WEB DATABASE PUBLISHING

1. Basic concepts of WEB database publishing (WEBDBP)
2. WEBDBP structures
3. CGI – concepts
4. Cold Fusion
5. API - concepts
6. Structure of Oracle Application Server
7. Listeners
8. ORB - WRB
9. Cartridges
10. DAD
11. PL-SQL cartridge

Basic concepts of WEB database publishing

Merging of two current technologies:
internet/intranet publishing + database management

some benefits of internet/intranet publishing:

- no special investment required for clients
- universal interface
- wide area access
- platform independence

some benefits of database management

- large amount of data
- integrity checking
- flexibility

some drawbacks:

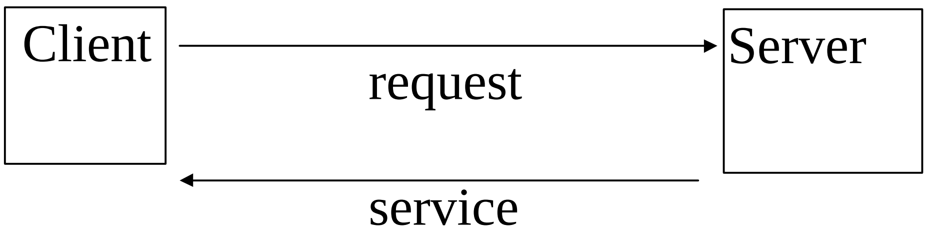
- not the fastest solution
- not every data types can be managed
- security difficulties

key words:

WEB	HTML	HTTP	CGI	API
SQL	ODBC	JDBC	Java	CGI
API	Transaction	Middleware		Internet
Intranet	Corba			

WEBDBP structures

It is based on the client-server technology



This basic structure is called *2-tier structure*

The main basic communication modes:

- RPC, remote procedure call
- Message-based

Logical functional components:

- Presentation Services
- Presentation Logic
- Business Logic
- Distribution Services
- Database Logic
- Database Services
- File Services

The 2-tier structures

Fat clients: Most of the functional modules of the application are performed on the clients

Lite clients: Only few functional modules of the application are performed on the clients

Remote Data Client-Server Architecture

Client		Server
Presentation		Data services
Business logic	⇔	File services
Data logic		

Remote Presentation Client-Server Architecture

Client		Server
Presentation		Business logic
	⇔	Data logic
		Data & file services

Split Logic Data Client-Server Architecture

Client		Server
Presentation		Business logic
Business logic	⇔	Data logic

Data logic Data & file services

The 3-tier Structures

Usual distribution:

1. tier: Presentation logic, lite client
2. tier: Business logic, application server
3. tier: Database logic, database server

The 3-tier structure can be extended to n-tier structure, containing several special application servers

The WEBDBP is based on the 3-tier client-server model

The three layers:
1. tier : the client, it is a browser
	the browser is connected to the web-server
2. tier : the web-server
	it is connected to both the browser as to the database
	it can pass static and dynamic HTML documents to the clients
3. tier : the database - server
	it is connected to the web server

A special module is required to create a connection between the database and the web server

Processing of a user-request

1. user sends a request from the browser to the web server
2. the web server detects that a database access is required
3. the web server passes control to the server extension modul
4. the extension modul formulates a command for the database
5. the extension modul sends a request to the database server
6. the database server processes the request
7. the database server sends the result back to the extension program
8. the extension modul generates a result page
9. the result page is transfered back to the web server
10. the web server sends the result page to the client

The web server can processes several requests parallel

The database server can processes several requests parallel

Mixed WEB Database Systems

It is based on the 2-tier structure instead of the 3 tier structure.

The browser downloads a database client program and executes it as a client site extension module. This database client uses a session-oriented protocol with the database server. This communication channel is independent from the web server.

