НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Кафедра інформаційних і дистанційних технологій

"**ЗАТВЕРДЖУЮ**"
Декан факультету
Глазунова О.Г.

РОЗГЛЯНУТО І СХВАЛЕНО
на засіданні кафедри інформаційних
і дистанційних технологій
Протокол № 12 від "14" травня 2019 р.
В.о.завідувача кафедри
Кузьмінська О.Г.

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Розробка веб-застосунків

галузь знань 05 Соціальні та поведінкові науки, 12 інформаційні технології
спеціальність 051 Економіка, 122 Комп'ютерні науки та інформаційні технології
спеціалізація Економічна кібернетика, Інформаційні управляючі системи та технології, Комп'ютерний еколого економічний моніторинг
Факультет інформаційних технологій
Розробники: доцент, канд.екон.наук, Мокрієв Максим Володимирович
(посада, науковий ступінь, вчене звання)

Київ – 2019 р.

1. Опис навчальної дисципліни

Розробка веб-застосунків

(назва)

Галузь знань, напрям підготовки, спеціальність, освітній ступінь		
Освітній ступінь	*Магістр*	
Галузь знань	*05 Соціальні та поведінкові науки* *12 інформаційні технології*	
Спеціальність	*051 Економіка* *122 Комп'ютерні науки та інформаційні технології*	
Освітня програма	*Економічна кібернетика* *Інформаційні управляючі системи та технології* *Комп'ютерний еколого економічний моніторинг*	
Характеристика навчальної дисципліни		
Вид	Обов'язкова	
Загальна кількість годин	120	
Кількість кредитів ECTS	4	
Кількість змістових модулів	2	
Курсовий проект (робота) (за наявності)		
Форма контролю	*Іспит*	
Показники навчальної дисципліни для денної та заочної форм навчання		
	денна форма навчання	заочна форма навчання
Рік підготовки (курс)	*1*	*2*
Семестр	*2*	
Лекційні заняття	30 *год.*	*год.*
Практичні, семінарські заняття	*год.*	*год.*
Лабораторні заняття	30 *год.*	*год.*
Самостійна робота	60 *год.*	*год.*
Індивідуальні завдання	*год.*	*год.*
Кількість тижневих аудиторних годин для денної форми навчання	4 *год.*	

2. Мета та завдання навчальної дисципліни

Предметом вивчення дисципліни є сучасні інформаційні технології розробки багатофункціональних веб-додатків і веб-систем, здатних працювати як на стороні користувача, так і сервера.

Метою курсу є засвоєння необхідних знань з веб-технологій та формування практичних навичок застосування широкого спектру інформаційних технологій для створення сучасних веб-систем, здатних виконувати обробку даних як на стороні користувача, так і серверу.

Завданням курсу є отримання теоретичних знань із веб-технологій та веб-програмування; повного розуміння технологій створення різноманітних веб-систем та веб-застосунків; отримання практичних навичок, необхідних для створення багатофункціональних веб-застосунків та прогресивних веб-застосунків.

Навчальна дисципліна забезпечує формування ряду компетентностей:

- загальні компетентності:

ЗК2.Здатність до абстрактного мислення, аналізу, синтезу та встановлення взаємозв'язків між явищами та процесами.

ЗК4.Здатність використовувати інформаційні та комунікаційні технології в професійній діяльності.

ЗК7.Здатність свідомо та соціально-відповідально діяти на основі етичних міркувань і принципів академічної доброчесності.

ЗК8. Здатність проводити дослідження та презентувати результати.

- фахові компетентності:

ФК 11. Вміння планувати і розробляти проекти у сфері економіки, здійснювати її інформаційне, методичне, матеріальне, фінансове та кадрове забезпечення.

ФК 12. Здатність створювати та впроваджувати сучасні інформаційні системи на підприємствах різних сфер діяльності;

ФК 15. Здатність системно аналізувати на основі створених моделей економічні об'єкти та процеси, інтерпретувати отримані результати і на підставі зроблених висновків виробляти обґрунтовані управлінські рішення на всіх рівнях господарської ієрархії управління, усвідомлювати їх наслідки

Програмні результати:

4. Знання основних методів та засобів використання матеріальних, інформаційних, фінансових та інших ресурсів в індивідуальній та спільній науково-дослідній діяльності.

6. Знання в галузі інформатики й сучасних інформаційних технологій; знання основ системного аналізу та управління підприємствами (установами), методів обробки економічної інформації в різних сферах економічної діяльності, сучасних методів та моделей прогнозування розвитку економічних систем.

3. Програма та структура навчальної дисципліни для:

– повного терміну денної форми навчання;

Назви змістових модулів	Тижні	Кількість годин			
		Денна форма			
		Усього	в тому числі		
			лек	лаб	сам.р.
Змістовний модуль 1. РОЗРОБКА ФРОНТЕНДУ					
Тема 1. Вступ до веб-застосунків	1	8	2	2	4
Тема 2. Як працює веб та що таке веб-застосунки	2	8	2	2	4
Тема 3. Особливості роботи з HTML5	3	8	2	2	4
Тема 4. Особливості роботи з CSS3	4	8	2	2	4
Тема 5. Автоматизація розробки веб-застосунків	5	9	2	2	5
Тема 6. Основи програмування на JavaScript	6	8	2	2	4
Тема 7. Використання JavaScript бібліотеки jQuery	7	11	3	3	5
Разом за змістовим модулем		60	15	15	30
Змістовний модуль 2. РОЗРОБКА БЕКЕНДУ					
Тема 8. Використання архітектурного шаблону MVC	8	8	2	2	4
Тема 9. Основи програмування на PHP	9	8	2	2	4
Тема 10. Робота PHP з базами даних	10	8	2	2	4
Тема 11. Фреймворки для PHP	11	8	2	2	4
Тема 12. Створення прогресивних веб-застосунків	12	9	2	2	5
Тема 13. Робота з каркасом NodeJS	13	9	2	2	5
Тема 14. Клієнтська оптимізація веб-застосунків	14	10	3	3	4
Разом за змістовим модулем		60	15	15	30
Усього		120	30	30	60

4. Теми семінарських занять

№ з/п	Назва теми	Кількість годин
1		
2		
...		

5. Теми практичних занять

№ з/п	Назва теми	Кількість годин
1		
2		
...		

6. Теми лабораторних занять

№ з/п	Назва теми	Кількість годин
1	Налаштування робочого середовища програміста	2
2	Налаштування хостінгу	2
3	Робота з HTML5	2
4	Робота з CSS3	2
5	Робота з фреймворками для верстання	2
6	Створення односторінкового веб-застосунку	2
8	Модульний контроль	2
9	Основи програмування на PHP	2
10	Створення застосунку: Постановка задачі	2
11	Створення застосунку: Бекенд	2
12	Створення застосунку: Фронтенд (виведення інформації)	2
13	Створення застосунку: Фронтенд (маніпуляція даними)	2
14	Основи створення прогресивних веб-застосунків	2
15	Основи роботи з Node.js	2
16	Модульний контроль	2

7. Контрольні питання, комплекти тестів для визначення рівня засвоєння знань студентами.

- Контрольні питання

8. Методи навчання.

Засвоєння матеріалу забезпечується на лекціях, лабораторних заняттях та самостійній роботі у комп'ютерних класах, обладнаних локальними мережами, інтернетом і новітнім програмним забезпеченням. Лекції супроводжуються використанням презентацій, навчальних фільмів та мультимедійного обладнання для полегшення засвоювання матеріалу.

9. Форми контролю.

Контроль знань у слухачів курсу "Розробка веб-застосунків" передбачає такі контрольні заходи:

- самоконтроль - є первинною формою контролю знань, який обов'язково забезпечується дистанційним курсом шляхом надання студентам переліку питань (питань та відповідей на них), а також тестів для самоперевірки;
- поточний контроль - здійснюється через систему оцінки безпосередньо викладачем лабораторних занять та виконаних завдань для самостійної роботи;
- модульний контроль - здійснюється в автоматизованому режимі, основною формою якого є тестування;
- підсумковий контроль – це іспит в кінці семестра, які складаються очно в період призначений деканатом або за індивідуальним графіком, який затверджується навчальним планом. Основною формою підсумкового контролю є тестування та співбесіда, робота над практичним завданням та співбесіда.

10.Розподіл балів, які отримують студенти.

Оцінювання студента відбувається згідно положенням «Про екзамени та заліки у НУБіП України» від 20.02.2015 р. протокол № 6 з табл. 1.

