

DEZVOLTARE WEB CU PHP

Pentru începătorii în programare și în PHP care vor să devină profesioniști

Autor: Flavius Aspra

15 februarie 2012

O inițiativă *Yet Another Project*
Homepage: <http://yet-another-project.github.com/>

Dezvoltare web cu PHP –

Pentru începătorii în programare și în PHP care vor să devină profesioniști

Flavius Aspra

Copyright © 2010-2012, **Flavius Aspra**.

Toate drepturile rezervate lui **Flavius Aspra**.

Nicio parte din această lucrare nu poate fi repusă către descărcare electronică fără acordul autorului.

Redistribuirea sa în format printat este admisă, atâta timp cât distribuitorul nu are niciun fel de câștiguri financiare de pe urma acestei activități sau a altor activități conexe.

Redistribuirea sa, de orice natură ar fi ea, trebuie să se facă integral și fără modificări aduse lucrării.

Flavius Aspra, <flavius@php.net>

URL: <http://flavius.github.com/>

Cuprins

Introducere	4
Scopul acestei cărți	4
De ce am nevoie? Premize	4
Convenții folosite	5
Cum să înveți eficient programare	7
Comunitatea	8
Exercițiile	8
Cum pot ajuta?	9
O privire de ansamblu a capitolelor	9
1 Rețelistică	11
1.1 Noduri, TCP/IP, rețele	11
1.2 Domenii	13
1.3 Protocoale	14
1.3.1 Primul exercițiu de hacking	15
1.4 Instalarea mediului de dezvoltare	18
2 Controlul fluxului de execuție și de date	26
2.1 O altfel de reîmprospătare	26

Introducere

Acest capitol conține informații foarte importante despre cum să studiezi, unde să ceri ajutor, cum să raportezi greșeli și în general cum să profiți la maxim de materialul prezentat. Recomand citirea sa cu atenție.

În ultimii ani, utilizarea Internetului a crescut rapid. Numărul de dispozitive conectate la Internet crește în mod exponențial de la an la an, iar web-ul¹ a devenit scena principală. Web-ul nu mai este static demult, avem rețele sociale,² feed-uri,³ bloguri și microbloguri,⁴ ș.a.m.d. Observăm că mai toate activitățile noastre informaționale s-au mutat pe web.

Însă aceste activități devin din ce în ce mai complexe, la fel ca și aplicațiile⁵ care le susțin. Iar aceste aplicații trebuie dezvoltate de cineva – de programatori.

PHP este unul dintre cele mai folosite limbaje pentru crearea de aplicații web dinamice. Succesul său se datorează în special simplității sale, însă acest lucru e cu două tășuri: pe de o parte este ușor accesibil, pe de cealaltă parte poți face foarte ușor greșeli majore.

Scopul acestei cărți

Pentru a scrie aplicații PHP bune⁶ este cerut simțul critic al programatorului, însă există o mare problemă: începătorul care alege PHP ca primul său limbaj de programare nu are un simț critic dezvoltat sau gândirea analitică necesare în programare. În combinație cu accesibilitatea *aparentă* a limbajului PHP, acest lucru se dovedește fatal pe *termen lung*.

Scopul acestei cărți este să-ți dezvolte gândirea autonomă, productivă, critică, cât și capacitatea de analiză și sinteză, capacități atât de vitale în programare.

Lucrarea de față nu este nici pe departe completă – nici nu vrea să fie. Tot ceea ce vrea este să-ți ofere o fundație bună de start. Acest lucru se face punând accent pe *terminologie* și pe explicarea *modului de funcționare* a noțiunilor și tehnologiilor prezentate.

După cum probabil intuiești deja, scopul acestei cărți este să te susțină să devii bun în perspectivă, fiind orientată spre viitor, pe termen lung.

¹World Wide Web

²hi5, facebook, lastFM, delicious

³RSS, Atom

⁴twitter

⁵Termenul „aplicație” nu este tocmai corect, în sensul tradițional, însă autorul consideră că limba este ceva maleabil, care se schimbă în timp în funcție de nevoi.

⁶performante, sigure, complexe, mentenabile

De ce am nevoie? Premize

Această lucrare pleacă de la premiza că știi deja (X)HTML,⁷ eventual și CSS, dar acesta din urmă nu este necesar pentru înțelegerea lucrurilor prezentate sau pentru învățarea PHP. Ar trebui studiat oricum, căci fără el nu este posibil *design*-ul de *website*-uri aspectuoase. Acolo unde va fi nevoie, vor fi prezentate și noțiunile JavaScript⁸ necesare.

Un lucru important de care ai nevoie este răbdare. Citește cu atenție și încearcă să înțelegi tot, căci informația este comprimată și uneori pare să nu aibă nicio aplicabilitate practică, însă e doar o iluzie – tot ce scrie în acest ghid scurt este important. Nu uita că *mă rezum doar la fundație, la cunoștințele de bază*.

Așa cum spune și coperta acestei cărți, plec de la premiza că *vrei să devii profesionist în PHP*. Dacă *nu* asta este intenția ta, atunci lucrarea de față nu este potrivită pentru tine. În particular, în timp ce urmezi această carte pentru prima oară, nu o face pentru a-ți rezolva problema ta imediată de care tocmai te-ai lovit, ci încearcă să înțelegi noțiunile expuse și să rezolvi exercițiile prezentate. Vei vedea că asta îți salvează mult timp și frustrare pe *termen lung*, și că rezolvarea eventualei probleme imediate de care te-ai lovit este de fapt marginală carierei tale de *programator profesionist*.

Ca un viitor profesionist ce ești, citește cu atenție acest material, și încearcă să înțelegi nu numai conceptele de care te lovești, ci și implicațiile lor. Analizează-le, atât pe el însele, cât și în relație cu celelalte concepte introduse. Cu cât sintetizezi mai mult atunci când întâlnești



ceva nou, cu atât vei ajunge să *jonglezi* cu noțiunile învățate mai rapid, lucru care îți va permite să fii inovativ.

De exemplu, în primul capitol vei învăța despre rețelistică. Întreabă-te pe parcursul întregii cărți ce efecte au limitările HTTP asupra posibilităților sau asupra securității.

Un alt lucru de care ai nevoie este stăpânirea limbii române. Experiențele noastre cu cursanții ne-au învățat că mulți dintre cei ce aspiră a fi programatori nu îndeplinesc această premiză. Unii dintre ei au avut dificultăți nu pentru că nu ar ști să vorbească, ci pentru că nu au realizat intensitatea cu care punem accent pe acest aspect. Nu te teme, prima fază de tutelare are exact acest scop: să te aducă pe linia de plutare.

Și nu în ultimul rând, ai nevoie de stăpânirea relativ bună a limbii engleze. Fără ea oricum nu ai avea succes în programare – de multe ori va trebui să citești documentații în engleză, ba mai bine, să îți documentezi aplicațiile în engleză. Apropo de documentație, toate proiectele de succes au o documentație bună în engleză. Nu te impacienta dacă nu te simți pregătit să scrii ceva în engleză, până la capitolul 3 inclusiv vei avea șansa să îți îmbunătățești engleza doar citind.

Convenții folosite

Pentru a-ți face lectura cât mai plăcută, cartea de față respectă anumite convenții, atât de natură tipografică, cât și inerente comunității din jurul lucrării.


În primul rând, pe prima pagină se află un *link* către pagina de start a proiectului. Această pagină va fi numită mereu *pagina **php**ro*. Următoarea secțiune îți va descrie despre ce este vorba în detaliu.

În al doilea rând, când spun wikipedia, mă refer la versiunea originală în engleză a site-ului <http://en.wikipedia.org/>. În carte nu voi face referire la niciun *site* în română.

„Pagina PHP” și „manualul PHP” se referă la paginile oficiale <http://php.net/> și respectiv <http://www.php.net/manual/en/>. Demn de menționat este că nu voi folosi decât *site*-urile oficiale ale produselor despre care vorbesc.

⁷În ciuda credințelor populare, HTML nu este un limbaj de programare.

⁸Un limbaj de programare pentru programarea clientului, a *browser*-ului, în contrast cu PHP, cu care se programează „serverul”. În ciuda credințelor din folclor, JavaScript și Java sunt limbaje complet diferite și de sine stătătoare.

 Urmarea link-urilor, în special cele către wikipedia și către manualul PHP, *nu* este opțională. Informațiile prezentate în acele pagini *fac parte* din cartea de față.

Cuvintele importante sunt scrise *cursiv*, termenii și noțiunile importante sunt scrise *înclinat*, iar cuvintele care fac referire la nume de funcții, comenzi sau instrucțiuni care trebuie introduse într-un fișier sau ca comandă, sau cuvinte cheie specifice unui anumit limbaj sunt scrise cu font neproportional⁹.

Apăsările de taste sunt scrise în chenare, astfel `ENTER` înseamnă să apeși enter o dată, iar `CTRL+F` înseamnă să apeși tasta `CTRL` și în timp ce o ții apăsată, să apeși `F`.

Codul sursă, de obicei în PHP, va arăta în felul următor:

Listing 1: Convenție listare

```
1 <?php
2 phpinfo () ;
```

Numerele de pe marginea stângă reprezintă numerele liniilor de cod corespunzătoare și nu trebuie scrise. Ele deserveșc unei mai ușoare identificări în explicațiile din text.


Codul sursă va conține și caracterul '. Acesta trebuie văzute ca apostrof. Altfel spus, copierea și lipirea codului sursă prezentat nu va funcționa pur și simplu. Acesta trebuie scris de tine însuși, deoarece caracterul respectiv este (intenționat) greșit. Procedez astfel pentru a împiedica copierea codului sursă. Acesta trebuie înțeles.


Atunci când introduc termeni noi, încerc să ofer și traducerea în engleză, în paranteză. Exemplu:


Un astfel de atac se numește *man in the middle*, deoarece atacatorul se află la mijloc, între cele două capete (en. *endpoints*).

iar la sfârșitul cărții poți găsi o referință a tuturor acestor termeni.

Există trei feluri în care marchez paragrafe:

 Paragrafele care te atenționează asupra unui lucru important sunt marcate ca atare, ca cel de față.

 În unele locuri vreau să-ți atrag atenția asupra unei practici de programare bune.

 În același timp, câteodată îți atrag atenția asupra unor lucruri care, deși sunt posibile, nu ar trebui făcute.

Exercițiile sunt marcate cu un creion, iar numărul de steluțe reprezintă dificultatea lor, între 0 (nicio steluță) și 3 (trei steluțe). Exemplu:



Exercițiu de dificultate 1*

Eu sunt un enunț.

Exercițiile de dificultate zero necesită doar înțelegerea exemplelor și explicațiilor imediat anterioare și mici modificări sau adăugiri. Cu cât solicitarea inteligenței și a capacității de sinteză a cursantului crește, cu atât crește și numărul steluțelor.