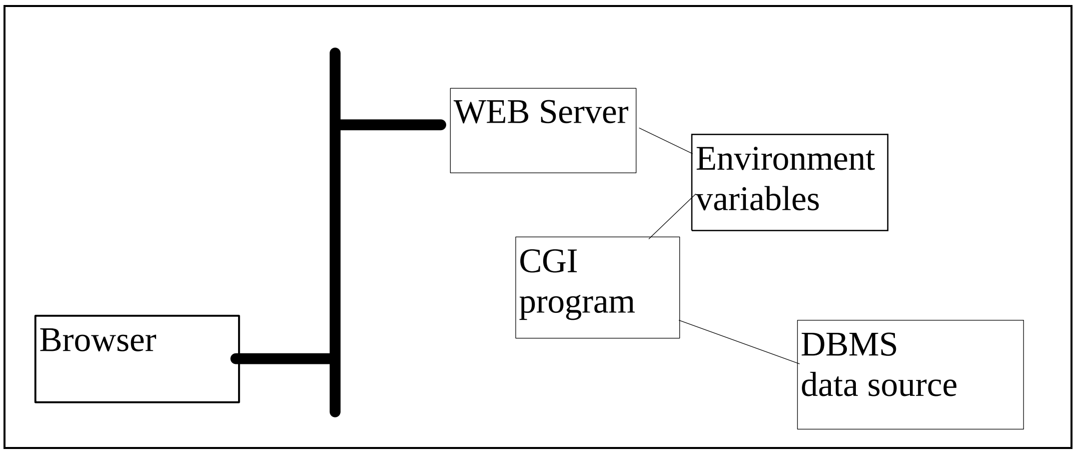
```
Browser                      ---   web server
client database application ---   database server
```

The client database program is based on the usual interface. It can perform the usual input validations.

CGI - concepts

The WEB Server can access and span an external program that can perform among others database manipulation too. The output of the external program is posted back to the Web Server.

The Web Server communicates with a CGI program through environment variables and through the operating system's standard input/output.



You can develop CGI programs in many languages but PERL, C, shell scripts are the most common.

Data transfer to the CGI program

Two methods of the data transfer:

GET method
POST method

In the case of GET method the program receives the data in the QUERY_STRING variable. The program parses this variable to get the values.

In the case of POST metyhod, the parameters are passed through the standard input chanel. The program reads data from the stdin.

Drawbacks of the CGI-DB solution:

- every connection too the data source, DBMS needs a new process, new allocations
- CGI programs lacked any method for co-operating with each others

COLD FUSION

Cold Fusion is a database application development tool that enables the rapid creation of dynamic, database WEB applications.

Cold Fusion is based primary on the CGI concepts. It allows developing schemas, so called templates of HTML pages to describe the database connection at a higher level.

The database connection is described with the special HTML-DBML language instead of a low-level procedural language.

Benefits:

- no code required
- based on database connection standards
- flexible

it allows

- connection to data sources via ODBC
- querying the database
- updating, inserting, deleting data
- validation rules for field entries
- formatting of field entries
- implementing conditional statements
- dynamic data manipulations

- variable handling

Structure of the Cold Fusion

client:

browser

request html page

server:

web server

cgi connection to cold fusion server

connection:

cold fusion server reads the given template file

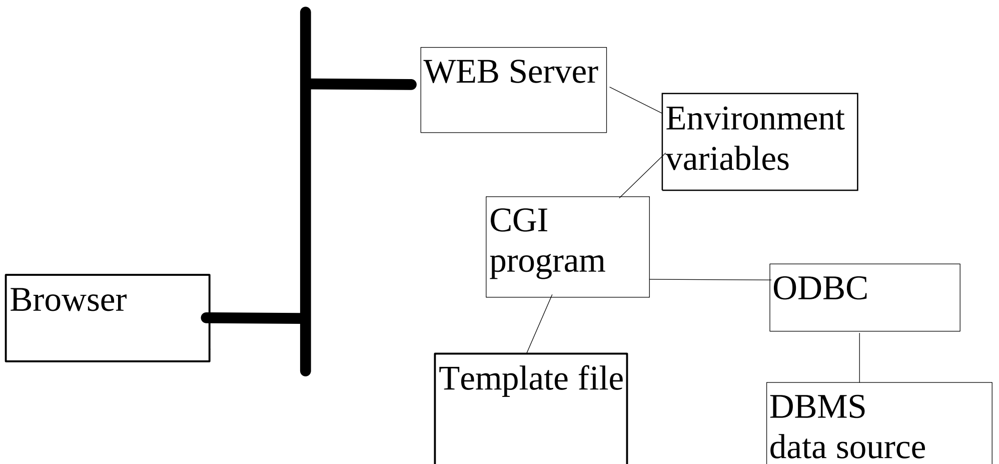
performs the actions given in the template file

