Лабораторна робота № 5.

Тема роботи: "Активні елементи в WEB-сторінках. Взаємодія WEB-сторінок з сервером . Частина 1"

Мета роботи: Набуття навичок в створенні та роботі з активними елементами в WEB-сторінках. Дослідження взаємодії сторінок із сервером. Поняття про скрипти.

Теоретичні відомості.

Створення форми

Для функціонування форм необхідні *обробники (handlers')* — спеціальні додатки на Web-сервері, як приймають і обробляють дані з полів форм. Одержавши введену користувачем інформацію, обробник може також повернути відповідь у вигляді сторінки підтвердження. Таким чином, обробники є посередниками між формою і Web-сервером.

Майте на увазі, що коли користувач посилає форму на Web-сайт, інформація що міститься в ній зберігається на сайті (або в іншому визначеному автором сайту місці) і вже не зможе бути змінена.

Самі дані вводяться через форми HTML. Форма є гіпертекстовою сторінкою з одним або декількома полями даних і спеціальною кнопкою для передачі введеної інформації.

Синтаксис

FORM – тег, що визначає форму для заповнення в HTML документі. У одному документі може бути визначено декілька форм для заповнення, але вкладені **FORM** оператори не дозволені. Формат оператора **FORM** виглядає таким чином:

<FORM ACTION="URL" METHOD="POST">...</FORM>

Його атрибути наступні:

ACTION
URL серверу запитів, куди буде відісланий зміст форми після підтвердження. Якщо це поле відсутнє, буде використаний URL поточного документа.

METHOD
HTTP/1.0 метод використовується для пересилання змісту заповненої форми на сервер. Існують наступні методи:

- **GET** - це метод за замовчанням, який дописує до URL вміст заповненої форми.
- **POST –** це метод, коли вміст заповненої форми пересилається не як частина URL, а як вміст тіла запиту.

ENCTYPE
задає тип кодування вмісту заповненої форми. Цей атрибут діє тільки коли використовується метод **POST і** навіть в цьому випадку має тільки одне можливе значення (яке є значенням за замовчанням) - **application/x-www-form-urlencoded.**

Усередині **FORM** оператора може знаходитися все, що завгодно, окрім іншого оператора **FORM.** Згідно специфікації, усередині оператора **FORM** використовуються теги **INPUT, SELECT, і TEXTAREA.**

Тег INPUT

Тег INPUT використовується створення простого елементу введення всередині FORM. Це одиночний тег, і він не має закриваючого тега - тобто він використовується подібно тегу IMG.

Атрибути для тега INPUT наступні:

TYPE повинен бути один з:

- "hidden" : користувачу не пропонується поля для введення, але вміст тега передається при підтвердженні і пересиланні форми. Це значення може бути використане для передачі інформації при взаємодії клієнта і серверу.
- "image" : картинка, по якій ви можете клацнути мишею, що приводить до негайного підтвердження і відсилання форми.
- "text" поле введення тексту, значення за замовчанням
- "password" (поле введення пароля; символи, що вводяться, виглядають як зірочки)
- "checkbox" (кнопка, що приймає положення on (включено) і off (вимкнено))
- "radio" (кнопка, що приймає положення on і off)
- "submit" (кнопка, дія якої зводиться до відсилання зісту заповненої форми на сервер запитів)
- "reset" (кнопка, яка встановлює у всіх інтерфейсних елементах значення за замовчанням)

NAME

символічне ім'я (воно не видиме) для цього поля введення. Це поле повинне бути присутнім для всіх полів введення окрім "submit" і "reset", оскільки воно використовується в рядку запиту при ідентифікації поля введення при відправленні її на сервер після підтвердження форми.

VALUE

для полів введення тексту або пароля, може бути використано для визначення початкового змісту поля. Для checkbox або radio button, VALUE задає значення кнопки, коли вона знаходиться у позначеному стані (непозначені кнопки ігноруються при відсиланні запиту); значення за замовчанням для checkbox або radio button - "on". Для типів "submit" і "reset", VALUE може бути використане для відображення напису на цих кнопках.

CHECKED (значення не потрібне)

визначає, що дана кнопка типу checkbox або radio button позначена за замовчанням; це поле працює тільки для кнопок типу checkbox і radio button.

