# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №4**

**«Використання технології AJAX.**

**Опис даних мовою XML.**

**Робота з ПЗ Denwer»**

**з дисципліни «Програмування інтернет застосувань»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняв:

ст. викл.Сенів М.М.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Використання технології AJAX. Опис даних мовою XML. Робота з ПЗ Denwer.

**Мета роботи:** Оволодіти технологією Ajax та методами створення сценаріїв взаємодії з сервером.

## 1. Теоретичні відомості

**AJAX (Asynchronous JavaScript And XML)** — підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX — один з компонентів концепції DHTML.

Про AJAX заговорили після появи в лютому 2005-го року статті Джесі Джеймса Гарретта (Jesse James Garrett) «Новий підхід до веб-застосунків».

AJAX — це не самостійна технологія, а швидше концепція використання декількох суміжних технологій. AJAX підхід до розробки призначених для користувача інтерфейсів комбінує кілька основних методів і прийомів:

Використання DHTML для динамічної зміни змісту сторінки:

Використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю

альтернативний метод — динамічне підвантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON), динамічне створення дочірніх фреймів

Використання цих підходів дозволяє створювати набагато зручніші веб-інтерфейси користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем. AJAX — асинхронний, тому користувач може переглядати далі контент сайту, поки сервер все ще обробляє запит. Браузер не перезавантажує web-сторінку і дані посилаються на сервер без візуального підтвердження (крім випадків, коли ми самі захочемо показати процес з’єднання з сервером). Використання AJAX стало найпопулярніше після того, як компанія Google почала активно використовувати його при створенні своїх сайтів, таких як Gmail, Google Maps і Google Suggest. Створення цих сайтів підтвердило ефективність використання даного підходу.

**Порівняння класичного підходу та AJAX**

Класична модель веб-застосунку:

* Користувач заходить на веб-сторінку і натискає на який-небудь її елемент
* Браузер надсилає запит серверу
* У відповідь сервер генерує повністю нову веб-сторінку і відправляє її браузеру і т. д.

З боку сервера можлива генерація не всієї сторінки наново, а тільки деяких її частин, з подальшою передачею користувачу.

Модель AJAX:

* Користувач заходить на веб-сторінку і натискає на який-небудь її елемент.
* Браузер відправляє відповідний запит на сервер.
* Сервер віддає тільки ту частину документа, яка змінилася.

Поширеною мовою опису даних є XML. Мова XML призначена для опису та зберігання даних ієрархічного типу. Вона використовується для зберігання та обміну форматизованими даними між різними програмними системами:

* дані Web-серверів;
* дані результатів запитів до серверів баз даних;
* дані Web-сервісів;
* новини [RSS](http://en.wikipedia.org/wiki/Rss) блоків.
* мова XHTML є підмножиною XML.

## 2. Хід роботи

1. Створимо новий сайт на віртуальному диску Denwer. Для цього в папці: Z:\home створимо папку нашого сайту: labs.ua. Перезавантажимо Denwer: на робочому столі запустимо ярлик: Restart Demwer.
2. В першу чергу нам потрібно створити на нашому віртуальному хостінгу xml-файл з даними. В програмі notepad++ створюємо новий файл з наступним текстом:

<?xml version="1.0" encoding="windows-1251" ?>

<root>

<data>

Це відповідь з сервера, яка знаходиться в XML-файлі

і була отримана за допомогою JavaScript.

</data>

</root>

Зберігаємо його в каталозі: Z:\home\labs.ua\www під назвою: data.xml.

Тепер створимо веб сторінку під назвою index.html в цьому ж каталозі, з наступним вмістом:

</head>

<body>

<h1>Web-додаток з допомогою технології Ajax</h1>

<p>Дана сторінка демонструє використання

Ajax-технології (Асинхронного Javascript і XML)

для обновлення вмісту сторінки зчитуванням

xml-файла, який знаходиться на сервері.