Оцінка національна	**Рейтинг здобувача вищої освіти, бали**
Відмінно	**90 − 100**
Добре	**74 − 89**
Задовільно	**60 − 73**
Незадовільно	**0 − 59**

Для визначення рейтингу студента (слухача) із засвоєння дисципліни $\mathbf{R_{ДИС}}$ (до 100 балів) одержаний рейтинг з атестації (до 30 балів) додається до рейтингу студента (слухача) з навчальної роботи $\mathbf{R_{НР}}$ (до 70 балів): $\mathbf{R_{ДИС} = R_{НР} + R_{АТ}}$ **.**

11.Методичне забезпечення

Викладання навчальної дисципліни забезпечується сучасними технічними засобами навчання, які побудовані на новітніх інформаційно-комунікаційних

технологіях (мультимедійний комп'ютер, мультимедійний проектор, інтерактивний комплекс SMART Board, авторські засоби мультимедіа).
На заняттях і під час самостійній роботі студентів використовуються методичні рекомендації щодо вивчення дисципліни, ілюстративні комп'ютерні дидактичні матеріали, які розроблені на кафедрі, а саме:
- Опорні конспекти лекцій.
- Навчальні посібники.
- Робоча навчальна програма.
- Збірка тестових і контрольних завдань для поточного і модульного оцінювання навчальних досягнень студентів.
- Засоби підсумкового контролю (комп'ютерні програми поточного тестування, комплект завдань для підсумкового контролю).
- Електронний навчальний комплекс на платформі Moodle.

12. Рекомендована література

Основна:

1. Тузовский А.Ф. Проектирование и разработка web-приложений. Учебное пособие для СПО. - Москва: Юрайт, 2018. - 220с.
2. Разработка интернет-приложений. Учебное пособие для СПО. Е.Г. Сысолетин, С.Д. Ростунцев. - Москва: Юрайт, 2018. - 90с.
3. Дино Эспозито. Разработка современных веб-приложений: анализ предметных областей и технологий. - Издательство: Диалектика, 2017. - 464с.
4. Никсон Р. Создаём динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. - СПб.: Питер, 2016. - 768с.

Допоміжна:

1. Разработка веб-приложений с помощью PHP и MySQL. Люк Веллинг, Лора Томсон. - Издательство: Диалектика-Вильямс, 2017. - 848с.
2. Давид Скляр. Изучаем PHP 7: руководство по созданию интерактивных веб-сайтов. - Издательство: Диалектика, 2016. - 464с.
3. Этан Браун. Изучаем JavaScript: руководство по созданию современных веб-сайтов. - Издательство: Диалектика, 2018. - 368с.
4. Дэвид Макфарланд. Новая большая книга CSS. - Издательство: Питер, 2018. - 720с.
5. Владимир Дронов. Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS. - Издательство: БХВ-Петербург, 2018. - 768с.
6. Дакетт Д. Javascript и jQuery. Интерактивная веб-разработка. - Издательство: ЭКСМО, 2018. - 640с.
7. Крис Аквино, Тодд Ганди. Front-end. Клиентская разработка для профессионалов. Node.js, ES6, REST. - Издательство: Питер, 2018. - 512с.
8. Сильви Морето. Bootstrap в примерах. - Издательство: ДМК Пресс, 2017. - 314с.
9. Фримен А. Angular для профессионалов. - Издательство: Питер, 2017. - 800с.

10. Стоян Стефанов. React.js. Быстрый старт. - Издательство: Питер, 2018. - 304с.

13. Інтернет-ресурси:

1. Презентації про HTML5 з використанням HTML5:
   a. Презентація "Вав-ефект від HTML5", URL: http://www.htmlfivewow.com
   b. Презентація "CSS3+HTML5", URL: http://cssing.org.ua/examples/css3html5/#slide1
2. Что такое HTML5 – Различия между HTML и HTML5, URL: https://www.hostinger.ru/rukovodstva/chto-takoe-html-i-ih-razlichiya
3. Сайт про веб-розробки на технологіях HTML5, URL: https://html5book.ru/
4. Сайт навчання основам веб-технологій та веб-програмування, URL: https://www.w3schools.com/
5. Книжки з програмування: як читати і що саме, URL: https://dou.ua/lenta/articles/programming-books/
6. Що читають Front-End розробники: блоги, підручники та новини, URL: https://beetroot.academy/uk/blog/shho-chitayut-front-end-rozrobniki-blogi-pidruchniki-ta-novini/

Конспект лекцій

Тема 1. Вступ до веб-застосунків

Коли ми говоримо про створення та використання веб-застосунків, потрібно розуміти, що все це створюється на безі веб-технологій. Якщо коротко, то це набір технологій, які початково розроблялися для використання у веб-сервісах інтернету. Бурхливий розвиток цих технологій дозволив вийти за рамки інтернету, розширити традиційне сприйняття інтернету, що в свою чергу генерує нові технології, які розширюють межі.

Розвиток Веба привів до створення мов програмування елементів гіпертекстових документів на стороні клієнта (наприклад, JavaScript). Це привело до необхідності створення на сервері систему попередньої обробки файлу, що відправлявся. Веб-сервер ускладнився таким чином, що з'явилися різні прийоми динамічної генерації сторінок HTML, що потребувало виконувати на сервері програмні процедури. У запиті URL вставили виклик процедур, а на сервері реалізували технологію CGI (Common Gateway Interface). Тепер в запиті URL вказується процедура, яку треба виконати на сервері. Процедуру CGI можна написати на будь-якій мові, аби вона сприймала стандартне введення і стандартний вивід. У технології Java для цього створюються аплети, сервлети, використовується мова JSP(Java Server Pages).

Причини зростання ролі веб-застосунків зрозумілі - вони не вимагають установки програмних засобів у користувача і їх набагато простіше "підлаштовувати під цього самого користувача", такі застосування більше керовані з обох боків, менше вимог до клієнтського пристрою. Багато застосувань вже використовують для взаємодії з користувачем веб-інтерфейс, у веб-застосунки закладається функціональність, порівнянна з традиційними програмними продуктами.

У основі реалізації корпоративних інформаційних систем на базі архітектури Інтернет/Інтранет лежить принцип "відкритої архітектури", що багато в чому визначає незалежність реалізації корпоративної системи від конкретного виробника. Усе програмне забезпечення таких систем реалізується у вигляді аплетів або сервлетів (програм написаних на мові JAVA) або у вигляді CGI модулів (програм написаних, як правило, на Perl або C++).

Під клієнтською платформою доцільно розуміти не лише системне оточення на клієнтській стороні, але і спосіб організації призначеного для користувача інтерфейсу і його взаємодії з бізнес-логікою, розділеною у рамках додатка на клієнтську і серверну частину. У застосунку здійснюється взаємодія між клієнтською і серверною частиною, і це є визначальним для клієнтської платформи.

При забезпеченні веб-доступу до наявних баз даних (БД), можливий ряд технологічних і організаційних рішень. Практика використання веб-технологій для доступу до існуючих БД надає широкий спектр технологічних рішень, по різному пов'язаних між собою, що перекривають, взаємодіють і доповнюють. Вибір конкретних рішень при забезпеченні доступу, що залежить від специфіки конкретної СУБД і від ряду інших чинників, як платформа, сервер, наявність фахівців, здатних з мінімальними витратами освоїти певну гілку технологічних рішень, існування інших БД.

У загальному випадку інформаційна система, реалізована з використанням архітектури клієнт-сервер, включає Web-вузли з інтерактивним інформаційним наповненням, реалізованих за допомогою технологій Java, JavaBeans, JavaScript, PHP, ASP, Perl, що взаємодіють з базою даних, з одного боку, і з клієнтським місцем з іншого. База даних, у свою чергу, є джерелом інформації для інтерактивних програм реального часу.

Тема 2. Як працює веб та що таке веб-застосунки

Розробка веб-застосунків являє собою широке поняття, що охоплює процес написання веб-сторінки чи сайту. Веб-сторінки пишуться за допомогою HTML, CSS і JavaScript. Ці сторінки можуть являти собою простий текст і графічні елементи, що буде нагадувати документ. Сторінки можуть також бути інтерактивними або відображати інформацію, що змінюється. Інтерактивні серверні сторінки трохи складніше писати, але за їх допомогою створюються більш насичені сайти. Сьогодні більшість сторінок є інтерактивними, вони надають сучасні онлайн сервіси, такі як кошик покупця, динамічна візуалізація і навіть складні соціальні мережі.

Веб-сторінка
⇩
Веб-сайт
⇩
Веб-застосунок

Нагадаємо основні відмінності між поняттями веб-сторінка, веб-сайт та веб-застосунок.

Веб-сторінка - файл, який містить інформацію описану мовою HTML (або XHTML) і може включати гіперпосилання на інші web сторінки.

Кожна web сторінка має URL адресу за допомогою якої вона може бути отримана.