La evaluarea dificultății exercițiilor procedez în felul următor: în primul rând, plec de la premiza că cursantul știe toate noțiunile din capitolul anterior, chiar dacă nu le-a sintetizat pe toate. În același timp, plec de la premiza că restul capitolelor trecute au fost bine sintetizate.

⁹ *monospace*

❧ Dacă trebuie să sari cu mai mult de un capitol înapoi pentru a revizui ceva, atunci este un indiciu că nu ai trecut prin toate stadiile de studiu în mod consecvent. Secțiunea următoare îți va explica care sunt aceste stadii.

Dacă observi încălcări ale acestor convenții, te rog să le raportezi pe pagina de greșeli a [phppro](#).

Cum să înveți eficient programare

Momentan cartea de față nu acoperă încă materia așa cum aș vrea – nu este completă. Însă subiectele abordate sunt acoperite complet, cel puțin la nivel conceptual.

Cartea în sine nu este gândită pentru a fi folosită singură, ci în paralel cu comunitatea [phppro](#). În particular, unele exerciții chiar nu sunt gândite pentru a fi rezolvate de cititor singur, ci cu susținerea tutorilor de pe [phppro](#).

Pe [phppro](#) găsești și ajutor sub formă de idei și indicii pentru rezolvarea exercițiilor.

Învăță *terminologia*, înțelege-o și folosește-o. Dacă setul de scule de programare folosite este lancea ta de programator, atunci terminologia este vârful lancei. Care este diferența dintre un toiag tocit, și o lance fără vârf? Exact, nici una. Nu te apuca să folosești termeni pe care nu-i înțelegi, ci documentează-te înainte. Cu o lance ascuțită:

- te vei putea înțelege mai ușor cu alți programatori; tu îi vei înțelege pe ei, și ei pe tine
- pe măsură ce termenii înțelegi de tine devin mai complecși, vei putea acumula cunoștințe din ce în ce mai complexe bazate pe cele anterioare, în ritm exponențial. La început ți se va pare frustrant, însă dacă vrei să devii bun, oricum va trebui să înveți termenii odată și-odată. Deci de ce să nu faci totul ca la carte de la bun început?
- un programator profesionist știe mai mult de un singur limbaj de programare; ai fi uimit dacă ai afla câți termeni și câte concepte sunt comune multor limbaje. Dacă știi terminologia, chiar dacă ai învățat-o în (cu) PHP, vei putea trece la un nou limbaj cu mult mai puține eforturi. Primul limbaj (învățat corect) este cel mai greu, apoi ți se va pare floare la ureche

Urmează [link](#)-urile în timp ce studiezi; această carte nu este și nu va fi niciodată „completă” – se pleacă de la premiza că citești și înțelegi ce se află la acele link-uri *înainte* de a trece mai departe.

Notele de subsol sunt importante; dacă acestea introduc termeni neexplicați anterior sau în imediata vecinătate, atunci trebuie reținute și făcute legături atunci când termenii respectivi sunt introduși pentru prima oară.

Plec de la premiza că cititorul meu are un anumit nivel de inteligență. Asta nu înseamnă că nu iau în serios orice nelămurire. Însă mă aștept ca noțiunile prezentate să fie citite cel puțin, și apoi înțelese. Nu are rost să citești o carte dacă ... nu o citești cu trup și suflet. Atunci când ai o problemă, ia-o gradual, netrecând la următorul stadiu până nu îl îndeplinești pe cel anterior.

❧ Stadiile¹⁰ sunt: citire, înțelegere, sinteză, imaginație (jonglarea cu noțiunile), inovație.

A sintetiza înseamnă a face legături cu toate celelalte noțiuni deja învățate. De exemplu vei învăța ce înseamnă un *array*, iar peste câteva capitole vei face cunoștință cu obiecte. Dacă vei sintetiza cum trebuie, îți vei da seama singur că este foarte posibil să ai un *array* de obiecte.

A jongla cu noțiunile are ca efect practic faptul că cititorul știe să pună în practică și să combine lucrurile învățate de ca și cum acele noțiuni ar fi fost inventate de el.


❧ Îți poți ușura procesul de sinteză asimilând terminologia încă din momentul introducerii ei.

¹⁰Noțiunea de *stadiu de învățare* este extinderea autorului a sistemului japonez shu-ha-ri (en. *retain-detach-transcend*, jp.). Detalii pe <http://www.makigami.info/cms/japanese-learning-system-japan-36>.


Această sinteză e foarte importantă, și de fapt, o faci de când erai copil. De exemplu, ai văzut-o pe mama ta tăind legumele cu cuțitul. Mai târziu, la joacă, ai avut nevoie să tai o ață, și nu aveai decât un cuțit în apropiere. Ți-ai dat seama că poți tăia ața cu cuțitul, deși nu este o legumă. Altfel spus, ai sintetizat scopul uneltei „cuțit”: să taie ceva.

Lucrarea de față explică foarte bine noțiunile, de la zero, însă sinteza îți este lăsată ție. Motivația mea de a proceda așa este următoarea: după cum sugerează subtitlul cărții – *Pentru începătorii în programare și în PHP care vor să devină profesioniști* – scopul meu e să te îndrum pe calea profesionalismului. Pe de cealaltă parte, sunt un darwinist convins, și dacă nu reușești nici să devii profesionist, nici să vezi utilitatea acestei cărți, atunci e mai bine așa. Ultimul lucru pe care îl vreau este să te susțin să devii ceva în care nu ai avea succes.

Capacitatea de sinteză pe care o vei fi având la sfârșitul cărții mai are încă un efect pozitiv asupra viitorului profesionist din tine: în programare, vei fi confruntat cu nevoia de a reutiliza codul pe care-l scrii, astfel încât să nu fii nevoit să rescrii același cod iar și iar, doar pentru că trebuie să-l personalizezi puțin. Însă pentru a putea face codul atât de flexibil încât să-l poți adapta cu ușurință, trebuie să prevezi cazuri „imprevizibile”; altfel spus, să te gândești la imposibil.

 Nu copia pur și simplu exemplele din carte, pentru că riști să te trezești la un moment dat că nu ești în stare să scrii ceva de unul singur. În schimb citește cu atenție codul și explicațiile de dinaintea și după el, apoi *închide cartea* și scrie totul din minte, argumentându-ți (pe baza explicațiilor pe care le-ai citit) de ce faci un lucru într-un anumit fel, sau de ce îl faci de fapt.

Știu că este mai ușor să copiezi, dar vor veni vremuri când va trebui să inventezi singur un script. Deci obișnuiește-te de pe acum să scrii singur, și de ce nu, să faci greșeli. Atunci când faci o greșală și PHP îți spune asta, citește cu atenție mesajul de eroare, apoi corectează-ți codul, și ține minte pentru fiecare fel de greșală ce eroare generează, pentru ca în viitor să poți identifica mai rapid greșelile pe care le faci pe baza mesajelor de eroare pe care ți le arată PHP.

 Această *putere de imaginație*, în combinație cu *capacitatea ta de analiză și sinteză*, și pe o fundație solidă a *înțelegerii conceptelor și termenilor* cu care intri în contact, sunt cheia succesului garantat.

Comunitatea

Dezvoltare web cu PHP – Pentru începătorii în programare și în PHP care vor să devină profesioniști nu este pur și simplu o carte, ci o comunitate și o serie de servicii pe care această comunitate le oferă. Cartea de față constituie doar scheletul, fundația studiului. Pentru a beneficia deci de aceste servicii, cititorul cărții trebuie să fie și cursant în cadrul comunității.

Pagina [phpro](#) este pagina de start a comunității. Printre serviciile oferite se numără:

- verificarea soluțiilor exercițiilor și oferirea de indicii acolo unde cursantul s-a blocat, individual, pentru fiecare cursant în parte, exact acolo unde are nevoie
- clarificarea nelămuririlor pe care cursantul le are în urma citirii explicațiilor
- articole care întregesc conceptele prezentate în carte; excursuri
- garanția că cursanții¹¹ au într-adevăr potențialul de a deveni profesioniști
- servicii care sunt folosite în viața reală a unui programator

Comunitatea [phpro](#) nu este un loc unde poți primi ajutor la problemele de care te-ai lovit pe cont propriu. Altfel spus, comunitatea noastră este strict una de studiu.

¹¹În special cei care au reușit să ofere soluții la primele trei exerciții din capitolul 2, eventual cu susținerea tutorilor

Exercițiile

Exercițiile sunt parte integrantă a studiului. Scopul exercițiilor nu este numai de a te testa, ci și de a te învăța lucruri noi. De fapt, unele exerciții au menirea exclusivă de a te învăța ceva.

Indiferent de menirea fiecărui exercițiu, poți apela la comunitatea [phpro](#) pentru susținere, sfaturi și indicii la exerciții. În fapt, chiar va trebui să o faci la unele exerciții – vei avea nevoie de asta.

Desprinzându-te de comunitatea [phpro](#), riști să studiezi ceva de unul singur și să ai impresia că ai înțeles totul corect, însă lucrurile învățate se pot așterne greșit în mintea ta, și la un moment dat te vei lovi tu însuși de probleme din cauza asta.

Având însă permanent, la fiecare exercițiu, un tutore lângă tine care te îndrumă, șansele ca un concept de programare să fie înțeles și aplicat greșit scad considerabil.

Unele exerciții vor fi direct legate de comunitate și de serviciile pe care aceasta le oferă. În capitolul patru de exemplu, exercițiile îți vor cere să formezi echipe cu alți cursanți, și să concurezi împotriva altor echipe, folosind scule de programare așa cum sunt folosite în viața reală a unui programator, precum un *bug tracker* sau un *revision control system*.

Însă pentru a primi acces la aceste servicii pe care comunitatea [phpro](#) le oferă gratis, trebuie să rezolvi toate exercițiile anterioare sub tutela comunității, dovedind astfel că ai potențialul unui programator bun.

Cum pot ajuta?

Atât programatorii experimentați, cât și începătorii, pot ajuta, iar ajutorul lor este apreciat în egală măsură.

Punctul de întâlnire pentru toți este [phpro](#), unde poți găsi îndrumare despre ce poți face, sau unde poți raporta ce ai de raportat.

De la cititorii avansați mă aștept la critică constructivă, sfaturi sau idei. *Feedback*-ul mă bucură, însă vreau să atrag atenția asupra unui lucru: există situații în care, atunci când trebuie să explici ceva, trebuie să faci compromisuri între corectitudinea tehnică și ușurința cu care noțiunile pot fi acumulate de cititor (en. *the learning curve*), iar cu compromisurile suntem obișnuiți din programare. Așa se face că pe alocuri ofer explicații nu tocmai corecte, care sunt corectate apoi. Asigură-te că ai citit tot conținutul relevant (și mai ales notele de subsol) înainte de a raporta o greșeală – cel mai probabil explicațiile sau definițiile sunt reluate și șlefuite undeva.