</p>

<p id="xmlData">

Для отримання відповіді з серверу перейдіть по ссилці:

<a href="data.xml" onclick="ajaxRead('data.xml'); return false">Перегляд data.xml</a>

</p>

</body>

</html>

Запустимо у браузері веб-сторінку: <http://www.labs.ua/> (Результати показано на рис. 2.1).

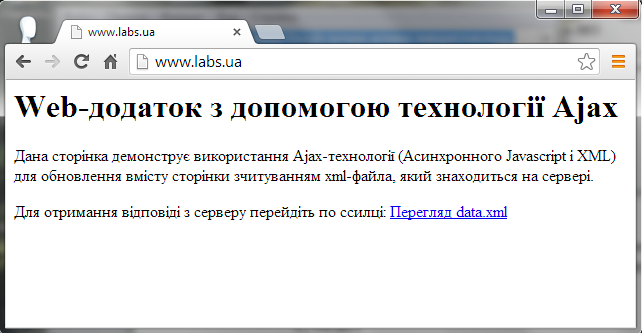


Рис. 2.1. Веб-сайт <http://www.labs.ua/>

Далі напишемо скрипт, який буде відправляти запит та получати дані з xml-файла:

<script type="text/javascript">

<!--

function ajaxRead(file){

var xmlObj;

Далі йде обов’язкова частина скрипту – перевірка доступності методів. Всі браузери описують об’єкт по-своєму. При наявності window.XMLHttpRequest (для Mozilla, Chrome, Netscape і Safari) створюємо екземпляр об’єкта XMLHttpRequest. Якщо цей метод недоступний, перевіряємо наявність window.ActiveXObject (для Internet Explorer 5+) і створюємо екземпляр об’єкта ActiveXObject. Якщо браузер не має жодної можливості, то завершуємо функцію і повертаємося:

if(window.XMLHttpRequest){

xmlObj = new XMLHttpRequest();

} else if(window.ActiveXObject){

xmlObj = new ActiveXObject("Microsoft.XMLHTTP");

} else {

return;

}

Методом open відкриваємо з’єднання з сервером і вказуємо тип передачі даних GET, URL файлу даних (в нашому випадку змінна file, яка була відіслана як параметр функції ajaxRead - data.xml) і асинхронності (по замовчуванню):

xmlObj.open ('GET', file, true);

xmlObj.send ('');

В останньому рядку просто відсилаємо порожній рядок на сервер. Інакше стан об'єкта XMLHttpRequest ніколи не стане рівним 4, тобто не буде "завершено", і сторінку не можна буде відновити. Для POST-запиту відправили б дані.

1. Опрацювання відповіді відбувається наступним чином:

При кожній зміні стану об'єкта *XMLHttpRequest* відбувається подія *onreadystatechange*. Якщо стан змінюється ми використовуємо конструкцію:

xmlObj.onreadystatechange = function(){ : }.

За допомогою *if* переконуємося , що дані доступні , потім запускаємо функцію оновлення . Вона має два параметри: *ID* елемента у веб-сторінці ( оновлюваний елемент в поточній веб-сторінці) і дані для заповнення цього елемента . Оновлюватися буде тег *p* , що має атрибут *id = " xmlData "* . За допомогою властивості *responseXML* оголошуємо , що це *XML*-файл. *xmlObj.responseXML* є об'єктом *DOM*. Цей об'єкт можна інтерпретувати як об'єкт document віддаленого *XML* -файла. Іншими словами , *xmlObj.responseXML* - це об'єкт *document*, якби запускали скрипт в самому файлі *data.xml* . Через метод *getElementsByTagName* шукаємо дані по імені тега. В даному випадку це перший тег з ім'ям *data*. Конструкція:

xmlObj.responseXML.getElementsByTagName('data')[0]