Web сторінки можуть бути статичними або динамічними:

-статичні (html або htm) сторінки передаються браузеру без обробки;

-динамічні сторінки попередньо обробляються на сервері в результаті чого виходить HTML документ, який і передається браузеру.

Веб-сторінка

Веб-сайт - це набір логічно пов'язаних веб-сторінок, зображень (images), відео або інших цифрових даних, які доступні на веб-сервері.

Веб-сайт має деяку початкову сторінку (homepage), використовуючи яку можна перейти до інших сторінок сайту.

Сторінки web сайту зазвичай розділені по каталогах, які створюють ієрархічну структуру сайту. Якщо початкова сторінка сайту в URL не вказується, то веб-сервер використовує сторінку зі стандартним ім'ям (default або index). Тобто, для входу на сайт достатньо вказати його доменну адресу.

Веб-сторінки та веб-сайт

Веб-застосунок — розподілений застосунок, в якому клієнтом виступає браузер, а сервером — веб-сервер. Браузер може бути реалізацією так званих тонких клієнтів — логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у відображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-застосунки є міжплатформовими сервісами. Унаслідок цієї універсальності і відносної простоти розробки веб-застосунки стали широко популярними в кінці 1990-х — початку 2000-х років. Далі підходи до створення веб-застосунків тільки удосконалювалися. На сьогодні веб-застосунки все більше за можливостями прирівнюються до звичайних прикладних програм.

Веб-застосунки

Істотною перевагою побудови веб-застосунків для підтримки стандартних функцій браузера є те, що функції повинні виконуватися незалежно від операційної системи клієнта. Замість того, щоб писати різні версії для Microsoft Windows, Mac OS X, GNU/Linux й інших операційних систем, застосунок створюється один раз для довільно обраної платформи і на ній розгортається.

Інший (менш універсальний) підхід полягає у використанні Adobe Flash або Java-аплетів для повної або часткової реалізації призначеного для користувача інтерфейсу. Оскільки більшість браузерів підтримують ці технології (зазвичай, за допомогою додаткових модулів-плагінів), Flash- або Java-застосунки можуть легко виконуватись. Оскільки вони надають програмістові більший контроль над інтерфейсом, то здатні обходити багато несумісностей у конфігураціях браузерів, хоча несумісність між Java або Flash реалізаціями клієнта може спричиняти різні ускладнення. У зв'язку з архітектурною схожістю з традиційними клієнт-серверними
застосунками, певним чином «товстими» клієнтами, існують суперечки щодо коректності зарахування подібних систем до веб-застосунків; альтернативний термін «Насичений інтернет-застосунок» (англ. Rich Internet Application).

Тема 3. Особливості роботи з HTML5

HTML5 це мова для структурування та презентації контенту для World Wide Web. Це п'ята версія стандарту HTML і станом на 2012 рік знаходиться в стадії розробки. Її метою було поліпшити мову підтримкою новітніх мультимедіа зберігаючи при цьому легкість читання для людей а також забезпечити сумісність із різноманітними пристроями та платформами. HTML5 покликаний підняти не тільки HTML4, але XHTML1 і DOM2HTML (зокрема, JavaScript).

Суть HTML5

HTML5 – це також спроба визначити єдину мову розмітки, яка може бути записана в будь-якому HTML або XHTML синтаксисі. Вона розширює, удосконалює і раціоналізує розмітку для документів, а також надає і API-інтерфейс розмітки для побудови складних веб-застосувань.

Загалом можна сказати наступне:

HTML5 – це, швидше, нова платформа для створення веб–застосувань, ніж стандарт, який продовжує традиції попередників.

HTML5 регламентує взаимодію з JavaScript через посередництво об’єктної моделі документу.

HTML 5 підтримує всі елементи HTML4.

В практичному плані, HTML5 додає безліч нових синтаксичних особливостей. До них відносяться теги <video>, <audio> і <canvas>, з метою інтеграції із SVG контентом. Ці функції спроектовані так, щоб забезпечити можливість легко включати і обробляти мультимедіа та графічний контент на веб-сторінках без необхідності вдаватися до власних плагінів та API. Інші нові елементи, такі як

<section>, <article>, <header> і <nav>, призначені для збагачення смислового змісту документів. Нові атрибути були введені для тих же цілей, а деякі елементи й атрибути були вилучені. Деякі елементи, такі як <a>, <cite> і <menu> були змінені, переглянуті або стандартизовані. DOM API є невід'ємною частиною специфікації HTML5.

Як вже відмічалося раніше, HTML5 робить великий тиск на семантику веб документу. На практиці це означає, що багато чого не буде видимим для кінцевого користувача, натомість буде добре помітним для пошукових серверів та програмістів.

Перше з чим стикаються при розробці веб-сайту - це загальна структура документу. До якої структури сайту ми зазвичай звикли (тут будемо говорити про найбільш загальний підхід без унікальних дизайнів).

Шаблон

Які підходи можливі для створення такої сторінки? Якщо підходити методологічно до питання дизайну веб-сторінок, то можна виділити такі групи (школи) дизайну:

текстовий;
фреймовий;
табличний;
контейнерний.

Чітких меж між вказаними школами, виділити неможна. Підходи до дизайну, як і абсолютна більшість веб-технологій, стандартів та правил з’явилися і розвивалися еволюційно.

Текстовий підхід до дизайну сайтів є першим та найпростішим. Суть дуже проста: основним вмістом веб-сторінки є структурований текст, способи оформлення, в більшості, обмежуються простим форматуванням тексту, зміною шрифтів, кольорів тексту тощо. Оскільки до зовнішнього виду сучасної веб-сторінки ставляться певні вимоги (всі ми звикли до такого шаблону , як на схемі), обійтись тільки текстовим верстанням досить складно. Фактично цим підходом в чистому виді нихто не користується, навіть з огляду на те, що час завантаження сторінок при цьому мінімальний а верстання дуже просте. Крім того, загальні елементи для всіх веб-сторінок сайту не підвантажуються, а прописуються для кожної сторінки. Тобто кожного разу при відкритті наступної сторінки вантажаться всі її елементи, навіть у випадку, коли міняється фактично, у порівнянні з попередньою сторінкою, тільки її змістовна частина.

В якості рішення, яке ліквідовує недоліки текстового підходу до дизайну сайтів, стала ідея використання фреймів – елементів веб-сторінок, містиме яких підвантажується із зовні. Інакше кажучи, фрейм відображає містиме іншої веб-сторінки, адресу якої прописують у параметрах фрейму. Таким чином, веб-сторінка складається з набору фреймів, при цьому підвантажується тільки потрібний контент. Структуру фреймів, правда, при цьому змінити не можна. Як і текстовий дизайн, фреймовий наразі практично не застосовується.

Ідея табличного верстання сайтів лежить на поверхні. На веб-сторінці розміщується html-таблиця, в клітинки якої і розміщують контент. Відповідно, візуально веб-сторінку можна розділити на необхідну кількість блоків ("шапка" сайту, панель навігації і т.д.). Табличний дизайн є дуже популярним та розповсюдженим, особливо між початківців. Основною перевагою підходу є його зрозумілість та можливість уникнути проблем сімісності. С врахуванням того, що до кожної клітинки таблиці можні застосувати власний CSS стиль, можна говорити про те, що підхід ще може посперечатися про свої перспективи.

Контейнерний підхід при верстанні та дизайні сайтів заключається у використанні елементів - контейнерів для розташування контенту. Контейнери (парний тег) є блочними елементами, до кожного контейнеру можна застосувати індивідуальний стиль. Також контейнер може відображати вміст зовнішньої, по відношенню до нього веб-сторінки, при цьому, на відміну від фреймів, контейнери стандартизовані консорціумом W3C. У порівнянні ж з табличним підходом, створювані сторінки "легші" з точки зору кількості коду.

На сьогодні, мабуть, більшість сайтів використовують гібридні підходи. Наприклад, основний поділ сторінки робиться контейнерами, а для додаткової розбивки використовуються таблиці та форматування тексту.

В HTML5 для семантичного поділу сторінки використовують такі теги:
<section> Елемент <section> визначає основний документ або розділ. В даному контексті це тематичне групування вмісту, як правило, з заголовком. Наприклад, розділами можуть бути глави, вкладки в діалоговому вікні з вкладками або пронумеровані розділи дисертації. Головна сторінка веб-сайту може бути розбита на розділи для вступу, виводу новин, контактної інформації.
<nav> Елемент <nav> це розділ навігаційних посилань, що містить посилання на інші сторінки. Не всі групи посилань повинні братися в тег <nav> — тільки раздели з основною навігацією. Зокрема, в підвалинах сторінки часто міститься короткий список посилань, на кшталт: умови використання сайту, головна сторінка, сторінка з авторськими правами, тощо. Для подібних випадків достатньо тега <footer>, без використання <nav>.
<article> Елемент <article> задає компонент сторінки, призначений для самостійного розповсюдження або повторного використання, наприклад в

синдикації. Це може бути повідомлення форуму, журнальна або газетна стаття, запис у блозі, коментар користувача, інтерактивний віджет, ґаджет або будь-який інший незалежний контент.

