Am ales sa încercăm să facem acest laborator ceva mai dinamic, și cu partea practică mai intensă (deși nu excludem teoria *cu totul*).

Ținta

Scopul nostru este să construim o galerie de fotografii. Pentru început, ne folosim de câteva fotografii existente, dar, în continuarea laboratorului (sau acasă, ca temă opțională) putem încerca să mergem mai departe și să folosim un stream public de pe Flickr ca sursă de fotografii.

Vom folosi PHP, javascript și jQuery, biblioteca PHP GD pentru manipularea de imagini, script-ul Lightbox pentru afișarea de galerii, și apoi AJAX și JSON pentru a optimiza cererile noastre.

Pornim de la un director care are imagini (pentru simplitate, toate imaginile au fost redimensionate la aceleași dimensiuni - în cazul real cel mai probabil acest lucru nu se întâmplă), și de la un template pentru afișarea galeriei.

Despre path-uri, URL-uri și protocoale

Pentru a înțelege discuția din acest laborator trebuie să ne uităm mai atent la cum ne putem referi la fișierele / script-urile / imaginile cu care lucrăm.

Path (în românește, cale) - este calea de pe sistemul curent (în cazul nostru, al server-ului) de fișiere. Există două tipuri de cale (două moduri în care poate fi exprimată calea către un fișier)

- **cale absolută**: descrie calea completă până la fișier: /home/student1/public_html (echivalentul Linux pentru C:\Documents and Settings\student1\public_html)
- **cale relativă**: descrie calea unui fișier relativă la script-ul / css-ul care este interpretat în momentul dat. Adică în loc să spună *fișierul se găsește în folder-ul home, apoi în folder-ul student1, apoi în public_html*, poate spune *fișierul se găsește în directorul css din directorul părinte al scriptului curent*. De ex: ../css/fisier.jpg

URL-ul are mai multe secțiuni (în forma lui generală: scheme://domain:port/path?query_string#fragment_id). Când vrem să apelăm un script din radacina serverului web, de exempu

http://141.85.252.118/phpinfo.php

referința se face către folder-ul rădăcină al server-ului web, cu care acesta este configurat (în cazul nostru, și al celor mai multe servere bazate pe Apache, /var/www/).

Când vrem să apelăm un script folder-ul nostru public_html, spre exemplu:

http://141.85.252.118/~student1/check.php

Apache va identifica formatul ~username si va folosi radacina pentru fiecare utilizator, în cazul acesta /home/student1/public_html/check.php

Atenție! Multe browsere web pot naviga și un director ftp (adică pot deschide URL-uri ftp://141.85.252.118/). Este important de înțeles diferența între cele două protocoale - pentru mai multe, citiți aici.

Prima variantă - pagini generate cu PHP

Directory traversing

Pentru a putea sa afisam imaginile din directorul nostru, în primul rând trebuie să le găsim. Pentru

asta, vom folosi câteva funcții puse la dispoziție de PHP pentru a face ceea ce se numește Directory Traversing.

Folosind funcția opendir și readdir, deschideți folder-ul care conține fotografiile, și tipăriți pe câte o linie numele fotografiilor găsite în acest folder (atenție la . și .., care sunt și ele listate de readdir)

Folosind acest cod, haideți să vedem cum putem să afișăm toate pozele în pagina galerie_task1.php.

Folosiți codul de la punctul anterior pentru a genera tag-uri *img* pentru fiecare dintre imagini. Folosiți clasa *fotografie* pentru fiecare din imagini.

Generarea de thumbnail-uri

Avem acum toate imaginile într-o pagină. Puteți să mutați din imagini din folder-ul nostru pentru a vedea că pagina generată reflectă conținutul folder-ului. Totuși, galeria noastra este ineficientă și nu este tocmai user-friendly - încarcă toate fotografiile - dacă utilizatorul nu este interesat să vadă toate fotografiile? Dacă am avea 100 de fotografii, am vorbi de o pagină generată de câțiva MB!

