

Programmazione di siti web dinamici

ANALISI DEL PROBLEMA

Inconvenienti delle tecnologie e attuali

14/01/2006

Jean-Vincent Loddo
Laboratoire d'Informatique de Paris Nord (LIPN)

Sommario

1. Il passaggio da «statico» a «dinamico»
2. La correttezza degli script/programmi (codice)
3. La correttezza del html/xml generato (risultati)
4. L'inadeguatezza del modello di funzionamento (sessioni, cookies)
5. Il problema dello stato globale (basi di dati)
6. Altri problemi e difficoltà

Sommario

1. Il passaggio da «statico» a «dinamico»
2. La correttezza degli script/programmi (codice)
3. La correttezza del html/xml generato (risultati)
4. L'inadeguatezza del modello di funzionamento (sessioni, cookies)
5. Il problema dello stato globale (basi di dati)
6. Altri problemi e difficoltà

Il passaggio da «statico» a «dinamico»

Definizione

- **sito statico** = insieme finito di pagine html memorizzate in file sul disco
- **sito dinamico** = insieme finito di programmi (script) che generano dinamicamente («al volo»), come **risultato del calcolo**, le pagine html richieste dall'utente
- Esempio: *HyperNietzsche*
 - => base dati in costante evoluzione
 - => necessariamente dinamico

Il passaggio da «statico» a «dinamico»

Evoluzione del concetto di sito web

- **Contenitore di informazioni**
 - ipertesto (informazione = testo + immagini)
 - interazione semplice (link = nesso)
- **Interfaccia utente**
 - di una applicazione remota
 - interazione complessa (form, javascript, applet)

Definizione

- **sito statico** = insieme finito di pagine html memorizzate in file sul disco
- **sito dinamico** = insieme finito di programmi (script) che generano dinamicamente («al volo»), come **risultato del calcolo**, le pagine html richieste dall'utente
- Esempio: *HyperNietzsche*
 - => base dati in costante evoluzione
 - => necessariamente dinamico

Il passaggio da «statico» a «dinamico»