<aside> Цей елемент представляє розділ сторінки, який не має прямого відношення до вмісту сторінки і його можна відокремити від контенту. В поліграфії такі ділянки часто виділяють плашкою. Тег <aside> може використовуватися для виводу цитат, бокових панелей, реклами, навігації через <nav> та іншого контенту, який считается можна відокремити від основного вмісту сторінки.

<hgroup> Елемент <hgroup> задає заголовок розділу та застосовується для групування кількох тегів <h1>–<h6>, коли заголовок включає кілька рівнів, таких як підзаголовки, альтернативні назви або гасла.

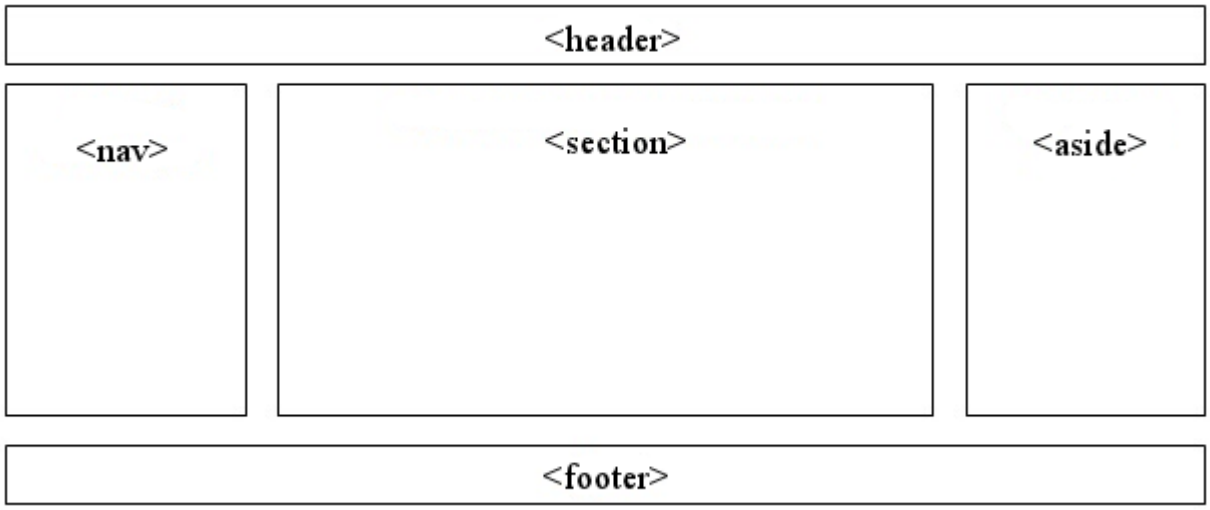
<header> Представляє собою групу з вступних або навігаційних засобів. Елемент <header> зазвичай містить заголовок розділу (теги <h1>–<h6> або <hgroup>), але це не обов'язково. <header> також може використовуватися для обгортання розділу змісту, форми пошуку, або відповідних логотипів.

<footer> Задає нижній колонтитул для розділу змісту або підвал для сторінки. Елемент <footer> зазвичай містить інформацію про розділ, таку як: ім'я автора, посилання на відповідні документи, авторські данні тощо. Колонтитули не обов'язково повинні виводитися в кінці розділу, як це звичайно робиться.

<time> Представляє собою або час в 24-годинному форматі, або точну дату, яку при бажанні можна сіміщати з часом та вказівкою часового поясу.

<mark> Відмічає фрагмент документу або виділяє його у довідкових цілях.

З врахуванням цього, представлену раніше структуру можемо зробити так:



Тема 4. Особливості роботи з CSS3

Кас-кадні таб-лиці стилів (англ. Cascading Style Sheets або ско-ро-че-но CSS) — спеціальна мова, що ви-ко-ри-сто-вується для відо-бра-жен-ня сторінок,

на-пи-са-них мо-ва-ми розмітки даних. Найбільш часто CSS ви-ко-ри-сто-ву-ють для візу-аль-ної пре-зен-тації сторінок, на-пи-са-них HTML та XHTML.

CSS ви-ко-ри-сто-вується ав-то-ра-ми та відвіду-ва-ча-ми веб-сторінок для того щоб визна-чи-ти ко-льо-ри, шриф-ти, верст-ку та інші ас-пек-ти вигля-ду сторінки. Одна з го-лов-них пе-ре-ваг — мож-ливість розділити зміст сторінки (або кон-тент, на-пов-нен-ня, за-зви-чай HTML, XML або подібна мова розмітки) від вигля-ду до-ку-мен-ту (що описується в CSS).

Таке розділення може по-к-ра-щи-ти сприй-нят-тя та до-ступність кон-тен-ту, за-без-пе-чи-ти більшу гнучкість та кон-троль за відо-бра-жен-ням кон-тен-ту в різних умо-вах, зро-би-ти кон-тент більш струк-ту-ро-ва-ним та про-стим, при-бра-ти по-вто-ри та ін. Один і той же HTML або XML до-ку-мент може бути відо-бра-жен-ний по-різному в за-леж-ності від ви-ко-ри-ста-но-го CSS.

Що дає використання CSS?

- Відображати один і той же документ в різних стилях.
- Декілька дизайнів сторінки для різних пристроїв. Наприклад, на екрані дизайн буде розрахований на велику ширину, під час друку меню не виводитиметься, а на смартфоні меню буде внизу, під вмістом.
- Зменшення часу завантаження сторінок сайту за рахунок перенесення правил відображення в окремий CSS-файл. В цьому випадку браузер завантажує тільки структуру документа і дані, що зберігаються на сторінці, а стильові правила цих даних завантажуються браузером тільки один раз і кешуються.
- Простота подальшої зміни дизайну. Не потрібно правити кожну сторінку, а лише змінити CSS-файл.
- Додаткові можливості оформлення. Наприклад, за допомогою CSS-розмітки можна зробити так, щоб меню було завжди видно при скролінгу сторінки, або прибрати підкреслення у посилань.
- Дозволяє створювати складну і пропрацьовану техніку дизайну.

Правила в CSS працють по каскаду (пріорітету, вазі). Це дозволяє отримати передбачуваний результат у випадку, коли до одного елемента, одночасно, застосовуються декілька стильових правил.

Правило має наступну структуру:

Всі CSS-правила складаються з селектора і блоку оголошень (укладеного у фігурні дужки). Всередині блоку оголошень може знаходитися одне або кілька оголошень, розділених крапкою з комою. Оголошення - це рядок, складений з css-властивості та її значення.

Кожне правило починається з селектора (покажчика), що вказує на ті html-елементи, до яких ми будемо застосувати css-правило. У блоці оголошень відбувається найцікавіше - ми встановлюємо правила відображення обраних нами елементів, визначаємо їх властивості - розмір, колір, грані, поля, положення на екрані і т.д.

Тема 5. Автоматизація розробки веб-застосунків

Повторне використання (reuse) - це використання для нових розробок будь-яких порцій формалізованих знань, здобутих під час реалізації завершених розробок програмних систем.

Накопичений досвід розроблення систем програмного забезпечення може бути зафіксовано в різних формах, починаючи від конкретних параметризованих програмних модулів і кінчаючи програмними архітектурами та середовищами.

Класи

В об'єктно-орієнтованому програмуванні клас — це спеціальна конструкція, яка використовується для групування пов'язаних змінних та функцій.

Клас можна порівнювати з формою для випічки печива — форма одна, а печива можна випекти безліч. Печиво — це конкретні об'єкти, екземпляри класу печиво, яке може бути з різною начинкою.

Інший приклад, можна створити загальний клас Людина з полями Ім'я та Прізвище, рік народження, професія, зарплата. При створенні ж на основі класу конкретного екземпляру, дані поля заповнюються конкретними даними про певну людину. Обробкою цих даних можуть займатися відповідні методи. Наприклад, можна створити метод для обчислення віку людини, тощо. Тобто, через методи реалізується поведінка об'єктів.