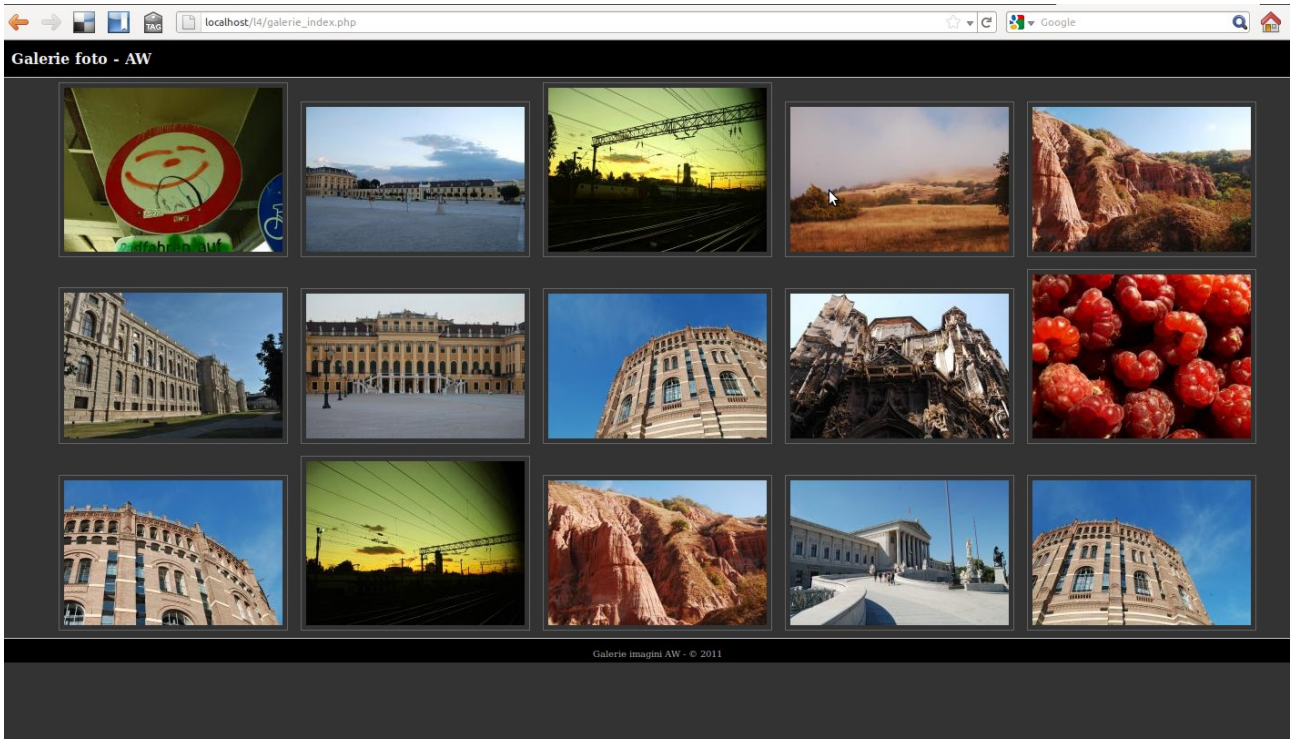
O soluție poate fi să avem o serie de thumbnail-uri - imagini de dimensiuni / calitate mai mică care să aibă link-uri către imaginile in dimensiunea reală.

Pentru a genera thumbnail-uri, un algoritm general este următorul:

1. Verificăm dacă avem un folder special (în general în folder-ul cu imaginile) pentru thumbnails (să îi spuneam .thumbnails - în linux un fișier / folder cu . înainte este considerat hidden) - dacă nu există, trebuie creeat (funcții relevante - file_exists, is_dir, mkdir
2. Pentru fiecare fișier, verificăm dacă are un thumbnail deja generat în folder-ul respectiv (file_exists ne ajută și aici)
3. Dacă există thumbnail-ul, putem trece la următoarea imagine
4. Dacă nu există, trebuie creeat. Pentru a creea un thumbnail nou, folosim biblioteca GD
    1. Creem un obiect de tip imagine pentru tipul de imagine curent (în cazul nostru o imagine jpeg), folsind funcția imagecreatefromjpeg. Avem nevoie de un obiect de tip imagine pentru a putea apoi opera cu el si a creea thumbnail-ul.
    2. Stabilim un nume pentru thumbnail - putem folosi un format care să nu ne oblige să lucrăm cu extensia, spre exemplu thumb_numeoriginal.jpg
    3. În general, avem o arie în care trebuie să ne încadrăm cu thumbnail-ul, dar trebuie să păstrăm raportul între dimensiunile imaginii - deci trebuie să calculăm dimeniunile thumbnail-ului. Vom folosi getimagesize pentru a obține dimensiunile imaginii originale, și vom genera un thumbnail care are 120px lățime și o înălțime calculată cu regula de trei simplă.
    4. Creăm un obiect imagine nou, în care să generăm thumbnailul, folosind noile dimensiuni calculate și funcția imagecreatetruecolor
    5. Copiem conținutul redimensionat (și reeșantionat) imagecopyresampled
    6. Salvăm thumbnail-ul cu numele de la punctul 2, folosind imagejpeg
    7. Opțional, putem să folosim imagedestroy pentru a elibera memoria de cele două obiecte imagine create (în caz contrar, s-ar putea să depășim limita de memorie alocată script-ului nostru).

Scrieți o funcție PHP care să primească ca parametru calea unei imagini și să scrie în folder-ul .thumbnail din directorul cu imaginea thumbnail-ul acesteia, conform algoritmului de mai sus.

Modificați pagina anterioară pentru a afișa thumbnail-urile, nu imaginile la dimensiunea lor originală. Folosiți clasa *thumbnail* pentru tag-ul *img*. Thumbnail-urile trebuie să aibă link-uri spre fișierele imagine originale. Folosiți funcția scrisă la punctul anterior pentru a genera thumbnail-ul în cazul în care el nu există.



Lightbox

Acum, când dăm click pe unul din thumbnail-uri, browserul aduce imaginea originală, și atât. Pentru a avea o galerie cu un UI și UX mai plăcut, putem folosi lightbox.

Integrați lightbox cu galeria noastră curentă.

Navigație și pagina de detalii

Lightbox este excepțional când vrem să prezentăm doar imaginea, eventual și un mic text, dar în general vom vrea să avem o pagină de detalii pentru fiecare fotografie, și să putem să navigăm printre fotografii.

Ca să putem face acest lucru, în primul rând trebuie ca link-urile de pe thumbnail-uri să nu mai activeze lightbox.

Dezactivați lightbox

Apoi, avem nevoie de o pagină de detalii - folosiți galerie_detaliu.php ca start - are deja două butoane pentru navigație.

Utilizarea parametrilor din HttpRequest

Ca să putem afișa o anume fotografie, trebuie să primim, de la pagina cu thumbnail-uri un parametrul - numele imaginii, sau un ID după care să o găsim. Cel mai la îndemână este să exploatăm modul în care se formează o cerere de tip GET pentru asta. Putem să formăm, pentru link-uri, un URL care să arate cam așa:

$url = "http://server/galerie_index.php?nume_fisier=" . $nume_fisier;

Ca să putem să folosim acest parametru în script-ul țintă, putem să căutăm numele în vectorul special $_GET.

```
if (isset($_GET['nume_fisier'])) {
  $nume_fisier = $_GET['nume_fisier'];
} else {
  $nume_fisier = 'default.jpg';
}
echo $nume_fisier;
```

Folosiți transmiterea parametrilor prin cereri GET pentru a transmite script-ului galerie_detaliu.php care imagine să o afișeze. Afișați, desigur, imaginea respectivă în galerie_detaliu.php.

Navigație

Desigur, utilizatorul poate oricând să se întoarcă la prima pagină, la galerie_index.php. Dar, în cazul cel mai comun va fi interesat să nu facă asta, și să se poată mișca cu ușurință între fotografii. Acest lucru ne pune într-o oarecare dificultate, noi neavând o ordine predefintă a imaginilor. Mai mult, parcurgerea unui director costă timp, mai ales dacă directorul este mare.

