обробки полягає у наступному. веб-сервер, отримавши запит, передає його на виконання файлу index.php, який знаходиться безпосередньо в htdocs. Оскільки в URL не міститься жодних параметрів після index.php, то застосовуються налаштування за умовчанням, вказані в конфігураційному файлі routes.php:

```
$route['default_controller'] = 'welcome';
```

Завдяки цьому, управління передається у контролер Welcome.

3.2.6 Створення контролеру

Контролер Welcome знаходиться в файлі application/controllers/Welcome.php (за домовленістю назва файлу і класу має співпадати; назва записується з великої літери). Контролер – це клас що містить основну логіку та делегує роботу.

Код файлу welcome.php має вигляд:

```
<?php
class Welcome extends CI_Controller {
  function __construct(){
    parent::__construct();
    // власний код конструктора
  }
  function index(){
    $this->load->view('welcome_message');
  }
}
```

Як видно, файл містить клас Welcome, успадкований від CI_Controller – базового класу для всіх контролерів проекту (*system/core/Controller.php*). Суттєвим методом в ньому є index(). Саме цей метод викликається за замовчуванням у випадку, коли URL не містить інших сегментів. В цьому методі записано код:

$this->load->view('welcome_message');

який означає, що необхідно передати управління компоненту подання (view) з назвою файла welcome_message.php з метою генерації сторінки HTML.

Як правило, файли подання містять код HTML з найпростішими вставками мовою PHP: операторами та функціями виведення (echo, print), а також циклів (for, foreach тощо). Це обумовлено власне метою подання – об‘єднання статичного коду HTML та динамічних даних, отриманих в результаті взаємодії програми з моделями (базою даних,

файлами тощо).

Вибір контролера, якому передається управління при обробці запиту користувача, визначається URL-адресою запиту. Типова структура URL-адреси має наступний фіксований формат :

http://[your-site-url]/[controller-class]/[controller-method]/[arguments].

Першим параметром є назва контролера проекту, якому передається управління, далі метод у класі контролера, і остання частина – параметри, які отримає метод контролера.

Наприклад:

http://example.com/portfolio/latest/asc/10 ,

де portfolio – клас контролера,
latest – метод контролера,
asc та 10 – параметри методу.

Слід зауважити, що у прикладі після назви сайту не вказано index.php. Як правило, на практиці позбуваються використання index.php засобами веб-сервера. Наприклад, для веб-сервера Apache необхідно в кореневому каталозі проекту створити файл .htaccess і вписати в цей файл наступні інструкції:

```
RewriteEngine on
RewriteCond $1 !^(index\.php|assets|robots\.txt)
RewriteRule ^(.*)$ /index.php/$1 [L]
```

При цьому завдяки використанню модуля modrewrite встановлюються правила обробки запиту перед передачею його веб-застосунку (тут CodeIgniter). А саме: RewriteCond вказує, які файли пропускати без обробки (index.php, assets, robots.txt), а RewriteRule задає правило, що будь-який запит, крім вказаного в RewriteCond необхідно модифікувати шляхом додавання перед ним рядку index.php. Таким чином, CodeIgniter розпізнає вірний формат URL.

3.2.7 Створення моделі

Моделі призначені для об'єднання функцій взаємодії із зовнішніми даними (файлами, базами даних) у єдиний клас. Для використання моделі необхідно створити файл PHP визначеної структури в каталозі /application/models. При цьому назви файлу та класу мають співпадати, а назва записується з великої літери.

Структура класу включає назву класу, ознаку успадкування від універсальної моделі (CI_Model), а також конструктор __constructor(), в якому викликається код батьківського класу. Завантаження моделі з контролера відбувається за допомогою команди:

$this->load->model('model_name');

де model_name - назва класу моделі.

Після завантаження моделі виклик її функції відбувається за допомогою команди

$this->model_name->method(arguments).

Наведемо, приклад моделі:

```
<?php
class Testmodel extends CI_Model {
  function __construct() {
    parent::__construct();
    //завантаження бібліотеки бд
    //зробить бд доступною через об'єкт $this->db
    $this->load->database();
  }
  function get_data($id) {
    //строка запиту
    $sql = "SELECT field
            FROM table
            WHERE id = ".$this->db->escape($id);
    $q = $this->db->query($sql);
    $row = $q->row();
    return $row->field;
  }
}
```

В подальшому виклик методів класу моделі відбувається наступним чином:

```
$this->testmodel->get_data(10);
```

Слід зауважити, що CI виконує під‘єднання автоматично під час завантаження об‘єкта database, тому код класу моделі зазвичай містить лише команди доступу до таблиць бази даних. Налаштування підключення до БД знаходиться в файлі application/config/database.php.

Результати виконання запитів до БД можуть бути отримані у вигляді обєктів та масивів. Наведемо приклади.