Esempio di pagina HTML

```
<HTML>
  <HEAD>
    <META http-equiv="content-type" content=
      "text/html; charset=iso-8859-1">
    <META name="generator" content="Adobe GoLive 4">
    <TITLE>HyperNietzsche</TITLE>
  </HEAD>

  <BODY bgcolor="#B0E0E6">

    <DIV align="left">
      <P><FONT size="4"><B>Projet HyperNietzsche</B></FONT></P>
    </DIV>

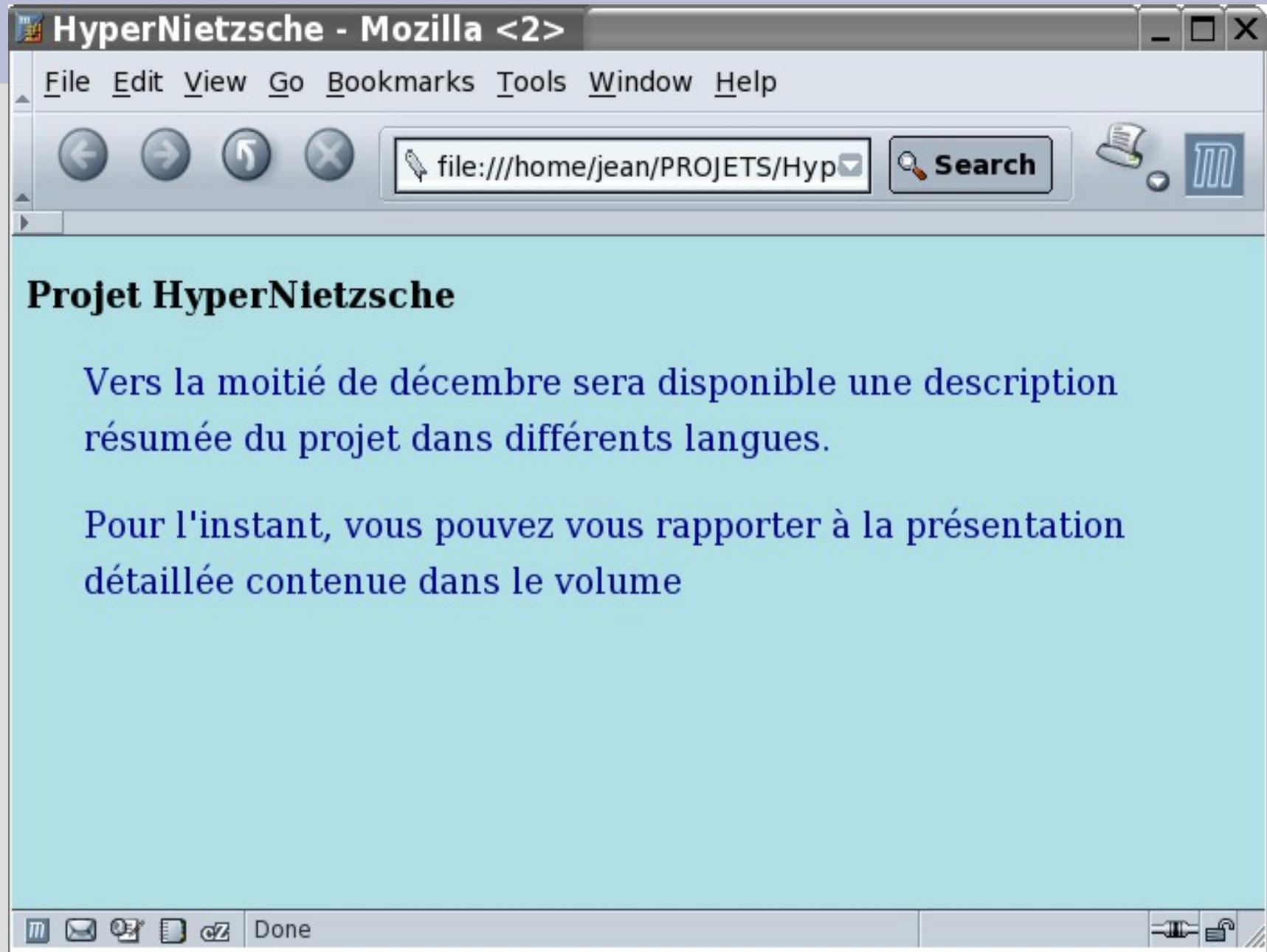
    <DIV align="left">
      <DIV style="margin-left: 2em">
        <P><FONT size="4" color="#00008B">Vers la moiti&eacute; de
          d&eacute;cembre sera disponible une description
          r&eacute;sum&eacute;e du projet dans diff&eacute;rents
          langues.</FONT></P>

        <P><FONT size="4" color="#00008B">Pour l'instant, vous
          pouvez vous rapporter &agrave; la pr&eacute;sentation
          d&eacute;taill&eacute;e contenue dans le volume</FONT></P>
      </DIV>
    </DIV>

  </BODY>
</HTML>
```

Il passaggio da «statico» a «dinamico»

Il codice **HTML** è interpretato da un browser



Il passaggio da «statico» a «dinamico»

Esempio di script PHP

```
<html>
<head> <title>Welcome to HyperNietzsche</title></head>

<body bgcolor="#bfe9b5">
<h1><font size="5">HyperNietzsche</font></h1>

<table border="0" cellpadding="0" cellspacing="2">
<tr>

<? // Récupération des langues utilisables
include("pg_base.php");

$result = pg_query("select * from langue where
    principal and (utilisable or terminé)
    order by alias");

$result_countt=0;
while ($rec = @pg_fetch_array($result,$result_countt++))
{
    $alias = ucfirst($rec["alias"]);
    echo '<td><center><a href="login.php?langue=',
        $rec["code"], '&alias=', $alias, '">',
        $alias, '</a></center></td>';
}
|
?>
</tr> </table>
</body> </html>
```

Il passaggio da «statico» a «dinamico»

Interpretazione dell'html generato da PHP

HyperNietzsche - Mozilla <2>

File Edit View Go Bookmarks Tools Window Help

http://www.hypernietzsche.org/navigate.php?pa Search

Manuscripts

transcriptions

Works

Editions

Authors

Essays

Submit a new contribution

Paths

Rhizomes

hyperNietzsche
(Version 0.6)