виділяє тільки вузол. Щоб виділити вміст вузла використовується властивість *firstChild.data.*

xmlObj.onreadystatechange = function(){

if(xmlObj.readyState == 4){

updateObj('xmlData',

xmlObj.responseXML.getElementsByTagName('data')[0].firstChild.data);

}

}

1. Функція оновлення *updateObj* за допомогою *getElementById* шукає вузол за

вказаною *ID* і заміняє його вміст. Її перший параметр *obj - ID* оновлюваного об'єкта; другий параметр *data* - нові дані, які повинні бути поміщені в оновлюваний об'єкт *obj.*

function updateObj(obj, data){

document.getElementById(obj). innerHTML = "<b>"+data+"</b>"; }

//-></script>

**В кінцевому результаті документ *index.html* буде мати такий вміст:**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html;charset=windows-1251"?>

</head>

<body>

<script type="text/javascript">

<!--

function ajaxRead(file){

var xmlObj;

//для Mozilla, Chrome, Netscape і Safari

if(window.XMLHttpRequest){

xmlObj = new XMLHttpRequest();

//для Internet Explorer 5+

} else if(window.ActiveXObject){

xmlObj = new ActiveXObject("Microsoft.XMLHTTP");

//нічого не вийде - завершуємо функцію

} else {

return;

}

//Опрацювання відповіді

xmlObj.onreadystatechange = function(){

//pfgbn "завершено",

if(xmlObj.readyState == 4){

updateObj('xmlData',

xmlObj.responseXML.getElementsByTagName('data')[0].firstChild.data);

}

}

//відкриваємо з’єднання з сервером і вказуємо тип передачі даних GET, URL //файлу і асинхронності

xmlObj.open ('GET', file, true);

//відсилаємо порожній рядок на сервер. Інакше стан об'єкта

//XMLHttpRequest ніколи не стане рівним 4

xmlObj.send ('');

}

//за допомогою getElementById шукає вузол за

//вказаною ID і заміняє його вміст

function updateObj(obj, data){

document.getElementById(obj).innerHTML = "<b>" + data + "</b>";

}

//-->

</script>

<h1>Web-додаток з допомогою технології Ajax</h1>

<p>Дана сторінка демонструє використання

Ajax-технології (Асинхронного Javascript і XML)

для обновлення вмісту сторінки зчитуванням

xml-файла, який знаходиться на сервері.

</p>

<p id="xmlData">

Для отримання відповіді з серверу перейдіть по ссилці:

<a href="data.xml" onclick="ajaxRead('data.xml'); return false">Перегляд data.xml</a>

</p>

</body>

</html>

Перейдемо по ссилці «Перегляд data.xml» основного вікна сайту, яке було зображено раніше на рис. 2.1 і отримуємо у відповідь результат у цьому ж вікні (без його перезавантаження), текст xml-файла, що знаходиться на сервері (рис. 2.2).

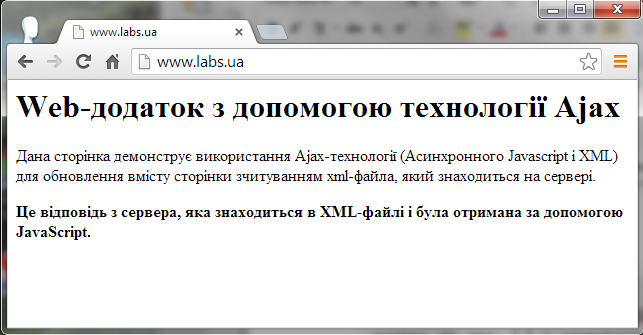


Рис. 2.2. Веб-сторінка <http://www.labs.ua/>

після виконання AJAX-запиту

## ВИСНОВКИ

На даній лабораторній роботі я оволодіти технологією Ajax та методами створення сценаріїв взаємодії з сервером, зокрема відправлення повідомлень на сервер, отримання та відпрацювання відповіді. Також я навчився настроювати та користуватися віртуальним хостінгом за допомогою ПЗ Denwer.