connection to data sources vie ODBC

performing data manipulation

generating the result page

web server passes the result page to the browser



Cold Fusion Components

1. Request HTML page
2. Template files
3. ODBC data sources

Request HTML pages

referencing to the cold fusion server:

URL = node:/cgi-path/dbml.exe?Template=tname&…

where

dbml.exe - name of the cold fusion server

tname - name of the template file

the POST parameter passing method is required

```
<FORM ACTION="…/cgi-path/dbml.exe?Template=in1.dbm"
METHOD=POST>
…..
<INPUT TYPE=SUBMIT …..>
</FORM>
```

The fields of the form are passed to the cold fusion server

Components

Template files

It combines normal HTML and DBML commands

<command parameter-list>

Special DBML commands

Insertion of new record:

<DBINSERT DATASOURCE = dsn TABLENAME = table … >

dsn : ODBC DSN identifier

table : table name

the cold fusion detects the field structure of the table and stores the parameter values into the fields of the new records

html page field and database table field are connected to each others upon the name (both must have the same name)

Components

Updating table records

<DBINSERT DATASOURCE = dsn TABLENAME = table …>

dsn : ODBC DSN identifier
table : table name

the cold fusion detects the field structure of the table and stores the parameter values into the fields of the new records
The table must have a key and there must exist parameter values for the key and the fields to be modified.

Creating Queries

<DBQUERY NAME = name DATASOURCE = dsn SQL = query …>

name: query identifier
dsn : ODBC DSN identifier
query : SQL SELECT command

the SELECT command can refer to dynamic parameters

dynamic parameter:

- html form parameter
- URL parameter
- CGI environment information

Query in ColdFusion

reference to a dynamic variable :

```
        #varname#
```

```
    SQL = " SELECT * FROM Books WHERE
price > #iprice# "
```

Displaying output values on the result page

```
    <DBOUTPUT QUERY = name …>
        display format description
    </DBOUTPUT>
```

The format description section may contain dynamic field variables that correspond to the fields of the result table.

field reference : #queryname.fieldname#

the output is generated for every records of the result table.

```
    <DBOUTPUT QUERY = phone …>
        <HR>
        #phone.firstname# #phone.lastname#
Phone: #phone.phone# <BR>
    </DBOUTPUT>
```

Query in ColdFusion

Presenting the results in tabular formats

```
	<DBTABLE QUERY = name >
		<DBCOL HEADER = heading1 TEXT = #field1# …>
		<DBCOL HEADER = heading2 TEXT = #field2# …>
		….
	</DBTABLE>
```

Basic input field validation and formatting for input field fname

- requiring entry

```
<INPUT TYPE=HIDDEN NAME=fname_required>
```

- simple type checking

```
<INPUT TYPE=HIDDEN NAME=fname_type>
```

- range checking

```
<INPUT TYPE=HIDDEN NAME=fname_range
```

```
        VALUE="MIN=v1 MAX=v2">
```

Template control flow

the cold fusion allows conditional processing of statements

```
    <DBIF value operator value >
        html part A
    <DBELSE>
        html part B
    </DBIF>
```