What is HyperNietzsche

Deutsch English Español Français Italiano Português

Il passaggio da «statico» a «dinamico»

Esempio di servlet Java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Hello extends HttpServlet
{
    public void doGet (HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException
    {
        PrintWriter out;
        String title = "Example Apache JServ Servlet";

        // set content type and other response header fields first
        response.setContentType("text/html");

        // then write the data of the response
        out = response.getWriter();

        out.println("<HTML><HEAD><TITLE>");
        out.println(title);
        out.println("</TITLE></HEAD><BODY bgcolor=#FFFFFF>");
        out.println("<H1>" + title + "</H1>");
        out.println("<H2> Congratulations, ApacheJServ 1.1.2 is working!<br>");
        out.println("</BODY></HTML>");
        out.close();
    }
}
```

Il passaggio da «statico» a «dinamico»

Interpretazione dell'html generato da Java

```
<HTML><HEAD><TITLE>Example Apache JServ Servlet
</TITLE></HEAD><BODY bgcolor="#FFFFFF">
<H1> Example Apache JServ Servlet </H1>
<H2> Congratulations, ApacheJServ 1.1.2 is working!<br>
</BODY></HTML>
```



Sommario

1. Il passaggio da «statico» a «dinamico»
2. **La correttezza degli script/programmi (codice)**
3. La correttezza del html/xml generato (risultati)
4. L'inadeguatezza del modello di funzionamento (sessioni, cookies)
5. Il problema dello stato globale (basi di dati)
6. Altri problemi e difficoltà

Una scelta critica:

Il linguaggio di programmazione

Costruire un sito dinamico

=> programmare degli algoritmi

=> scegliere un linguaggio di programmazione

- moderno, alto livello di astrazione, codice riutilizzabile, etc
- controlli statici (dinamici è... troppo tardi)

Prime (deprimenti) risposte tecnologiche (fine '90)

- **PHP, ZOPE (Python), ASP,...**
- **imperativi** (basso livello)
- **interpretati** (anziché compilati)
 - inefficienti
 - senza alcun controllo (nemmeno di sintassi!)
- a favore: **librerie** gigantesche

L'esempio dello script PHP

```
<html>
<head> <title>Welcome to HyperNietzsche</title></head>

<body bgcolor="#bfe9b5">
<h1><font size="5">HyperNietzsche</font></h1>

<table border="0" cellpadding="0" cellspacing="2">
<tr>

<? // Récupération des langues utilisables
include("pg_base.php");

$result = pg_query("select * from langue where
principal and (utilisabile or terminé)
order by alias");

$result_countt=0;
while ($rec = @pg_fetch_array($result,$result_countt++))
{
$alias = ucfirst($rec["alias"]);
echo '<td><center><a href="login.php?langue=',
$rec["code"], '&alias=', $alias, '">',
$alias, '</a></center></td>';
}

