Практичне завдання.

Тема: Обробка даних html-форм

ТЕОРЕТИЧНІ ВІДОМОСТІ

Форми в html

Для початку необхідно розібратися з тим, що собою являють форми. У цілому – це конструкції вигляду:

```
<formaction="action.php" method="post" name="someform">
Введіть деякий текст: <inputtype="text" name="sometext" value="Деякий
текст"><br>
Введіть пароль: <inputtype="password" name="password" value=""><br>
<inputtype="submit" name="sendform" value="Відправити форму">
</form>
```

Якщо додати такий код у звичайний html-файл, то ви отримаєте форму такого вигляду:

Уведіть деякий текст:

Уведіть пароль:

Відправити форму

Форми– це механізм, розроблений для відправлення даних, уведених користувачем на сервер. Таким чином, форми дозволяють користувачу через браузер взаємодіяти з віддаленим сервером, відправляючи туди дані, створювати динамічний контент чи навіть виконувати якісь нетривіальні та складні задачі (особливо у зв'язку з поширенням технології AJAX, що дозволяє обмінюватися даними із сервером без перевантаження сторінки, а це, у свою чергу, дає можливість розробляти повноцінні програми, аналогічні за властивостями до звичайних, із тією лише різницею, що виконуються вони у браузері), а не лише гортати статичні сторінки. Усі поля, що мають бути відправлені, необхідно розмістити між тегами <*form*></*form*>.

У тегах <*form*> є декілька атрибутів, необхідних для нормальної відправки даних. По-перше, *action* – атрибут, що вказує, на яку сторінку необхідно перейти при передачі даних. Атрибут *method*–метод, яким дані будуть пересилатися на сервер. У цілому існують два методи пересилання даних: *get* та *post*. Перший працює через адресний рядок браузера. Після відправлення даних таким методом адресний рядок набуває вигляду:

name.php?param1=value1&param2=value2&param3=value3.

Кожному полю форми відповідає значення, введене користувачем. Однак такий тип передачі даних має безліч недоліків. По-перше, не може йти мова про конфіденційність, оскільки будь-хто, подивившись на адресний рядок, зможе побачити всі дані, відправлені користувачем.

По-друге, деякі дані не можуть бути передані. Наприклад, так неможливо завантажити файл. Проте сторінка-результат передачі даних може бути за адресним рядком без жодних проблем додана до списку "обраних" посилань користувача, що дозволяє при наступному зверненні швидко повернутися до результатів, отриманих під час попередньої роботи. Наприклад, це необхідно для посторінкового виведення якихось даних, коли адресному рядку вказується як параметр номер сторінки, чи під час пошуку, коли параметри пошуку також можна зберегти і отримати той самий результат пошуку пізніше. Для вирішення цих проблем використовують метод передачі даних *post*.

У такому випадку всі дані, що відправляються, приєднуються до запиту, який відправляють браузером на сервер для завантаження сторінки. Однак, нажаль, втрачається можливість створити в обраному посиланні на результати виконання запиту. Та при додаванні деякої інформації на сервер це не є необхідністю, оскільки при кожному такому запиті знову відправляються дані, тобто пересилання файлів, додавання чи редагування записів виконувались при кожному запиті, що в більшості випадків може нашкодити. Для того щоб передавати дані, під час запиту використовуються елементи *input*. Для кожного такого елемента атрибутом слід зазначити тип (*type*), ім'я (*name*) та значення (*value*). Ім'я має бути унікальним, оскільки воно є необхідним для можливості розпізнавання сервером даних.

Тип може набувати значення:

- *text;*
- *password;*
- *checkbox;*
- *submit;*
- *reset та інш.*

Text – текстове поле, як у прикладі.*Password* – також текстове поле, однак для забезпечення конфіденційності замість символів, що вводяться, відображаються "*". *Submit* формує кнопку відправлення даних на сервер, а *reset* переводить форму у початковий стан. *Checkbox* – прапорець, що може бути виділений. Також існує можливість для завантаження файла, однак завантаження файлів на сервер через форми – це вже інша тема Окрім *input,* існують також *textarea* та *select*, що дозволяють створити область для введення тексту, та випадаючий список відповідно. Для першого елемента все досить просто – це парний тег,

між тегами, що відкривають та закривають, знаходиться текст, що за замовчуванням буде відображатися на сторінці. Для другого ж елемента слід навести приклад:

```
<selectname="someselect">
<optionvalue="true"<Істина></option>
<optionselectedvalue="false"<Брехня></option>
</select>
```

Як бачимо, *select* також парний тег, що містить між частинами, що його закривають та відкривають, список можливих. Кожний парний тег *option* являє собою один із елементів для вибору та містить між тегами, що відкривають та закривають, видиме значення для вибору. Також він може мати атрибут *selected*, що свідчить про те, що цей елемент за замовчуванням є вибраним. Може бути наявним атрибут *value*, який задає значення, що буде передане під час відправлення форми на сервер у випадку, коли обрано цей елемент із випадаючого списку.

