REPUBLQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB
BLIDA

MEMOIRE DE FIN D'ETUDE

Pour l'obtention du diplôme d'Ingénieur d'Etat en Informatique

Thème

L'IMPLEMENTATION DE L'APPLICATION RESSOURCES HUMAINES EN INTRANET SOUS ORACLE8i (pour Linux)

Organisme d'Accueil : Centre de Traitement Informatique (CTI) SONATRACH

Sujet proposé par : M.Daymellah Abd El Wahab

Encadré par :
Mme Nabila Khaldoune
M.Mustapha Oldache

Présenté par :
BOUKHORS ASSIA
BAKALEM MAHDIA

Promoteur : Mme S.Oukid-Khouas

Promotion 2002-2003

Remerciements

A monsieur le conseiller du directeur général de la SONATRACH M.Hassani

Notre passage dans votre centre nous à été très instructif et bénéfique.

Nous tenons à vous exprimer notre profonde gratitude.

A monsieur le directeur du Centre de Traitement Informatique M.Abdelwahabe

Nous tenons à vous exprimer notre profonde gratitude pour l'accueil.

A monsieur Mustapha Oldache

Nous vous somme très reconnaissant pour l'aide précieuse et les conseils que vous nous avez apporté, tout au long de notre projet.

A Mme Nabila Khaldoune

Nous vous adressons nos vifs remerciements et l'expression de notre profonde gratitude pour votre précieuse collaboration et votre aide tout au long de notre projet.

A Mme S.Oukid - Khouas

Nous vous adressons nos vifs remerciements et l'expression de notre profonde gratitude pour votre précieuse collaboration et votre aide tout au long de notre projet.

Aux membres du jury

Vous nous faites un très grand honneur de juger notre modeste travail.

A tous les enseignants de l'université de BLIDA

Il vous revient le mérite de nous avoir prodigué un enseignement profitable et une formation complète.

Dédicaces

A mes très chers parents

Je dédie ce mémoire à mon père et ma mère qui m'ont soutenue et guidé vers la réussite. Votre patience et votre amour m'ont permis de continuer vers le droit chemin.

A mes sœurs et mon frère

Qui m'ont aidé et encouragé, surtout Nadia.

A Mahdia

Pour ton enthousiasme tout au long de notre projet.

A mes amis

De le promotion, Lamine, Chouäbe, Ouafid, Zohir, Rafika, Amel, Rouf, Farida, Ratiba, à toute ma famille.

ASSIA

A mes très chers parents

Je dédie ce mémoire à mon père et ma mère qui m'ont soutenue et guidé vers la réussite. Votre patience et votre amour m'ont permis de continuer vers le droit chemin.

A mes sœurs et mon frère

Qui m'ont aidé et encouragé, Ismail, Abdelkrim, Nasredine, Mohamed et ma sœur Djawida et surtout sa petite fille Amina et bien sûr, sans oublier mes belles sœurs et leurs enfants.

A Assia

Pour ton enthousiasme tout au long de notre projet.

A mes amis

De la promotion, à tous mes amis, à toute ma famille, Amina, Nadjiba.

MAHDIA

Résumé

Le sujet qui nous à été proposé par le CTI consiste à implémenter l'application Ressources Humaines développée sous un environnement client/serveur deux tiers avec l'outil de développement Developper 2000, sous une architecture multi-tiers sécurisée, faisant intervenir :

- Un client léger (navigateur Web),
- Un serveur d'application (conteneur de servelets et/ou serveur J2EE),
- Un serveur Web (Apache),
- Et la base de données sous ORACLE8i,

Il s'agit d'exploiter directement la base de données Ressources Humaine, puis concevoir une application du type INTRANET, à l'aide des outils inhérents au développement multi-tiers fournis avec le logiciel ORACLE8i disponible au sein de l'entreprise conjointement sur la plate-forme LINUX.

Summary

The subject which had been proposed to us by the CTI consists of implementing the application the human resources developed underneath a client/server environment two third with developing tools Developper 2000, under an architecture multi-ones, given great care which leads to:

- A light client (Navigator Web).
- Application server (container of Servlets and or server J2EE).
- An apache Web server.
- And the basic data under ORACLE8i.

It is a matter of exploiting directly the basic data of Humans Resources, the predicting an Intranet-type application with the means of inherent tools to multi-ones development provided withier Oracle8i which is provided withier the enterprise tightly linked on the stage LINUX.

Présentation de l'entreprise « SONATRACH»

Historique :

L'entreprise Nationale SONATRACH a été crée le 31.12.1963, son rôle principal était le développement du secteur des hydrocarbures. Les missions et les prérogatives de l'entreprise Nationale SONATRACH ont été élargies le 22 septembre 1966. Aussi sa mission qui se limitait à l'origine au transport et à la commercialisation des hydrocarbures a été élargie à tous les domaines de l'industrie pétrolière, à savoir la recherche, la production, le transport, la transformation et la commercialisation des Hydrocarbures.

Depuis le 24 février 1971, date de la nationalisation des hydrocarbures, l'entreprise a pris en charge l'ensemble du domaine minier et s'est vue confier le développement de toutes les branches de l'industrie pétrolière.

Missions principales de la SONATRACH :

Sous l'autorité d'un Directeur général, l'administration de la SONATRACH a notamment pour mission :

- Le développement, la conservation et valorisation des réseaux énergétiques nationaux,
- La reconstitution et l'accroissement des réserves d'hydrocarbures,
- L'intensification des efforts d'exploitation et capitalisation des études réalisées dans ce domaine, pour une meilleure connaissance du sous-sol et la mise en évidence des réserves d'hydrocarbures potentielles,
- Diversification des marchés et des produits à l'exportation,
- Approvisionnement énergétique National à moyen terme, compte-tenu des réserves nationales,
- Adaptation de l'outil commercial aux exigences du marché énergétique pour une meilleure maîtrise de ses mécanismes et des performances commerciales accrues,
- Développement et maintenance des complexes de perfection, de transport et de conditionnement des hydrocarbures,
- Développement des techniques modernes de gestion nationale par le biais de la formation continue,

Présentation du Centre de Traitement Informatique (C.T.I) :

Le centre de traitement informatique (CTI) est le centre de traitement principal de l'entreprise SONATRACH. Il constitue :

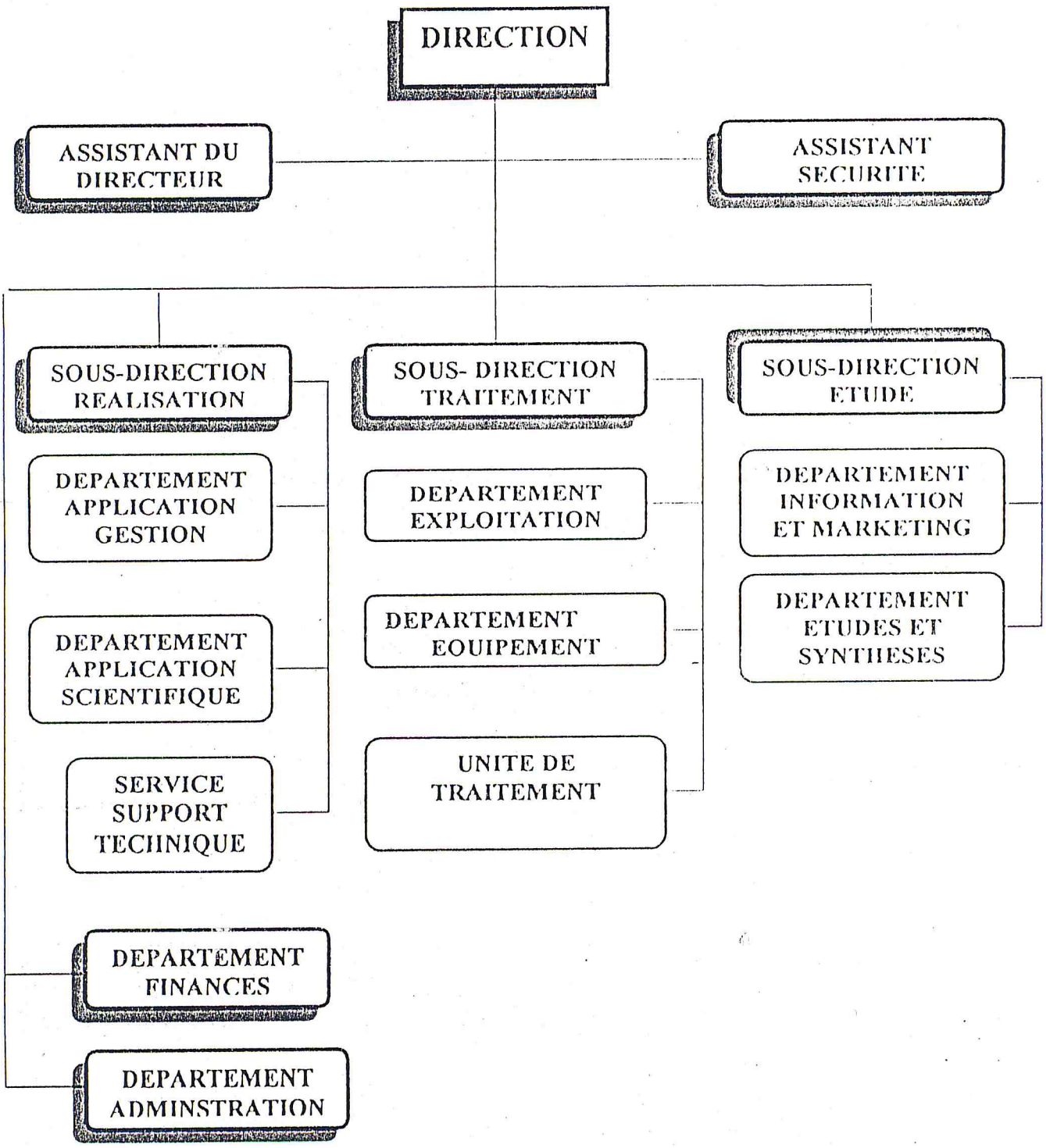
- L'outil privilégié de la direction générale en matière informatique,
- Le centre de traitement des directions centrales,