?>
</tr> </table>
</body> </html>
```

Errori :

- sintattici
- di tipo (semantica statica)
- nei «dialoghi» tra parti dell'applicazione

Una scelta critica:

Il linguaggio di programmazione

Risposte meno deprimenti

- **Java**
 - imperativo, semi-compilato, tipato, oggetti
- **OCaml, Haskell**
 - compilati, tipati, oggetti (OCaml)
 - funzionali
 - programmi concisi
 - tipaggio molto efficace

ma ancora insoddisfacenti...

Lista dei problemi e osservazioni (in costruzione)

(1) correttezza degli algoritmi codificati

→ linguaggio compilato, meglio se funzionale

Sommario

1. Il passaggio da «statico» a «dinamico»
2. La correttezza degli script/programmi (codice)
3. **La correttezza del html/xml generato (risultati)**
4. L'inadeguatezza del modello di funzionamento (sessioni, cookies)
5. Il problema dello stato globale (basi di dati)
6. Altri problemi e difficoltà

Un secondo problema evidente:

Il codice html generato

(a) l'**html** prodotto dinamicamente (da un codice) è lui stesso un **codice** che sarà interpretato da un navigatore

- incontestabilmente codice (Javascript, applet Java,...)
- una **stringa** (parola) di un linguaggio formale (XHTML)
- definito da una DTD (grammatica formale)
- strutturato (sintassi astratta => **albero**)

(b) considerato come semplice stringa da PHP, JAVA,...

- il programmatore costruisce stringhe (tramite l'operazione di concatenazione: echo, out.println), **non alberi**
- nessun controllo (statico) è possibile

Un secondo problema evidente:

Il codice html generato

Eppure si potrebbe :

- utilizzare l'**xhtml** (**html** come dialetto di **xml**)
- costruire e manipolare alberi **xhtml** anziché stringhe (costruttori e distruttori adeguati)
 - **XDuce** (University of Pennsylvania, Paris 7)
 - **CDuce** (ENS-Paris)
 - **Xtatic**, **Scala**
 - approccio funzionale
 - pensarli come sotto-linguaggi (esempio **OCamlDuce**)

Lista dei problemi e osservazioni (in costruzione)

(1) correttezza degli algoritmi codificati

→ linguaggio compilato, meglio se funzionale

(2) correttezza dell'HTML generato

→ manipolare alberi *xhtml* come XDuce e CDuce

Sommario

1. Il passaggio da «statico» a «dinamico»
2. La correttezza degli script/programmi (codice)
3. La correttezza del html/xml generato (risultati)
4. L'inadeguatezza del modello di funzionamento (sessioni, cookies)
5. Il problema dello stato globale (basi di dati)
6. Altri problemi e difficoltà

L'origine delle sessioni

- Perché sessioni o cookies