У сучасних об'єктно-орієнтованих мовах програмування (в тому числі в php, Java, C++, Oberon, Python, Ruby, Smalltalk, Object Pascal) створення класу зводиться до написання деякої структури, що містить набір полів та методів (серед останніх особливу роль грають конструктори, деструктори, фіналізатори). Практично клас може розумітися як якийсь шаблон, за яким створюються об'єкти — екземпляри цього класу. Усі примірники одного класу створені за одним шаблоном, тому мають один і той же набір полів та методів.

Мова JavaScript хоча і є об'єктноорієнтованою, але має свою специфіку. Тому до останнього часу не йшлося про класичне використання класів. Проте, з часу специфікації ECMAScript 5 говорять про повноцінне використання класів у JavaScript.

На сьогодні вже майже ніхто не створює для розповсюдження окремі класи. Класи розрослися до бібліотек і далі використовуються в такому вигляді. Прикладом може бути клас написаний на PHP для роботи з файлами формату MS Excel, який має назву PHPExcel (який ще до цього багато хто називає класом). Наразі це велика бібліотека PHPOffice.

Бібліотеки

Бібліотека — це сукупність підпрограм, методів чи об’єктів, що можуть використовуватися у розробці програмного забезпечення. Зазвичай складові бібліотеки мають пов’язаний функціонал, як-от графічна візуалізація даних на сторінці або розробка адаптивних інтерфейсів, і набір інструментів для його реалізації. Бібліотеки на сайт здебільшого додаються двома способами: в проект сайту додається файл із завантаженою бібліотекою або шляхом CDN(Content Delivery Network) — це коли бібліотека підвантажується зі стороннього серверу по ссилці.

Фреймворки

Фреймворк — це таке програмне забезпечення, по-суті схоже чимось на велику бібліотеку, що полегшує розробку та об’єднання різних компонентів великого програмного проекту. Але фреймворк відрізняється від бібліотеки тим, що бібліотека не впливає на архітектуру основного програмного продукту і не накладає на нього ніяких обмежень. Фреймворк може включати в себе допоміжні програми, коди бібліотеки, сценарії мови та інше ПЗ, що значно полегшує розробку об’єднання різних компонентів великого програмного проекту і робить HTML більш гнучкішим.

Часто буває важко розрізнити, що є бібліотекою, а що фреймворком. Але різниця все ж є. Коли ви створюєте свою програму з підключеними бібліотеками, то викликаєте ці бібліотеки , коли вони вам необхідні. Функції фреймворка, на відміну від бібліотеки, не викликаються вами, а навпаки, ваш код викликається з нього. Фреймворк можна уявити собі у вигляді напівфабрикату програми, до якого ви дописуєте потрібну функціональність самі.

CMS системи

CMS(Content Management System) — це програмне забезпечення для керування вмістом сайту. Основна функція CMS — відображати сторінки сайту користувачам, формуючи їх вміст “на льоту” з попередньо визначених шаблонів з дизайном та контентом (різноманітними матеріалами, які зберігаються в базі даних). Інша функція — допомогти власникові сайту без яких-небудь спеціальних навичок керувати сайтом, тобто публікувати нові сторінки, новини, викладати відео, робити посилання на зовнішні ресурси і так далі. Серед CMS особливо популярні конструктори сайтів (simplesite, wix, tilda ets) — дозволяють будь-якому користувачу зробити власний сайт узагалі не знаючи програмування методом drug-n-drop.

CMS присутні практично у всіх мовах програмування, підтримуваних ООП і не тільки: Java, PHP, Python, Ruby, Perl, JavaScript та інші.

CMF системи

CMF (Content Management Framework) - це, згідно з найбільш поширеним визначенням, фреймворк-система для управління вмістом сайту, а також інструментарій для створення систем управління контентом або ж веб-додатків взагалі. CMF може включати різноманітні бібліотеки кодів, допоміжні програми, мову програмування, мову сценаріїв. Об'єднання різних компонентів програмного проекту зазвичай відбувається за рахунок використання єдиного API (application programming interface - інтерфейс прикладного програмування).

Ряд CMS, що надають API для розширення своєї функціональності, претендують на звання CMF.

Тож, можна сказати, що CMF - це поняття більш широке ніж CMS, і кожна CMF є CMS, однак не кожна CMS - це CMF.

Як правило, на основі CMF створюються CMS - готові системи управління вмістом, а ті, в свою чергу, є основою для створення повноцінних сайтів.

CRM системи

Не менш популярні у звичайних користувачів CRM(Customer Relationship Management) — але вони навідміну від CMS призначені для більш тіснішої взаємодії із замовником, клієнтами. CRM спрямовані на автоматизацію, спрощення, пришвидшення процесів між ними. Наприклад, це програми бухгалтерського обліку, складські програми, програми оптимізації маркетингу тощо.

Тема 6. Основи програмування на JavaScript

В наш час веб-переглядачі (веб-браузери) надають потужні інструменти та ресурси для розробників, щоб реалізувати динамічні, “живі” і багаті на функціонал веб-сайти та аплікації. Відповідно, залишається все менше причин реалізувати декстопну програму (програму, що потребує інсталяції на вашому комп’ютері) і більшість задач можна реалізувати у вигляді веб-програми. Це ще одна причина чому веб є на стільки перспективним напрямком.

Завдяки швидкому прогресу веб-браузерів з’явилась окрема спеціалізація у вебі: фронтенд програміст. Даний спеціаліст займається розробкою багатого і динамічного функціоналу на стороні браузера. Його основна мова програмування – JavaScript. З її допомогою він створює так звані “живі” аплікації: веб-сайти, які без перевантаження і навігації постійно надають користувачу оновлену інформацію. Це і чати, і динамічні ленти новин, і будь-які інші веб-сторінки, що безперебійно працюють без перевантаження.

Дехто думає, що фронтенд розробник – це щось середнє між верстальщиком та бекенд програмістом. Я ж хочу сказати, що JavaScript це повноцінна мова програмування і в складних великих проектах вимагає неабиякого досвіду та знань. І, вважаю, що однаково складно є освоїти на хорошому рівні одну із серверних мов і JavaScript у браузері.

Об'єктна модель документа (Document Object Model - DOM) є стандартом, запропонованим веб-консорціумом, і регламентує спосіб представлення вмісту документа (зокрема веб-сторінки) у вигляді набору об'єктів. Під вмістом розуміється все, що може знаходитися на веб-сторінці: малюнки, посилання, абзаци, текст і т. д.

На відміну від об'єктної моделі браузера (BOM), яка унікальна для кожного браузера, об'єктна модель документа є стандартом і повинна підтримуватися всіма браузерами. І хоча на практиці підтримка DOM реалізована не в повній мірі, тим не менш необхідно прагнути слідувати вимогам цього стандарту як виробникам браузерів, так і розробникам веб-сайтів.

DOMB DOM документ представляється у вигляді деревовидної структури, що є однією з найбільш уживаних структур в програмуванні. Це забезпечує уніфікований спосіб навігації по документу.

Об'єктна модель описує кожний HTML- документ як набір окремих об'єктів – зображень, абзаців, списків і т. д. до найбільшого рівня, навіть до окремих символів. Кожний об'єкт може мати властивості, визначені у вигляді атрибутів. Наприклад, абзаци <P> має атрибут вирівнювання ALIGN, який може набувати

значень left, right, center. В об'єктній моделі атрибут називають властивістю об'єкта. Об'єкт має також свої методи і події, які можуть відбуватися з ним і впливати на нього. Наприклад, зображення <img> має подію OnMouseOver, яка відбувається тоді, коли користувач розміщує над ним вказівник миші. Можна керувати станом об'єктів, використовуючи методи з деякого набору стандартних методів. Все це й складає концепцію DOM як платформо-незалежного програмного інтерфейсу, який дає змогу програмам та скриптам керувати вмістом HTML-документів, змінювати їх структуру та оформлення.

Об'єкти з однаковим набором властивостей, методі і подій об'єднуються у класи однотипних об'єктів. Класи – це описи можливих об'єктів. Самі об'єкти створюються тільки після завантаження документа браузером, або як результат роботи програми. Про це слід пам'ятати, щоб не звернутися до об'єкта, якого немає.

Об'єкти мають фіксовані імена і певні властивості. Наприклад, вікну браузера відповідає об'єкт Window, а HTML-документу, завантаженому в браузер, – об'єкт Document. Звичайні властивості – це змінні з фіксованими іменами, які мають певні значення. Одні властивості можна лише переглядати, інші можна змінювати. Для доступу до властивостей об'єкта у мовах сценаріїв використовують такий синтаксис:

Об'єкт.властивість

Наприклад, значенням властивості Document.Location є URL-адреса HTML-документа.