- Le prestataire de service des structures opérationnelles, Le CTI n'exerce pas de tutelle directe sur les structures de l'Entreprise en matière informatique. Il est l'interlocuteur de l'Entreprise pour toute relation en la matière avec les organismes et institutions extérieures.

Son organisation repose sur les principes suivants :

- La distribution fondamentale entre les activités de réalisation de produits informatique et celle de leur utilisation,
- Le regroupement des entités homogènes des activités de base pouvant être intégré en ensembles cohérents,
- La dotation d'une autonomie de fonctionnement aux structures internes leur permettant la recherche de rentabilité,
- La minimisation de la dépendance des structures internes par l'utilisation des équipements logiciels adéquats,
- La promotion continue de l'informatique par la mise à disposition des utilisateurs de compétences et de moyens adaptés,
- La possibilité de prendre en charge la gestion des équipements informatique des structures opérationnelles qui lui seraient confiés,

Organisation du Centre de Traitement Informatique



Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Introduction générale

La SONATRACH en tant que l'une des principales sociétés pétrolières dans le monde n'a pas voulu rester en marge des avancées technologique dans le domaine de l'Internet. Pour cela, elle a confiée au CTI (Centre de Traitement Informatique de SONATRACH) la mission de mettre en place l'Intranet de société, reliant tout d'abord les sites de l'entreprise à Alger, puis en second lieu de rendre présente la SONATRACH sur le World Wide Web en mettant en service un site Web qui renseignerait le monde sur l'état actuel de la société.

Le serveur Web mis en œuvre sur une Sparc station, a été réalisé par une équipe du CTI. Il peut être consulté à l'adresse www.sonatrach.dz. Des informations diverses sur l'entreprise y sont présentes et mises à jour régulièrement.

Le concept d'Intranet ayant été étendu au domaine applicatif, il est devenu courant actuellement de développer dans le World Wide Web des applications d'intérêt général telleque les banques de données ou des applications de commerce sur l'Internet. C'est dans ce contexte que le CTI nous propose de réimplémenter l'application Ressources Humaines développé sous forme client/serveur sous une architecture deux tiers avec l'outil Developper 2000 en une application sous une architecture trois tiers sécurisée, faisant intervenir :

- Un client léger (navigateur Web),
- Un serveur d'application (conteneur de Servlets et/ou serveur J2EE),
- Un serveur Web,
- Et une base de données sous ORACLE,

A l'aide des outils inhérents au développement multi-tiers fournis avec les logiciels ORACLE8i, JAVA conjointement sur les plate-formes LINUX et SOLARIS.

Chapitre 1: La base de données Ressources Humaines

1.1 Introduction

La première étape de notre conception d'une interface d'accès à une base de données est la compréhension de l'application Ressources Humaines dans l'architecture actuelle (deux_tiers), cette compréhension est basée essentiellement sur la connaissance de la base de données Ressources Humaines.

I.2 Définition du Ressources Humaines

L'application Ressources Humaines est l'une des plus importante application de gestion dans les entreprises, elle traite les problèmes humains et sociaux.

La gestion des ressources humaines a considérablement évolué au cours de ces dernières décennies. Elle a essentiellement intégré les recherches concernant le facteur humain dans une gestion des entreprises.

La gestion des Ressources Humaines porte sur l'utilisation efficace et efficiente de la ressource la plus importante de l'entreprise : ses employés. Elle englobe les activités nécessaires à l'acquisition et au maintien d'un effectif productif, répondant aux objectifs de l'organisation. Elle se penche non seulement sur les besoins de Ressources Humaines et leurs coûts.

1.3 La pratique de Ressources Humaines

La pratique de Ressources Humaine a pour objectif de traiter des quatre grands types de relations humaines que l'on trouve au sein des organisations : les relations de l'homme avec son travail, les relations de forces entre les différents groupes humains, les relations hiérarchiques et les relations entre individus.

On peut distinguer trois types de services du personnel dans une entreprise : dans le premier type, le service des Ressources Humaines s'occupe la charge du recrutement et de la rémunération mais n'a pas intégré au processus décisionnel de l'entreprise. Le second type traduit une vision à moyen terme du rôle du service du personnel : celui-ci établit une politique de gestion prévisionnelle (pour une durée de trois à cinq ans) de la promotion, de la formation, de l'évaluation et elle est consultée avant que les grandes orientations de l'entreprise ne soient décidées. Le troisième type est celui du développement. Les Ressources Humaines sont considérées comme un facteur déterminant dans le développement de l'entreprise. La fonction personnelle doit alors concilier le développement de l'entreprise et celui des individus de manière optimale.

Quelque soit la politique adoptée, le service du personnel remplit des tâches dans quatre grands domaines : La gestion du personnel sur un plan individuel ou collectif, la gestion des relations sociales internes et externes (relations avec le syndicat, les pouvoirs publics, etc.), les obligations légales et réglementaires auxquelles sont soumises les entreprises en manière de personnel (fiches de paie, médecine du travail, etc), et la gestion des équipements sociaux.

En fonction de la politique choisie, ces différents domaines constituent une réflexion stratégique à plus ou moins long terme.

1.4 Fondement d'un système d'information Ressources Humaines

Pour mener à bien sa politique des Ressources Humaines, toute organisation a besoin d'un système d'information adapté en vue de lui permettre un diagnostic précis de ces Ressources Humaines d'une part, et du fonctionnement des ensembles des domaines d'activité Ressources Humaines d'autre part.

Dans ce cadre, le système d'information Ressource Humaines doit être adapté aux objectifs organisationnels, tout en étant rigoureux dans la manière dont il utilise et coordonne ses indicateurs.

En général, il est opportun d'associer les informations d'ordre internes à des renseignements pris à l'extérieur de l'organisation.

Le système d'information constituera une base de données qu'il faudra nourrir d'inventaires de toutes sorte en liaison bien entendue avec les objectifs généraux et les besoins opérationnels.

1.5 Présentation de la base de données

La base de données des Ressources Humaines gérée par le département administratif du Centre de Traitement Informatique (CTI), trace les mouvements des agents de la SONATRACH depuis leur recrutement jusqu'à leur retraite, les informations concernant les agents, les diplômes, les fonctions. Les projets développés sont donc les composantes de cette base de données.

La base de donnée des Ressources Humaines étant la base de donnée technique (données important à manipuler et à consulter) et évolutive, à nécessiter d'être recréer sous un environnement plus fiable et plus efficace.