- **http** è un protocollo di rete **senza stato**
- il problema è che spesso bisogna **conservare lo stato** della conversazione con l'utente (esempi: lingua, carrello della spesa)
- **problemi semantici** (p.e. senso dei pulsanti *back*, *forward*, *clone*)
- **problemi di programmazione**
 - il programmatore deve preoccuparsi di trasmettere certi dati (link, form, session, cookies) dalla costruzione di una pagina all'altra
 - deve **programmare** l'analisi dello stato per “riprendere il filo” del dialogo interrotto
 - più cerca di semplificarsi la vita “fattorizzando” le pagine da produrre, più complica la fase di analisi del dialogo interrotto

L'origine delle sessioni

- Nel modello di calcolo tradizionale
 - il ciclo di vita tipico di un programma
 - il programma inizia (è caricato in memoria)
 - calcola, interagisce (con l'utente o altri programmi)
 - restituisce un risultato (per esempio una stringa contenente html)
 - termina
 - una seconda esecuzione non ha un rapporto stretto con la precedente
 - le sessioni o cookies possono essere viste come dei «patch» di questo schema di funzionamento tradizionale
 - si **rattoppa** uno schema inadeguato (vedi problemi semantici)

Lista dei problemi e osservazioni (in costruzione)

(1) correttezza degli algoritmi codificati

→ linguaggio compilato, meglio se funzionale

(2) correttezza dell'HTML generato

→ manipolare alberi *xhtml* come XDuce e CDuce

(3) principio di funzionamento inadeguato

→ pensare ad un ciclo di vita non classico

Sommario

1. Il passaggio da «statico» a «dinamico»
2. La correttezza degli script/programmi (codice)
3. La correttezza del html/xml generato (risultati)
4. L'inadeguatezza del modello di funzionamento (sessioni, cookies)
5. **Il problema dello stato globale (basi di dati)**
6. Altri problemi e difficoltà

Il problema dello stato globale

Un quarto problema:

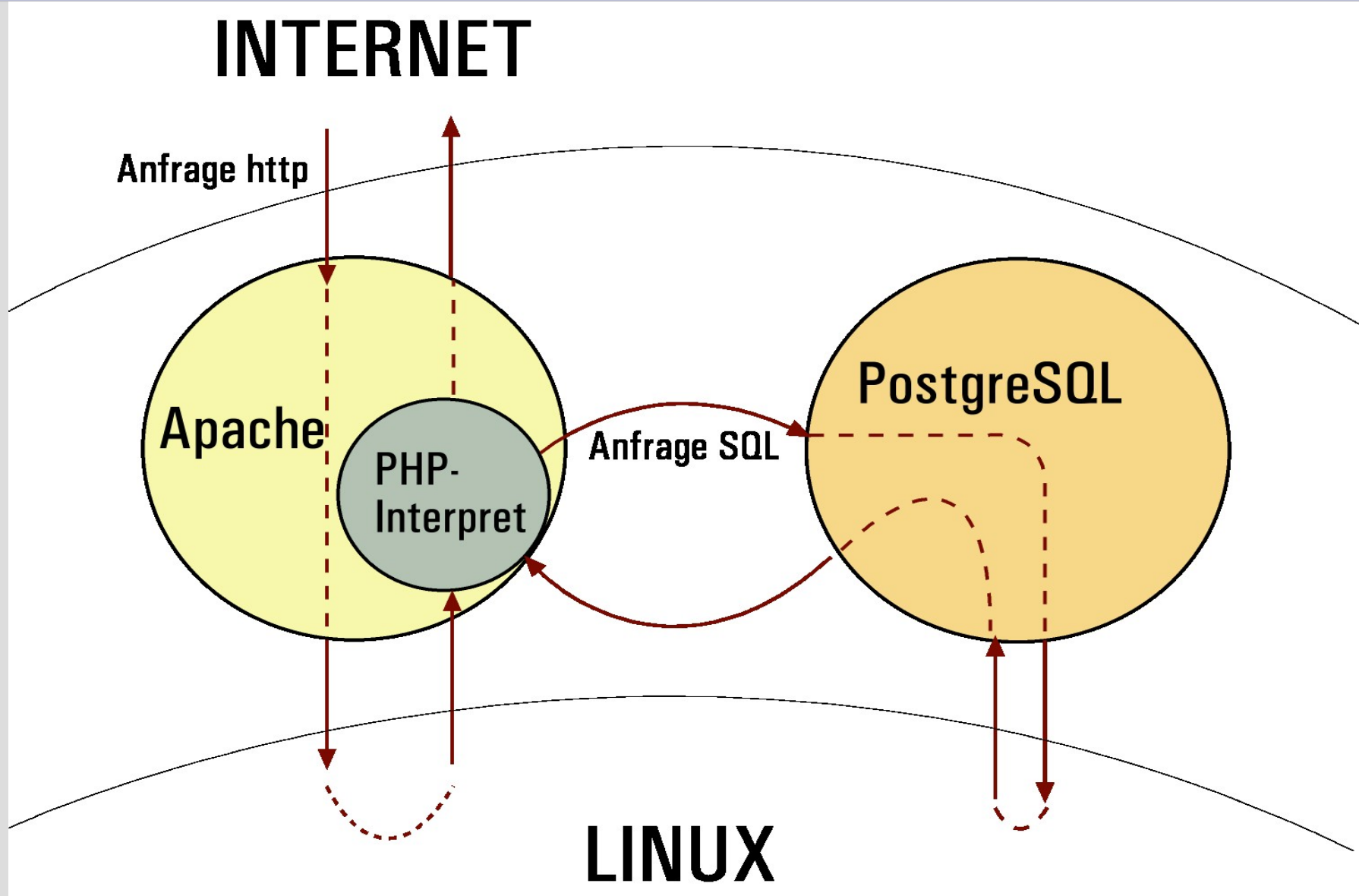
Le basi dati

L'esecuzione del programma può necessitare l'interrogazione di una base dati (semplici file, MySQL, PostgreSQL, Oracle, etc)

- esiste cioè uno «**stato globale**» condiviso da tutti gli utenti (e che tutti contribuiscono a far evolvere)
- le interrogazioni possono essere multiple per una stessa pagina => problema di **scalabilità**
- il **modello relazionale** può non essere il più adatto

HyperNietzsche => troppe query per ogni pagina

La struttura tipica di una applicazione web



L'esempio di script PHP con query