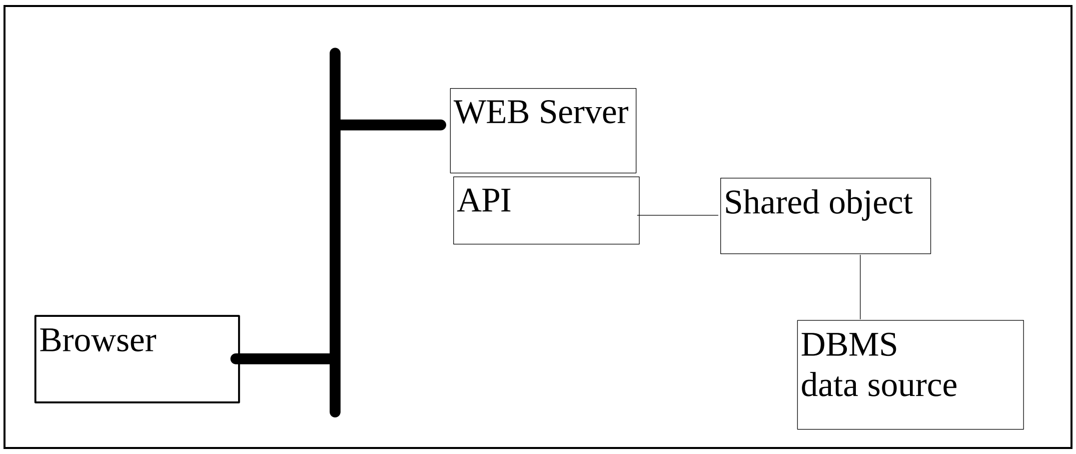
Sending SMTP mails:

```
    <DBMAIL TO = to FROM = from SUBJECT
= subject …>
        letter text
    </DBMAIL>
```

The letter text may contain dynamic variables too.

API - concepts

In this structure the server exrension program is implemented as a part of the server, as a DLL or a shared object



Server API’s are the methods that you can use to extend the server functionality.

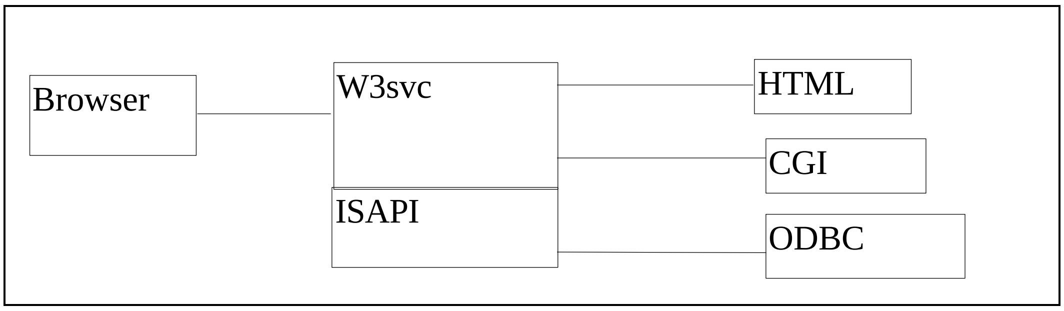
- it is faster than CGI

API’s are specific for the WEB server.

- NSAPI   Netscape
- MS ISAPI Microsoft
- ….