- **Résumé d'existant**

Un agent arrive à la SONATRACH suivant plusieurs mode d'arrivé, ça peut être par stage effectué au niveau de la SONATRACH, par recrutement ou par transfert. Il possède un diplôme, et maîtrise des langues, il est affecté à une section d'un certain service d'un département, d'une direction, d'une branche. Un agent peut avoir un projet de développement au sein de la SONATRACH pour un certain délai. L'agent suit un système de travail, une formation dans un établissement et/ou une formation séminaire dans une certaine spécialité. Des établissements disposes aussi de formation séminaires.

Les agents des structures qui n'ont pas l'accès à la base de données se déplacent jusqu'à la direction générale pour apporter les imprimés des masques de saisie relative aux informations des agents.

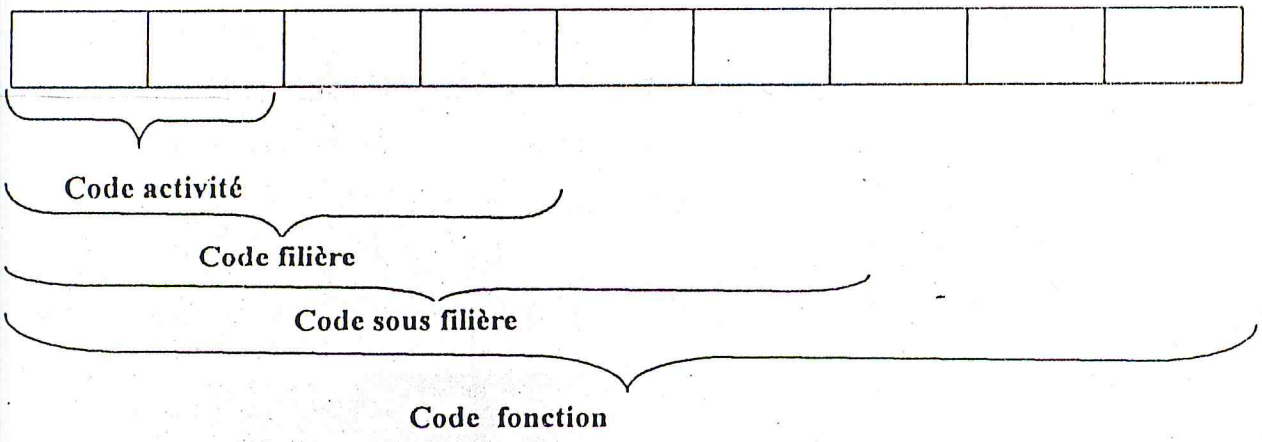
➢ **Quelques règles de gestion**

- Un agent peut occuper une ou plusieurs fonctions.
- Un agent peut pratiquer un ou plusieurs système de travail
- Un agent peut ne pas détenir un diplôme comme il peu en détenir plusieurs.
- Un agent peut suivre un ou plusieurs formations.
- Un agent peut maîtriser une ou plusieurs langue.

➢ **Quelques contraintes de la base de données**

La contrainte de système de travail : NN, SR, NR.

- Contrainte valeur du diplôme : un nombre entre 45 et 200.
- Contrainte niveau de maîtrise : « A », «B », « C », « D ».
- Contrainte de type formation : un nombre entre 1, 2, 3, 4, 5.
- Contrainte mode d'arrivé : entre 1 et 12.
- Contrainte de code matricule : sur 9 positions telque :



- La contrainte matricule : le matricule est composé de 6 positions, les 5 premières sont des chiffres de [0-9] et la dernière position est une lettre différente de (« i », « o », « « z »)

La figure suivante schématise le modèle conceptuel de données (MCD):

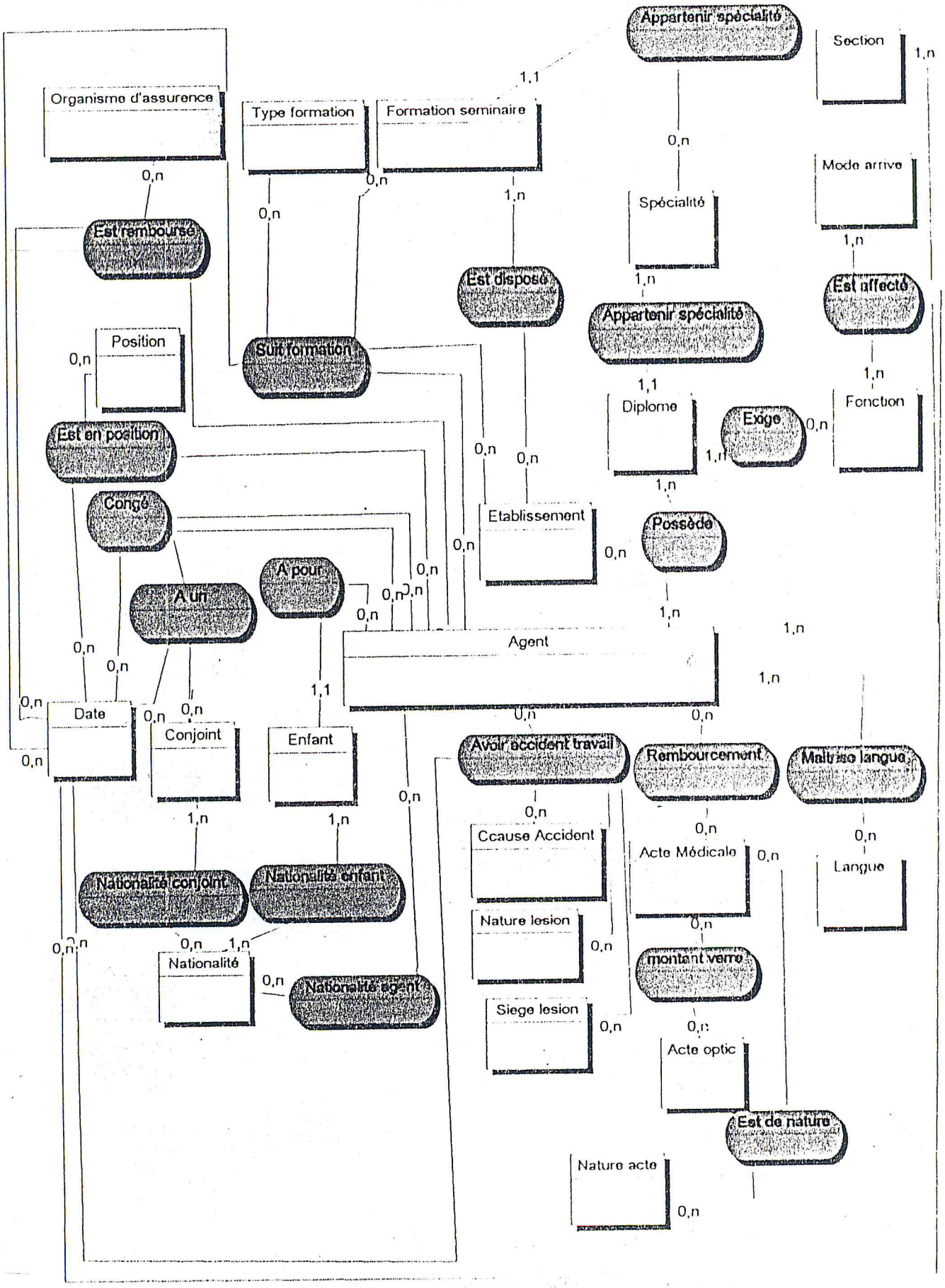


Figure 1 : Le Modèle Conceptuel de Données

1.5.1 Les tables de la base de données des Ressources Humaines

La base de données Ressources Humaines contient des références (données) associées aux différentes entités. Ces entités sont représentées sous forme de tables. [Annexe D]

1.6 Critique de l'application Ressources Humaines

L'application Ressources Humaines à été développée avec l'outil FORMS d'ORACLE du logiciel Developpeur/2000, sous une architecture 2 tiers, elle est installée sur quelques postes. Le Developper 2000, qui est propriétaire à oracle nécessite beaucoup d'espace disque, ce que rend le client lourd, de plus, il faut installer le ce dernier dans chaque poste client a fin accéder à cette application, et enfin la formation des agents est nécessaire pour une bonne utilisation de l'application.