Властивістю об'єкта може бути інший об'єкт. При цьому перший об'єкт називають також батьківським (parent), а другий – нащадком (child). Якщо ми хочемо звернутися до властивості або методу об'єкта Object2, який міститься в об'єкті Object1, то слід записати:

- Object1.Object2.властивість
- Object1.Object2.метод()

Тема 7. Використання JavaScript бібліотеки jQuery

Вперше в 2006 році випущений Джоном Ресігом, jQuery поставив себе як крос-платформну бібліотеку JavaScript, що полегшило написання рішень на стороні клієнта.

В той час, коли це відбулося, це було особливо корисним через невідповідності, які існували в реалізаціях JavaScript в Internet Explorer, Firefox і, зрештою, Google Chrome (якого ще не було випущено до 2008 року).

Як визначено у Вікіпедії:

jQuery - це крос-платформна бібліотека JavaScript, розроблена для спрощення сценаріїв HTML на стороні клієнта. jQuery - це найпопулярніша бібліотека JavaScript, що використовується сьогодні, з установкою на 65% з 10 мільйонів найбільших сайтів, що продаються в Інтернеті. jQuery - це безкоштовне програмне забезпечення з відкритим кодом, ліцензоване за ліцензією MIT.

Крім того, сам веб-сайт jQuery говорить:

jQuery - це швидка, маленька та багатофункціональна бібліотека JavaScript. Це робить такі речі, як переміщення HTML-документів та маніпулювання, обробка подій, анімацію та Ajax, набагато простіше за допомогою простого API, який працює в безлічі браузерів. З поєднанням універсальності та розширюваності, jQuery змінило спосіб, у який мільйони людей пишуть JavaScript.

jQuery - javascript бібліотека, використання якої робить розробку javascript коду набагато швидше, простіше і приємніше.

Давайте розберемося.JQuery - бібліотека Javascript, що фокусується на взаємодії Javascript і HTML. Була опублікована на комп'ютерній конференції «Barcamp» у Нью-Йорку Джоном Ресигом в 2006 році. У чому головна перевага jquery? Вона закладена на рівні ядра - це вибір елементів об'єктної моделі документів. Крім того, завдяки наявності плагінів, базова функціональність jquery може бути розширена.

Основне завдання jQuery – це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та XPath селекторів. Також даний фреймворк надає інтерфейси для Ajax-застосувань, обробників подій і простої анімації. Завдяки тому, що обсяг програмного коду jquery менше, ніж обсяг стандартного коду Javascript, скорочуються часові витрати на розробку елементів веб-сторінки. Сам програмний код більш зрозумілий у порівнянні з Javascript.

В ядро jquery закладено такий основний функціонал:

функції ядра;
робота із селекторами;
робота з атрибутами;
обхід дерева DOM;
маніпуляції елементами;
робота з CSS-властивостями елементів;

робота з подіями;
візуальні ефекти;
взаємодія з ajax;
утиліти.

Для маніпулювання потрібними елементами сторінки в Javascript є кілька способів знайти їх на сторінці серед безлічі інших об'єктів. Ці способи вимагають запам'ятовування великої кількості інформації, у той час як для пошуку елемента за допомогою jquery необхідно лише пам'ятати ID елемента, з яким ви прагнете працювати.

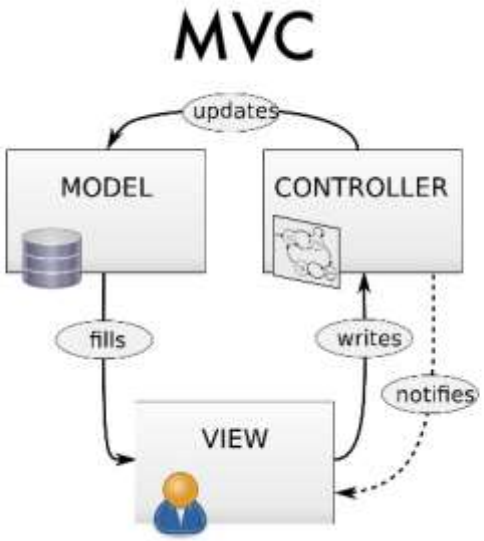
Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів. Наприклад, так:

```
$('#test') //знаходимо елемент з id="test"
.text('Клікни по мені') //додаємо до нього текст "Клікни по мені"
.addClass('myAlert') //додаємо клас "myAlert"
.css('color','red') //встановлюємо колір тексту червоним
.attr('alert','Привіт, світ!') // додаємо атрибут "alert" із значенням "Привіт, світ!"
.bind( // додаємо в обробник події click функцію, яка відкриє модальне
'click', // вікно із текстом, що вказаний в атрибуті "alert" ("Привіт, світ!")
function(){alert($(this).attr('alert'))}
);
```

Так історично склалося, що браузери по різному обробляють код, написаний на Javascript. Відсутність єдиного стандарту призвело до того, що програмістам доводиться писати зайві рядки коду для сумісності з усіма популярними браузерами. Бібліотека jQuery розроблялася таким чином, щоб її можна було використовувати незалежно від браузера. jQuery надає розробникові базовий набір функцій, який буде працювати у всіх браузерах.

Тема 8. Використання архітектурного шаблону MVC

Концепція MVC (Model-View-Controller: модель-вид-контролер) дуже часто згадується в світі веб програмування в останні роки. Кожен, хто хоч якось пов'язаний з розробкою веб додатків, так чи інакше стикався з даними акронімом. Сьогодні ми розберемося, що таке - концепція MVC, і чому вона стала популярною.



MVC - це не шаблон проекту, це конструкційний шаблон, який описує спосіб побудови структури нашого застосування, сфери відповідальності та взаємодія кожної з частин в даній структурі. Вперше вона була описана в 1979 році, звичайно ж, для іншого оточення.

Тоді не існувало концепції веб застосунків. Tim Berners Lee (Тім Бернерс Лі) посіяв насіння World Wide Web (WWW) на початку дев'яностих і назавжди змінив світ. Шаблон, який ми використовуємо сьогодні, є адаптацією оригінального шаблону до веб розробки.

Шалена популярність даної структури у веб застосунках склалася завдяки її включенню в два середовища розробки, які стали дуже популярними: Struts і Ruby on Rails. Ці дві середовища розробки намітили шляхи розвитку для сотень робочих середовищ, створених пізніше.

Додаток розділяється на три основних компоненти, кожен з яких відповідає за різні завдання. Давайте детально розберемо компоненти на прикладі.

Контролер (Controller)
Контролер керує запитами користувача (одержувані у вигляді запитів HTTP GET або POST, коли користувач натискає на елементи інтерфейсу для виконання різних дій). Його основна функція - викликати і координувати дію необхідних ресурсів і об'єктів, потрібних для виконання дій, що задаються користувачем. Зазвичай контролер викликає відповідну модель для задачі і вибирає відповідний вид.

Модель (Model)
Модель - це дані і правила, які використовуються для роботи з даними, які представляють концепцію управління додатком. У будь-якому додатку вся

структура моделюється як дані, які обробляються певним чином. Що таке користувач для додатка - повідомлення або книга? Тільки дані, які повинні бути оброблені відповідно до правил (дата не може вказувати в майбутнє, e-mail повинен бути в певному форматі, ім'я не може бути довшим X символів, і так далі).

Модель надає контролеру представлення даних, які запросив користувач (повідомлення, сторінку книги, фотоальбом, тощо). Модель даних буде однаковою, незалежно від того, як ми хочемо представляти їх користувачеві. Тому ми вибираємо будь-який доступний вид для відображення даних.

Модель містить найбільш важливу частину логіки нашого застосування, логіки, яка вирішує завдання, з якою ми маємо справу (форум, магазин, банк, тощо). Контролер містить в основному організаційну логіку для самого додатка (дуже схоже на ведення домашнього господарства).

Вид (View)
Вид забезпечує різні способи представлення даних, які отримані з моделі. Він може бути шаблоном, який заповнюється даними. Може бути кілька різних видів, і контролер вибирає, який підходить якнайкраще для поточної ситуації.

Веб застосунок зазвичай складається з набору контролерів, моделей і видів. Контролер може бути влаштований як основний, який отримує всі запити і викликає інші контролери для виконання дій в залежності від ситуації.

Тема 9. Основи програмування на PHP

PHP означає Препроцессор гипертекста PHP. Це серверна мова програмування, створена спеціально для динамічних сторінок Web. Сьогодні PHP використовується сотнями тисяч розробників. Кілька мільйонів сайтів написано на PHP, що складає понад 20% доменів інтернету.

Що вміє PHP? "PHP може все», - заявляють його творці. В першу чергу PHP використовується для створення скриптів, що працюють на стороні сервера, для цього його, власне, і придумали. PHP здатний вирішувати ті ж завдання, що і будь-які інші CGI - скрипти, у тому числі обробляти дані HTML-форм, динамічно генерувати HTML сторінки і т.п. Але є й інші області, де може використовуватися PHP. Всього виділяють три основні області застосування PHP.