```
<html>
<head> <title>Welcome to HyperNietzsche</title></head>

<body bgcolor="#bfe9b5">
<h1><font size="5">HyperNietzsche</font></h1>

<table border="0" cellpadding="0" cellspacing="2">
<tr>

<? // Récupération des langues utilisables
include("pg_base.php");

$result = pg_query("select * from langue where
    principal and (utilisable or terminé)
    order by alias");

$result_countt=0;
while ($rec = @pg_fetch_array($result,$result_countt++))
{
    $alias = ucfirst($rec["alias"]);
    echo '<td><center><a href="login.php?langue=',
        $rec["code"], '&alias=', $alias, '">',
        $alias, '</a></center></td>';
}
|
?>
</tr> </table>
</body> </html>
```

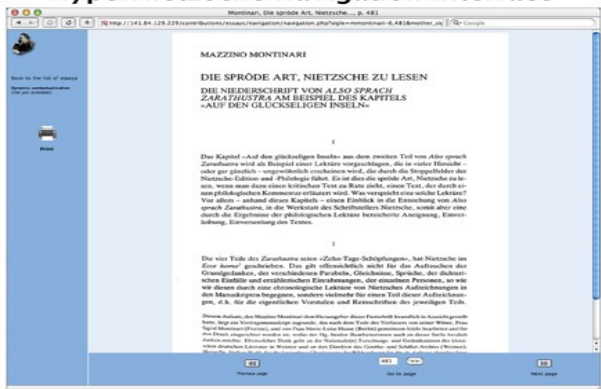
Il problema dello stato globale

Un quarto problema:

Le basi dati

- Applicazione spezzata in due parti
 - Problemi di correttezza
 - Comunicazioni a base di stringhe generate **dinamicamente**
 - Nessun controllo statico delle interrogazioni
 - Nessun controllo statico sull'uso delle risposte
 - Conversioni da implementare
 - Problemi di efficienza
 - Dialoghi costosi
 - Diverse interrogazioni per una stessa pagina

HyperNietzsche Navigation Interface

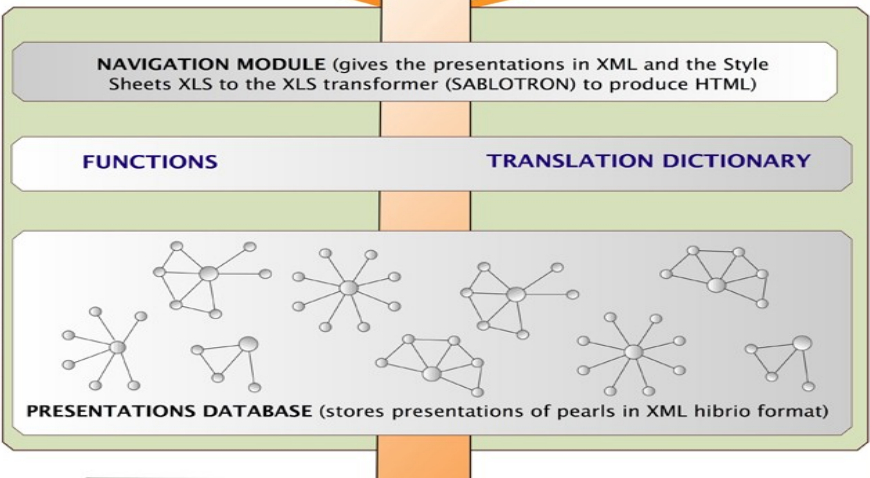


HyperNietzsche Submission Interface

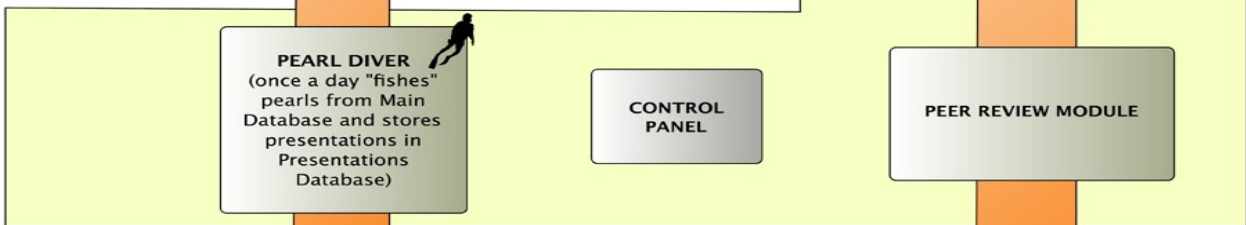


Il problema dello stato globale L'ingranaggio HyperNietzsche

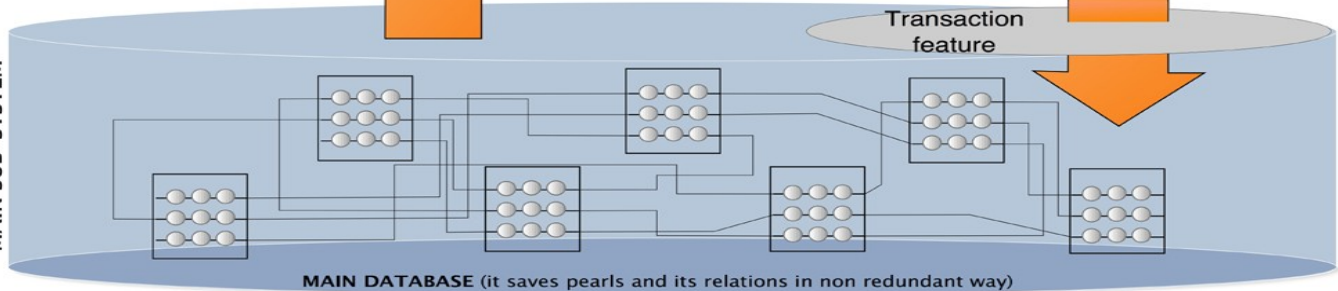
PRESENTATIONS SUB-SYSTEM



ADMINISTRATION SUB-SYSTEM



MAIN SUB-SYSTEM



- un sistema di caching pesante
- la rivincita dello statico!
- il modello relazionale non pare il più adatto

Lista dei problemi e osservazioni (in costruzione)

- (1) correttezza degli algoritmi codificati
 - linguaggio compilato, meglio se funzionale
- (2) correttezza dell'HTML generato
 - manipolare alberi *xhtml* come XDuce e CDuce
