

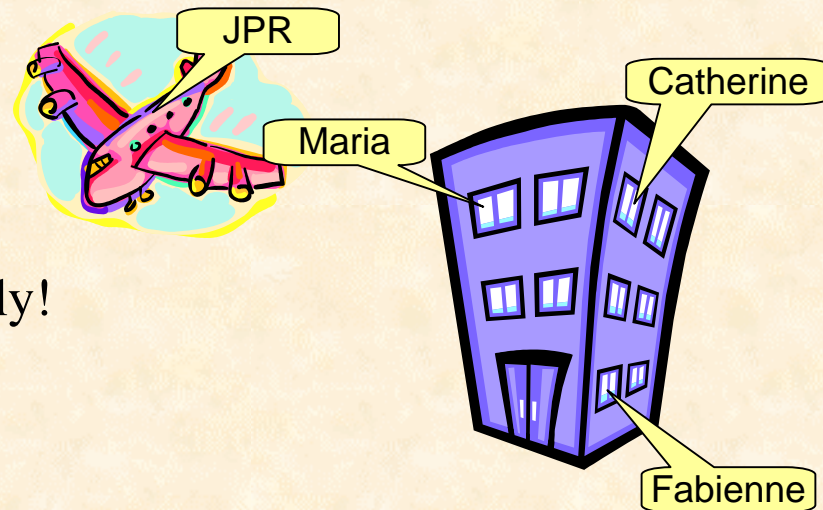
Experiences in developing a typical Web/Database application

J-P. Rosen
Adalog
An AXLOG Group company

rosen@adalog.fr

The need

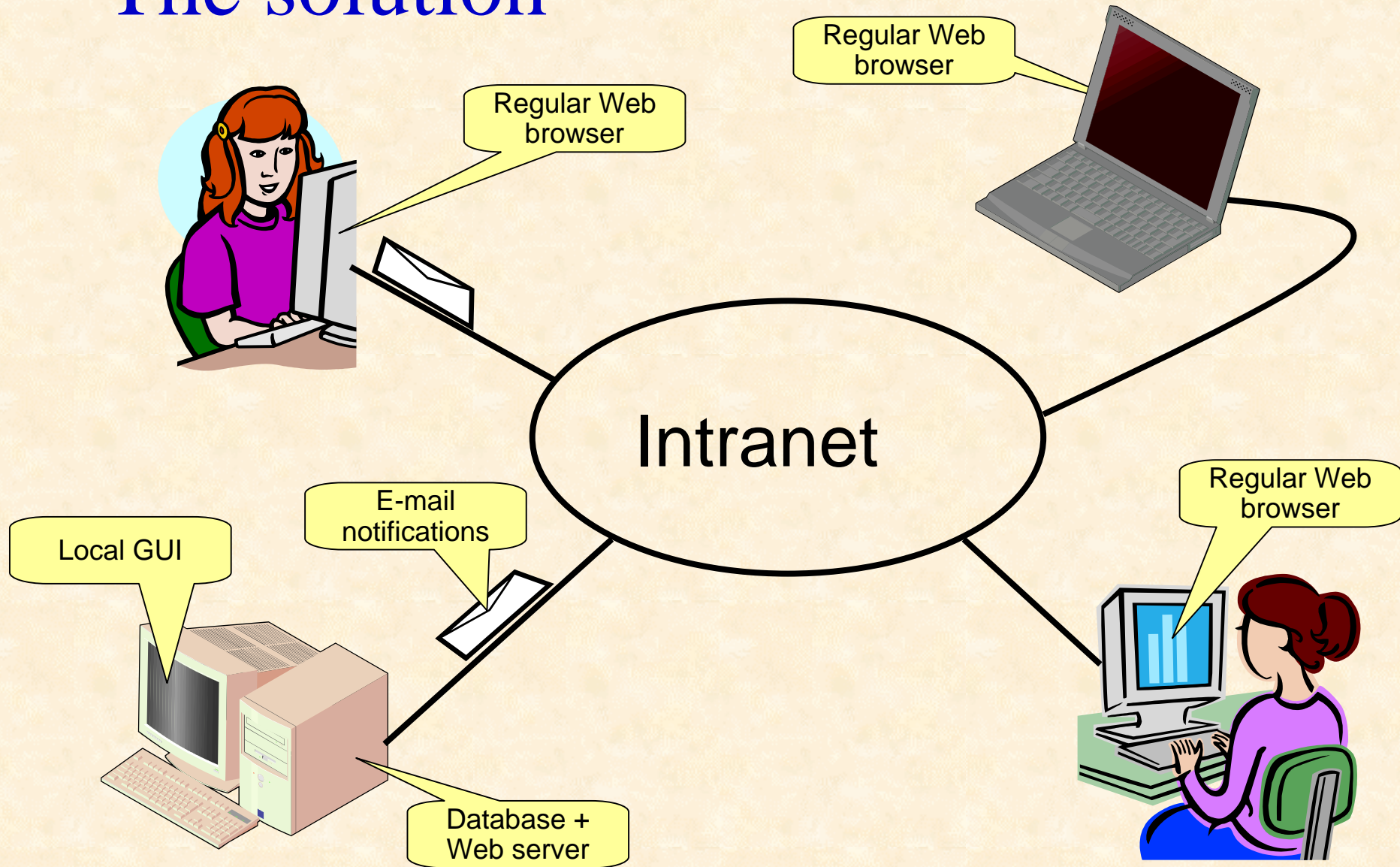
- Managing the registration process to Adalog's training sessions
 - + Several persons in charge
 - + In various locations, not available at the same times
 - + Must answer the phone immediately!
- Pinging people
 - + Prepare hand-outs
 - + Reserve restaurant
 - + ...
- Managing mailing
 - + Classical database extraction