Запит з декількома результатами (Object Version)

```
$query = $this->db->query('SELECT name, email FROM
my_table');
foreach ($query-> result_object() as $row) {
   echo $row->name;
   echo $row->email;
}
echo 'Total Results: ' . $query->num_rows();
```

Запит з декількома результатами (Array Version)

```
$query = $this->db->query('SELECT name, email FROM
my_table');
foreach ($query->result_array() as $row) {
   echo $row['name'];
   echo $row['email'];
}
```

Запит з єдиним результатом (Object Version)

```
$query = $this->db->query('SELECT name FROM my_table
LIMIT 1');
$row = $query->row();
echo $row->name;
```

Запит з єдиним результатом (Array version)

```
$query = $this->db->query('SELECT name FROM my_table
LIMIT 1');
$row = $query->row_array();
echo $row['name'];
```

Запит типу Insert

```
$sql = "INSERT INTO mytable (title, name) VALUES
(".$this->db->escape($title).",          ".$this->db-
>escape($name).")";
$this->db->query($sql);
echo $this->db->affected_rows();
echo $this->db->insert_id()
```

3.2.8 Створення подання

Подання у Codelgniter - це повна HTML-сторінка або її фрагмент (наприклад: заголовок, меню, основний вміст, нижня частина). Основне призначення подання - відділити логіку отримання даних від візуального вигляду. Як правило, подання містить мінімум PHP-коду: лише той, що призначено для виведення даних (echo, print

тощо). Крім того, допустимим є використання циклів PHP (for, foreach тощо) для виведення даних, що мають вигляд списків, дерев та інших складних структур.

Наприкла, якщо потрібно відобразити список в поданні, необхідно передати в подання параметр

```
$data['todo_list'] = array('Clean House', 'Call Mom',
'Run Errands');
```

а в поданні записати код

```
<ul>
  <?php foreach ($todo_list as $item):?>
     <li><?php echo $item;?></li>
  <?php endforeach;?>
</ul>
```

3.2.9 Бібліотеки та помічники

В CI для швидкої розробки застосунків передбачені **бібліотеки** та **помічники**. Програміст може розширювати існуючі та створювати власні бібліотеки та помічники.

Помічники та бібліотеки дозволяють вирішувати деякі розповсюджені задачі. Різниця між ними в тому, що кожний файл помічника – це просто об'єднана в одну категорію колекція функцій, кожна з яких виконує одну специфічну задачу незалежно від інших функцій. Помічники пишуться не в ООП-форматі, а є простими процедурними функціями.

Є помічник URL, що допомагає створювати посилання, помічник Form, який допомагає створювати елементи форми, помічник Text з операціями форматування тексту, помічник Cookie для встановлення та зчитування куків, помічник File для роботи з файлами тощо. Файли помічников завантажуються за допомогою команди $this->load->helper('name');, де name - це імя помічника без розширення .php та постфікса _helper. Після завантаження поміника його функції стають доступні як звичайні функції PHP.

Серед бібліотек слід виділити бібліотеки для роботи з вхідними та вихідними даними застосунку, електронною поштою, фтп, створення сесій, засоби двонаправленного шифрування даних, валідації форм, пагінації, бібліотеки для завантаження файлів та для роботи з зображеннями, язикову бібіліотеку для локалізації застосунку тощо.

Для їхнього завантаження використовується команда

```
$this->load->library('classname');
```

де classname - це ім'я класу, який треба завантажити. Для використання бібліотеки використовується команда

```
$this->someclass->some_function();
```

Детальна документація по фреймворку CodeIgniter може бути знайдена у дистрибутиві фреймворку.

3.2.10 Створення модулю для відображення статичних сторінок

Розглянемо приклад створення модулю Pages, що відображатиме статичні сторінки.

Для цього створимо файл application/controllers/Pages.php з наступним кодом.

```
<?php
class Pages extends CI_Controller {
  public function view($page = 'home'){
  //код
  }
}
```

Ми створили клас з назвою Pages з методом view, що приймає один аргумент $page.

Створимо два подання (шаблони сторінок), які міститимуть header та footer майбутніх сторінок.

Створимо файл *application/views/templates/header.php* з наступним вмістом:

```
<html>
  <head>
    <title>Static pages</title>
  </head>
  <body>
    <h1><?php echo $title; ?> - labs</h1>
```

Хедер містить звичайний HTML код, який ви бажаєте відобразити до завантаження головного подання. В цьому коді також виводиться вміст змінної $title, яку ми визначимо у контролері.

Створимо також файл *application/views/templates/footer.php* з кодом:

```
    <p><strong>ZNTU &copy; 2017</strong></p>
  </body>
</html>
```

Шаблони статичних сторінок розмістимо в каталозі

application/views/pages/. Створимо тут два файли *home.php* та *about.php*. Всередині цих файлів розмістіть будь-який зміст (наприклад, “Hello World!”).

Додамо логіку до контролеру. Раніше ми створили метод view(), що приймає ім'я сторінки, яку необхідно завантажити:

```
public function view($page = 'home'){
if(!file_exists(APPPATH.'/views/pages/'.$page.'.php')
){
    show_404(); // такої сторінки не існує!
  }
  $data['title'] = ucfirst($page);
  $this->load->view('templates/header', $data);
  $this->load->view('pages/'.$page, $data);
  $this->load->view('templates/footer', $data);
}
```

Тепер, якщо запитана сторінка існує, вона завантажується, включаючи хедер та футер, та відображається користувачеві.

Якщо сторінка не існує , відображається помилка «404 Page not found». show_404() – це вбудована функція CodeIgniter, що відображає стандартну сторінку помилки.