În privința calității cărții, există trei mari probleme:

- Nu cred în cacofonii. Consider că propria imaginație e singura vinovată dacă „vezi” alte lucruri când citești. Ca atare, refuz să le corectez. Ba mai rău, corectarea lor prin folosirea virgulei sau reformulări mai mult ar îngreuna inteligibilitatea.
- Folosesc xenisme. Resursele în limba engleză sunt cele mai acurate și cele mai actuale, din acest motiv nu încerc să evit folosirea lor. Asta îți va permite, pe *termen lung*, să te poți ajuta singur. Pentru a articula un xenism pun cratimă, și apoi particula specifică. De exemplu *web-ul*; însă: *Internetul* – deoarece cuvântul internet există în limba română.
- Este foarte posibil să întâlnești formulări ciudate, cu ordinea cuvintelor inversată, și altfel de greșeli similare. Cauza acestui lucru este că 90% din timp vorbesc germana, lucru care-și lasă amprenta. Corecturile sunt binevenite.

O privire de ansamblu a capitolelor

1. Rețelistică – noțiunile de rețelistică sunt necesare pentru a înțelege mai ușor apoi lucruri legate de securitate, optimizare sau servicii web

2. Controlul fluxului de execuție și de date – te învață constructele pentru controlul informațiilor în cadrul aplicației
3. Reutilizarea și modularizarea codului – împărțirea codului în funcții și fișiere, separarea logicii aplicației de vizualizare
4. Baze de date și lucrul în echipă – cum să lucrezi în echipă, de la anumite reguli de comunicare, până la mailing lists și git; documentarea proiectului; debugging și profiling pentru a lua cele mai bune decizii; database design până la și inclusiv 3rd normal form
5. Securitatea aplicațiilor web – XSS, sql injection, CSRF
6. Programare orientată pe obiecte – OOP, concepte generale, câteva patterns (helper, strategy, factory, singleton), test-driven development, SPL
7. ajax, json, servicii (REST, SOAP, XML-RPC), XML, PDO și alte delicii, și ca „ultima frontieră”: php internals.

Începând cu capitolul 4 proiectele se vor realiza în echipe, iar tutorii vor avea doar rolul de consilieri. Proiectele rezultate astfel vor aparține respectivilor programatori, și vor putea fi folosite pentru portofoliile cursanților.

Rețelistică

Cu PHP controlezi ce se întâmplă pe server,¹ server care este conectat la Internet. Deci folosirea PHP este strâns legată de rețea. Din acest motiv, înțelegerea noțiunilor fundamentale de rețelistică este indispensabilă.² Nu voi intra în toate detaliile, doar în ce ai nevoie ca programator și pentru a-ți seta mediul de programare în PHP, pentru a crea website-uri sigure, și pentru a înțelege noțiunile necesare în folosirea extensiilor PHP care lucrează cu rețeaua.

Noduri, TCP/IP, rețele

Rețelistica (en. *networking*) se ocupă cu comunicarea între așa-zise noduri (en. *nodes*). Un nod poate fi un calculator personal, un router, un server, practic orice dispozitiv digital conectat la rețea.

Înainte de a vedea printr-un exemplu practic prin ce noduri din rețea trec informațiile atunci când cauți ceva pe web cu google, trebuie să faci cunoștință cu câteva definiții generale.

Există două mari tipuri de interfețe pentru interacțiunea cu un sistem de operare (en. *operating system*)³:

- **GUI** (en. *graphical user interface*)

Este cea mai comună printre utilizatorii ocazionali, este ușor de utilizat și nu necesită cunoștințe avansate de IT pentru a o folosi. Însă te limitează în folosirea programelor

- **CLI** (en. *command line interface*)

Este destinată profesioniștilor. Necesită cunoștințe mai avansate, însă este mult mai puternică și flexibilă

Sub MS Windows, interfața CLI este cunoscută și ca *MS-DOS Prompt*. Pentru a-l lansa în execuție, du-te în Start -> Run, și tastează cmd. Apare o fereastră neagră și neprietenoasă. Sub GNU/Linux aceasta este cunoscută sub numele de *consolă*, *terminal* sau *shell* (unul dintre ele fiind *bash*).

Deschide interfața CLI a sistemului tău și introdu comanda `tracert google.com` pentru Windows sau `traceroute google.com` pentru GNU/Linux. Figura 1.1 îți dă un exemplu de *output*.

După cum vezi, tot ceea ce îi trimit eu lui google.com și ce îmi trimite el înapoi trece prin alte 11 noduri din rețea. Dacă un infractor (en. *cracker*) deține controlul asupra unuia dintre aceste noduri,

¹În acest capitol vei vedea că termenul „server“ este deseori folosit greșit, ca și în acest paragraf.

²De exemplu pentru înțelegerea implicațiilor de securitate, folosirea extensiilor precum *cURL* sau cereri asincrone – *AJAX*. Nu îți face griji dacă nu înțelegi toți acești termeni deocamdată.

³abv. OS

```

[flav@evolution ~]$ traceroute google.com
traceroute: Warning: google.com has multiple addresses; using 74.125.87.104
traceroute to google.com (74.125.87.104), 30 hops max, 52 byte packets
 1 192.168.2.1 (192.168.2.1)  2.335 ms  2.294 ms  2.419 ms
 2  .surfer.at (212.17.116.17)  11.060 ms  11.924 ms  8.802 ms
 3  at-graz-pe-r17a-ge-0-1.upc.at (212.17.116.17)  18.417 ms  8.693 ms  9.504 ms
 4  212.17.99.105 (212.17.99.105)  20.566 ms  23.196 ms  19.794 ms
 5  at-vie01a-rd1-ae-0-945.aorta.net (213.46.173.109)  18.772 ms  19.733 ms  37.257 ms
 6  de-fra03a-rd1-xe-3-3-0.aorta.net (213.46.160.253)  32.951 ms de-fra03a-rd1-xe-9-2-0.aorta.net (
213.46.160.122)  34.155 ms de-fra03a-rd1-xe-3-3-0.aorta.net (213.46.160.253)  32.797 ms
 7  72.14.219.9 (72.14.219.9)  76.874 ms  35.361 ms  36.016 ms
 8  209.85.254.108 (209.85.254.108)  37.675 ms 209.85.248.12 (209.85.248.12)  123.352 ms 209.85.254
.108 (209.85.254.108)  37.697 ms
 9  72.14.236.250 (72.14.236.250)  36.230 ms 72.14.232.103 (72.14.232.103)  75.219 ms 72.14.236.250
(72.14.236.250)  37.268 ms
10  209.85.248.41 (209.85.248.41)  37.304 ms 209.85.248.47 (209.85.248.47)  34.590 ms 209.85.248.39
(209.85.248.39)  36.018 ms
11  72.14.232.221 (72.14.232.221)  35.400 ms 72.14.238.101 (72.14.238.101)  35.990 ms 72.14.232.221
(72.14.232.221)  55.344 ms
12  hb-in-fl04.le100.net (74.125.87.104)  38.132 ms  41.000 ms  37.746 ms
[flav@evolution ~]$

```

Figura 1.1: Ruta către un nod

poate vedea absolut orice „vorbesc” eu cu google.com. Un astfel de atac se numește **man in the middle**, deoarece atacatorul se află la mijloc, între cele două capete (en. *endpoints*).

Bineînțeles că nu numai eu aș fi victima unui astfel de atacator, ci oricine trimite pachete de date (en. *data packets*) oricui altcuiva (nu neapărat numai lui google.com), atâta timp cât pachetele de date trec prin nodul controlat de el.

Toate nodurile din rețeaua globală numită Internet au o adresă unică numită adresă IP (en. *internet protocol*). Ea este de forma xxx.xxx.xxx.xxx, unde xxx este un număr între 0 și 255. Unele adrese IP au o semnificație specială. De exemplu, fiecare nod are adresa IP 127.0.0.1. Ea nu este utilă în Internet, deoarece orice pachet de date trimis la acea adresă ajunge înapoi la expeditor, fără nici măcar a trece pragul sistemului local. Încearcă de exemplu comanda `tracert 127.0.0.1`.

Alt bloc de adrese IP speciale sunt toate adresele între 192.168.0.0 și 192.168.255.255. Ele sunt rezervate rețelelor locale (en. *local area network*).⁴

Un astfel de exemplu este chiar rețeaua mea locală⁵. După cum vezi în figura 1.1, pachetele de date care pleacă de la mine trec mai întâi prin nodul 192.168.2.1. Aceasta este adresa *router-ului*⁶ meu. *Topologia* rețelei din perspectiva mea arată ca în figura 1.2.

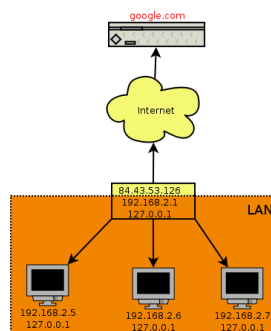


Figura 1.2: Exemplu de topologie a unui LAN

Misiunea unui *router* este să distribuie pachetele pe care le primește altor calculatoare din LAN-ul meu. Asta îmi permite să am acces la Internet cu mai multe calculatoare, însă din exterior LAN-ul este văzut ca având o singură adresă IP: 84.43.53.126, și deci ca un singur calculator. Dacă nu puneam acest *router* între calculatoarele mele și Internet, nu puteam conecta la Internet decât un singur calculator, care avea adresa IP 84.43.53.126.


Există deci două tipuri de adrese IP, cele interne rețelei mele, precum 192.168.2.1 sau 192.168.2.6 și

⁴abv. LAN

⁵Un bun punct de start este http://en.wikipedia.org/wiki/Private_network. De acolo poți urma link-uri interesante către noțiunile ce urmează a fi introduse, și multe alte lucruri neprezentate aici.

⁶<http://en.wikipedia.org/wiki/Router>

cele externe rețelei, publice, precum 84.43.53.126 sau una dintre adresele lui google.com, 74.125.87.104. Cele dintâi se numesc *adrese private*⁷ sau adrese LAN, cele din urmă sunt în schimb *adrese publice*.

 Internetul este o rețea de LAN-uri și WAN-uri (en. *wide area network*) interconectate. Asta transformă Internetul în cea mai mare WAN, o rețea globală.

Dacă nu ai un *router* la mijloc, între tine și Internet, atunci îți poți vedea adresa IP externă prin interfața pusă la dispoziție de sistemul tău de operare. Sub windows, introdu comanda `ipconfig /all` în CLI, sub GNU/Linux `ifconfig -a`.