În mod normal, informații despre imagini ar fi stocate într-un tabel a unei baze de date - vom discuta despre acest aspect într-un alt laborator. Pentru moment, vom improviza o sursă de date într-un fișier - vom parcurge toate fișierele și vom crea un fișier text în același director (.index.txt) unde vom scrie pe fiecare linie un număr de ordine, un caracter de separare, și numele imaginii. Această operație va fi rulată o singură dată.

Creați acest fișier. Modificați galerie_index.php să transmită numărul de ordine al imaginii, nu numele ei (va trebui să citiți într-un array fișierul text). Apoi, la fiecare încărcare a galerie_detaliu.php, citiți acest fișier într-un array și puneți link-urile necesare pe cele două butoane (anterioara / următoarea imagine). Legați logic ultima imagine de prima (astfel încât array-ul nostru poate fi o lista circulară).

AJAX

La fiecare apăsare a butoanelor de navigație, pagina este reîncărcată pentru o nouă poză. Putem rafina acest comportament folosind AJAX. AJAX folosește un obiect special javascript pentru a creea și a trimite o cerere către server, în timp ce pagina încărcată de browser rămâne neschimbată. O cerere AJAX are asociat, de regulă, un *handler* (o funcție care este apelată atunci când se primește răspunsul la cerere). În mod normal, în javascript, handler-ul trebuie să verifice că răspunsul a fost recepționat corect, și că a fost returnat codul 200, care înseamnă că totul a fost în regulă - și apoi să facă ceva util.

Noi vom folosi jQuery pentru a simplifca aceste cereri (în spate, jQuery folosește exact aceeași metodă, dar ne ajută pentru că scăpăm de o parte din problemele administrative). Obiectivul nostru este ca atunci când unul din cele două butoane este apăsat, să obținem URL-ul imaginii cerute, să schimbăm atributul *src* al imaginii, și să schimbăm, desigur, parametrii care vor fi folosiți pe cele două butoane (ca să se refere la poza curentă).

Pentru a putea face asta, avem nevoie să scriem un script php către care să se facă cererea AJAX, și care să calculeze (pe server) și să ne întoarcă, într-un format ușor de procesat, aceste date (src imagine nouă, informație pentru link anterioare, informație pentru link următoare).

În jQuery, puteți folosi .attr() pentru a modifica un atribut al unui tag, .get() pentru a face o cerere AJAX de tipul GET.

Un format ușor de folosit pentru a pasa informație între server și client prin cereri AJAX este formatul JSON. PHP poate genera JSON dintr-un Array prin funcția json_encode. jQuery poate primi un string JSON și să îl citească într-un obiect folosind .parseJSON().

Implementați naviația galeriei cu AJAX.

API-uri externe

Multe din aplicațiile web care aparțin generației Web2.0 pun la dispoziție API-uri (application programming interface) pentru ca utilizatorii să le poată folosi datele, sau capabilitățile și în alte aplicații, nu doar prin interfața definită de aplicația originală a furnizorului. Aceată deschideră deschide posibilități extraordinare pentru aplicații Web - spre exemplu, putem să ne luăm stream-ul de imagini geo-tagg-uite de pe Flickr și să le afișăm thumbnail-urile pe o hartă Google.

Până acum am servit imagini dintr-un folder al nostru local. Ca experiment pentru voi, încercați să obțineți imagini dintr-un stream Flickr pe care să le prezentați cu Lightbox sau cu o navigație implementată cu jQuery, sau o combinație cu PHP.

Pentru a putea face asta, ar trebui să înțelegeți ce este JSONP (care practic este o metodă de a face o cerere AJAX către un domeniu dinafara domeniului de origine al paginii curente (mai multe despre same-origin policy, aici). Puteți citi documentația pentru API-ul Flickr, și puteți să vă uitați și la câteva exemple de aici sau aici. Veți avea nevoie de o cheie API (se poate obține de aici, aveți nevoie de un cont Yahoo!, Facebook sau Google). Puteți folosi imaginile date la laborator ca imagini de test pentru photostream-ul Flickr (să le uploadați acolo).