Перша область, як уже говорилося, - це створення додатків (скриптів), які виконуються на стороні сервера. PHP найбільш широко використовується саме для створення такого роду скриптів. Для того щоб працювати таким чином, знадобиться PHP-парсер (тобто обробник PHP - скриптів) і веб-сервер для

обробки скрипта, браузер для перегляду результатів роботи скрипта, ну, і, звичайно, який-небудь текстовий редактор для написання самого PHP-коду. У цьому курсі ми будемо обговорювати створення саме серверних додатків, як приклад використання мови PHP.

Друга область - це створення скриптів, що виконуються в командному рядку. Тобто за допомогою PHP можна створювати такі скрипти, які будуть виконуватися, незалежно від веб-сервера і браузера, на конкретній машині. Для такої роботи буде потрібно лише парсер PHP (в цьому випадку його називають інтерпретатором командного рядка (CLI, командний рядок користувача)). Цей спосіб роботи підходить, наприклад, для скриптів, які повинні виконуватися регулярно за допомогою різних планувальників задач або для вирішення завдань простої обробки тексту.

І остання область - це створення GUI-додатків (графічних інтерфейсів), що виконуються на стороні клієнта. В принципі це не найкращий спосіб використовувати PHP, особливо для початківців, але якщо ви вже досконально вивчили PHP, то такі можливості мови можуть виявитися вельми корисні. Для застосування PHP в цій області буде потрібно спеціальний інструмент - PHP-GTK, який є розширенням PHP.

Отже, можливості застосування PHP достатньо широкі та різноманітні. Тим не менше існує безліч інших мов програмування, здатних вирішувати схожі задачі. Чому варто вивчати PHP? Що це нам дає? По-перше, PHP дуже простий у вивченні. Достатньо ознайомитися лише з основними правилами синтаксису і принципами його роботи, і можна починати писати власні програми, причому братися за такі завдання, вирішення яких на іншій мові вимагало б серйозної підготовки.

По-друге, PHP підтримується майже на всіх відомих платформах, майже у всіх операційних системах і на найрізноманітніших серверах. Це теж дуже важливо. Навряд чи комусь захочеться переходити, наприклад, від роботи під ОС Windows до роботи під Linux або від сервера IIS до серверу Apache тільки для того, щоб вивчити ще одну мову програмування.

Тема 10. Робота PHP з базами даних

Одним із значних переваг PHP є підтримка широкого кола баз даних. Створення скрипта, що використовує бази даних – дуже просто. В даний час PHP підтримує наступні бази даних:
MySQL, Adabas D, Ingres, Oracle (OCI7 і OCI8), dBase, InterBase, Ovrimos, Empress, FrontBase, PostgreSQL, FilePro (тільки читання), mSQL, Solid, Hyperwave, Direct MS-SQL, Sybase, IBM DB2, Velocis, Informix, ODBC, Unix dbm.

Також в PHP включена підтримка DBX для роботи на абстрактному рівні, так що ви можете працювати з будь-якою базою даних, що використовують DBX. Крім того, PHP підтримує ODBC (Open Database Connection standard), таким чином, ви можете працювати з будь-якою базою даних, що підтримує цей всесвітньо визнаний стандарт. Далі в цьому уроці докладніше про роботу PHP з СУБД.

Отже, для початку відповімо на запитання: що таке MySQL?

MySQLБаза даних MySQL стала найпопулярнішою в світі базою даних з відкритим вихідним кодом - завдяки її високій продуктивності, надійності і легкості використання. Існує більше 6 мільйонів установок цієї бази даних, починаючи від великих корпорацій і до спеціалізованих вбудованих додатків.

MySQL - це одна з найпопулярніших і найпоширеніших СУБД (система управління базами даних) в інтернеті. Вона не призначена для роботи з великими обсягами інформації, але її застосування ідеально для Інтернет сайтів, як невеликих, так і досить великих.

MySQL відрізняться хорошою швидкістю роботи, надійністю, гнучкістю. Робота з нею, як правило, не викликає великих труднощів. Підтримка сервера MySQL автоматично включається в поставку PHP.

Важливим чинником є її безкоштовність. MySQL розповсюджується на умовах загальної ліцензії GNU (GPL, GNU Public License).

На сьогодні СУБД MySQL належить фірмі ORACLE. Детальніше про цю СУБД можна дізнатися на її офіційному сайті www.mysql.com.

Раніше для довготривалого зберігання інформації ми працювали з файлами: поміщали в них деяку кількість рядків, а потім витягували їх для подальшої роботи. Завдання тривалого зберігання інформації дуже часто зустрічається в програмуванні Web-додатків: підрахунок відвідувачів на сторінці, зберігання повідомлень у форумі, віддалене управління змістом інформації на сайті і т.д.

Тим часом, професійні прийоми роботи з файлами дуже трудомісткі: необхідно піклуватися про розміщення в них інформації, про її сортуванні, видобування, при цьому не потрібно забувати, що всі ці дії будуть відбуватися на сервері хост-провайдера, де з дуже великою ймовірністю встановлено один з варіантів Unix - отже, потрібно так само піклуватися про права доступу до файлів і їх розміщення. При цьому обсяг коду значно зростає, і зробити помилку в програмі дуже просто.

Всі ці проблеми вирішує використання бази даних. Бази даних самі дбають про безпеку інформації та її сортування і дозволяють витягувати і розміщувати

інформацію за допомогою одного рядка. Код з використанням бази даних виходить більш компактним, і налагоджувати його набагато легше. Крім того, не потрібно забувати і про швидкість - вибірка інформації з бази даних відбувається значно швидше, ніж з файлів.

Примітка. Додаток на PHP, що використовує для зберігання інформації базу даних (зокрема MySQL) завжди працює швидше додатка, побудованого на файлах. Справа в тому, що бази даних написані на мові C + +, і написати на PHP програму, яка працювала б з жорстким диском ефективніше бази даних - завдання непідсильне за визначенням, оскільки програми на PHP в принципі працюють повільніше, ніж програми на C + +, так як PHP - інтерпретатор, а C + + - компілятор.

Таким чином, основна перевага бази даних полягає в тому, що вона бере на себе всю роботу з жорстким диском і робить це дуже ефективно.

У дистрибутив PHP входить розширення, що містить вбудовані функції для роботи з базою даних MySQL. У цій частині лекції ми познайомимося з деякими основними функціями для роботи з MySQL, що будуть потрібні для розв'язання задач побудови web-інтерфейсів з метою відображення і наповнення бази даних. Виникає питання, навіщо будувати такі інтерфейси? Для того щоб вносити інформацію в базу даних і переглядати її вміст могли люди, не знайомі з мовою запитів SQL. При роботі з web-інтерфейсом для додавання інформації в базу даних людині потрібно просто ввести ці дані в html-форму і відправити їх на сервер, а наш скрипт зробить все інше. А для перегляду вмісту таблиць досить просто клацнути по посиланню і зайти на потрібну сторінку.

Тема 11. Фреймворки для PHP

Створення сучасного web-сайту є доволі таки не тривіальною задачею. Адже прочитавши книгу про PHP та виконавши кілька вправ у вас все одно не буде потрібних навичок для створення готового продукту за який можна буде не соромлячись брати гроші. Але проблема в тому ,що приклади з книжок та різних освітніх сайтів допоможуть лише на початку. Проте згодом кількість коду буде тільки збільшуватись і збільшуватись, наприклад, для отримання даних з форми. Ми повинні:

- Перевірити тип отриманих значень
- Впевнитись, що значення мають потрібний нам формат
- Видалити з тексту заборонені теги
- Замінити службові символи SQL – запитів (захист від SQL Injection)
- Якщо параметри зв’язані між собою перевірити ці ж зв’язки

Певна річ код повинен бути не тільки написаний, але і відповідно протестований. Це ж добрий кусок роботи! І що най головніше рутинної, яку ніхто не любить тай при виконанні якої можна допустити кілька помилок, які можуть вилізти вже на етапі релізу проекту.

Якщо ми пишемо бекенд на PHP, то ми маємо три варіанти реалізації свого проекту:

Використовувати чистий PHP, підключаючи лише потрібні бібліотеки. Цей варіант най праце затратний, проте ви можете добитись найбільшої гнучкості вашого сайту реалізувавши будь-який функціонал при цьому забезпечити максимальну продуктивність.