Dacă însă ai un router la mijloc, și nu știi cum să-i accesezi interfața de configurare, atunci trebuie să îi ceri unui alt calculator de pe Internet să-ți spună ce adresă IP vede el că ai, pentru că ce vede el va fi mereu adresa ta Internet, externă.



What is my IP Address?*

Caută pe web cu un motor de căutare *what is my ip address* și compară adresa afișată de acea aplicație web cu ce adresă IP îți raportează OS-ul tău. Ce poți deduce din această comparație, legat de topologia rețelei tale?

Domenii

Teoretic, este suficient să știi adresa IP a unui nod pentru a comunica cu el. De exemplu, am aflat că una dintre adresele IP ale lui google.com este 74.125.87.104. Deci putem intra bine mersi pe adresa `http://74.125.87.104`.

Însă oamenii nu pot reține ușor numere. De aceea s-a inventat DNS (en. *domain name system*), un sistem care convertește nume precum google.com, în adrese IP. Atunci când introduci google.com în browserul tău, acesta trimite o cerere la serverul DNS și primește înapoi adresele IP asociate cu google.com. Abia apoi se conectează la acea adresă IP, și îi cere informații.

Sub MS Windows îți poți afla serverele DNS cu aceeași comandă `ipconfig` și parametrul `/all`. Sub GNU/Linux, acestea se află în mod normal în fișierul `/etc/resolv.conf`.

Să presupunem că serverele tale DNS se deconectează de la rețea.⁸ Atunci nu vei mai putea accesa `http://google.com`, pentru că browserul tău nu mai poate afla adresa IP corespunzătoare. Însă asta nu te împiedică să introduci manual adresa IP, dacă o știi.

Pentru a afla dacă un nod este conectat la rețea se folosește comanda `ping`⁹, având primul parametru adresa IP pe care vrei să o verifici.



Ping your DNS server

Află adresa IP a serverului tău DNS și trimite-i un pachet de date cu `ping`.¹⁰

Poți interoga manual serverul DNS cu comanda `nslookup` (en. *nameserver lookup*). Exemple: `nslookup google.com`, `nslookup 192.0.32.10`. Este deci posibil să afli atât adresele IP pornind de la un nume precum google.com, însă și numele, pornind de la o adresă IP de Internet (nu de intranet).

⁷le poți numi și „intranet“, analog cu cele publice, „internet“.

⁸O glumă comună este să spui că administratorul serverului s-a împiedicat din greșeală de stecher și l-a tras din priză fără să își dea seama.

⁹o numim comandă, însă în realitate este un program, și se numește `ping.exe`

¹⁰Acest exercițiu nu trebuie rezolvat în cadrul programului de tutelare, este un exercițiu suplimentar, pentru tine.

Deși este posibil ca mai multe domenii să *pointeze* către aceeași adresă IP, o adresă IP nu poate „pointa” (prin *reverse DNS*) decât spre un domeniu.¹¹

Totuși, din ce este compusă o adresă precum `www.google.com`? `.com` se numește TLD (en. *top level domain*). Ceea ce se află în stânga sa este un *subdomeniu*, până la următorul punct. Astfel, google este un subdomeniu al lui `.com`, iar `www` este un subdomeniu al lui `.google.com`.

Dacă vrei să ai un domeniu al tău, trebuie să cumperi unul de la o firmă numită **domain name registrar**. Pentru a învăța PHP nu ai nevoie de un domeniu, poți face totul pe calculatorul tău personal, așa cum vei vedea în acest capitol.

Protocoloale

Internetul este „făcut” din așa-numite *servicii*. Cele mai cunoscute servicii sunt web-ul pentru documente interconectate,¹² *pop3* sau *imap* pentru e-mail, *irc* (en. *internet relay chat*) pentru chat, sau *ftp* (en. *file transfer protocol*) pentru transferul de fișiere.

Pentru a folosi unul dintre aceste servicii, ai nevoie de un program special numit *client*. Nu există un client care să „știe” să acceseze toate serviciile, pentru că fiecare serviciu este diferit de celelalte. Vom vedea mai târziu de ce.

Astfel, avem clienți pentru *www*, clienți pentru *ftp*, clienți pentru *irc*, ș.a.m.d.

Serviciul *world wide web*, sau pe scurt *web*-ul, este atât de răspândit, încât oamenii au dat un nume special clienților *www*: *browser*. Există mai multe browsere, create de diferite firme. Printre cele mai cunoscute se numără:

- **Firefox**, creat de Mozilla
- **Internet Explorer**, creat de Microsoft
- **Opera**, creat de Opera Software
- **Google Chrome**, creat de Google
- **Safari**, creat de Apple

Colocvial vorbind, te poți referi la client și ca fiind calculatorul pe care rulează clientul, sau chiar la utilizatorul uman din spate care îi dă instrucții clientului. În explicații voi încerca să nu mă exprim colocvial, ci corect din punct de vedere tehnic.

În continuare vom clarifica ce se întâmplă atunci când ceri o pagină web cu ajutorul unui browser de la un server.

Imaginează-ți că un calculator stă conectat la rețea și nu face nimic din „proprie inițiativă”. Acest calculator se numește *server*. Pe server, care este *mașina fizică*, rulează un *program* numit *daemon*. Acesta așteaptă conexiuni din exterior, pe care să le deservească. Administratorul serverului va spune colocvial *the server is up and running*, dar ce vrea să spună de fapt este că *serverul este conectat la Internet iar cel puțin un daemon așteaptă noi conexiuni*.

Să nu uităm că fiecare serviciu este diferit. Așa cum avem diferiți clienți pentru diferite servicii, tot la fel avem și *daemon*-uri diferite pentru fiecare serviciu.

Exemple de astfel de *daemon*-uri: *apache* pentru serviciul *www*, *UnrealIRCd* pentru *irc*, *postfix* pentru e-mail, ș.a.m.d. Reține că acestea sunt programe, la fel cum este și *firefox.exe*. În contrast cu asta noțiunile de *client* și *daemon* sunt clasificări generice pentru tipuri de software.

Ceea ce nu am încetat să sugerez până acum devine evident: cele două părți, clientul serviciului pe care doresc să-l utilizez, și daemonul care e capabil să-i răspundă clientului meu cu informații utile, trebuie să comunice unul cu celălalt.

¹¹Tehnic este posibil, însă practic poate cauza probleme.

¹²Prin link-uri.

Această comunicare dintre cele două programe se va desfășura într-un limbaj comun numit *protocol*. Fiecare serviciu are un protocol specific lui. Astfel avem un protocol irc pentru serviciul irc, protocolul ftp pentru serviciul ftp, sau protocolul http pentru serviciul web, ș.a.m.d. De aici vine acel „http://“ care precede orice adresă web. Această adresă se numește **URL** (en. *uniform resource locator*), o formă de **URI** (en. *uniform resource identifier*).

De obicei pe un server rulează mai multe *daemon*-uri în paralel și care așteaptă să deservească cereri. Dar cum să știe sistemul de operare al serverului pentru ce daemon este destinat un anumit pachet de date? Din câte știm până acum, nu are cum, pentru că OS-ul nu înțelege noțiunea de „protocol“. Totuși OS-ul trebuie să paseze datele primite pachet cu pachet daemon-ului corect.

Pentru a rezolva această problemă fiecare pachet de date este „însemnat“ cu un *port*. Un port este un număr între 1 și 65535 care îi permite sistemului de operare să paseze pachetul de date procesului¹³ corect.

Atunci când administratorul serverului (en. *sysadmin*) lansează în execuție un daemon, acesta îi spune OS-ului că ascultă (en. *listen*) pe un anumit port. Din acel moment, OS-ul „știe“ că acel port îi „aparține“ acelui daemon. Astfel, sysadmin-ul poate rula pe sistemul său mai mulți daemons concomitent, chiar și pentru servicii diferite, fără a-și face griji că pachetele de date ar putea ajunge la daemon-urile greșite.

Important de menționat este și că fiecare serviciu are un port standard. Serviciul www are 80 ca port standard. Astfel, URL-urile `http://example.com:80` și `http://example.com` sunt absolut echivalente.

În toate cele explicate până acum nu am intrat în detalii importante precum **TCP/IP**, însă este recomandată citirea și înțelegerea acelor noțiuni, cât și a celorlalte articole pe care le poți urma de acolo. Foarte importantă este și înțelegerea modelului **OSI** (en. *open system interconnection*).

1.3.1 Primul exercițiu de hacking

Înainte de a trece la conținutul propriu-zis, vreau să clarific de ce această secțiune folosește termenul *hacking*.

În primul rând, pentru a face acest capitol mai atractiv pentru toți cititorii care s-au plictisit de teoria uscată¹⁴ de rețelistică prezentată până acum, când ei vor de fapt să învețe PHP. Nu este ceva iluzoriu, ceea ce vei face în continuare chiar are de-a face cu hacking. Însă hacking implică mult mai multe¹⁵ cunoștințe – paginile anterioare doar ți-au prezentat sumar niște concepte de bază.

În al doilea rând, pentru a-mi crea ocazia de a explica termenul de hacker. În zilele noastre, *mass-media*, în încercarea sa de a crea¹⁶ senzaționalul, a redefinit *hacker* ca *infractor*. Însă nu asta este semnificația originală a cuvântului. Un hacker este o persoană care înțelege foarte bine ce face. Poți fi un hacker în orice domeniu, dar atunci când devii un hacker într-un domeniu, știi tot ce mișcă în branșa ta. Un hacker în calculatoare știe totul despre calculatoare, începând de la electronica procesorului, și terminând cu analiza psihologică a omului în relația sa cu calculatorul (sau, pe larg, cu tehnologia în general).¹⁷

O altă concepție greșită despre hacking este că înseamnă să intri în sisteme care nu-ți aparțin (*să le spargi*). Nu asta caracterizează hackerul, ci după cum am mai spus, înțelegerea lucrurilor cu care are de-a face. Te poți numi hacker la fel de bine dacă personalizezi un program scris de altcineva pentru nevoile tale. În astfel de cazuri se spune că *ai hack-uit codul sursă*, că l-ai înțeles mai întâi, pentru a-l putea modifica. După cum vezi, hacking înseamnă *înțelegere*.

Felul în care aplici această înțelegere depinde de tine. Asta nu îi absolvă pe cei care folosesc ceea ce știi pentru lucruri ilegale să fie catalogați drept infractori. Pe de cealaltă parte, hackerii adevărați reacționează acid când sunt priviți ca niște infractori, deoarece ei știu că *hacking* este de fapt ceva

¹³Un program care este executat. Gândește-te la ce listează „task manager“.

¹⁴Trebuie să recunoaștem că nu a fost deloc uscată.