The constraints

- Use free software
- User interface usable by casual users
- Availability on Windows and Linux
- Independent of any particular DBMS
- Easily modifiable
- Deal with concurrent accesses
- Efficiency *is not* a concern
- Reliability *is* a concern

The solution



AWS

- Ada Web Server (Pascal Obry)
 - + A set of packages for managing http/https connections
 - + Facilities for managing pages
 - + Facilities for building pages
- Basic behaviour

```
procedure Start (Web_Server : in out HTTP;  
                 Callback    : in      Response.Callback;  
                 Config      : in      AWS.Config.Object);  
  
type Callback is access  
  function (Request : Status.Data) return Response.Data;
```

AWS - Dispatchers

- A call-back procedure appropriate to the basic behaviour
- Analyzes the request and:
 - + Dispatches to other functions according to the URI (URI dispatcher)
 - + Considers the URI as a file name and returns the corresponding file (Page dispatcher)
 - + Dispatches to other functions according to the method (Method dispatcher)
 - + Dispatches to other functions according to the host name (virtual host dispatcher)

AWS – Templates parser

- Tags and Vector-Tags

- + Tag: associates a value to a name (string)
- + Vector-tag: associates several values to a name (string)
- + Referred to in template as `@ @_name_@`

- Conditions

```
@@IF@@ <condition>  
...  
@@END_IF@@
```

Content included only if <condition>
is true

- Tables

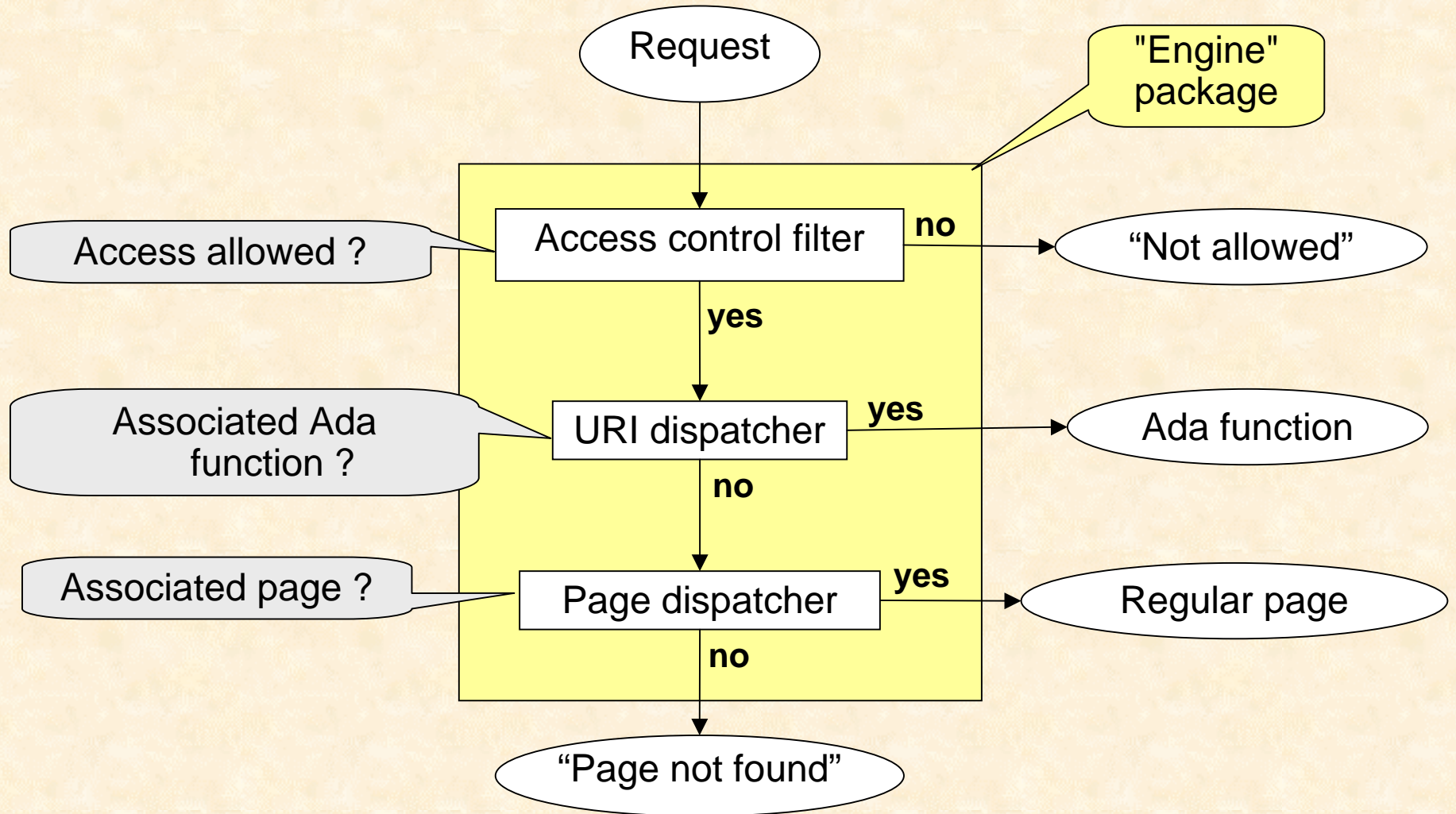
```
@@TABLE@@  
...  
@@END_TABLE@@
```

Repeat contents, picking up one value
from each vector-tag for each iteration

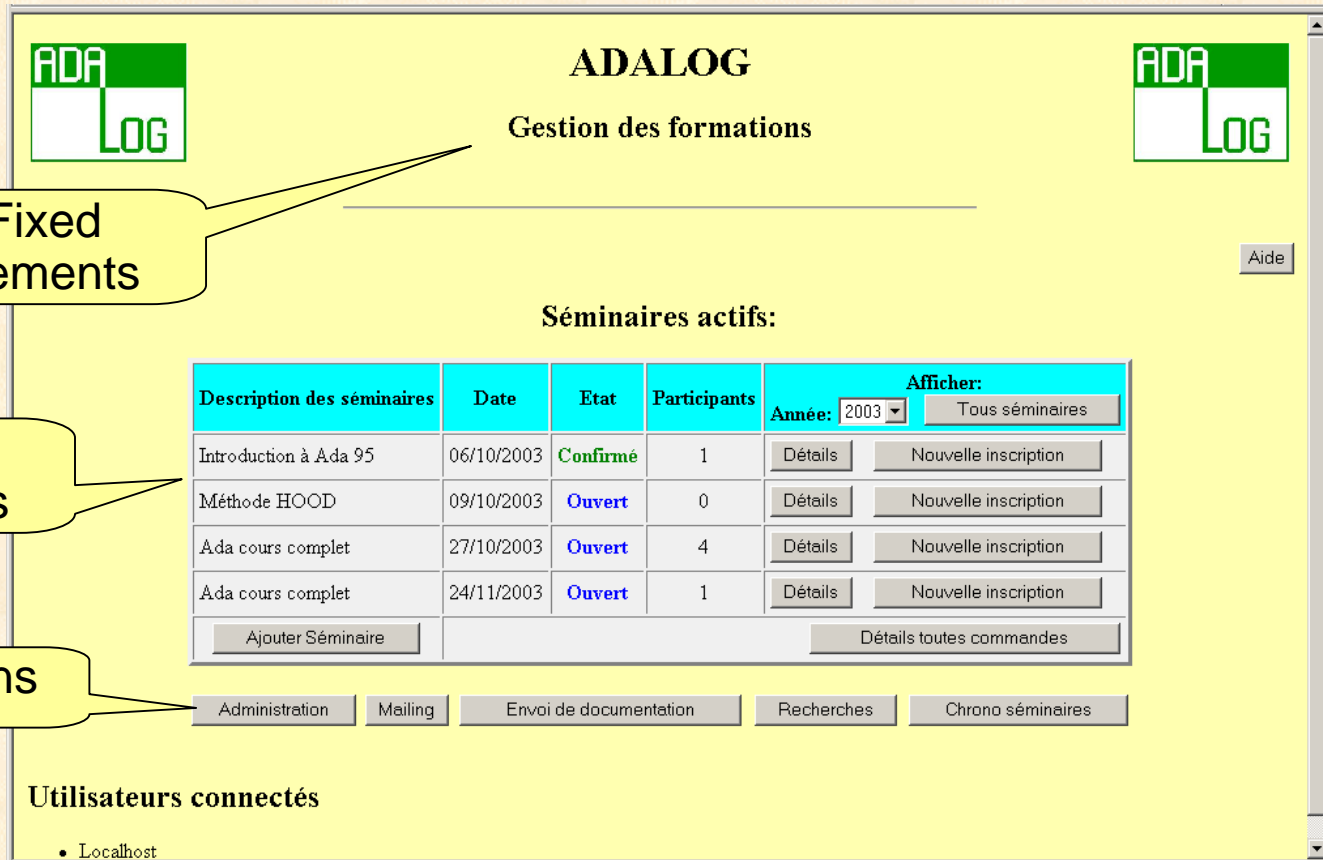
AWS – Other features

- Session management
- Mail client
- HTTP client
- Server push
- SOAP support (server and client)
- LDAP support
- JABBER support
- ... and more

Gesem filters and dispatchers



A typical page



ADALOG
Gestion des formations

Séminaires actifs:

Description des séminaires	Date	Etat	Participants	Afficher:	
				Année: 2003	Tous séminaires
Introduction à Ada 95	06/10/2003	Confirmé	1	Détails	Nouvelle inscription
Méthode HOOD	09/10/2003	Ouvert	0	Détails	Nouvelle inscription
Ada cours complet	27/10/2003	Ouvert	4	Détails	Nouvelle inscription
Ada cours complet	24/11/2003	Ouvert	1	Détails	Nouvelle inscription

Ajouter Séminaire Détails toutes commandes

Administration Mailing Envoi de documentation Recherches Chrono séminaires