SIZE

фізичний розмір поля введення в символах; це поле діє тільки для елементів введення тексту або пароля. Якщо розмір не визначений, виставляється 20. Багаторядкові поля введення тексту можуть бути задані за допомогою SIZE = ширина, висота; наприклад SIZE=60,12. Атрибут SIZE не повинен використовуватися для завдання багаторядкових полів введення, оскільки можна скористатися тегом TEXTAREA.

MAXLENGTH

максимальна кількість введених символів, які прийматимуться для введення, визначається тільки для полів введення тексту і пароля (і лише в однорядкових елементах). За умовчанням - необмежена. Мається на увазі, що поля введення повинні мати смугу прокрутки.

Тег SELECT

Всередині <FORM> ... </FORM>, може бути присутнім будь-яка кількість тегів SELECT, вільно перемішаних з іншими HTML елементами (включаючи INPUT і TEXTAREA елементи) і текстом (але не додаткових елементів FORM). Тег SELECT в багатьох графічних клієнтах представляється як список.

На відміну від INPUT, SELECT має відкриваючий і закриваючий теги. Усередині оператора SELECT дозволена тільки послідовність тегів OPTION, за кожним з яких слідує деяка кількість простого тексту (без HTML виразів), наприклад:

```
<SELECT NAME="a-menu">
<OPTION> Перший розділ.
<OPTION> Другий розділ.
</SELECT>
```

Атрибути оператора SELECT наступні:

NAME

символічне ім'я для елементу SELECT. Це поле повинне бути присутнім, оскільки воно використовується при відправленні запиту (аналогічно оператору INPUT).

SIZE

якщо SIZE рівний 1 або якщо цей атрибут ігнорується. Якщо SIZE = 2 або більше, SELECT буде представлений як вікно вибору; значення SIZE тоді визначатиме, скільки елементів списку будуть видимі.

MULTIPLE

якщо присутній (необов’язковий), задає, що SELECT повинен дозволяти множинний вибір із списку. Наявність MULTIPLE примушує SELECT бути представленим як список вибору, незалежно від значення SIZE.

Атрибути OPTION наступні:

- *SELECTED* визначає, що ця опція вибрана за замовчанням. Якщо SELECT дозволяє множинний вибір (за допомогою атрибуту MULTIPLE), як SELECTED можуть бути помічені декілька опцій.

Тег TEXTAREA

Тег TEXTAREA може бути використаний для розташування багаторядкового поля введення з необов'язковим вмістом за умовчанням у формі. Атрибути тега TEXTAREA наступні:

- NAME символічне ім'я поля введення.
- ROWS число рядків в полі введення(висота).

- COLS число стовпців в полі введення (ширина).

TEXTAREA має смуги прокрутки, таким чином може бути введено будь-яку кількість тексту. Елемент TEXTAREA складається як з відкриваючого так і закриваючого тегів. TEXTAREA без змісту за замовчанням виглядає приблизно так:

```
<TEXTAREA NAME="foo" ROWS=4 COLS=40> ... </TEXTAREA>
```

Підтвердження і відсилання форми

Для методу GET

Коли натискається кнопка submit, вміст форми буде доданий до URL в наступній формі:

action?name=value&name=value&name=value

(тут "action" - URL, задане атрибутом ACTION в тезі FORM, або URL поточного документа, якщо атрибут ACTION не був заданий).

Нестандартні символи в прикладах "name" або "value" будуть проігноровані, при цьому маються на увазі також "=" та and "&". Це означає, що включення "=", які розділяють імена і значення (names і values), і включення "&", які розділяють пари name/value, не ігноруються.

Для полів введення тексту і пароля, значенням буде те, що введе користувач. Якщо користувач залишить це поле порожнім, значення value також буде порожнім, але в рядку запиту буде присутній фрагмент "name=".

Для кнопок типу checkbox і radio button, значення value визначається атрибутом VALUE у тому випадку, коли кнопка позначена. Непозначені кнопки при складанні рядка запиту ігноруються вцілому. Декілька кнопок типу checkbox, при необхідності, можуть мати один атрибут NAME (і різні VALUE). Кнопки типу radio button призначені для того, щоб поводитися за принципом "одна зі всіх" і повинні мати однаковий атрибут NAME і різні атрибути VALUE.