- ORACLE FORMS

Outil utilisé pour générer des applications basées sur des formulaires appelées application FORMS. Ces formulaires applicatifs peuvent servir à saisir des données, émettre des requêtes, et se déplacer parmi les enregistrements.

- PL/SQL

C'est un langage procédural, extension de SQL : utilisation du SQL au sein de boucles et de tests.

1.7 Conclusion

Les Ressources Humaines sont un élément essentiel pour le bon fonctionnement de l'entreprise. Sa gestion est lourde, elle engendre des frais qui augmentent les charges de l'entreprise.

La gestion des Ressources Humaines permet de fournir des renseignements appropriés, valables et opportuns en réponse aux questions de Ressources Humaines qui se posent au niveau des décideurs, elle assure également une bonne gestion de la formation des agents par l'établissement d'un plan adéquat, et au plus, avoir une meilleure exploitation en automatisant la gestion administrative du personnel, sans oublier, qu'elle réduit le temps et le coût du traitement de la paie.

Chapitre 2 : Les architectures d'accès aux bases de données

2.1 Introduction

Une grande partie des données délivrées sur le Web est par essence dynamique, par exemple, les utilisateurs peuvent suivre les cours de la bourse et les dernières prévisions météorologiques à temps réel.

L'accès aux bases de données sur réseau est l'une des applications majeures de l'Intranet, qui considéré une passerelle entre le système d'exploitation du serveur et les fichiers de données ou plus généralement de la base de données. Cette passerelle pourra se trouver sur le même serveur que les données ou sur un serveur différent. Ceci déterminera 2 types d'architecture : de 2 tiers ou 3tiers.

Pour faciliter aux utilisateurs l'accès aux données que ce soit l'application sous l'architecture 2-tiers ou 3-tiers, des techniques sont mises en œuvre grâce à la fusion du Web et du SGBD qui permettent de construire une architecture client serveur de toute nouvelle génération.

2.2 Les architectures d'accès

2.2.1 Architecture 2-tiers

Une architecture 2-tiers est composée de deux éléments, un client et un serveur où le tiers fait référence non pas à une entité physique mais logique, et que l'on peut organiser comme suit [FMC98]:

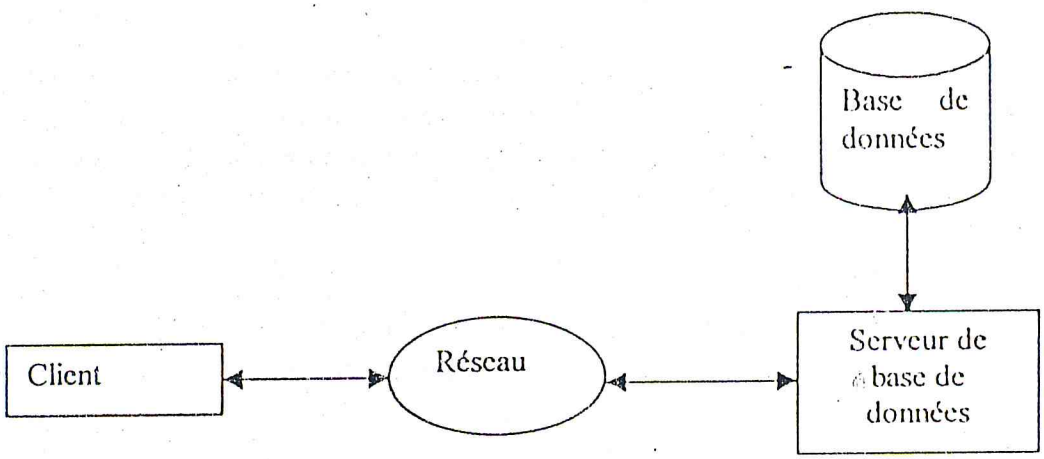


Figure 1 : Structure d'architecture 2-tiers

Bien que ce schéma puisse paraître simpliste, une analyse détaillée des éléments de cette architecture et leurs fonctions attachées met en évidence certains points essentiels sur lesquels on attire l'attention :

- La présentation est à la charge du client exclusivement (le client qui gère la présentation).
- Le calcul (processing) est réparti entre le client et le serveur.
- Le logiciel client est généralement spécifique au serveur.
- Les données sont stockées ou accessibles via le serveur. Dans le cadre d'une topologie d'accès à une base de données, le serveur traitera les requêtes en provenance du client qui se feront en général en langage SQL.

C'est parce que le client assume des tâches de présentation et de processing, et donc fait communiquer avec le serveur sans intervention d'un autre processus que le client est dit 'lourd', car il communique avec le serveur via un frontal.

Plus précisément, il existe deux possibilités d'architecture 2-tiers, le schéma précédent sera plus exactement l'un des deux suivantes [FMC98] :

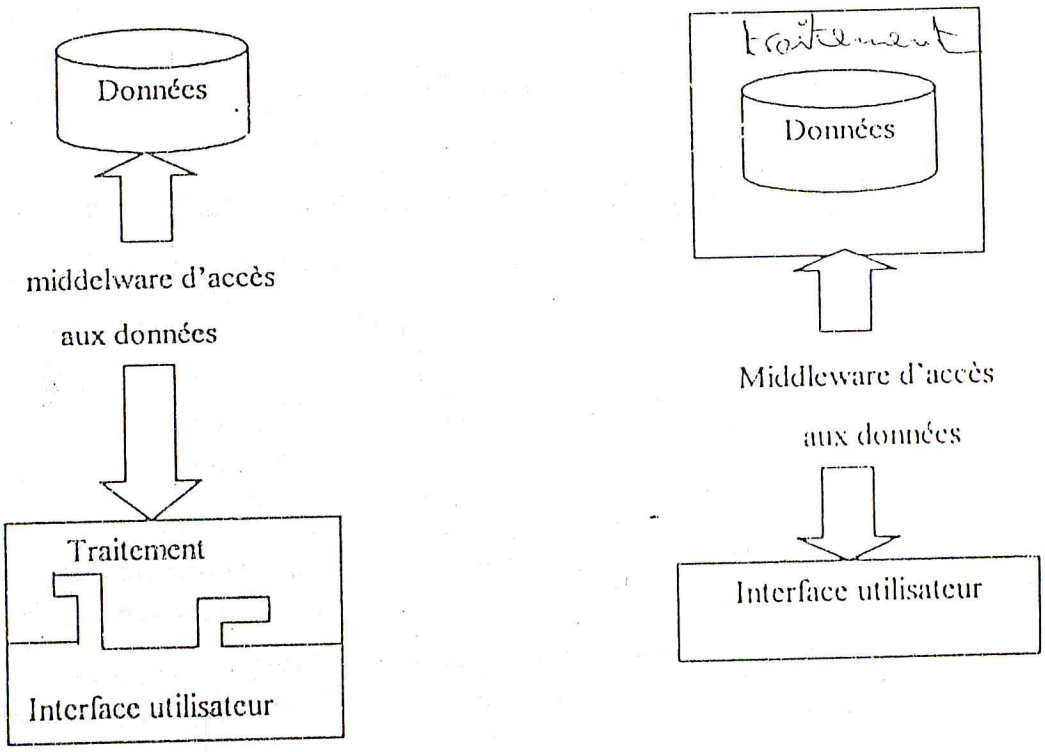


Figure 2 : possibilité d'architectures 2-tiers

Le premier type d'architecture ne différencie pas réellement l'interface utilisateur des traitements : généralement les objets de l'interface utilisateur lancent des requêtes SQL vers le moteur de base de données et récupèrent les résultats directement. Un middelware transporte les requêtes entre les deux parties.

Le second type d'architectures ne différencie pas les traitements des données : c'est le cas des procédures stockées pour un SGBD relationnel.

➢ **Les avantages d'une architecture 2-tiers**

- Le développement d'une architecture 2-tiers peut être réalisée rapidement.
- La plupart des outils de développement dans l'architecture 2-tiers sont robustes et mènent d'eux-mêmes à des techniques RAD (*Rapid Application Developemnet*) qui peuvent être utilisées pour s'assurer que les spécifications des utilisateurs sont précisément et totalement prises en compte.

➢ **Les inconvénients d'une architecture 2-tiers**

- Pas d'indépendance nette entre l'interface utilisateur et les traitements fonctionnels, rendant délicate l'évolution des applications.
- Difficulté de relocalisation des couches de traitement consommatrices de calcul. De même, il est assez délicat de modifier la source de données, son code d'accès étant trop lié à l'interface homme machine (IHM).
- Difficulté de remplacement du middleware de transport, du fait de l'impact sur toutes les couches du logiciel.
- Réutilisation délicate du logiciel développé selon cette architecture.