It is strongly coupled with the WEB server

MIIS HTTP-ODBC structure



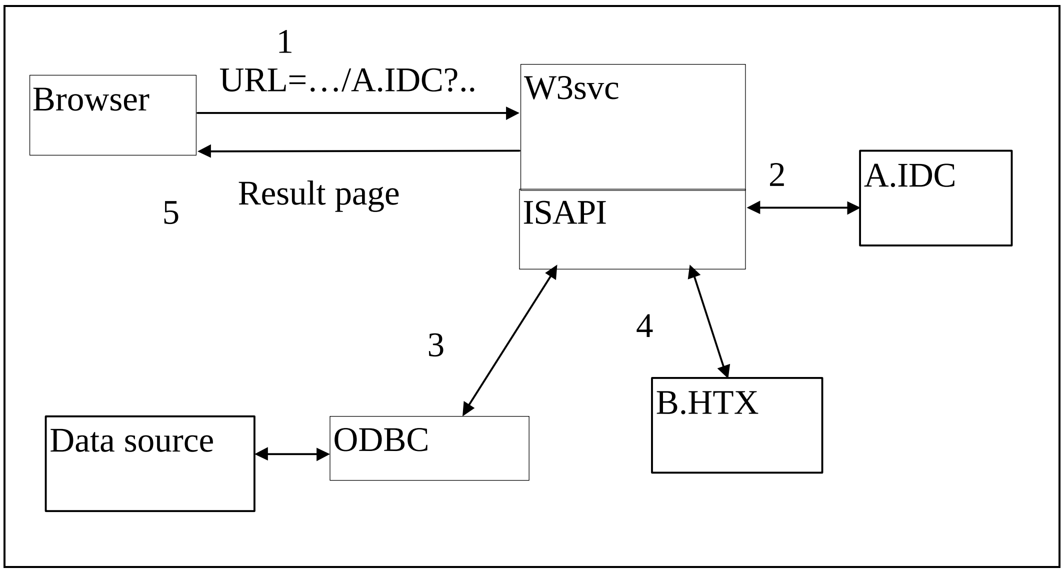
Two description files

Data Source description

.IDC

Template for display the result

.HTX



MSIIS ISAPI elements

IDC parameters

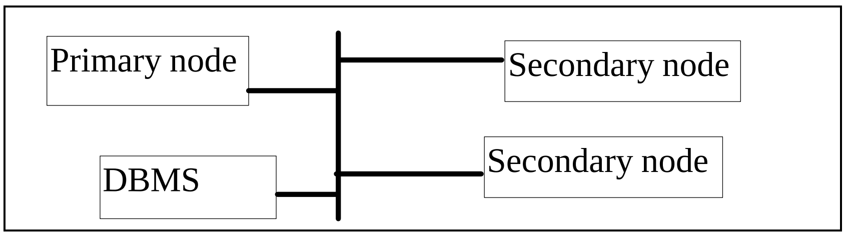
- data source (ODBC DSN)
- template file identifier
- SQL Statement

Template elements

- normal HTML commands
- special HTML commands:
    - <%begindetail%>……..<%enddetail%>
      loop on the result record set
    - <%name%>
      parameter, data field value
    - <%if%>……..<%endif%>
      conditional statements