Для методу POST

Вміст форми кодується так само, як для методу GET (див. вище), але замість додавання вмісту форми до URL, заданої атрибутом форми ACTION, як запит, вміст посилається блоком даних як частина операції POST. Якщо присутній атрибут ACTION, то значення URL, яке там знаходиться, визначає, куди послати цей блок даних. Цей метод особливо рекомендується при відправлені великих блоків даних.

Хід виконання:

1. Створити на своєму диску **X:** каталог **lr5**.

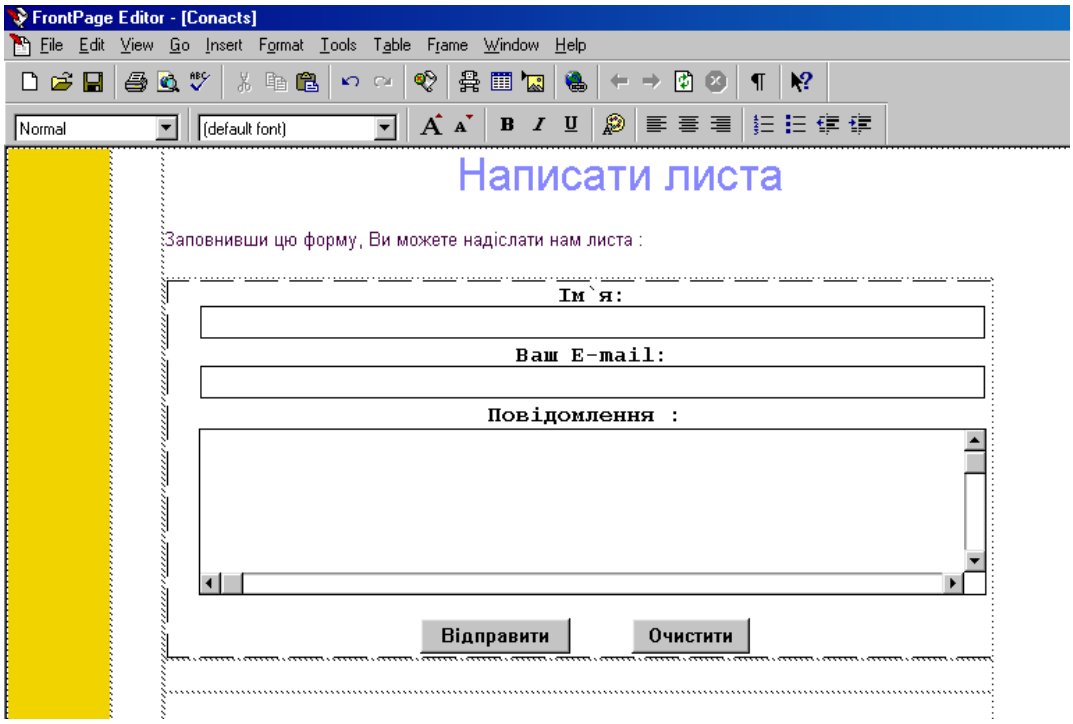
2. **<u>Дослідити роботу скрипту-лічильнику відвідувань.</u>** Для цього :

- Створити в каталозі l**r5** нову сторінку **count.html**.
- В сторінку вставити заголовок по центру “Дослідження лічильника завантажень.” Під ним розмістити текст “Відвідувань : ”. Справа від цього тексту вставити будь-яке зображення. Далі клацнути мишкою по зображенню і перейти на закладку “Source” – джерельний текст WEB-сторінки. Замінити теги зображення сторінки на такий текст **<img src="http://www.local.vsau/cgi-bin/ic.pl?../data/xxxxxx.txt" title="" alt="лічильник завантеження">** Замість **xxxxx** треба вставити свій логин в компп`ютерній мережі ВДАУ. Розуміти цю стрічку треба так: “Зображення цифр лічильника на моїй сторінці буде формувати скрипт **ic.pl** , що

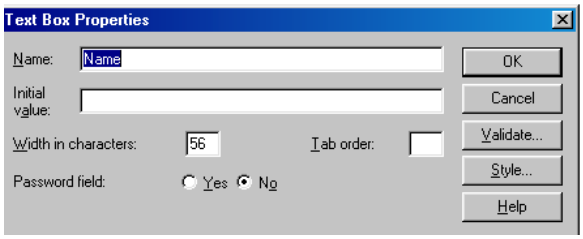
знаходиться на WEB-сервері ВДАУ **http://www.local.vsau** в його каталозі **/cgi-bin** . Після знаку запитання йде команда скрипту, куди йому (**../data**) записувати файл **xxxxxx.txt** , в якому він буде зберігати число завантажень".