¹⁵Și când spun asta, nu exagerez.

¹⁶În sensul de „a inventa“

¹⁷Cel mai renumit exemplu este **Kevin Mitnick**

constructiv și benefic unei minți sănătoase. Ei preferă ca infractorii să fie numiți *crackeri*, pentru a deosebi binele de rău.

.....

Atunci când introduci un URL într-un browser, acesta face multe lucruri în fundal. Unul dintre cele mai importante lucruri este comunicarea în limbajul HTTP (en. *hyper text transfer protocol*) cu *daemon*-ul de pe serverul dorit, asta după ce a aflat adresa IP a acestuia de la serverul DNS.

Noi vrem să facem manual ceea ce ar face un browser automat pentru noi. Pentru asta, avem nevoie de un program numit telnet. Este un program foarte simplu, tot ce face este să stabilească o *conexiune* cu *daemon*ul, și să-i transmită datele pe care le tastăm.

O dată ce conexiunea este stabilită, ambele endpoints (*daemon*ul și noi, cu ajutorul clientului telnet) pot scrie la grămadă date. De exemplu dacă noi scriem

foo

și *daemon*ul ne răspunde cu

bar

atunci toate datele comunicate arată ca un fișier cu conținutul:

foo

bar

Imaginează-ți că două endpoints comunică prin plicuri. Programul care inițiază conexiunea (numit client), scrie datele pe care urmează să le trimită într-un plic.

Dacă clientul și *daemon*ul doar schimbă date „așa cum sunt“, așa cum am face cu telnet, atunci doar acest plic este necesar. Însă dacă aplicațiile vor să se înțeleagă reciproc, ele trebuie să folosească un protocol de comunicare. Pentru web, acest protocol este HTTP.

Acest protocol de comunicare constituie *layer*-ul *application level* din **modelul OSI**.¹⁸

În astfel de cazuri, clientul va trebui să pună în plicul nostru încă un plic¹⁹ cu datele structurate specifice protocolului folosit, în cazul nostru cele specifice protocolului HTTP.


Majoritatea sistemelor de operare moderne vin cu programul telnet preinstalat. Deci deschide interfața CLI și lansează programul, pasându-i ca prim parametru numele serverului, și *port*-ul ca al doilea parametru, deci:

telnet example.org 80

După ce s-a conectat, îi poți cere *daemon*ului ce document vrei, însă trebuie să o faci în limbajul HTTP, altfel nu va „înțelege“ ce vrem de la el. Altfel spus, folosind analogia cu plicul, noi ca oameni, deoarece știm că vrem să comunicăm în HTTP, și deoarece telnet nu are noțiunea de *application layer*, avem responsabilitatea de a ne „împacheta“ singuri cererea în încă un plic cu formatul specific HTTP, plic pe care i-l pasăm lui telnet, care ne va pune plicul pe rând în toate celelalte *layere* („plicuri“) din modelul OSI aflate sub *application layer*: presentation, session, transport, network, iar electronica (de exemplu placa de rețea) se va ocupa de data link layer și physical layer.²⁰

De exemplu, îi vom cere pagina de start:

GET / HTTP/1.1 




Host: example.org 



¹⁸http://en.wikipedia.org/wiki/OSI_model

¹⁹sau mai multe, în funcție de protocolul de comunicare folosit

²⁰Lucrurile sunt mult mai complexe, telnet singur nu ar putea face astfel de lucruri fără susținere din partea sistemului de operare. Aici am simplificat imaginea pentru o mai ușoară înțelegere.

Simbolul  înseamnă că trebuie să apeși tasta . Deci pentru a termina cererea HTTP, apeși de două ori  la rând. Este posibil ca programul tău telnet să nu afișeze ce tastezi. În acest caz, trebuie să scrii orbește, și să o faci exact ca mai sus, *inclusiv* scrisul cu litere mari/mici.

GET este metoda HTTP (en. *http method*) pe care o folosim pentru a cere resursa (documentul) dorit.

Protocolul²¹ HTTP ne dictează că după GET urmează un spațiu și apoi calea către documentul dorit, urmat de protocolul folosit, aici HTTP, un slash, și versiunea protocolului.


Prima linie GET / HTTP/1.1 se numește *request line*. După ea urmează o serie de *request headers*, precum Host mai sus.

Request line și request headers împreună sunt cunoscute și ca cererea HTTP (en. *HTTP request*).

Daemonul îți va răspunde cu un text care arată cam așa, și care constituie răspunsul HTTP (en. *HTTP response*):

```
HTTP/1.1 200 OK
Server: Apache/2.2.3 (Red Hat)
Last-Modified: Tue, 15 Nov 2005 13:24:10 GMT
ETag: „b300b4-1b6-4059a80bfd280\
(1) Accept-Ranges: bytes
Content-Type: text/html; charset=UTF-8
Connection: Keep-Alive
Date: Tue, 15 Dec 2009 11:52:46 GMT
Age: 2528
Content-Length: 438

<HTML>
<HEAD>
  <TITLE>Example Web Page</TITLE>
</HEAD>
<body>
(2) <p>You have reached this web page by typing &quot;example.com&quot;;
    &quot;example.net&quot;;
    or &quot;example.org&quot;; into your web browser.</p>
    <p>These domain names are reserved for use in documentation and
    are not available for registration.
        See <a href=„http://www.rfc-editor.org/rfc/rfc2606.txt\>
        RFC 2606</a>, Section 3.</p>
</BODY>
</HTML>
```

Întreaga comunicație cu daemonul este deci compusă din trei segmente mari, fiecare separate printr-o linie goală, rezultată din apăsarea de două ori a tastei : *HTTP request*, *HTTP response headers* (segmentul (1) din outputul de mai sus), și *HTTP response body* (segmentul (2)).

Felul în care se face cererea este parte din specificația limbajului de comunicare HTTP. Acest limbaj, ca și multe altele, au fost stabilite prin așa-numitele **RFC**-uri (en. *request for comments*).

Folosind analogia cu plicurile, cele șapte niveluri din modelul OSI nu sunt totul: protocolul HTTP însuși mai definește încă trei plicuri care trebuie puse în „plicul” cu datele HTTP:

- *HTTP request*
- *response headers*

²¹Formatul

- *response body*

Chiar dacă cele trei „segmente“ sunt scrise fie de client, fie de daemon, noi privim *întreaga comunicație* ca pe un fișier unitar, și din acest motiv putem spune că este constituită din aceste trei segmente („plicuri“).

După ce a primit răspunsul HTML, un browser ar face alte lucruri precum:

- crearea de cereri noi pentru fiecare element care face referire la resurse externe precum imagini, frame-uri, scripturi Javascript, ș.a.m.d. În urma acestui pas imaginile ar părea că fac parte din document, însă ele sunt de fapt resurse diferite și de sine stătătoare, cu URL-uri diferite.
- executarea scripturilor Javascript

Deoarece pentru fiecare resursă externă este necesară crearea unei noi conexiuni TCP/IP și comunicarea cu serverul în limbajul HTTP, protocolul HTTP este un *protocol stateless*²².



HTTP e stateless

Faptul că HTTP este un protocol stateless are implicații majore asupra structurii fundamentale în care îți vei concepe aplicațiile.

Gândește-te la aceste implicații și explică-le în 200-300 cuvinte.

Însă clientul nostru telnet nu știe toate aceste lucruri, el nu înțelege limbajul de comunicare HTTP pe care l-am folosit,²³ cum nu înțelege nici limbajul de formatare HTML, în consecință nici nu poate afișa imagini sau executa scripturi Javascript. El doar s-a conectat la daemon pe portul dorit de noi, și ne-a oferit posibilitatea de a comunica cu el în HTTP pentru a primi codul HTML al paginii.

Dacă în acest moment am vrea să cerem daemonului un alt fișier, și nu cel standard, ar trebui să nu-i cerem resursa „/“ (care urmează după „GET“ în cererea anterioară, și care este numită și *root*), ci numele resursei cu tot cu calea sa absolută. De exemplu, pentru fișierul „http://example.org/contact.html“, cererea ar arăta astfel:

```
GET /contact.html HTTP/1.1
```

```
Host: example.org
```

Motivul pentru care trebuie să specificăm la ce domeniu ne referim în câmpul (en. *header field*) *Host* (aici: example.org) este că un daemon HTTP poate deservi mai multe domenii simultan, și deci trebuie să-i spunem la care din ele ne referim. De exemplu, este posibil ca http://example.org și http://www.example.org să fie două site-uri complet diferite, independente, cu fișiere diferite. Întâmplător, administratorul care a setat acel apache 2.2.3 (știm că acel program cu acea versiune deservește acel site din response header-ul „Server“) a făcut setările astfel încât cele două hostname-uri complet diferite example.org și www.example.org să facă referire către același director de pe server. Din acest motiv nu contează către ce hostname trimitem cererea, daemonul ne va răspunde cu aceleași fișiere.

Instalarea mediului de dezvoltare

În această secțiune vom instala cele două scule necesare creării și testării aplicațiilor web, sau mai bine spus, scripturilor PHP. Mă voi limita doar la instrucțiuni pentru Microsoft Windows XP²⁴. Instalarea sub GNU/Linux este ușoară: nu trebuie decât să instalezi pachetele de genul apache sau httpd și PHP din repoziitoriul pus la dispoziție de distribuția ta, așa cum ai instala orice alt program. Directivele de

²²http://en.wikipedia.org/wiki/Stateless_protocol

²³Protocolul de comunicare este „plicul“ ce corespunde *application layer* în modelul OSI.

²⁴Instalarea sub Windows Vista sau Windows 7 este asemănătoare.

configurare sunt foarte similare, deci o citire a instrucțiunilor următoare nu este de prisos, chiar dacă folosești GNU/Linux.

Pașii și instrucțiunile de configurare sunt aceleași, atât pentru windows, cât și pentru GNU/Linux.

În primul rând, intră pe pagina oficială a daemonului apache²⁵ și urmează linkul Download a ultimei versiuni, momentan²⁶ 2.2.19²⁷.



Figura 1.3: Pagina oficială Apache HTTPD

Alege un *mirror*²⁸ și apasă change. Mirror-ul selectat automat ar trebui să fie unul dintre cele mai bune. Apoi selectează versiunea „Win32 Binary without crypto“.

Introdu datele ca în figura 1.4 și treci la pasul următor.