В наступному коді завантажуються подання в тому порядку, в якому вони повинні бути відображені. Другий параметр метода view() використовується для передачі значень до подання. Кожне значення масиву $data асоціюється змінній з ім'ям її ключа. Таким чином значення $data['title'] в контролері еквівалентно змінній $title у поданні.

Таким чином, переходячи в браузері за посиланням http://[your-site-url]/index.php/pages/view ми за замовченням побачимо сторінку home. За посиланням http://[your-site-url]/index.php/pages/view/about ми побачимо сторінку about.

Для того, щоб встановити створений контролер в якості контролеру за замовченням зробимо наступне.

Відкриємо файл налаштування маршрутизації at*application/config/routes.php* та додаму наступні строки:

```
$route['default_controller'] = 'welcome'';
$route['welcome'] = 'welcome';
$route['(:any)'] = 'pages/view/$1';
```

CodeIgniter зчитує правила маршрутизації зверху до низу та переправляє запит за першим знайденим правилом. Кожне правило це регулярний вираз (ліва частина) асоційований з контролером та його

методом (права частина)

3.3 Завдання на лабораторну роботу

3.3.1 За допомогою phpMyAdmin створити базу даних "last_name", де last_name – прізвище студента латиницею. Створити в базі таблицю "settings" з двома полями: "name" та "value". Додати два записа: один з назвою "welcome_title" та другий "welcome_content".

3.3.2 Розгорнути та налаштувати PHP-фреймворк CodeIgniter.

3.3.3 Ознайомитися зі структурою фреймворку. Підключити його до розробленої бази даних.

3.3.4 В модулі welcome модифікувати подання шляхом підключення до нього фреймворку Bootstrap.

3.3.5 Створити модель для модуля welcome, що витягає з бази заголовок сторінки та її вміст. Передати отримані значення в подання модуля.

3.3.6 Створити модуль перегляду статичних сторінок, для оформлення яких використати Bootstrap.

3.4 Зміст звіту

3.4.1 Тема та мета роботи.

3.4.2 Результати роботи.

3.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

3.5 Контрольні запитання

3.5.1 Що таке клієнт-серверна архітектура?

3.5.2 Опишіть особливості протоколу HTTP.

3.5.3 Дайте характеристику мови PHP.

3.5.4 Які обмежувачі вказують блок з PHP кодом? Як «вбудувати» PHP код до HTML?

3.5.5 Як включити зміст вказаного файлу в поточний?

3.5.6 За допомогою яких команд можна вивести інформацію в консоль (на екран)? Яка конструкція використовується для скороченого запису виводу строки?

3.5.7 Як створити та видалити змінну? Як задається тип змінної? Як перевірити, чи існує змінна?

3.5.8 Які типи даних існують в PHP? Як задати константу?

3.5.9 Вкажіть синтаксис умовного оператору if, циклу for, while, foreach, switch.

3.5.10 Як працювати з масивами (створити масив, додати елемент, видалити елемент, функції роботи з масивами)?

3.5.11 Як працювати з рядками (створити рядок, перетворити масив на рядок та навпаки, функції роботи з рядками)?

3.5.12 Як створити функцію? Як передати в функцію аргументи за значенням та за посиланням? Як задати значення параметрів за замовченням?

3.5.13 Що таке POST та GET запит?

3.5.14 Як в PHP отримати передані параметри запиту?

3.5.15 Які є суперглобальні масиви в PHP?

3.5.16 Як оголосити клас? Як створити об’єкт класу? Як отримати доступ до поля та до метода класу?

3.5.17 Що таке інтерфейс та абстрактний клас?

3.5.18 Що таке статичний метод, поле або константа?

3.5.19 Що таке фреймворк? Опишіть архітектурний шаблон MVC та його призначення. Що таке контролер, модель, подання?

3.5.20 Як виконати запит до бази даних?

Лабораторна робота № 4
Створення веб-застосунку за допомогою фреймворку CodeIgniter

4.1 Мета роботи

Отримати навички створення веб-застосунків за допомогою MVC фреймворку CodeIgniter з використанням мови програмування PHP та мови доступу до баз даних MySQL.

4.2 Основні теоретичні відомості

4.2.1 Взаємодія з базою даних

Сьогодні при розробці серйозних веб-застосунків практично завжди використовуються бази даних. У них зберігається інформація, необхідна для роботи сайту – від самого вмісту (контенту) і до логінів і паролів користувачів і різних налаштувань сайту.

Незважаючи на те, що в базі даних можна зберігати не тільки текстову інформацію, а й, скажімо, картинки, якісь документи тощо, як правило вона все ж використовується саме для зберігання текстової інформації, а інформація іншого типу зберігається у вигляді файлів.

Для створення повноцінного веб-застосунку необхідно приділити увагу розробці запитів для спілкування застосунку з базою даних.

SQL (Structured Query Language) – це декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних. SQL є стандартною мовою для доступу до БД.

MySQL є популярною системою управління базами даних з відкритим кодом, яка зазвичай використовується в веб-застосунках завдяки своїй швидкості, гнучкості та надійності.

Припустимо, є БД IMBD (Internet Movies DataBase), з структурою, наведеною на рис. 4.1.