- Запустити режим просмотру сторінки. На місті зображення має з`явитись число, що буде свідчити про кількість завантажень цієї сторінки.
- Відкрийте цю сторінку через провідник Windows – при цьому має завантажитись WEB-браузер. Натисніть на WEB-браузері кнопку "Перезавантажити" – "Reload". Занотуйте, як веде себе лічильник. В висновках напишіть пояснення принципу роботи лічильника, користуючись лекційним матеріалом.

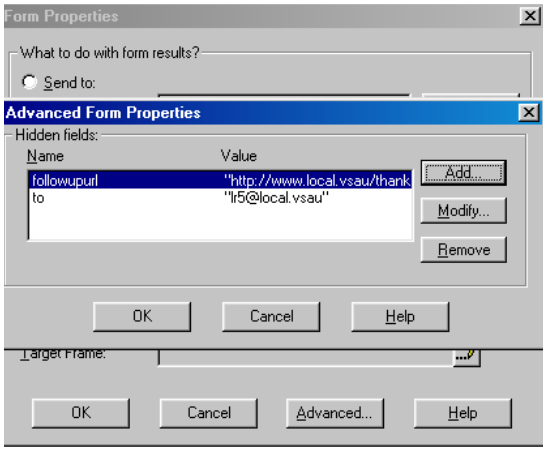
3. **Дослідити роботу простої поштової форми, її активних елементів та простих Java-аплетів.**
4. Для цього створити в каталозі **lr5** нову сторінку з назвою form.html. Створити поштову форму за таким зразком:



- В якості об`єкта, що має на малюнку пунктирну лінію, розмістити з меню FrontPage Insert - Form Field об`єкт – One-Line Text Box. Ви одержите форму з двома кнопками Submit і Reset, а також з об`єктом Text Box (це прямокутник на екрані). Вказати мишкою на ліву границю форми і, користуючись кнопкою Enter, розсунути форму до необхідних розмірів.
- Позначити об`єкт Text Box і скопіювати, утворивши на формі його другий екземпляр.
- Вставити на форму з того ж пункту меню об`єкт Scrolling Text Box.
- Надати одержаній формі вигляду, наведеному на рис. вище.
- Настроїти окремі об`єкти форми. Для цього треба двійчи клацнути по необхідному об`єкту і вікні його властивостей внести відповідні настройки. Отже, для першого Text Box вони будуть такі (В полі Name вказати **Name**)



- Для другого Text Box в полі Name вказати **Email**.
- Для об`єкту Scrolling Text Box в якості імені вказати **Message**.
- Тепер треба настроїти власне саму форму. Для цього обрати її властивості (Form Properties) і натиснути на першій формі настройки кнопку Options. В полі Action написати шлях до скрипту поштової форми http://www.local.vsau/cgi-bin/forms.pl. Метод відсилки встановити **POST**. Далі повернутись на першу форму настройки і натиснути кнопку **Advanced**.
-



Далі, натиснувши кнопку **Add**, внести такі величини:

Name : followupurl
Value : http://www.local.vsau/thanks.htm

Дали повторно її натиснути і додати ще такі :

Name : to
Value : lr5@local.vsau

Тут ми створили два прихованих поля - **followupurl** і **to**. При цьому вони вже «заповнені» відповідною інформацією **http://www.local.vsau/thanks.htm** і **lr5@local.vsau**.