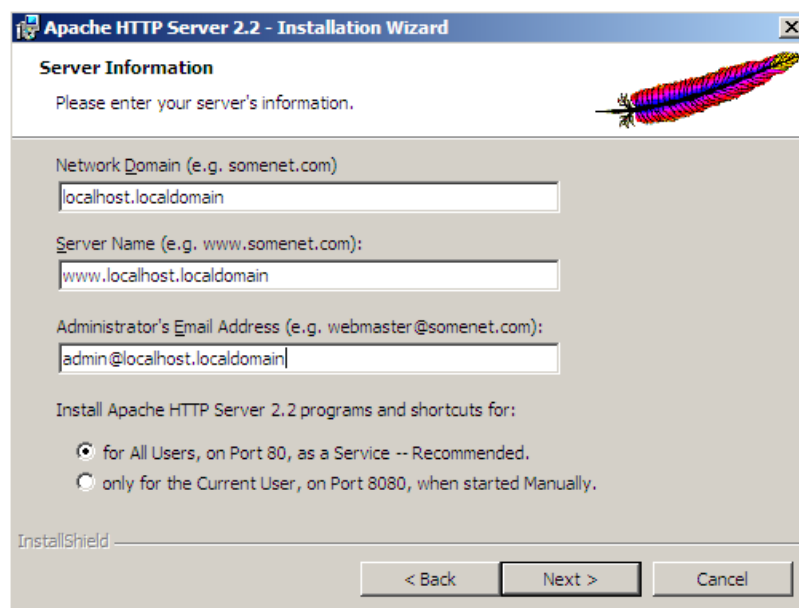


Figura 1.4: Setarea inițială apache httpd

Alege modul de instalare custom, deoarece vrem să decidem exact unde și ce se instalează, ca în 1.5.

În figura 1.6 avem un arbore cu toate componentele instalate. Apache HTTP Server 2.2.14 este părintele tuturor componentelor. Selectează-l și apasă pe butonul Change... pentru a schimba locul de

²⁵<http://httpd.apache.org/>

²⁶08.08.2011

²⁷Tot timpul este bine să alegi ultima versiune, cea stabilă.

²⁸ Un mirror este o oglindire a unor fișiere. Mai multe firme sau instituții au ales să pună la dispoziție ogindiri ale fișierelor de instalare pentru apache. Oricine poate face un mirror, însă e recomandat să înștiințezi autorul oficial, pentru ca acesta să pună un link către mirror-ul tău. Toate fișierele de pe un mirror sunt identice, deci teoretic nu contează ce mirror alegi, însă este recomandat să alegi un mirror apropiat ție din punct de vedere geografic, pentru ca descărcarea să fie mai rapidă.

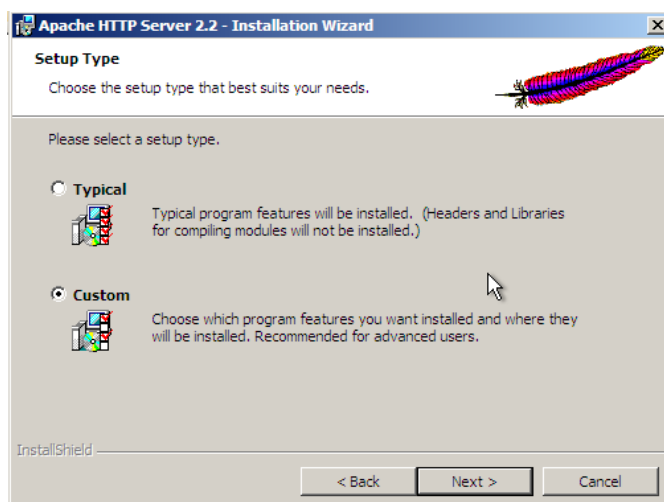


Figura 1.5: Setare personalizată

instalare.

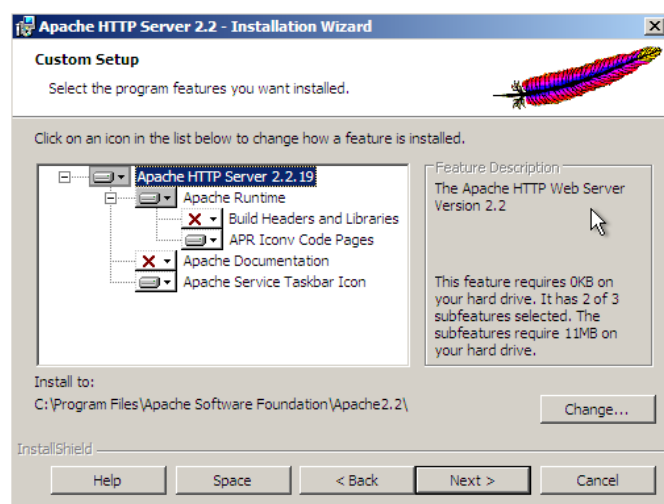


Figura 1.6: Setare httpd - Componentele instalate

Crează un nou director `C:\webdev\` unde vor fi instalate toate sculele de programare web, introducând manual calea `C:\webdev\apache` exact ca în 1.7.

Confirmarea te va aduce la dialogul anterior, iar în partea de jos ar trebui să scrie că programul va fi instalat la acea locație.

Finalizează instalarea. La vizitarea adresei <http://localhost/> ar trebui să vezi mesajul `It works!`. Asta înseamnă că daemonul apache este setat corect și poate servi pagini statice în formatul html, iar calculatorul tău este acum un server.

Noi însă vrem să generăm în mod dinamic cod HTML cu PHP. Deci intră pe pagina oficială PHP²⁹ și urmează linkul de download, sub Windows binaries³⁰. Selectează tipul de executabil VC9 x86 Thread Safe pentru versiunea PHP 5.3, ca în 1.8, apoi descarcă arhiva .zip care conține PHP, salvând-o în `C:\webdev\`.

După dezarhivare, conținutul său va arăta ca în Figura 1.9.

²⁹<http://php.net>

³⁰În părțile ce urmează se folosește versiunea 5.3.6, dar tu trebuie să alegi ultima versiune stabilă. Important este doar să selectezi versiunea compilată cu VC9, care este thread-safe.

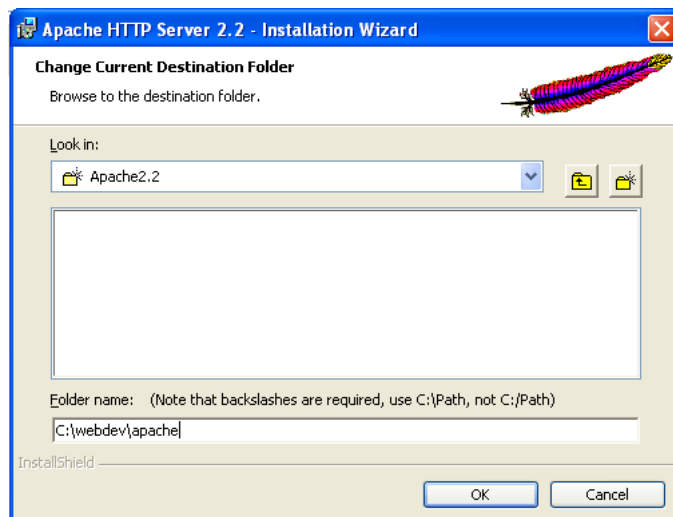


Figura 1.7: Setare httpd - Calea instalării

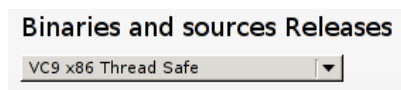


Figura 1.8: PHP - Tipul *build*-ului PHP

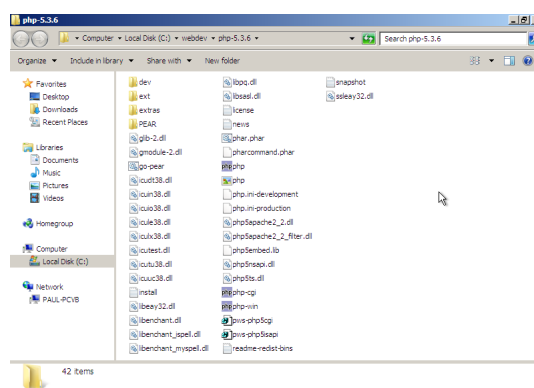



Figura 1.9: PHP - What you get

Înainte de a trece la integrarea lui PHP în apache, îți recomand să instalezi un editor text ideal pentru începătorii în programare: **Notepad++** ³¹, de pe pagina oficială.

Notepad++ este foarte bun ca editor deoarece, fiind un editor text, vezi exact ce faci, preluând  controlul, așa cum ar trebui să o facă orice programator. Pe lângă asta, Notepad++ îți și colorează textul dacă recunoaște limbajul în care scrii acel text, precum PHP sau HTML.

❧ Dacă ești cumva tentat să folosești editoare WYSIWYG³² precum Dreamweaver, atunci ar trebui să te oprești acum - programarea nu e pentru tine. De ce? În afară de faptul că un astfel de editor nu te-ar stimula să înveți, ți-ar și crea mai multe probleme. Cu PHP, tu ca programator vei prelua controlul și vei genera HTML. Pe lângă asta, astfel de programe nu sunt destul de puternice ca o minte umană, și mai și generează cod greșit.

³¹<http://notepad-plus.sourceforge.net/>

³²what you see is what you get

PHP poate fi folosit și ca limbaj de scripting general, nu doar pentru generarea dinamică de HTML. Însă pentru asta trebuie setată calea către directorul php care conține fișierul de configurare php.ini, asupra căruia voi reveni puțin mai târziu.

Sub Windows XP, click dreapta pe My Computer și apoi Properties. Sub Windows 7, în start -> run introdu comanda:

`C:\Windows\System32\SystemPropertiesAdvanced.exe`

Alege tabul Advanced, apoi click pe Environment Variables în partea de jos a dialogului, așa cum vezi în Figura 1.10.

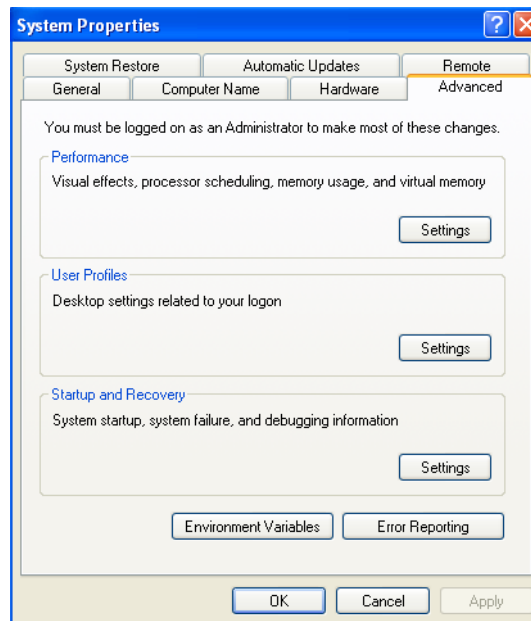


Figura 1.10: Proprietățile avansate ale sistemului Windows XP

Se va deschide un dialog similar cu cel din Figura 1.11.