2.2.2 Architecture 3-tiers ou multi-tiers

L'architecture logicielle à 3-tiers ou plus effectue nécessairement une séparation dans le codage et les outils mis en œuvre entre l'interface utilisateur, le traitement et les données.

L'architecture 3-tiers est composée de trois éléments ou plus précisément dans ce cadre là de trois couches. En effet dans ce contexte, il est plus adéquat de parler de couche fonctionnelle où à chacune d'elle est attachée un élément/entité logique.

Dans le modèle 3-tiers, il faut distinguer trois couches/éléments [GEO02] :

- La couche présentation associée au client qui de fait est dit ''léger'' dans la mesure où il n'assume aucune fonction de traitement à la différence du modèle 2-tiers.
- La couche fonctionnelle liée au serveur, dans de nombreux cas est un serveur Web muni d'extension applicative.
- La couche de données liée au serveur de base de données (SGBD).

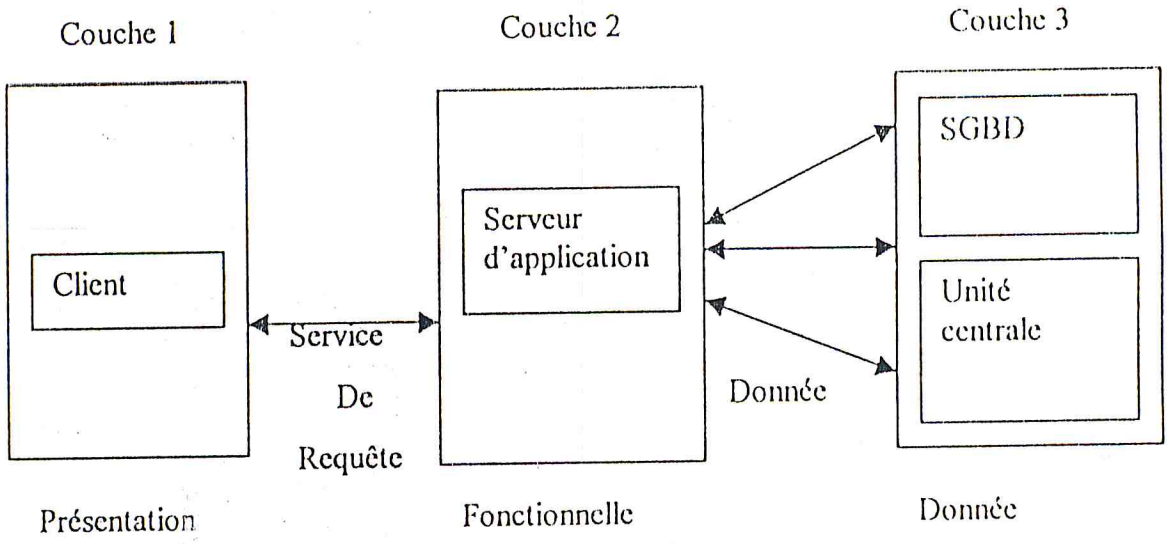


Figure 3 : Structure générique d'une architecture 3-tiers

Comme dans l'architecture 2 tiers, il existe deux possibilités d'architecture 3-tiers, le schéma précédent sera plus exactement l'un des deux suivantes :

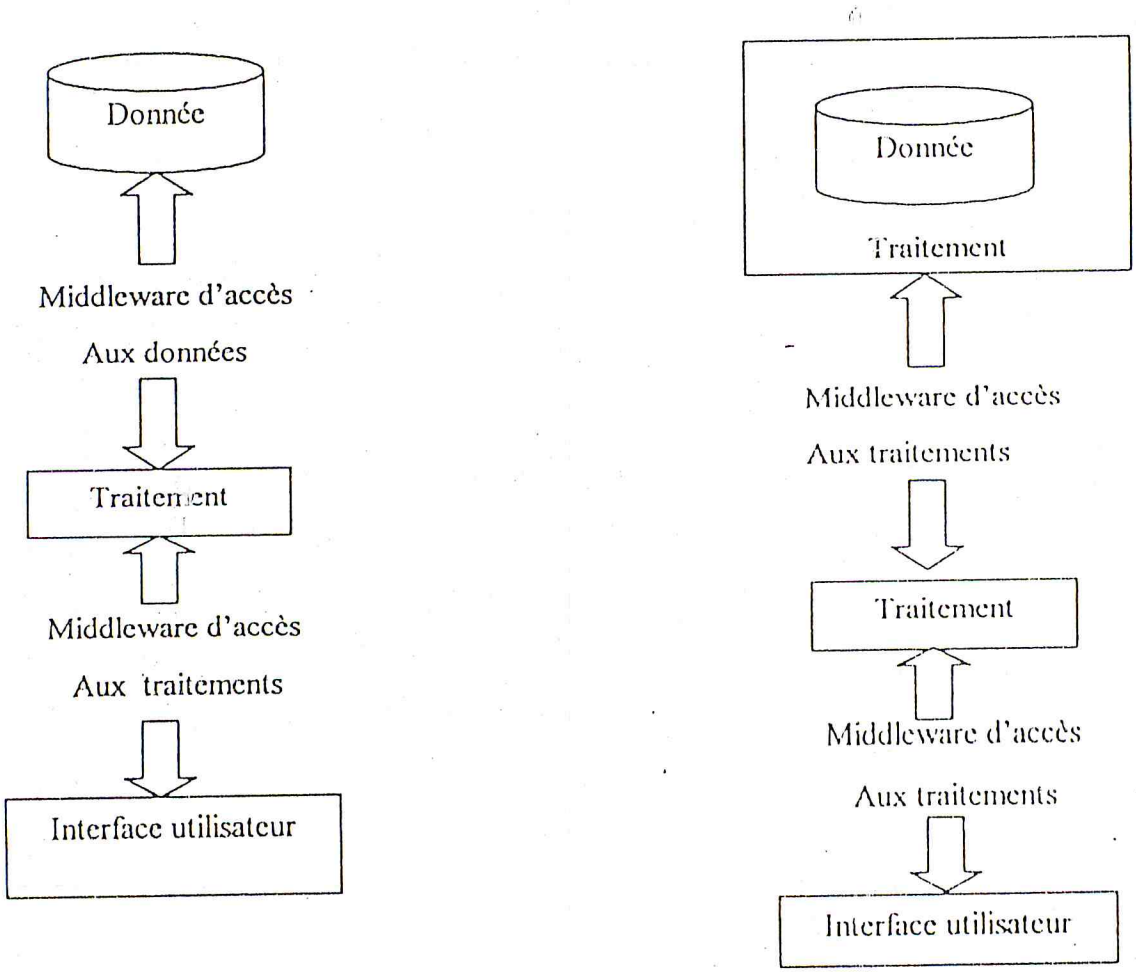


Figure 4 : Possibilités d'architecture 3-tiers

➢ **Les avantages d'une architecture 3_tiers**

- Interchangeabilité des différentes couches par le respect d'interfaces précises entre les couches (amenées par un middleware standard ou spécifique).
- Autonomie des couches les unes par rapport aux autres si l'une d'entre elles évolue (modification de l'implémentation et non de l'interface d'appel).
- Indépendance des différentes couches par rapport à la localisation physique. Le niveau ''traitement'' pouvant être redécoupé à l'infini, cette architecture permet le déploiement d'application distribuée à grande échelle.
- Indépendance par rapport aux fournisseurs de middleware ou SGBD.
- Capacité d'identification de composants réutilisables ou de réutilisation de composants existants, soit au niveau technique soit au niveau fonctionnel.
- Capacité accrue de réparation en sous-projet de réalisation.

➢ **Les inconvénients d'une architecture 3-tiers**

- Une expertise de développement à acquérir qui semble plus longue que dans le cadre d'une architecture 2-tiers.
- Les coûts de développements d'une architecture 3-tiers sont plus élevés que pour du 2-tiers.