Використовувати готове рішення. На сьогодні майже на кожен тип задач є своя CMS (Collaboration Management System). Тут можна обійтись без програмування використовуючи стандартні теми, шаблони змінюючи лише контент. Якщо не вистачаю функціоналу можна написати плагін або ж знайти готове рішення. На щастя майже для всіх стандартних задач з якими не може впоратись «гола CMS» існує багато плагінів чи віджетів. Це рішення звичайно хороше ,але продуктивність може дещо впасти в порівнянні з першим варіантом.

Використовувати фреймворк! В принципі фреймворк можна назвати додатковою бібліотекою. Але ось саме тут є суттєві відмінності. Бібліотеку ви використовуєте для розширення функціоналу ,а фреймворк окрім цього ще визначає архітектуру (взаємозв’язки) додатку. В принципі використання фреймворку це щось середнє між 1 та 2 варіантом. З однієї сторони ви серйозно обмежені в свободі дій в порівнянні з першим варіантом ,але ці обмеженні ніщо в порівнянні з тим які ви готові рішення отримаєте.

Тема 12. Створення прогресивних веб-застосунків

Вражаюче, як швидко змінюються технології. Ще вчора все дивувалися стрімкому розвитку нативних мобільних додатків, а сьогодні ми спостерігаємо нову еволюцію.

Ніхто не міг і припустити, що курс розвитку мобільних додатків зміниться так кардинально. Так про що ж мова?

У 2015 році компанія Google заявила про створення нової технології - прогресивних веб-додатках / Progressive Web App (PWA). Однак тоді мало хто сприйняв це нововведення серйозно. Проте сьогодні PWA стає популярним трендом.

З 2005-го веб-розробка перейшла від статичних сайтів до динамічних документів, що створюються серверними (LAMP, ASP.NET) або клієнтськими інструментами, і почав застосовуватись адаптивний веб-дизайн. Незважаючи на перевагу швидкого розгортання, спроби створення веб-застосунків для пристроїв на зразок iPhone 2007-го року, у порівнянні з нативними застосунками були невдалими. Нативні застосунки надавали кращий юзер експіріенс і швидше завантажувались через різницю швидкостей читання з SSD та мережі. Запаковані ресурси та прямий доступ до апаратного забезпечення дозволяли нативним застосункам виконуватись набагато швидше і надавати більше функцій. Проте, з середини 2010-х, постійні вдосконалення в HTML5, CSS3, та JavaScript, значно функціональніші і сумісні зі стандартами браузери, і потужні процесори такі як A10 та Snapdragon 820 зробили гібридні веб-застосунки життєздатною альтернативою.

PWA - це назва групи застосунків, які використовують стек веб-технологій (JS + HTML + CSS) і дозволяють з'єднати простоту використання веб-сайту зі специфічними для нативних застосунків операційної системи UX і технічними можливостями. Основне призначення PWA збільшити конверсію, кількість користувачів і зручність використання веб-застосунків на мобільних пристроях.

Progressive Web Apps є логічним продовженням Accelerated Mobile Pages, таким чином, якщо ви раніше створювали AMP застосунки, то вам однозначно варто оновити свій застосунок до норм PWA застосунків. Якщо ви до цього нічого не чули про AMP, то це не стане для вас проблемою під час вивчення PWA.

Під прогресивністю в даному випадку мається на увазі те, що застосунок з обмеженим функціоналом зможе працювати на будь-якому гаджеті, в будь-якому місці, але при переході користувача на більш досконалий пристрій або браузер, розширюються функції PWA. І при всьому цьому його не треба завантажувати багаторазово на кожен з пристроїв.

Тема 13. Робота з каркасом NodeJS

ackend, back-end (бекенд) — це та «задня» частина сайту, яка невидима відвідувачеві, але яка безпосередньо відповідає за роботу сайту. Якщо порівняти сайт з кораблем, то бекенд — це машинне відділення.

Для простоти можна вважати, що бекенд — це рушій сайту з усіма його зв’язками з механізмами хостинга — базою даних, веб-сервером, поштовими службами.

Бекенд і фронтенд сайту пишуться на різних мовах: для бекенду характерні мови PHP, Python, Ruby, Go, а для фронтенду — JavaScript, HTML, CSS. Тому у розробників сайтів є поділ спеціалізацій — програмісти бекенду і фронтенду.

Проте, з 2009 року частина фронтенд технологій перемістилася до бекенду. Програмісти подумали, чому б не використати на сервері вже знайому мову JavaScript, замість того, щоб вивчати нові технології. Подумали і придумали Node.JS

node.js

Node.js — платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше Javascript застосовувався для обробки даних в браузері на сторонні користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Що ж таке насправді Node.js і чому він став настільки популярним за короткий термін?

Що таке Node.js? Це не веб-сервер. Він сам по собі абсолютно нічого не робить. Це не Apache і не Nginx. Він не містить якихось config-файлів. Найголовніше - це не мова програмування.

Так, з усіма «не» начебто розібралися. Йдемо далі. Давайте нарешті дамо визначення цьому Node.js. Node.js - це середовище для виконання вашого JavaScript коду, це просто ще один спосіб виконувати код на вашому комп'ютері.

Тобто ставиться Node.JS і Node.JS виконує JavaScript на сервері, на робочому столі, на ноутбуці, на мобільних пристроях і так далі. JavaScript де завгодно.

Node.js був створений Райаном Далом в 2009 році.

Що ж таке зробив Райан і чому Node.JS отримав таке велике визнання? Адже насправді до Node.JS існували інші спроби зробити те ж саме. Node.JS створений на основі віртуальної машини V8. Ця віртуальна машина була створена компанією Google для браузера Chrome і вона вміє виконувати JavaScript.

Не просто вміє, вона робить це дуже добре. Вона виконує JavaScript швидко, вона підтримує практично всі можливості сучасного JavaScript, найсучаснішого стандарту який зараз прийнятий, плюс можна за допомогою спеціальних прапорів включити можливості майбутнього стандарту. Крім того V8 дуже економно витрачає пам'ять, дуже добре оптимізована в цьому плані, дозволяє профілювати процесор, пам'ять, дивитися, що відбувається.

Тема 14. Клієнтська оптимізація веб-застосунків

Клієнтська оптимізація - це оптимізація процесу завантаження клієнтським додатком вмісту веб-сторінок. Основна мета такої оптимізації - досягнення максимальної швидкості завантаження сторінок сайту браузером клієнта, адже навіть незначні зміни часу завантаження можуть мати серйозні наслідки для задачі, покладеної на сайт.

При побудові високопродуктивних сайтів повинен бути присутнім і клієнтський, і серверний підхід, вони багато в чому доповнюють один одного. Головна відмінність клієнтського підходу полягає в тому, що в якості об'єкта оптимізації розглядаються сторінки сайту, одержувані браузером клієнта, що складаються з HTML-документа, що містить виклики зовнішніх об'єктів, а також самі зовнішні об'єкти (найчастіше це файли CSS, файли JavaScript і зображення).

Може здатися, що клієнтська оптимізація є лише складовою частиною серверної оптимізації, однак це не так. Різні технологічні рішення клієнтської області сайту при однаковому навантаженні на сервер можуть забезпечувати абсолютно різні характеристики клієнтського швидкодії.

При виключенні з розгляду усіх факторів, що відносяться до серверного програмного забезпечення і каналу передачі даних, можна укласти, що збільшення швидкості завантаження сторінки на різних стадіях завантаження принципово можливо за рахунок обмеженої кількості методів.

Більшість наведених в тут методів оптимізації є універсальними і можуть бути застосовані практично в будь-якому випадку, на будь-якому сайті. Але тільки вибір найбільш підходящого плану оптимізації може привести до найкращого результату при вирішенні кожної конкретної задачі.

Аналіз

Перед оптимізацією сайту необхідний ретельний аналіз його клієнтської продуктивності, а також чітко сформульована мета оптимізації, адже в подібному удосконаленні важливий тільки результат, а не процес.

Процедуру аналізу веб-сайту можна розділити на кілька основних стадій:

- аналіз веб-сторінок та їх компонентів,
- аналіз стадій завантаження веб-сторінок,
- аналіз характеристик браузерів, за допомогою яких веб-сторінки зазвичай завантажуються.

Метою клієнтської оптимізації може бути рішення подібних задач:

досягнення мінімально можливого часу завантаження будь-якої конкретної сторінки;

досягнення мінімально можливого часу завантаження групи сторінок, що переглядаються в довільному порядку;

забезпечення мінімально можливого часу з моменту запиту сторінки до моменту появи у користувача можливості переглядати сторінку і взаємодіяти з нею.

Це далеко не повний перелік можливих цілей. Іноді і зовсім потрібно досягати компромісу і вибирати між кількома взаємо-виключають варіантами оптимізації. У таких ситуаціях краще мати максимум можливої інформації про ваших веб-сайтах і їх відвідувачах.