Utilisateurs connectés

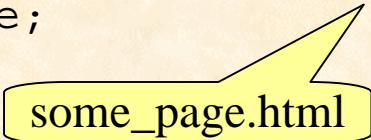
- Localhost

The page design pattern

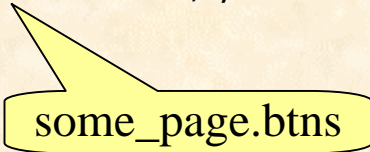
```
with AWS.Response;  
package Pages.Some_Page is  
  function Build (<parameters>) return AWS.Response.Data;  
end Pages.Some_Page;
```

```
package body Pages.Some_Page is  
  My_Name : constant String := "some_page";  
  
  function Build (<Parameters>)  
    return Response.Data is ...  
  
  function Buttons (Request : in AWS.Status.Data)  
    return Response.Data is ...  
  
  function Page (Request : in AWS.Status.Data)  
    return Response.Data is ...
```

```
begin  
  Engine.Register(My_Name, Page'Access, Buttons'Access);  
end Pages.Some_Page;
```



some_page.html



some_page.btns

Reliability

- Every page has an exception handler:

exception

```
when Occur : others =>
```

```
  return Pages.Error.Build
```

```
    (Unit          => "pages." & My_Name,
```

```
     Subprogram   => "Name of subprogram",
```

```
     Occur        => Occur);
```

ADALOG

Erreur du programme

Aide

Unité : Pages chrono
Sous-Programme : Page
Exception : CONSTRAINT_ERROR
Message : pages-chrono.adb.29 explicit raise