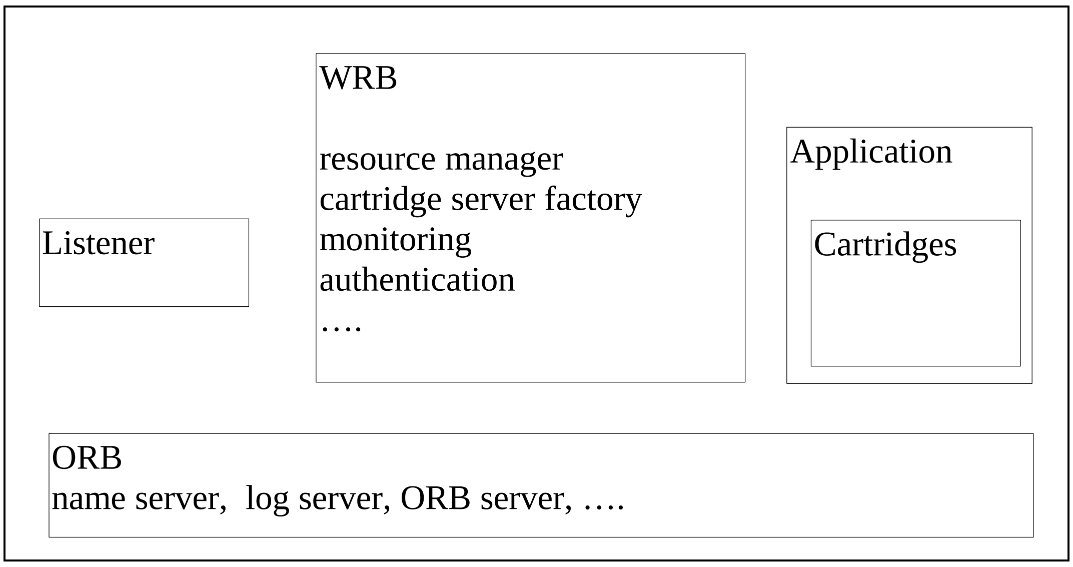
Structure of Oracle Application Server

Effective connection to the Oracle DBMS
Easy, flexible programming tool

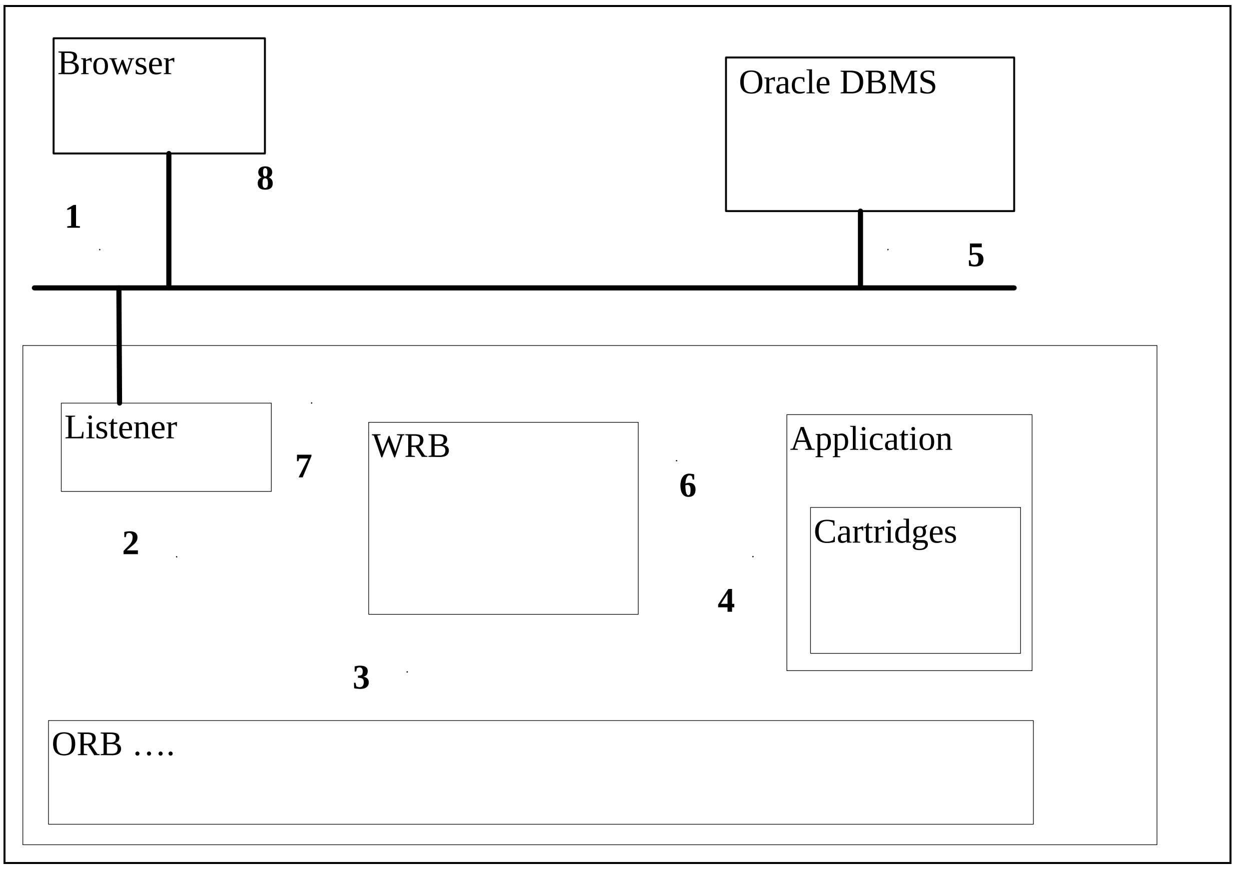


primary node : it contains the WRB, listeners and cartridges
secondary node : it contains some of the components mentioned above

Components of the OAS



Steps of operation



Request			
1.	browser	2.	listener
3.	WRB(ORB)	4	cartridges
5	DBMS		
Result			
5	DBMS	6	cartridges
7	WRB (ORB)	8	listener
9	browser		

Application Programming Models

An application can include several instances of the same cartridge
The work is distributed among the instances, the load is balanced

- request-response model : each client request is processed individually.
  two subsequent request from the same client may be responded by two different instances

- session model : subsequent requests from the same client are handled by the same instance. the connection breaks after a given amount of time-out.