2.3 Les technologies d'accès

Plusieurs solutions permettent d'accéder à une base de données, dont les principales technologies sont :

2.3.1 Les anciennes technologies

➢ **CGI (Common Gateway Interface)**

- **Définition**

Les premiers serveurs HTTP ne comportaient pas de mécanismes prédéfinis pour générer des réponses d'une façon dynamique, des interfaces étaient utilisées pour appeler d'autres programmes capables de traduire les requêtes en un contenu exécutable.

Le premier standard pour le contenu dynamique basé sur l'interface CGI qui est une norme spécifiant le mécanisme selon lequel un serveur Web transmet l'information de requête à des programmes externes, ces derniers sont à leur tour exécutés par le serveur Web pour générer une réponse à l'exécution.

Perl est un langage le plus prisé actuellement pour écrire de tels programmes CGI et il y a aussi les langages C et C++ [NYM02].

- **Principe de CGI**

A chaque fois qu'une requête de contenu dynamique est reçue par le serveur Web, ce dernier doit lancer un nouveau processus pour exécuter le programme CGI correspondant à la requête.

Le navigateur envoie alors au programme l'information de la requête, puis attend les résultats qui seront transmis dans la réponse au navigateur.

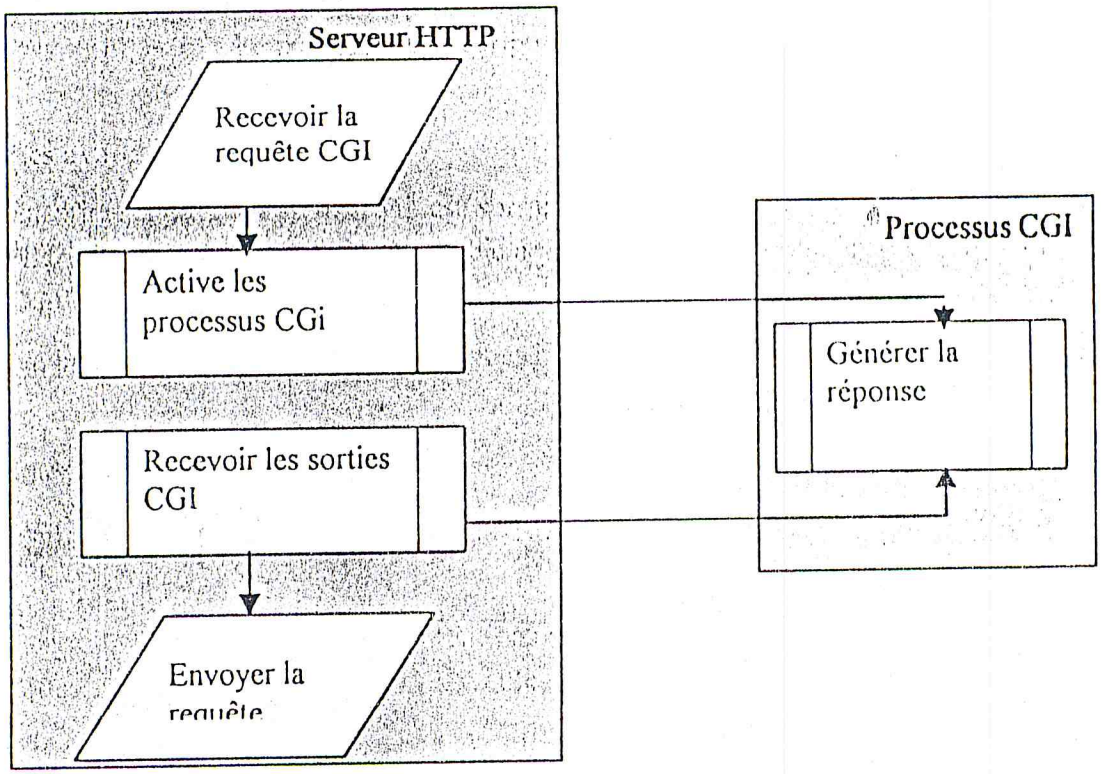


Figure 5 : Le processus serveur pour l'exécution d'un programme CGI

➢ **Les avantages de CGI**

- La principale avantage de CGI est son universalité et sa simplicité de mise en œuvre. En effet un programme CGI est accessible à partir de n'importe quel serveur http du marché, et peut être écrit dans plusieurs langage comme C, perl ou shell.
- Un investissement minimal en matériel et logiciel.

➢ **Les inconvénients de CGI**

- CGI pose des problèmes qui limitent son utilisation dans le cas du déploiement à grande échelle d'application Web.
- Les programmes CGI ne s'exécutent pas au niveau du serveur Web c'est à dire un nouveau processus doit être lancé pour les exécute (*multi-processing*).
- Crée et établir une communication avec le processus séparer entraîne une surcharge certaine.

- Chaque processus accapare une partie des ressources mémoire de machine locale (zone mémoire, et d'autres ressources système doivent être allouées) d'où le coût très élevé.
- Les programmes CGI n'exécutent qu'une seule requête à la fois.

➢ **Serveur http étendus**

Extensions au serveur http, afin de pouvoir exécuter les requêtes des clients au sein du serveur http lui-même. Un ensemble de fonctions supplémentaires est en général fourni sous forme d'A P I (*Application Programming Interface*) et les applications sont développées en utilisant cette dernière.

➢ **ISAPI de Microsoft**

L'A P I de Microsoft, permet de substituer à des exécutables CGI une ou des DLLs (*Dynamic Link Library*). Ces DLLs pourront être chargées dynamiquement en mémoire lors de la première requête, et restent disponibles pour de futurs autres clients. La référence à ces DLLs se fera de manière standard, par l'intermédiaire d'un URL se terminant par l'extension .dll.

➢ **NSAPI de Netscape**

Suivant le même principe, NSAPI est une extension du serveur http Netscape standard, l'interfaçant avec des applications clients. Les fonctions de service d'application, qu'offre le serveur en standard, sont définies dans un fichier de configuration et peuvent ainsi facilement être étendues.

➢ **Les avantages de http étendu**

- Diminuer la consommation de ressources (espace mémoire et temps machine) sur le serveur
- Les A P I offrent de bien meilleures performances.

➢ **Les inconvénients de http étendu**

- Les API dépendent d'une plate-forme spécifique.
- La mise au point de programmes basés sur ces A P I est complexe
- Leur mise en œuvre demande une bonne maîtrise de la programmation.
- Le débugging est difficile.
- Toute erreur dans l'implémentation du code peut entraîner le plantage du serveur.
- Chaque modification de la librairie dynamique comprenant le programme à exécuter nécessite le redémarrage du serveur Web.