Une erreur s'est produite dans le programme.
Merci de donner ci-dessous la description précise des manipulations qui ont abouti à ce problème.
Les informations figurant ci-dessus seront automatiquement jointes au message, il est inutile de les répéter

Envoyer le rapport d'anomalie Annuler

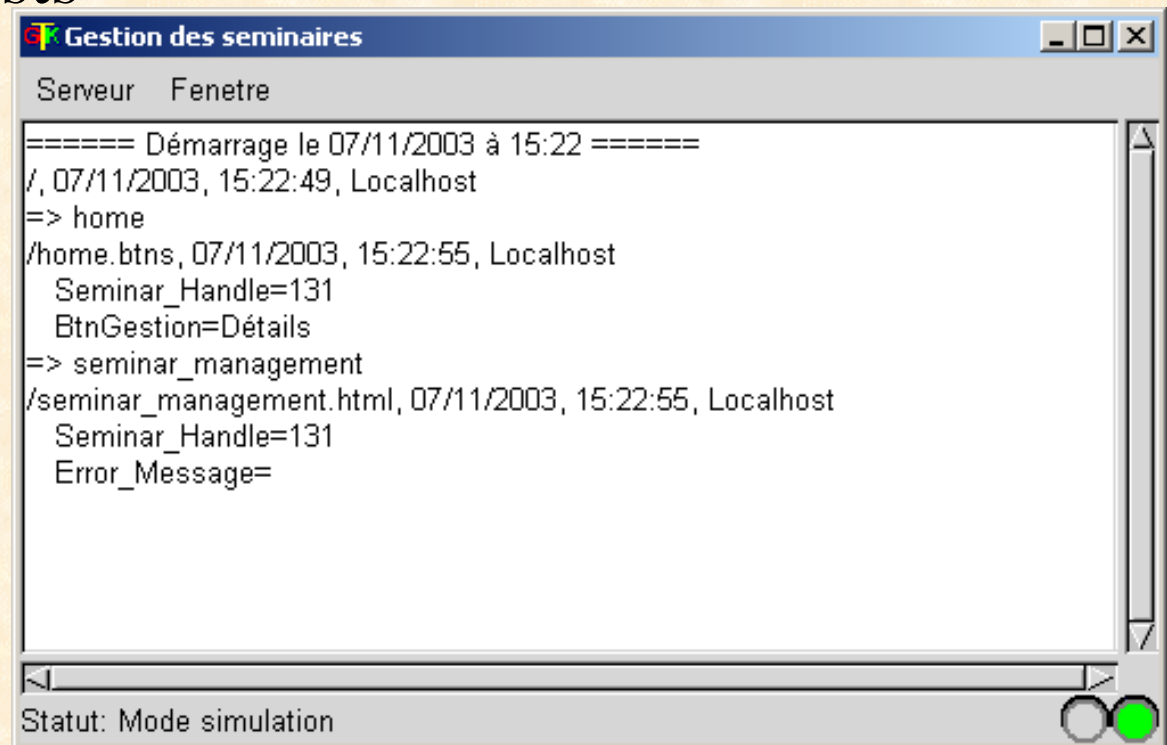
Concurrency

- Concurrent access is extremely unlikely, but possible
 - + Recognize users from their IP address
 - + Use a global lock:
 - Only one user can modify at any one time
 - "Modify" button on each page to grab the lock
- But beware of "back" button
 - + Display a page
 - + Modify it (get lock)
 - + Validate (release lock)
 - + Back page: the page is modifiable, but the user doesn't own the lock !
 - + Checked by the access control filter => page expired

Local interface (1)

- Manages the application
 - + Stop, lock database...
 - + Shows uncommitted transactions
- Monitors requests
 - + Clear window
 - + Save content to file

- Plain GTK
- Generated automatically with GLADE



```

Gestion des seminaires
----- Démarrage le 07/11/2003 à 15:22 -----
/, 07/11/2003, 15:22:49, Localhost
=> home
/home.btns, 07/11/2003, 15:22:55, Localhost
  Seminar_Handle=131
  BtnGestion=Détails
=> seminar_management
/seminar_management.html, 07/11/2003, 15:22:55, Localhost
  Seminar_Handle=131
  Error_Message=

Statut: Mode simulation
  
```

Local interface (2)