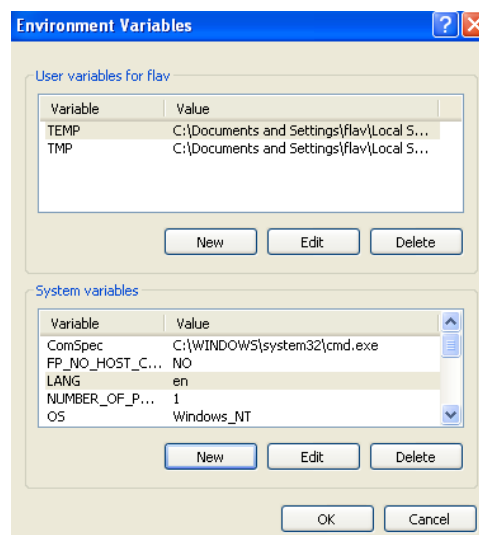


Figura 1.11: Variabilele de mediu ale unui sistem Windows

Click pe New sub System variables, și introdu datele ca în Figura 1.12. Această nouă variabilă de mediu îl va ajuta pe PHP să-și găsească fișierul de configurare php.ini.

În acest moment, PHP este setat și l-am putea lansa în execuție folosind calea absolută

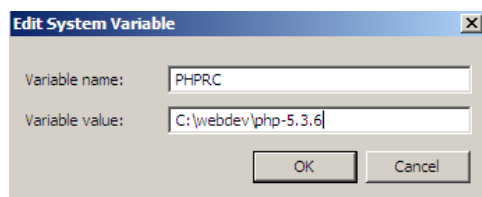


Figura 1.12: Variabilele de mediu ale unui sistem Windows

C:\webdev\php-5.3.6\php.exe, ceea ce poate deveni obositor cu timpul. Însă putem integra php.exe în sistem astfel încât să fie recunoscut ca comandă.

Pentru ca asta să se întâmple, identifică variabila sistemului numită Path și adaugă-i la sfârșit: ;C:\webdev\php-5.3.6\, ca în Figura 1.13.

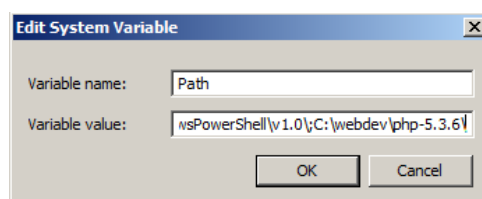


Figura 1.13: Variabila de mediu PATH

Acel ';' de la început este foarte important. PATH conține o listă de directoare în care windows se va uita pe rând atunci când lansezi în execuție un program ca pe o comandă. ';' are rolul de a separa aceste directoare.

Un singur lucru mai lipsește: php.ini. Arhiva pe care tocmai ai descărcat-o vine cu două astfel de fișiere, php.ini-development și php.ini-production. Fă o *copie* a fișierului php.ini-development și redenumeste-o php.ini.

Testează dacă ai setat totul cum trebuie deschizând o nouă instanță a promptului ms-dos. Acolo introdu comanda php --ini. Totul este corect dacă php este găsit de sistem, iar comanda de mai sus îți arată C:\webdev\php-5.3.6\php.ini ca *Loaded Configuration File*.

Acum vom trece la integrarea lui PHP ca modul apache. Vom spune că folosim SAPI-ul³³ (en. *server application programming interface*) apache2. În directorul în care ai instalat apache este un subdirector de configurare numit sugestiv conf/, unde rezidă toate fișierele de configurare pentru apache.

Fișierul principal de configurare al apache se numește httpd.conf. Este singurul fișier încărcat automat de apache pentru a citi configurația, celelalte fișiere trebuie incluse folosind directiva Include din acest fișier principal (en. *the master configuration file*). Liniile care încep cu caracterul diez (#) sunt comentarii, iar conținutul lor nu este luat în considerare, chiar dacă conține directive de configurare valide.

Deschide httpd.conf cu notepad++ și adaugă-i la sfârșit (combinația **CTRL+END**) directiva Include conf/extra/httpd-php.conf apoi crează un nou fișier cu **CTRL+N** cu acest conținut:

```
LoadModule php5_module "C:/webdev/php-5.3.6/php5apache2_2.dll"
AddType application/x-httpd-php .php
PHPIniDir "C:/webdev/php-5.3.6"
```

```
<IfModule dir_module>
    DirectoryIndex index.php index.html
</IfModule>
```

³³http://en.wikipedia.org/wiki/Server_Application_Programming_Interface

Acum apasă **CTRL+S** și salvează-l ca `httpd-php.conf` în directorul `C:\webdev\apache\conf\extra\`

Directiva `LoadModule` îi spune să încarce un fișier ca modul `apache`. `AddType` îl instruieste să interpreteze fișiere ce se termină în `.php` ca `application/x-httpd-php`, care este un tip MIME³⁴ (en. *multipurpose internet mail extension*). `PHPIniDir` îi spune lui PHP (de data asta modulului PHP) unde își găsește fișierul de configurare `php.ini`, așa cum variabila de mediu `PHPRC` pe care am setat-o anterior îi spune același lucru SAPI-ului CLI (adică lui `php.exe`).

Acel bloc condițional³⁵ `IfModule` verifică dacă `apache` are modulul „`dir`”, și dacă da, setează resursa de start corespunzătoare `root`-ului (atunci când este accesat „/” al unui director, așa cum ai văzut în capitolul anterior) fiecărui director. Fișierele sunt listate după ordinea priorității, dacă primul nu este găsit, `apache` va încerca să-l deservească pe al doilea.

O altă directivă importantă din `httpd.conf` este `DocumentRoot`. Acesta îi spune lui `apache` de unde să deservească fișiere. În mod standard, acest director se numește `htdocs` – *hypertext documents*.

Acum că avem totul setat, restartează daemonul `httpd`, în caz că e pornit, pentru a prelua toate schimbările din `httpd.conf` și fișierele incluse de acesta. Ar fi trebuit să-i dăm restart și dacă am fi modificat configurația `php` din `php.ini`.

Pentru a verifica instalarea și configurarea lui PHP ca modul `apache`, crează un nou fișier și salvează-l ca `C:\webdev\apache\htdocs\info.php`. Apoi introdu următorul text, numit și cod sursă (în limbajul PHP):

Listing 1.1: `phpinfo()` furnizează toate informațiile despre instalarea PHP curentă

```
1 <?php
2 phpinfo();
```

și salvează-l.

Acum intră pe adresa <http://localhost/info.php>. Ar trebui să vezi ceva similar cu Figura 1.14.

Felicitări, acum ai PHP instalat și poți învăța programare, fie folosindu-l pentru programare web ca modul `apache`, fie folosindu-l ca limbaj de scripting universal folosind SAPI-ul CLI (programul `php.exe`).

SAPI-urile există tocmai pentru a ușura integrarea lui PHP în toate aceste medii: pentru web putem folosi un SAPI specific daemonului folosit, de exemplu pentru `apache` avem SAPI-ul `php5apache2_2.dll`. Acest fișier DLL reprezintă însă doar interfața (SAPI-ul, așa cum îi spune și numele de *interface*) de comunicare dintre `apache` și ”adevăratul PHP”.

Pe același calapod, `php.exe` reprezintă interfața de comunicare dintre promptul `ms-dos` (sau `shell`, într-un sistem `*NIX`) și același ”adevărat PHP” ca și în cazul `apache`. Altfel spus, ambele, atât `php.exe`, cât și `php5apache2_2.dll` sunt două SAPI-uri, fiecare destinate integrării lui PHP în mediile lor specifice de execuție a scripturilor PHP.

În capitolul următor ne vom uita mai îndeaproape la limbajul PHP.

³⁴http://en.wikipedia.org/wiki/Multipurpose_Internet_Mail_Extensions

³⁵Se numește condițional deoarece ceea ce se află în interiorul său este inclus doar dacă condiția este adevărată, numele însuși conținând `If`.


<div> <div>PHP Version 5.3.6</div>  </div>	
System	
Build Date	Mar 17 2011 10:34:15
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--disable-isapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=D:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet" "--with-mcrypt=static"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,TS,VC9

Figura 1.14: Informații afișate de un apel la `phpinfo()`

Controlul fluxului de execuție¹ și de date²

Înțelegerea fluxului de execuție este primordială în crearea de aplicații care fac mai mult decât simple afișări – care iau decizii și îți fac site-ul dinamic. Indiferent de cât de simplă sau cât de complexă va fi aplicația ta, indiferent dacă vei folosi baze de date sau simple fișiere, cu siguranță vei folosi fluxul de execuție pentru a controla fluxul de date. Acest capitol te introduce în lumea datelor, iar apoi îți arată cum să le manipulezi.

O altfel de reîmprospătare

Vreau să stabilesc niște lucruri care probabil nu sunt evidente pentru tine, sau asupra cărora nu ai insistat prea mult când ai început să înveți să creezi *site-uri* statice cu HTML, eventual cu CSS sau poate chiar cu interacțiune în interiorul browserului cu JavaScript.

HTML este un limbaj de formatare (en. *markup language*). Cu el nu controlezi ceva, nu iei decizii, deci nu este un limbaj de programare. În HTML doar structurezi un document. În mod ideal îl structurezi semantic (**semantic HTML**), pentru a putea fi interpretat mai bine de motoarele de căutare web (en. *search engine*).

Există multe formate de fișiere, și chiar în capitolul anterior ai lucrat cu două astfel de formate – formatul de configurare specific apache, și protocolul HTTP. Dacă îți amintești, ți-am explicat pe rând ce înseamnă directive precum Include. Semnificația acestor directive, sau general spus, semnificația oricărei entități specifice unui limbaj (fie el de markup – HTML, de configurare – httpd.conf, de programare – PHP, sau un protocol de comunicare – HTTP) constituie *semantica* acelui limbaj.

Semantica unui anumit construct al unui limbaj (de orice natură ar fi el) este strâns legată de *contextul* în care se află acel construct. De exemplu, pentru a cere o resursă cu HTTP, am văzut că poți începe cererea cu GET, urmat de o cale absolută (care începe cu „/”) din cadrul Host-ului în cauză. Calea respectivă are semnificația pe care o vrem doar în contextul lui „GET”, ca dovadă că atunci când vedem căi de genul /script.php pe site-uri web, acestea nu sunt interpretate automat ca parametri GET, și deci browserul nostru nu este redirecționat către acele pagini.

Contextul a ceva înseamnă în ce punem acel ceva pentru a avea o anumită semantică. De exemplu, sarea pusă în contextul gătitului are semantica de *condiment*, dar dacă o pui pe o rană, are semantica de *ceva care provoacă durere*. Cu alte cuvinte, *context* înseamnă *circumstanțe* sau *mediul înconjurător*.