➢ **L'implémentation de CGI, NSAPI et ISAPI**

- **Sur l'architecture 2-tiers**

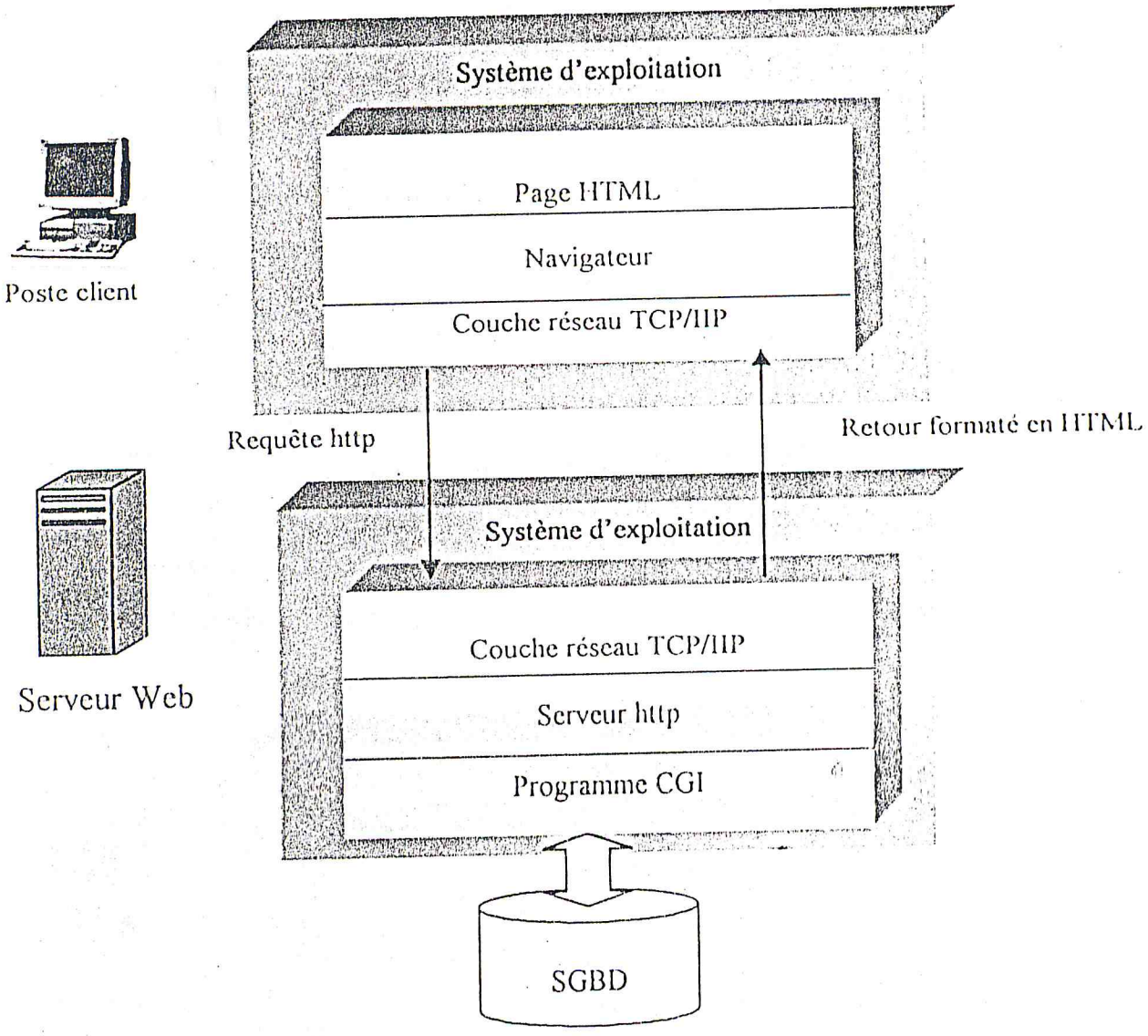


Figure 6 : L'architecture à base de CGI

Dans ce type d'architecture (2-tiers), l'interface utilisateur et les traitements sont regroupés dans les pages HTML qui contiennent directement des appels à la source de données, par l'intermédiaire du langage SQL, par exemple, le programme CGI, NSAPI, ISAPI [figure 7] ne servent que de passerelle entre le serveur http et la source de données. Il reçoit les requêtes en provenance du client et lui renvoie le résultat, sous forme HTML en effectuant un habillage.

Un grand nombre de produits de connectivité aux bases relationnelles sont conforme à cette architecture, qui possède pour principal inconvénient de ne pas séparer clairement la couche de présentation de celle des traitements sur les données.

De plus, cette intégration est réalisée directement dans le langage HTML, ce qui n'est pas du tout structuré (entraînant par la suite de nombreux problèmes de maintenance de développement).

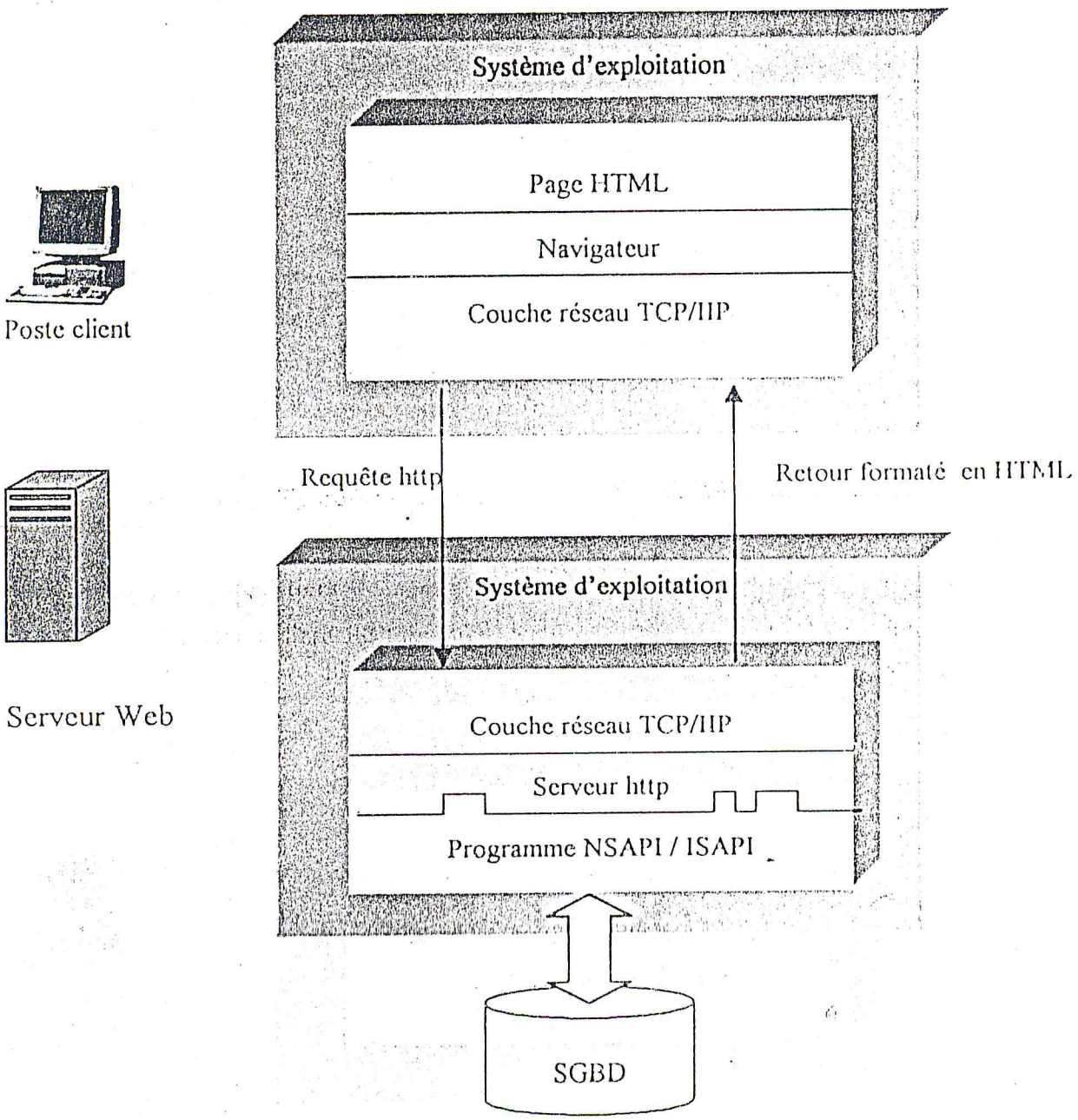


Figure 7 : L'architecture à base de NSAPI et ISAPI

- **Sur l'architecture 3-tiers**

Contrairement à l'architecture précédente (2-tiers), il est possible d'introduire un niveau supplémentaire et de séparer plus nettement les traitements en procédant de la manière suivante.

[Figure 8].