При створенні веб-застосунку можливі наступні запити до цієї БД:

1. скільки всього фільмів в базі даних (поле результату назвати Movies_number);
2. всі жанри фільмів;

3. вибрати всіх акторів, у яких прізвище починається з «Del»;
4. вибрати всіх режисерів, у яких в імені присутній «ron»;
5. вибрати всі фільми 2000 року;

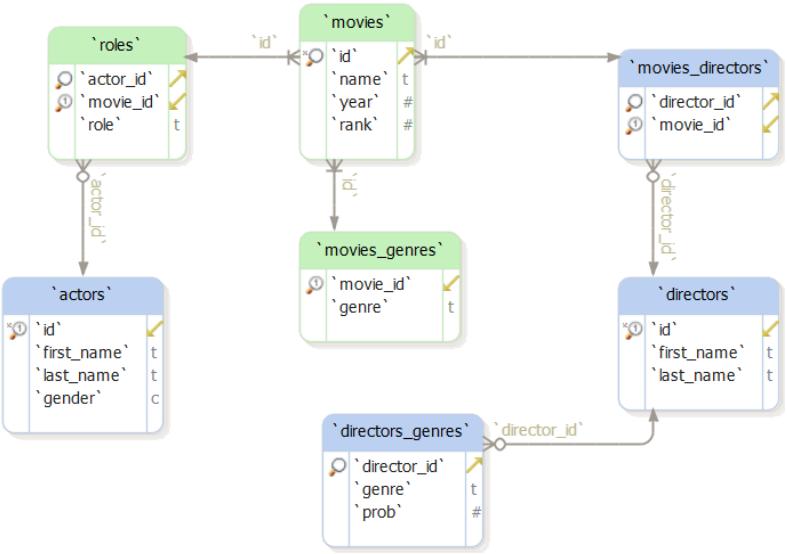

Рисунок 4.1 – Схема БД IMBD

6. скільки фільмів було знято у 2000 році;
7. рік випуску фільму «Braveheart»;
8. скільки акторів зіграло у фільмі «Lost in translation»;
9. імена і прізвища всіх акторів, які зіграли у фільмі «Lost in translation» і їх ролі;
10. знайти режисера фільму «Fight Club»;
11. скільки в базі фільмів, режисером яких був «Clint Eastwood»;
12. назви фільмів, режисером яких був «Clint Eastwood»;
13. імена режисерів, які є режисерами хоча б одного фільму жахів;
14. імя і прізвище кожного актора, який з'явився у фільмі режисера «Cristopher Nolan»;
15. найперший (або перші два) фільму Джеймса Камерона;
16. найпродуктивніший режисер;
17. фільми з рейтингом> = 9;

18. всі драми з рейтингом> = 8 з іменами і прізвищами їх режисерів в одному полі під ім'ям «Director»;

19. в яких фільмах знімався «Kevin Bacon»;

20. в яких фільмах знімався «Kevin Bacon» разом з «Donald (I) Sutherland» (або з «Kevin Costner»);

21. скільки років знімався в фільмах Kevin Bacon (різниця між останнім і першим фільмом);

22. ім'я та прізвище всіх акторів, які з'явилися в обох частинах фільму «Kill Bill: Vol. 1 »і« Kill Bill: Vol. 2 »;

23. 5 акторів, які знялися в найбільшій кількості фільмів, упорядкованих від найбільшої кількості до меншого;

24. 3 найбільш популярні жанри фільмів, відсортовані в порядку убування популярності;

25. прізвища всіх режисерів, які зняли більше комедій, ніж інших типів фільмів;

26. список років і кількості фільмів, знятих в цих роках, впорядковані за кількістю фільмів від самого продуктивного року;

27. яка роль є найчастішою;

28. яка роль є найчастішою за винятком «Himself» і порожнього рядка;

29. хто (ім'я та прізвище) грав роль «Trinity» у фільмі «The Matrix» (в базі Matrix, The);

30. в якому році було знято максимальну кількість фільмів (можливі кілька, упорядкувати по році);

31. актори і режисери (ім'я, прізвище, роль) для фільму «Lost in Translation», при цьому роль режисера повинна бути вказана як «Director»; упорядкувати на прізвище.

4.2.2 Створення складного модулю

Розглянемо приклад створення модуля новин в фреймворку CodeIgniter.

Створіть модель для модуля. В директорії application/models створіть новий файл News_model.php з наступним кодом:

```
<?php
class News_model extends CI_Model {
  public function __construct() {
    $this->load->database();
  }
```

```
}
```

У конструкторі моделі завантажується бібліотека бази даних. Це робить доступним клас бази даних через об'єкт $ this-> db.

Перед тим, як робити запити до бази даних, необхідно створити таблицю для новин та додати до неї кілька записів.

```
CREATE TABLE news (
  id int(11) NOT NULL AUTO_INCREMENT,
  title varchar(128) NOT NULL,
  slug varchar(128) NOT NULL,
  text text NOT NULL,
  PRIMARY KEY (id),
  KEY slug (slug)
);
```

Создайте метод для получения новостей из базы данных:

```
public function get_news($slug = FALSE) {
  if ($slug === FALSE) {
    $sql = "SELECT * FROM news";
    $query = $this->db->query($sql);
    return $query->result_array();
  }
  else{
    $sql = "SELECT * FROM news WHERE 'slug' =
".$this->db->escape$slug);
    $query = $this->db->query($sql);
    return $query->row_array();
  }
}
```

З цим кодом можна виконати два різних запити: отримати всі записи або отримати один елемент за його людино-зрозумілим посиланням.