Analog, „HTTP/1.1” are semantica de „protocolul și versiunea folosită” în contextul *request line*-ului HTTP al unui *request HTTP*.

¹en. *execution flow*

²en. *data flow*



Întrebări de sinteză**

Însă și GET însuși are semnificația pe care ai întâlnit-o în capitolul trecut doar într-un anumit context semantic.

1. Care este acest context semantic?
2. În ce context semantic are semantica întâlnită constructul Include?
3. În ce context semantic are sens comunicarea în limbajul (în protocolul) HTTP?

Determină contextul semantic în care următoarele constructe au semantica pe care o intuiești ca cunoscător al limbajului HTML:

4. `<td>`
5. `<tr>`
6. `<body>`
7. `<html>`
8. `href`

Dacă ai avut dificultăți majore la răspunderea întrebărilor de sinteză, te rog ia atitudine. În primul rând, plec de la premiza că citești cu atenție, și că reții tot ce-ți povestesc. Toate noțiunile pe care le introduc, le introduc pentru că astfel voi putea explica lucruri destul de complicate mai târziu, pe baza celor spuse aici. *Asta îți va permite să știi multe învățând cât mai puțin.* „Dezavantajul“ este că va trebui să fii concentrat la ce citești, și să sintetizezi singur



mult. Sinteza aceasta este un exercițiu perfect pentru tine ca viitor programator, deoarece atunci când vei programa vei fi confruntat cu această nevoie de a sintetiza lucruri. Metoda mea de predare, deși dură, te pregătește foarte bine pentru cariera ta de programator. Deci dacă simți că nu ești stăpân pe ce ai învățat până acum despre rețelistică și despre semantică, recitește acum, până nu te pierzi definitiv. Când recitești, urmează link-urile menționate – după cum am spus în capitolul Introducere, acestea nu sunt lectură opțională.

Pe lângă semantică, un limbaj mai are și o *sintaxă*. Regulile sintactice ale limbajelor sunt necesare pentru a crea contextul semantic în care vor exista constructele acelui limbaj.

Vreau să ilustrez asta cu un exemplu: în protocolul HTTP, GET trebuie să fie separat de cale printr-un spațiu. Asta este o regulă sintactică a limbajului, standardizată prin RFC-uri, însă practic separatorul ar putea fi orice altceva. Însă un separator trebuie să fie acolo, altfel calculatorul (browser-ul sau daemon-ul) nu ar putea decide unde începe cuvântul cheie „GET“, unde se termină, și unde începe calea către resursa pe care o dorim. Aceste reguli sintactice permit programelor precum daemon-uri și clienți HTTP să parseze (să „înțeleagă“) datele comunicate reciproc.

Există multe posibilități de a exprima sintaxa unui limbaj, însă acestea sunt mult prea complexe pentru noi. Însă există un standard nescris pentru a specifica sintaxa unor constructe simple, într-o singură linie. Ea se leagă de necesitatea unui anumit parametru.

De exemplu, sintaxa constructului GET ar putea fi:

`'GET' <RESOURCE> 'HTTP/1.1'`

RESOURCE este pus între `<` și `>`, ceea ce denotă că este un parametru necesar, care trebuie specificat. `'GET'` (inclusiv spațiul) și `'HTTP/1.1'` sunt puse între apostrofuluri pentru a arăta că sunt lucruri ce trebuie scrise exact așa cum sunt. RESOURCE este numele simbolic al parametrului, pe care îl putem refolosi în documentația limbajului (în cazul nostru, documentarea limbajului HTTP, sau mai bine spus, a unei cereri HTTP de bază).

Pentru a specifica că un parametru e opțional, îl punem între `[` și `]`. Astfel, sintaxa unei cereri HTTP ca cea pe care am făcut-o în capitolul anterior, împreună cu descrierea ei, ar putea arăta astfel:

```
'GET ' <resource> ' HTTP/' <version> <enter>
['Host: ' <name> <enter> ]<enter>
```

- resource = the absolute path to the resource
- version = the version of the HTTP protocol used;
currently only 1.0 and 1.1 are supported
- name = the hostname
- enter = press return once

Pe lângă lucrurile evidente pe care ni le spune această specificație sintactică, ne mai spune și un lucru care probabil ți-a scăpat: câmpul Host este opțional, dar dacă îl specificăm, atunci trebuie să specificăm și parametrul name, și să și apăsăm o dată enter.

Cu siguranță ai realizat că astfel de reguli pot fi incluse una în alta, creând reguli destul de complexe.


Un limbaj precum cel de mai sus, care folosește <,>[,] pentru a specifica un alt limbaj se numește un *metalimbaj*. *Meta* înseamnă *care descrie* - un limbaj care descrie limbajul.

Cel mai probabil ai întâlnit deja tag-ul HTML <meta>, de aici îi provine numele. Diferența este că <meta> nu se referă la limbaj (HTML în cazul de față), ci la informații. Ceea ce <meta> ne spune despre documentul HTML în care se află se numesc *metainformații* - *informații care descriu informațiile din documentul curent*.

Probabil ai folosit deja un forum pe web. Probabil că acel forum îți spunea la un moment dat că autorul unei intrări (al unui *topic* sau *thread*) se numește Xulescu. Ei bine, în timp ce intrarea propriu-zisă constituie informația, numele autorului este o metainformație (en. *metadata*) - o informație despre informație.

La ce îți folosește această cunoaștere despre metadata, metalimbaje, sintaxă și semantică? În primul rând, ai învățat primul cel mai important lucru pragmatic din viitoarea ta carieră de programator: să citești manualul (PHP sau orice altceva) – chiar dacă încă nu ești conștient de asta.

În al doilea rând, meta-ceva-urile te vor însoți în toate aplicațiile pe care le vei programa.

 Uită-te la toate aplicațiile pe care le folosești, vei vedea că toate au niște metadata. Singurul lucru care te-ar împiedica să vezi asta este că datele și metadatale se întretaie atât de mult încât e greu să le identifici pe fiecare.

Și în al treilea rând, pentru a te pregăti pentru exercițiile următoare care au menirea de a te dezgheța la minte puțin, deoarece, din păcate, nu știu nimic despre cititorul meu, însă trebuie să mă asigur cumva că este pe aceeași lungime de undă ca mine, ceea ce-i permite să asimileze cât mai eficient cunoașterea prezentată în continuare, fără risipă de cuvinte.



Reguli sintactice*

Fie regula sintactică

[A] A [A A] [A [A]] <C>

Partea I

Care dintre următoarele inputuri o respectă?

1. AABC
2. AAAAC
3. AC
4. AAAC
5. AAABC

Partea II

Detășează-te de nivelul abstract al acestei reguli sintactice, și fă o afirmație *pragmatică* despre B. Afirmația începe așa: *Într-un input valid, B este mereu ...*

Metalimbajul prezentat nu este bătut în cuie. În primul rând, caracterele speciale ale limbajului `<`, `>`, `[`, `]` și `'` pot fi schimbate în orice, atâta timp cât documentezi aceste schimbări aduse de tine.

Deasemenea, nu este decât un standard nescris, și îl poți extinde în ce fel ai nevoie. Din nou, important este doar să documentezi „extensiile” aduse metalimbajului astfel încât ceilalți programatori să îți înțeleagă specificația limbajului pe care îl descrii cu ajutorul acelei extensii proprii a metalimbajului.

De exemplu, să zicem că vrem să introducem un nou construct în acest metalimbaj care să însemne *simbolul din stânga mea poate apărea o dată sau de mai multe ori*, și ne decidem să folosim simbolul `'+'` pentru asta.

Astfel o regulă de genul:

`<FOO+> [BAR+]`

S-ar putea citi ca: *Unul sau mai mulți FOO urmat de zero sau mai mulți BAR*. Un astfel de simbol precum `'+'` se numește *cuantificator*. Ar fi la îndemână să stabilim, în documentația extensiei noastre adusă metalimbajului, că `'+'` poate cuantifica orice entitate din stânga sa, inclusiv o grupare `<>` sau `[]`.


Documentația ar suna așa:

`+` = repeat the entity on its left once or multiple times (a quantifier)
the entity can be any SYMBOL, `<required parameter>` or
`[optional parameter]`

Iar exemplul de mai sus, în care vrem ca FOO și BAR să fie separați de un eventual spațiu, ar deveni:

`<FOO ' '+> [BAR ' ']+`

Până acum definițiile noastre sintactice se limitau doar la o singură regulă (o singură linie), însă am putea introduce constructe în metalimbaj care ne-ar da voie să atribuim nume acestor reguli, și să refolosim acele nume în definițiile altor reguli sintactice, creând astfel interdependențe între reguli, și deci crea specificațiile unor limbaje foarte complexe.

 Majoritatea limbajelor de programare, inclusiv PHP, sunt definite în astfel de limbaje.



Sintaxa HTML***

Acum hai să aplicăm ce am învățat asupra limbajului HTML. În HTML avem tag-uri (ex. `<html>`) care au attribute și valori.

1. Crează o specificație sintactică a limbajului HTML folosind doar cuvintele cheie TAG, ATTRIBUTE și VALUE, care combinate cu `[]` și `<>` să reflecte cât mai corect sintaxa limbajului. Specificația trebuie să valideze orice text HTML valid. Un exemplu de input ar fi:

```
<form method="get">
  <checkbox name="hello" checked>
  <input type="submit">
</form>
```

Note:

- Regula creată *nu* trebuie să ia în calcul sintaxa specifică XHTML (în care de exemplu `''` ar fi greșit, doar `''` este valid).

- Se pleacă de la premiza că inputul este cel mai curat HTML posibil, că sunt folosite ” pentru a delimita valorile atributelor (dacă acestea există), că nu e mai mult de un spațiu acolo unde e nevoie de spațiu, ș.a.m.d. Pe scurt: folosește-ți intuiția pentru a decide ce înseamnă „cel mai curat HTML posibil“.
 - Deoarece caracterele < și > au o semnificație specială în limbajul HTML, pe care încerci să-l descrii sintactic folosind printre altele și caracterele < și > înseși, va trebui să le pui între apostrofuri, pentru a face diferența între <,> care ne spun în metalimbajul nostru că acel parametru este necesar, și '<' sau '>' care ne spun că ne referim la caracterul '<' respectiv '>' în limbajul pe care vrem să-l descriem (adică HTML însuși).
 - Va trebui să extinzi metalimbajul (nu uita să și documentezi extensiile aduse) pentru a ajunge la o rezolvare cât mai corectă și completă
 - Exercițiul este destul de dificil. Încearcă să te apropii cât mai mult de soluția cea mai corectă și completă.
-