- transaction model : all requests from the same URL are processed within the context of the same transaction. The client can send commit or roll-back messages.

Security schemes

Authentication schemes:

- basic authentication:
  users, password management,
  protection of files, directories
- digest authentication:
  similar to basic authentication, but it
  sends passwords encrypted across the
  network

Restriction schemes:

- IP based restrictions
  only request from specific IP
  addresses can access the protected
  files, directories
- Domain based restrictions
  similar to the domain based
  restrictions except it uses domain
  names instead of IP addresses

OAS Manager

The components of OAS Manager System:

- OAS Manager : to manage the WEB site
- OPAS Utilities : to install, to log, to perform some special administrator tasks

OAS Manager

The components of OAS Manager System:

- OAS Manager : to manage the WEB site
- OAS Utilities : to install, to log, to perform some special administrator tasks

menu of the OAS Manager

- site manger
- OAS
- Logging
- Security
- DB Access
- ORB configuartion
- HTTP Listeners
- Applications

Start and stop node manager manually

```
owsctl  start -nodemgr
owsctl stop -nodemgr
```

owsctl status -nodemgrl

Listeners

OAS Manager Listeners
- Node Manager Listener
- Administration Utility Listener

Node manager Listener

it is present on each node of the site
it is the first process of the OAS
default port is: 8888
it has a username/password protection
to manage it manually

start : owsctl start -nodemgr
stop owsctl stop -nodemgr
status: owsctl status -nodemgr

Administration Utility Listener

it is used to run some demos and samples
it is used to run log analyzer
default port is : 8889
it is managed by the OAS Manger web site

Listeners

The OAS listeners like other listeners can handle

- static page requests
- dynamic, CGI page request

and

- Oracle DBMS connection request

Listener parameter

- listener name
- port number
- host name
- root directory
- user id
- maximum number of connections
- URL of redirection server
- configuration directory
- initial file
- directory mappings

Directory mappings

It allows to define virtual file system by mapping specific virtual pathnames used in the URL to file system specific filenames of local directories.

http://node.uni.edu/test/a.html
root :   virtual : /

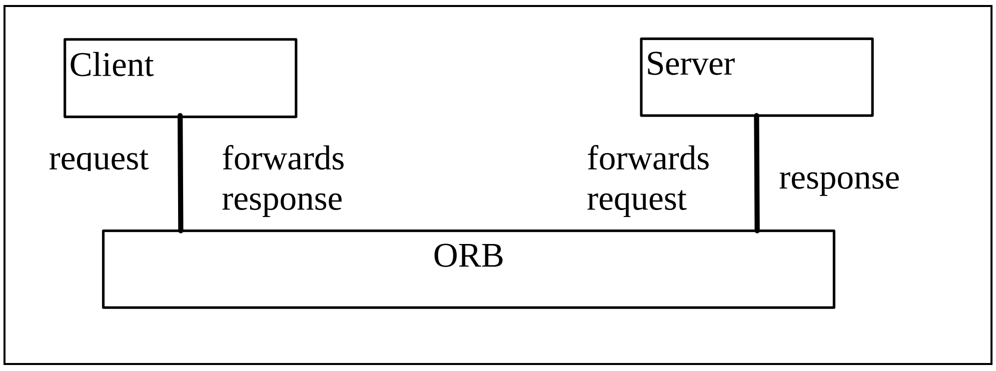
physical :   /usr2/locals/htp

ORB - WRB

Object Request Broker

ORB provides a distributed-object computing environment through which clients and servers interacts. Neither the client nor the server needs to be concerned with the location of the other or the transporting, converting data between the different sites.