- Problem: GtkAda allows only one task in the GUI
 - + But many tasks need to update the interface
- Solution: the main task manages the interface
 - + Usual pattern, terminate program when exiting main loop
 - + Updates are messages, queued in a FIFO
 - + How do you notify that a new message has arrived?
 - Avoid active loops
 - Schedule an idle-callback

Choosing the database interface

- Gate
 - + Didn't like embedded SQL
 - + Drives other Ada tools (emacs!) crazy
- mySQL binding
 - + Fear to depend on any special database system
- mySQL + ODBC
 - + DB independent
 - + Allows other applications to access DB
 - + Available for both Linux and Windows
 - + Somewhat low level, but easy to use a higher level binding.

Objects design pattern

```
with Globals, Data_Manager, AWS.Templates;  
use Globals;  
package Objects.Abstraction is
```

```
type Data is  
  record
```

```
    ..
```

```
  end record;
```

```
  -- Operations on Abstraction.Data
```

```
function Image (Item : Data) return Array_Of_Unbounded;
```

```
function Value (Item : Array_Of_Unbounded) return Data;
```

```
package Manager is new Data_Manager
```

```
  (Data      => Data,
```

```
   Data_Name => "my_data",
```

```
   Columns   => "col1, col2, col3");
```

```
subtype Handle is Manager.Handle;
```

```
type List is array (Positive range <>) of Handle;
```

```
function Associations (Item : Handle) return Translate_Table;
```

```
function Associations (Item : List)   return Translate_Table;
```

```
function Extract (Param : AWS.Parameters.List) return Data;
```

```
end Objects.Abstraction;
```

Ada
view

Database
view

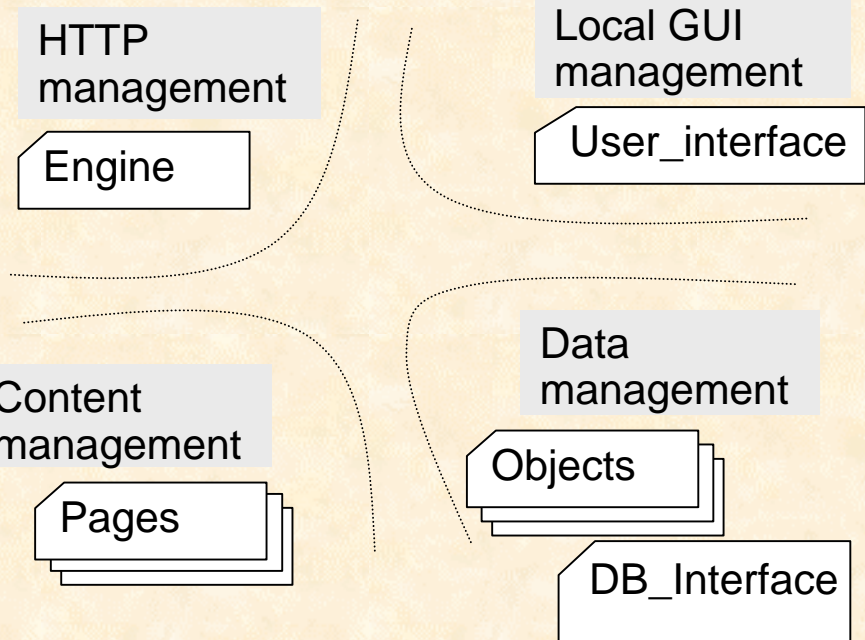
Templates
(HTML) view

AWS vs. Apache

- The application is an executable, not a set of scripts
 - + Must recompile when fonctionnalities are added/changed
 - + NOT when presentation changes (thanks to templates)
- Easy to deal with concurrent access
 - + Thanks to protected types!
- Possibility of having a control panel
- Easy to distribute
 - + Can even be made a single executable

Lessons learned (1)

- **Separate concerns**



- **Reliability**

- + Exceptions are great!

- **AWS is powerful enough**

- + No Javascript

- + The template parser is great!

Lessons learned (2)

- A web interface is difficult to manage
 - + User can close the browser at any time (even with uncommitted transactions), but the application is not aware!
 - + User can call "previous page" at any time: no global state
- Portability
 - + > 10_000 SLOC in 81 compilation units
 - + Network interface + GUI + Database interface
 - + **No** difference between Linux and Windows version
 - + Ada is great!



Questions