- (3) principio di funzionamento inadeguato
 - pensare ad un ciclo di vita non tradizionale?
- (4) gestione dello stato globale (BD)
 - integrazione in un linguaggio funzionale?

Sommario

1. Il passaggio da «statico» a «dinamico»
2. La correttezza degli script/programmi (codice)
3. La correttezza del html/xml generato (risultati)
4. L'inadeguatezza del modello di funzionamento (sessioni, cookies)
5. Il problema dello stato globale (basi di dati)
6. **Altri problemi e difficoltà**

Altri problemi e difficoltà

1. un solo programma per tutto il sito

- anziché un'applicazione spezzettata in una moltitudine di «moduli» (script/programmi)
 - difficile da **mantenere**
 - il cui rapporto è da ricercare nei link prodotti **dinamicamente**
 - poco chiaro semanticamente

2. una semantica chiara

- del rapporto causale tra le fasi di produzione delle pagine (le «tappe»)
- di come i dati «viaggiano» da una tappa all'altra

Altri problemi e difficoltà

3. problemi di **cooperazione** tra :

- il **programmatore** (algoritmica)
- il **web designer** (navigazione, mappa del sito)
- il **grafico** (interfaccia e look delle pagine)

4. possibile generalizzare siti web e **servizi web**?

- xml al posto di xhtml (semplice cambiamento di DTD!)
- possibile programmarli allo stesso modo?

Lista dei problemi e osservazioni CONCLUSIONE

- (1) correttezza degli algoritmi codificati
 - linguaggio compilato, meglio se funzionale
- (2) correttezza dell'HTML generato
 - manipolare alberi *xhtml* come XDuce e CDuce
- (3) principio di funzionamento inadeguato
 - pensare ad un ciclo di vita non tradizionale?
- (4) gestione dello stato globale (BD)
 - integrazione in un linguaggio funzionale?
- (5) unico programma, semantica formale, cooperazione, servizi web
 - da inventare