Тепер модель повинна бути прив'язана до відображенн, які використовуються для показу новин користувачеві. Це можна зробити в контролері. Створіть контролер application / controllers / News.php.

```
<?php
class News extends CI_Controller {
  public function __construct() {
    parent::__construct();
    $this->load->model('news_model');
    $this->load->helper('url_helper'); //site url
  }
  public function index() {
    $data['news']= $this->news_model->get_news();
```

```
        $data['title'] = 'News archive';
        $this->load->view('templates/header', $data);
        $this->load->view('news/index', $data);
        $this->load->view('templates/footer');
      }
      public function view($slug = NULL) {
        $data['news_item']      =      $this->news_model-
>get_news($slug);
        if (empty($data['news_item'])) {
          show_404();
        }
        $data['title'] = $data['news_item']['title'];
        $this->load->view('templates/header', $data);
        $this->load->view('news/view', $data);
        $this->load->view('templates/footer');
      }
}
```

Конструктор контролера завантажує модель, тому вона може бути використана у всіх методах цього контролера.

Далі, є два методи, для перегляду всіх новин і одного конкретного елемента. Ви можете побачити змінну $slug, яка передається методу моделі в другому методі. Модель використовує цю змінну для ідентифікації новини, яку слід повернути.

Тепер дані, отримані від моделі, повинні бути передані до відображення.

Створимо файл application / views / news / index.php і помістимо в нього код.

```
<a    href="<?php    echo    site_url('news/create');
?>">Create news</a>
<?php foreach ($news as $news_item): ?>
  <h3><?php echo $news_item['title']; ?></h3>
  <div class="main">
    <?php echo $news_item['text']; ?>
  </div>
  <p><a                href="<?php                echo
site_url('news/'.$news_item['slug']);         ?>">View
article</a></p>
<?php endforeach; ?>
```

Для формування правильного посилання на новини за допомогою функції site_url в файлі конфігурації application / config / config.php необхідно встановити наступне налаштування $config['base_url'] = 'http://localhost/ci/';

Файл відображення application/views/news/view.php міститиме код:

```
<?php
  echo '<h2>'.$news_item['title'].'</h2>';
  echo $news_item['text'];
```

Тепер необхідно додати правила маршрутизації для доступу до контролера:

```
$route['news/(:any)'] = 'news/view/$1';
$route['news'] = 'news';
$route['(:any)'] = 'pages/view/$1';
$route['default_controller'] = 'pages/view';
```

Направте браузер за адресою index.php / news і перевірте відображення сторінок новин.

Створений модуль вміє зчитувати дані з бази даних, але не вміє додавати нову інформацію. Розширимо наш контролер новин і модель, створену раніше, щоб додати цю функціональність.

Щоб записувати дані в базу, необхідно створити форму, через яку будемо отримувати інформацію для збереження. Нам потрібна форма з двома полями: одне для заголовка і інше для тексту. "Гарний" URL з заголовка новини буде формуватися безпосередньо в моделі.

Створимо нове відображення в application/views/news/create.php.

```
<?php if (isset($error)) echo $error; ?>
<form      method="post"      action="<?php      echo
site_url('news/create'); ?>">
  <label for="title">Title</label>
  <input type="input" name="title" /><br />
  <label for="text">Text</label>
  <textarea name="text"></textarea><br />
  <input  type="submit"  name="submit"  value="Create
news item" />
</form>
```

Повернемося до контролера новин. Тут необхідно зробити дві речі: переконатися в тому, що форма передана, а також в тому, що передані дані пройшли правила валідації:

```
public function create() {
  $data['title'] = 'Create a news item';
  if ($_SERVER["REQUEST_METHOD"] == "POST") {
    //process request
    $error = '';
    //validate data
    if (!$this->input->post('title')){
```

```
        $error .= '<p>Title is required</p>';
      }
      if (!$this->input->post('text')){
        $error .= '<p>Text is required</p>';
      }
      if (!$error){
        $this->news_model->set_news();
        $this->load->view('news/success');
      }else{
        $data['error'] = $error;
        $this->load->view('templates/header', $data);
        $this->load->view('news/create');
        $this->load->view('templates/footer');
      }
    }else{
       //form view
       $this->load->view('templates/header', $data);
       $this->load->view('news/create');
       $this->load->view('templates/footer');
    }
}
```

Залишається написати метод, який записує дані в базу даних. Відкрийте модель, створену раніше, і додайте до неї наступне:

```
public function set_news() {
  $title       =       htmlspecialchars($this->input-
>post('title'));
  $text        =       htmlspecialchars($this->input-
>post('text'));
  $this->load->helper('url');
  $slug = url_title($title, 'dash', TRUE);
  $sql = "INSERT INTO news (text, title, slug) VALUES
(".$this->db->escape($text).",           ".$this->db-
>escape($title).",".$this->db->escape($slug).")";
  return $this->db->query($sql);
}
```

Перед тим як додавати новини, необхідно додати додаткове правило маршрутизації:

```
$route['news/create'] = 'news/create';
$route['news/(:any)'] = 'news/view/$1';
$route['news'] = 'news';
$route['(:any)'] = 'pages/view/$1';
$route['default_controller'] = 'pages/view';
```

Перейдіть за адресою index.php / news / create і додайте новину.