Робота з даними, що були відправлені на сервер

Настав час розібратися з тим, яким чином можна працювати з даними, відправленими на сервер у php. Так само, як і під час передачі даних, може бути використаний один із двох методів *post* чи *get*. Php розрізняє дані, передані різними методами. Для отримання даних, відправлених через форму, існують два суперглобальні масиви *$_POST* та *$_GET*. Наявний і третій масив *$_REQUEST*, що не розрізняє методу передачі даних.

```
<formaction="foo.php" method="post">
Им'я: <inputtype="text" name="username"><br>
Email: <inputtype="text" name="email"><br>
<inputtype="submit" name="submit" value="Відправ мене!">
</form>
```

З форми такого вигляду можна отримати дані, звертаючись до елементасуперглобального масиву з індексом, еквівалентним до атрибуту *name* елемента вводу. Наприклад:

```
<?php
echo $_POST['username'];
echo $_REQUEST['username'];
?>
```

ЗАВДАННЯ

1 Реєстрація користувачів

Реалізувати реєстрацію користувачів у системі через стандартні *html-форми з клієнтською валідацією*. При реєстрації обов'язково повинні вводитися *ім'я користувача, пароль та деякі дані стосовно користувача, такі, як дата народження, стать* (можна

також додати свої). Необхідно звернути увагу на те, що для деяких полів, що можуть набувати лише одне з фіксованих значень, слід використовувати випадаючий список. Список зареєстрованих користувачів зберігати у файлі *data.log* у будь-якому зручному вигляді із збереженням реєстраційних даних.

Наприклад,

IP-адреса користувача / логін / пароль / дата народження/ …

```
138.50.78.01#User4#19/Feb/2000#260/…
122.35.0.115#User3#01/Aug/1998#parolyaNet/…
165.3.04.61#user1#19/Apr/2003#123/…
227.0.0.8#user2#10/Nov/2010#SumDU/…
```

2 Авторизація користувачів

Зареєстрований користувач має змогу авторизуватися на сайті. Для цього передбачити форму авторизації та алгоритм перевірки користувачів (зареєстрований чи ні). Вивести повідомлення про результат авторизації.

3. Клієнтська частина

1. Реалізацію здійснювати на односторінковому сайті, створеному за допомогою шаблона ***Twitter Bootstrap 4.4.*** Ознайомитися з документацією Bootstrap (https://www.w3schools.com/bootstrap4/default.asp, https://getbootstrap.com), завантажити дистрибутиви або підключити за допомогою посилань.

Bootstrap — це CSS/HTML-фреймворк для створення сайтів. Іншими словами, це набір інструментів для верстання. Він має ряд переваг, завдяки яким вважається одним з найпопулярніших.
Переваги **Bootstrap**:

- **Швидкість роботи** — завдяки безлічі готових елементів верстка з **Bootstrap** займає значно менше часу;
- **Масштабованість** — додавання нових елементів не порушує загальну структуру документів;
- **Легко налаштовувати** — редагування стилів проводиться шляхом створення нових css-правил, які виконуються замість стандартних. При цьому не потрібно використовувати атрибути типу “! important”;
- **Величезне співтовариство розробників;**
- **Широка сфера застосування** — Bootstrap використовується в створенні тем для практично будь-яких CMS (OpenCart, Prestashop, Magento, Joomla, Bitrix, WordPress і будь-які інші), в тому числі для односторінкових додатків. Особливою популярністю користується Bootstrap для створення одностроріночних або лендингів (landing page).

- **Адаптивність, висока швидкість і оптимізація, стандартизація інтерфейсів** - динамічні макети Bootstrap якісно відображаються на самих різних пристроях без необхідності внесення змін до розмітку.
- **Дизайн** - єдині шаблони і стильове оформлення елементів макета і всіх сторінок на сайті в цілому. Bootstrap-сторінки кросбраузерні і добре відображається у всіх основних браузерах. Регулярне оновлення і доповнення фреймворка найсучаснішими можливостями HTML і CSS вносить деякі обмеження у використанні з IE7 і IE8.
- **Простота і відкритість** - використовувати Bootstrap настільки просто, що з ним справляються навіть веб-розробники початківці, а відкритий вихідний код дозволяє брати участь в розробці, модифікувати під власні потреби або просто користуватися хорошим безкоштовним рішенням.
- **Вірний код** HTML, JavaScript і CSS в Bootstrap продуманий і протестований сотнями розробників зі всього світу.
- **Динамічна мова стилів LESS**, яка розширює можливості CSS: розробники можуть керувати кольорами, створювати вкладені колонки і змінні.

Розвиток Інтернету і вебсайтів обумовлює те, що половина користувачів Інтернету відвідує сайти з мобільних пристроїв і маленьких екранів. Тому, сучасний веб-розробник, блогер і веб-майстер повинен думати про функціонування сайту не лише на десктопах, але і на смартфонах і планшетах з тачскріном.

Концепція чуйного веб-дизайну, втіленого в Bootstrap, вирішує саме цю проблему: сайт однаково «відгукується» і відображає інформацію найбільш повним чином незалежно від типу екрану і розміру пристрою. Зміст і колірна гамма не змінюються, змінюється лише форма і спосіб згрупувати інформаційні та навігаційні блоки сайту найбільш зручним для користувача способом.

4. Захист і шифрування даних користувачів

Розглянути питання створення захисних и шифрованих даних.