Responsibilities of the ORB:
- finding appropriate object implementation to handle client reqiuests
- preparing those implementations to receive requests
- communicating the data
- returning results to the clients



Cartridges and Applications

cartridge:

- it contains configuration data to locate application logic
- it contains code to execute the application logic

cartridges provide runtime environment for specific programming languages like Java

- every cartridge has a virtual path to access it

some cartridge level parameters:

- virtual path for the catridge
- security schema
- main/max instances
- min/max threads per instance

Application:

it contains one or more cartridges of the same cartridge type

when application runs it creates instances of its cartridges within the cartridge server process

Application and cartridge lifecycles

if OAS starts up it starts up the minimum number of cartridge servers for each application

each cartridge sever then starts up the minimum number of cartridge instances

if minimum number is configured to 0, a cartridge server or cartridge instance is started up only it is required explicitly by a client request

a new cartridge instance is started up if there are more requests than cartridge instances

some cartridge are able to start up several threads per instances

Possible scenarios

scenario A : one thread and one instance in a cartridge server

scenario B : several threads and one instance in a cartridge server

scenario C : several instances but only one thread per instance in a cartridge server

Cartridges types

the following cartridges are supported in OAS4

- PL/SQL cartridge: it executes the PL/SQL procedures stored in the database
- JWEB cartridge : it runs Java applications. The access to the Oracle database can be processed by the following ways:
    - pl2java Java utility set
    - JDBC
- LiveHTML cartridge: it interprets server-side include SSI HTML documents
- Perl cartridge: it runs Perl scripts
- C cartridge : it runs C applications
- JCorba cartridge: it runs CORBA objects written in Java
- ODBC cartridge : to access ODBC databases like Informix, Sybase

Proposed scenarios:

PL/SQL, JWEB cartridges : scenario A or C
LiveHTML or Perl cartridges : scenario A
C, Jcorba cartridges : scenario A or B or C

DAD

Database Access Descriptor

A database may be local or remote

The DAD's contain the configuration data to needed by the OAS to connect to the Oracle database.

    configuration data
        - DAD name
        - database user name
        - use password
        - database location
        - database SID or connect string

TNS_ADMIN directory

    TNSNAMES.ORA configuration file

```
Example1.world =
  (DESCRIPTION =
    (ADDRESS =
          (PROTOCOL = TCP)
          (HOST = Production1)
          (PORT = 1521)
    )
    (CONNECT_DATA = (SID = SID1)
    )
  )
```

PL-SQL cartridge

It invokes stored procedures written in PL/SQL

Cartridge parameters:

- name
- virtual path
- DAD
- level or error messages
- protection

The DAD describes the access to the Oracle database

Identifications, activization of a PL/SQL cartridge:

```
http://host/virtual-path/stored-proc?parameters
```

steps of execution:

1. listener
2. WRB gets the request
3. cartridge server is activated
4. access to the database (DAD)
5. procedure execution
6. generation of result page
7. result page is passed back to the browser

Development of a PL/SQL cartridges

1. create a new pl/sql cartridge

2 create dad

3. create stored procedure

4. creating html file to invoke the procedure

A toolkit provided by the Oracle can be used to implement the HTML specific features in the PL-SQL environment.

Components, packages of the PLSQL Toolkit:

- htp (htf)	: procedures to generate HTML tags
- owa	: subprograms required by the PLSQL cartridge
- owa_util	: utility subprograms to perform dynamic SQL calls
- owa_pattern	: string manipulation
- owa_image	: retrieve coordinates where the user clicked on the image
- owa_cookie	: procedures to send or receive HTTP cookies

Sample PL-SQL applications

1. Creating and Loading the Stored Procedure onto the Database
2. Installing the PL/SQL Cartridge Packages and Creating a DAD
3. Reloading
4. Creating an HTML Page to Invoke the Procedure

Sample procedure
The current_users procedure retrieves the contents of the all_users table and formats it as an HTML table.

```
create or replace procedure current_users
AS
        ignore boolean;
BEGIN
        htp.htmlopen;
        htp.headopen;
        htp.title(`Current Users');
        htp.headclose;
        htp.bodyopen;
        htp.header(1, `Current Users');
        ignore := owa_util.tableprint(`all_users');
        htp.bodyclose;
        htp.htmlclose;
END;
```