Перевірте, чи відображається додана новина в списку.

Перевіримо, що буде якщо додати новину з вже існуючою назвою. Как необхідно змінити код, аби виправити невірну поведінку?

4.3 Завдання на лабораторну роботу

4.3.1 Розробити запити мовою MySql для запиту даних з бази даних IMBD.

4.3.2 Створити модуль новин, як це описано в роботі. Оформити макет за допомогою фреймворку Bootstrap.

4.3.3 Створити модуль наступного призначення для роботи з предметною областю IMBD: модуль перегляду та редагування даних про фільми.

а) Перегляд списку фільмів. Перелік існуючих об’єктів подати у вигляді таблиці з розбиттям інформації на сторінки (пагінація). Виводити по 10 записів на сторінку, передбачити переходи на першу сторінку, попередню, наступну та останню.

Зберегти останню переглянуту сторінку за допомогою сесії.

Передбачити можливість пошуку фільму за назвою, роком випуску, рейтингом та жанром.

б) Перегляд інформації про конкретний фільм.

в) Додання нового або редагування існуючого фільму.

Для фільму задається назва, рік випуску, рейтинг, один або декілька жанрів (чекбокси).

Додати можливість встановлення зображення для фільму тобто постеру, короткого опису, країни виробництва (селект), вікові обмеження перегляду (радіо перемикач).

Дані, що вводяться, перевірити на валідність. Рік випуску, назву фільму та рейтинг перевірити за допомогою регулярних виразів.

Передбачити випадок додавання вже існуючого фільму.

4.4 Зміст звіту

4.4.1 Тема та мета роботи.

4.4.2 Результати роботи.

4.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

4.5 Контрольні запитання

4.5.1 Що таке форми в HTML? Для чого вони потрібні? Що визначають атрибути action і method?

4.5.2 Які керуючі елементи форм існують? Наведіть їх і опишіть призначення.

4.5.3 Як відправити дані форми на сервер?

4.5.4 Як в PHP дізнатися тип запиту від клієнта?

4.5.5 Які суперглобальні масиви існують в PHP?

4.5.6 Як відправити файл за допомогою форми? Як обробити його в PHP?

4.5.7 Що таке валідація даних форми? Які типи валідації бувають?

4.5.8 Що таке регулярні вирази? Як їх використовувати?

4.5.9 Що таке кукі та сесія? Для чого смороду призначені.

4.5.10 Що таке MySQL? Які типи запитів бувають?

4.5.11 Опишіть синтаксис запиту SELECT.

4.5.12 Опишіть синтаксис запиту INSERT і DELETE

4.5.13 Опишіть синтаксис запиту UPDATE і REPLACE

4.5.14 Як вибрати дані відразу з декількох таблиць? Які типи об'єднань бувають?

4.5.15 Що таке підзапити?

4.5.16 Як з PHP виконати запит до БД?

Лабораторна робота № 5
Створення складних інтерфейсів веб-застосунків за допомогою JavaScript та AJAX запитів

5.1 Мета роботи

Отримати навички створення веб-застосунків за сценарної мови JavaScript, біліотеки jQuery та технології Ajax.

5.2 Основні теоретичні відомості

5.2.1 JavaScript

JavaScript (JS) – прототипно-орієнтована сценарна мова програмування.

JS дозволяє отримувати елементи HTML-сторінки, маніпулювати ними, динамічно створювати графічні інтерфейси і взаємодіяти з користувачем. JS взаємодіє з HTML-сторінкою за допомогою DOM (Document Object Model).

Підключати JS скрипт на сторінку можна в елементі head або в кінці елемента body використовуючи зовнішній файл

```
<script src="myScript.js"></script>
```

або безпосередньо сам скрипт

```
<script>
function myFunction() {
  alert("Hello World");
}
</script>
```

JS використовується для додання інтерактивність веб-сторінкам:

– динамічно змінює вміст HTML-сторінки, атрибути елементів і оформлення;

– реагує на події (приклад: приховування / відкриття частини інформації при натисканні кнопки миші);

– виконує валідацію форм;

– виконує обчислення на комп'ютері користувача.

Наведемо фрагмент коду, який змінює вміст елемента:

```
<p id="demo">JavaScript can change HTML content.</p>
<button type="button"
onclick="document.getElementById('demo').innerHTML  =
'Hello JavaScript!'">
Click Me!</button>
```

Наведемо фрагмент коду, який змінює атрибут елемента:

```
<img id="myImage" onclick="changeImage()"
src="pic_bulboff.gif" width="100" height="180">
<script>
function changeImage() {
  var image = document.getElementById('myImage');
  if (image.src.match("bulbon")) {
    image.src = "pic_bulb_off.gif";
  } else {
    image.src = "pic_bulb_on.gif";
  }
}
</script>
```

Наведемо фрагмент коду, який змінює CSS стилі елемента:

```
<p id="demo">JavaScript can change the style of an
HTML element.</p>
<button type="button" onclick="myFunction()">Click
Me!</button>
<script>
function myFunction() {
  var x = document.getElementById("demo");
  x.style.fontSize = "25px";
  x.style.color = "red";
}
</script>
```

Наведемо фрагмент коду, який валідує введені дані:

```
<input id="numb" title=”Please input a number between
1 and 10”>
<button type="button"
onclick="myFunction()">Submit</button>
<p id="demo"></p>
<script>
function myFunction() {
  var x, text;
  // Get the value of the input field with id="numb"
  x = document.getElementById("numb").value;
  // If x is Not a Number or less than one or greater
than 10
  if (isNaN(x) || x < 1 || x > 10) {
    text = "Input not valid";
  } else {
    text = "Input OK";
  }
  document.getElementById("demo").innerHTML = text;
```

```
}
</script>
```

В JS немає вбудованих функцій виведення.