Тепер треба настроїти кнопки форми. Для цього двійчи клацнути по кожній із них і в полі **Value Label** написати **Відправити** (тип кнопки вказати **Submit**), а в другій - написати **Очистити** і тип кнопки вказати **Reset**.
Залишилось лише додати підпрограми на мові Java, які мають **до відсилання даних з форми серверу, проконтролювати,** чи всі поля у листа заповнені. Для цього після тега <body> (натисніть в програмі FrontPage закладку просмотру кода **HTML**) треба додати такий текст:

```
<script language="JavaScript">
<!--
function Form_Validator(theForm)
{
```

```
if (theForm.Name.value == "")
{
    alert("Ви забули ввести своє ім`я в поле \"Ім`я\" ");
    theForm.server.focus();
    return (false);
}

if (theForm.Email.value == "")
{
    alert("Ви забули ввести адресу своєї електронної пошти в полі \"Ваш E-mail\" ");
    theForm.recipient.focus();
    return (false);
}

if (theForm.Message.value == "")
{
    alert("Ви забули написати текст листа");
    theForm.from.focus();
    return (false);
}
}
//-->
</script>
```

а в стрічку
`<form action="http://www.local.vsau/cgi-bin/forms.pl" METHOD="POST" onsubmit="return Form_Validator(this)">` вставити показану жирним шрифтом команду, що буде викликати наші підпрограми.

- Перевірте правильність виконання створеної Вами поштової форми. Для цього збережений файл відкрити через провідник WEB-браузером і натиснути кнопку **Відіслати**. При цьому має з`явитись повідомлення, що не заповнено поле **Ім`я**. Заповніть будь-яке ім`я і знову спробуйте відіслати листа. Має з`явитись повідомлення про друге незаповнене поле – адреса електронної пошти. Аналогічно перевірте і третє поле. Якщо все зроблене правильно і всі поля будуть заповнені, то при відсиланні листа, має завантажитись стрінка з повідомленням про відіслану пошту.
- Перевірте, як працює кнопка **Очистити.**.

І на останок. Що ж таки робить форма, коли натискається кнопка **Відправити**?
Фактично спочатку запускається Java-підпрограма контролю заповненості необхідних полів, а далі інфомація з усіх полів (в тому числі і прихованих!) передається скрипту **forms.pl**, що знаходиться за адресою http://www.local.vsau/cgi-bin/forms.pl**.** Далі скрипт відповідно до одержаної інформації “пише” електронного листа на вказану нами в прихованому полі адресу lr5@local.vsau і завантажує сторінку з подяками. Знову ж таки ту, що ми вказали в іншому прихованому полі **http://www.local.vsau/thanks.htm**.

Зміст звіту про виконання лабораторної роботи:

- Описати, яким чим створюється лічильник відвідувань і що для цього має існувати.
- Описати властивості досліджених елементів поштової форми.
- Навести фрагмент тексту поштової форми, що знаходиться між тегами <form> …</form >. Розібрати цей текст і олівцем написати коментарі до окремих його частин. Розібрати, яка частина цього тексту за що відповідає.

Питання для самоконтролю.

1. Як працює лічильник відвідувань сторінки ?
2. Де відбувається підрахунок кількості завантажень сторінок?
3. Що таке скрипт?
4. Що таке WEB-сервер і які його Ви знаєте функції?
5. Які активні елементи WEB-сторінок Вам відомі?
6. Які властивості мають активні елементи WEB-сторінок?
7. Чим відрізняються елементи Text Box і Scrolling Text Box і що у них спільного?
8. Що означає властивість кнопки Submit? Як вона перекладається ?
9. Що означає властивість кнопки Reset? Як вона перекладається ?
10. Що таке властивість Name? Чим вона відрізняється від властивості Value?
11. Що таке приховані об`єкти в формі? Для чого вони використовуються?
12. Завдяки якій команді об`єкти стають прихованими?
13. Назвіть всі приховані об`єкти (поля) Вашого проекту.
14. Що робить WEB-браузер при натисканні кнопки з властивістю Submit?
15. Хто проводить контроль правильності заповнення поштової форми - WEB-браузер чи WEB-сервер?
16. Назвіть імена скриптів, з якими Ви мали справу у цій лабораторній роботі, а також вкажіть де вони знаходяться?
17. Де чином скрипт тримає інформацію про число завантажень сторінки?
18. Чому всі студенти, користуючись одним скриптом, мають “окремі лічильники”?
19. Які дії виконує досліджуваний Вами поштовий скрипт?
20. “Хто” передає інформацію скрипту про зміст листа та інші його параметри?
21. “Хто” фактично відправляє електронний лист? Де це відбувається?