JS може «відображати» дані різними способами:

– запис у спливаюче вікно за допомогою window.alert ();
– запис в HTML вивід за допомогою document.write ();
– запис в елемент HTML за допомогою innerHTML (наприклад, document.getElementById ("demo"). InnerHTML)
– запис в консоль браузера за допомогою console.log () ;

Наведемо фрагмент коду валідації заповнення форми із записом помилки у спливаюче вікно:

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
  var x = document.forms["myForm"]["fname"].value;
  if (x == null || x == "") {
    alert("Name must be filled out");
    return false;
  }
}
</script>
</head>
<body>
<form name="myForm" action="demo_form.asp"
onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Валідація форм HTML частково може виконуватися автоматично браузером: якщо вказати для елемента форми атрибут required, то форма не буде відправлена на сервер, поки користувач не заповнить даний елемент.

Дії користувача і браузера на веб-сторінці генерують події. JS функції можуть бути встановлені у вигляді обробників подій: коли користувач взаємодіє з елементом сторінки, виконується задана функція.

Наведемо приклади можливих подій:

– HTML-сторінка закінчила завантажуватися;
– значення поля форми було змінено;
– була натиснута кнопка;
– покажчик миші пройшов над елементом.

Наведемо приклад завдання обробника події натискання на кнопку:

```
<button
onclick='getElementById("demo").innerHTML=Date()'>The
time is?</button>
```

Найбільш часто використовуються наступні події:

– Onchange – HTML-елемент був змінений;
– Onclick – користувач клікає на HTML елемент;
– Onmouseover – користувач проводить мишею над HTML елементом;
– Onkeydown – користувач натискає кнопку клавіатури;
– Onsubmit – користувач намагається відправити форму;
– Onload – браузер завершив завантаження сторінки.

5.2.2 Ajax

У класичній моделі веб-застосунку користувач натискає на існуючій сторінці на елемент, браузер формує і відправляє запит, сервер формує нову сторінку і відправляє її браузеру, браузер повністю перезавантажує сторінку.

AJAX (Asynchronous JavaScript and XML) – це технологія яка дозволяє створювати швидкі і динамічні веб-сайти.

AJAX дозволяє веб-сторінкам динамічно оновлювати свої частини шляхом фонового обміну даними з веб-сервером і без необхідності очікування повного перезавантаження сторінки. Користувач може продовжувати взаємодіяти зі сторінкою, поки дані вантажаться.

Наведемо приклад такого запиту:

```
<!DOCTYPE html>
<html>
<body>
<div id="demo"><h2>Let AJAX change this
text</h2></div>
<button type="button" onclick="loadDoc()">Change
Content</button>
</body>
```

```
</html>

function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200)
{
     document.getElementById("demo").innerHTML       =
xhttp.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

5.2.3 jQuery

jQuery – легка бібліотека JavaScript, девізом якої є «пиши менше, роби більше».

Призначення даної біблотеки – спрощення і прискорення написання JS. Вона допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними, виконувати AJAX запити.

Підключення бібліотеки проводиться таким чином:

```
<head>
<script src="jquery-1.12.0.min.js"></script>
</head>
```

или

```
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.1
2.0/jquery.min.js"></script>
</head>
```

Синтаксис jQuery спеціально підготовлений для вибору HTML елементів і виконання деяких дій над ними. Базовий синтаксис вибору елемента сторінки

$ ('selector').action (),

де знак $ позначає створення об'єкта jQuery,

селектор (selector) використовується для пошуку HTML елементів в DOM,

action () – дія, яку необхідно виконати над елементами.

Функція $ або jQuery вибирає елементи з DOM використовуючи CSS селектори і деякі додаткові специфічні. Наприклад:

$ (this) .hide () – сховати поточний елемент;

$ ("p").hide () – сховати всі <p> елементи;

$ (".test").hide () – сховати всі елементи з класом "test";

$ ("#test").hide () – сховати всі елементи з ідентифікатором "test".

Не можна намагатися використовувати DOM до повного завантаження сторінки. Скрипт може звернутися до ще не завантаженого вмісту, а це завжди призводить до помилок або несподіваних результатів. jQuery надає більш простий ніж стандартний підхід JS спосіб для запобігання передчасного виконання коду:

```
$(document).ready(function(){
   // jQuery methods go here...
});
```

або

```
$(function(){
   // jQuery methods go here...
});
```

jQuery також надає зручні засоби для роботи з подіями.

Наведемо приклад призначення обробника події:

```
$("p").click(function(){
  // action goes here!!
});
```

Для виклику події вручну необхідно виконати наступне:

```
$("p").click();
```

Одна з дуже важливих частин jQuery – це можливість маніпулювати DOM.

jQuery має множину методів для зручного доступу і зміни елементів і їх атірутов. Найбільш вживаними є методи:

– text() – встановлює або повертає текстовий вміст вибраних елементів;

– html() – встановлює або повертає вміст обраних елементів (включаючи HTML розмітку);

– val() – встановлює або повертає значення поля форми;

– attr() – встановлює або повертає значення атрибутів обраних елементів;

– css() – встановлює або повертає CSS стилі вибраних

елементів.

jQuery також спрощує роботу з XMLHttpRequest і надає зручні методи для виконання AJAX запитів.

Наведемо приклад:

```
$.ajax({
    "url": "foo/bar/mydata.txt",
    "type": "GET",
    "success": myAjaxSuccessFunction,
    "error": ajaxFailure
});
function myAjaxSuccessFunction(data) {
    // do something with the data
}
function ajaxFailure(xhr, status, exception) {
    console.log(xhr, status, exception);
}
```

5.3 Завдання на лабораторну роботу

5.3.1 Модифікувати подання списку фільмів за допомогою бібліотеки DataTables.

5.3.2 При редагуванні даних про фільм додати можливість модифікації відомостей про режисерів та виконавців ролей. За допомогою JS створити можливість додання максимум пятьох режисерів та будь-якої кількості акторів.

5.3.3 При введенні імен режисерів та акторів створити автодоповнення.

5.3.4 Реалізувати валідацію даних при редагуванні даних. Поле року випуску перевірити за допомогою регулярного виразу.

5.3.5 Реалізувати авторизацію та регистрацію користувачів. Дозволити редагування інформації лише авторизованим користувачаам.

5.4 Зміст звіту

5.4.1 Тема та мета роботи.

5.4.2 Результати роботи.

5.4.3 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

5.5 Контрольні запитання

5.5.1 Дайте характеристику мови JavaScript.

5.5.2 Як підключити JS скрипт до HTML сторінки?

5.5.3 Як вивести інформацію за допомогою JS?

5.5.4 Як оголосити змінну? Як визначити її тип? Які типи даніх є в JS?

5.5.5 Що таке обєкт в JS? Як з ним працювати (оголосити, створити обєкт, отримати доступ до методу, як адресуються обєкти)?

5.5.6 Як працювати з масивами в JS (створити, отримати доступ до елементу, пройти по масиву, визначити кількість елементів, методи масивів)?

5.5.7 Як працювати зі строками в JS (створити, визначити довжину, конкатинація строк, методи строк, перетворення чисел і строк)?

5.5.8 Яке призначення обєктів Math та Date? Які є методи для роботи з ними?

5.5.9 Як працювати з регулярними виразами в JS? Опишіть обєкт RegExp.

5.5.10 Опишіть синтаксис операторів if/else, for, while.

5.5.11 Як визначити функцію в JS? Як передати параметри в функцію?

5.5.12 Особливості ключового слова this в JS.

5.5.13 Поясніть поняття прототипу в JS.

5.5.14 Що таке DOM?

5.5.15 Які існують глобальні обєкти до яких може звертатися JS?

5.5.16 Як в JS отримати доступ до елементів сторінки?

5.5.17 Що таке ненавязливи JS?

5.5.18 Що таке події в JS? Поясніть спливання і перехоплення події.

5.5.19 Які події існують? Як створити обробник події?

5.5.20 Що таке jQuery? Які можливості надає ця бібліотека?

5.5.21 Який синтаксис jQuery? Поясніть селектори та дії, які можна виконувати над вибраними елементами сторінки.

5.5.22 Що таке AJAX? Як виконати AJAX-запит за допомогою jQuery.

ЛІТЕРАТУРА

1. Роббинс Дж. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дж. Роббинс. – 4-е издание. – Эксмо, 2014. – 516 с.

2. Томсон Л. Разработка Web-приложений с помощью PHP 5 и MySQL 5 / Л. Томсон, Л. Веллинг. – Вильямс, 2008. – 880 с.

3. Колисниченко Д.Н. PHP и MySQL. Разработка веб-приложений / Д.Н. Колисниченко. – СПб.: БХВ-Петербург, 2015. – 592 с.

4. Зандстра М. PHP: объекты, шаблоны и методики программирования / М. Зандстра. – 2-е изд.. – М.: Вильямс, 2009. – 480 с.

5. Никсон Р. Создаем динамические web-сайты с помощью PHP, MySQL и JavaScript / Р. Никсон. – Спб.: ПИТЕР, 2011. – 496 с.

6. Пауэрс Д. PHP. Создание динамических страниц / Д. Пауэрс. – М.: Рид Групп, 2012. – 640 с.

7. Кастаньетто Д. Профессиональное PHP программирование / Д. Кастаньетто, Х. Рават, С. Шуман, К. Сколло, Д. Велиаф. – СПб.: Символ-Плюс. – 2001. – 912 с.

8. Бранденбау Д. JavaScript: сборник рецептов / Д. Бранденбау. – СПб.: Питер, 2000. – 416 с.

9. Крейн Д. Ajax в действии / Д. Крейн. – М.: Вильямс, 2006. – 640 с.