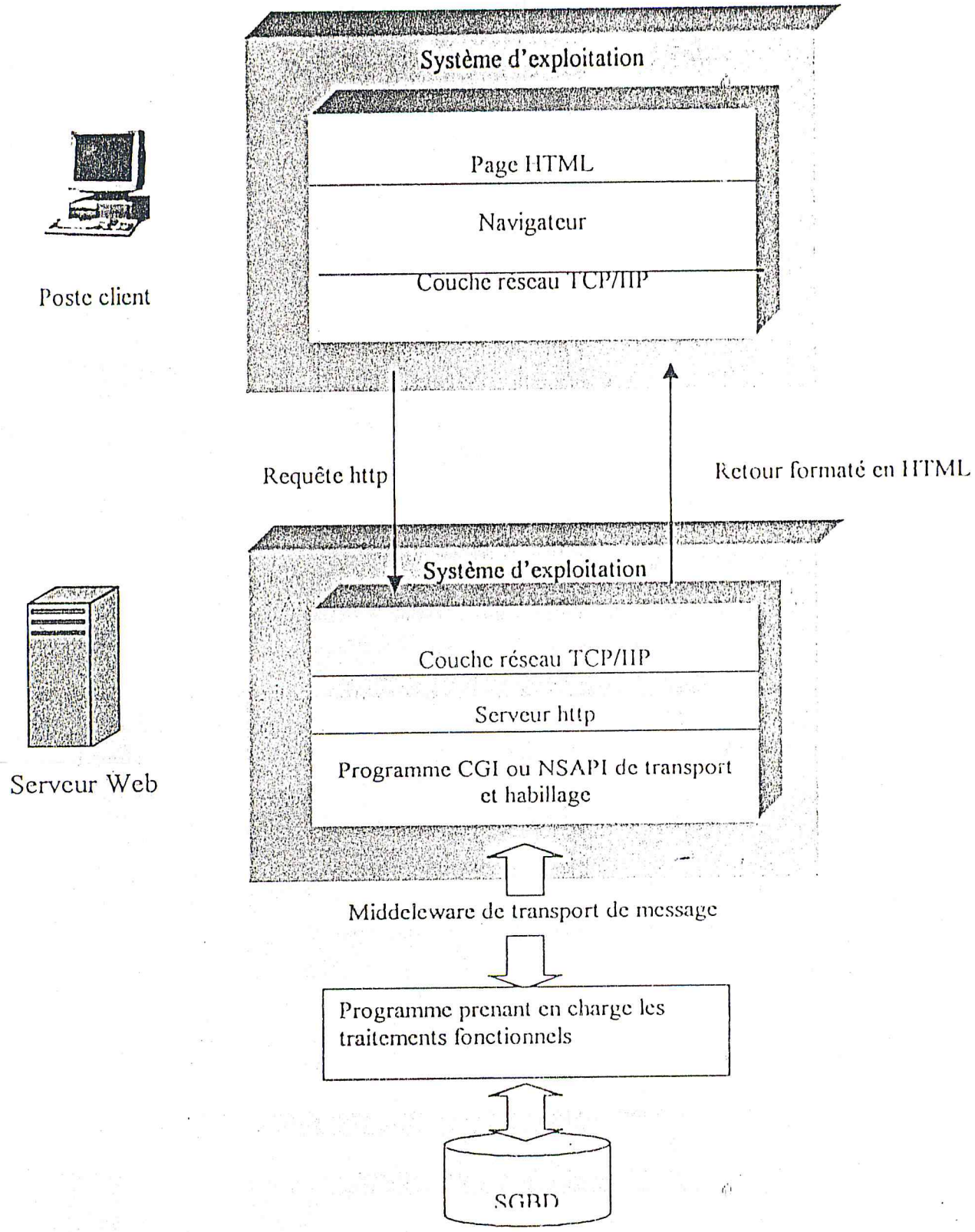


Figure 8 : Architecture 3-tiers à base de CGI ou de NSAPI/ISAPI

Dans le cas présent, logiciel s'exécutant sur le serveur contient l'ensemble des traitements de l'application. Ces traitements sont invoqués par l'intermédiaire de message identifiés précisément

(par opposition à du SQL dynamique par exemple). Les pages HTML contiennent donc simplement des numéros (ou noms) de messages. Le logiciel fonctionnel, qui lui renvoie des données en retour. Le logiciel CGI effectue un habillage en HTML avant de renvoyer la réponse au navigateur.

2.3.2 Les langages de scripts

➢ **ASP (Active Server Page)**

- **Définition**

ASP est un système de Microsoft qui supporte divers langages de scripts tels que PerlScript, JScript et VBscript, mais le langage par défaut de ASP est VBscript qui comporte un support pour accéder aux composants ActiveX. Ces composants sont des codes objets compilés pouvant, dans la pratique, encapsuler toutes sortes de fonctionnalités telles que l'accès à une base de données ou la manipulation de fichiers. Un grand nombre de composants ActiveX prêts à l'emploi sont disponible [DDL00].

➢ **Principe de ASP**

- Un scripte ASP s'exécute lorsqu'un navigateur appelle une page ayant cette extension (.asp) sur le serveur Web.
- Le serveur lance alors le module ASP et lui transmet le fichier. Le module parcourt le fichier à la recherche de la balise, au cours de cette action, dés que le module ASP rencontre un script encadré par la balise, il l'exécute et renvoie ensuite au serveur du code HTML que celui-ci transmet, à son tour, au navigateur.

➢ **Les avantages de ASP**

- Les fichiers ASP sont tout exécuté sur le serveur, ainsi les pages ne sont pas dépendantes d'un navigateur particulier. Elles sont uniquement limitées par les capacités du serveur.
- La modification d'une page ASP est facile. En plus dès que la modification est faite, le prochain qui aura à l'utiliser, utilisera la version mise à jour.
- La conservation des sessions pour chaque client se fait grâce à des variables stockées sur le serveur dans des répertoires virtuels.

➢ **Les inconvénients de ASP**

L'ASP à pour inconvénient majeur de n'être disponible qu'avec le serveur Internet de Microsoft (IIS '' *Internet Information Server* '') qui s'exécute sous le système d'exploitation Windows NT/2000.

➢ **PHP (Personal Home Pages)**

- **Définition**

Le signe PHP désignait les outils de création de pages personnelles comme ses fonctionnalités et son domaine d'application se sont étendus au cours des années, cette dernière appellation est devenue obsolète, et le sigle ne désigne plus maintenant que le système au sens large, tout comme JavaScript, PHP utilise la syntaxe C et fournit un support efficace pour le contrôle de type et l'accès aux bases de données. Il est doté également d'extension qui lui permettent de communiquer avec d'autres ressources comme la messagerie électronique et les annuaires [DDL00].

➢ **Principe de PHP**

- Un script PHP s'exécute lorsqu'un navigateur appelle une page ayant cette extension (.php) sur le serveur Web.
- Le serveur lance alors le module PHP et lui transmet le fichier. Le module parcourt le fichier à la recherche de la balise, au cours de cette action, dés que le module PHP rencontre un script encadré par la balise, il l'exécute et renvoie ensuite au serveur du code HTML que celui-ci transmet, à son tour, au navigateur.

➢ **Les avantages de PHP**

- La gratuité et la disponibilité du code source.
- La simplicité d'écriture de scripts.
- La possibilité d'inclure le script PHP au sien d'une page HTML (contrairement aux scripts CGI, pour lesquels il faut écrire des lignes de code pour afficher chaque ligne en langage HTML).
- La simplicité d'interfaçage avec des bases de données (de nombreux SGBD sont supportés, mais le plus utilisé avec ce langage est MySQL, un SGBD gratuit sur les plate-forme Unix et Linux, mais payant sous Windows).
- L'intégration au sien de nombreux serveurs Web (Apache, Microsoft IIS, ...).

➢ **Les inconvénients de PHP**

- Les problèmes dus à la gestion automatique des types de données.
- Pas d'environnement de développement.
- L'interprétation et l'exécution d'un script est très coûteux pour le serveur. Un nombre important et simultané de requêtes peut ralentir fortement les temps de réponses.

2.3.3 Les outils Java sous Oracle8i

➢ **JDBC (Java Data Base Connectivity)**

- **Définition**

JDBC est une couche logicielle standard offerte aux développeurs, en d'autres termes une A P I permettant l'accès à des bases de données relationnelles à partir du langage Java, elle est composée de classes et d'interfaces se trouvant dans le paquetage java.sql. il existe en fait deux types d'A P I :

- A P I **jdbc** : utilisée par le programmeur, répondant aux fonctionnalités SQL.
- A P I **jdbcDriver** : utilisée par le concepteur de pilote, dialogue entre la machine virtuelle Java et SGBD.

Cette spécification issue du travail commun de SunSoft et Intersolv définit essentiellement un certain nombre de classes pour se connecter, et gérer les accès aux données aussi bien en consultation qu'en mise à jour, JDBC a été complètement écrit en Java [GRE97].

➢ **Principe de JDBC**

L'interaction avec une base de données se déroule principalement en ces quatre phases successives :

- Chargement et configuration du client qui veut interroger une base de donnée.
- Connexion à la base de donnée.
- Exécution des commandes SQL.
- Inspection des résultats (si disponible).

A chacune de ces phases, correspond une classe Java que l'on trouvera dans le package java.sql :

- DriverManager envoie les demandes de connexion au driver adéquat
- Connection ouverture d'une session/connexion à une base de données.
- Statement exécution des ordres SQL standard.
- ResuttSet tableau des lignes résultant d'un ordre select.

JDBC fournit un moyen de communication de bas niveau avec les bases de données.

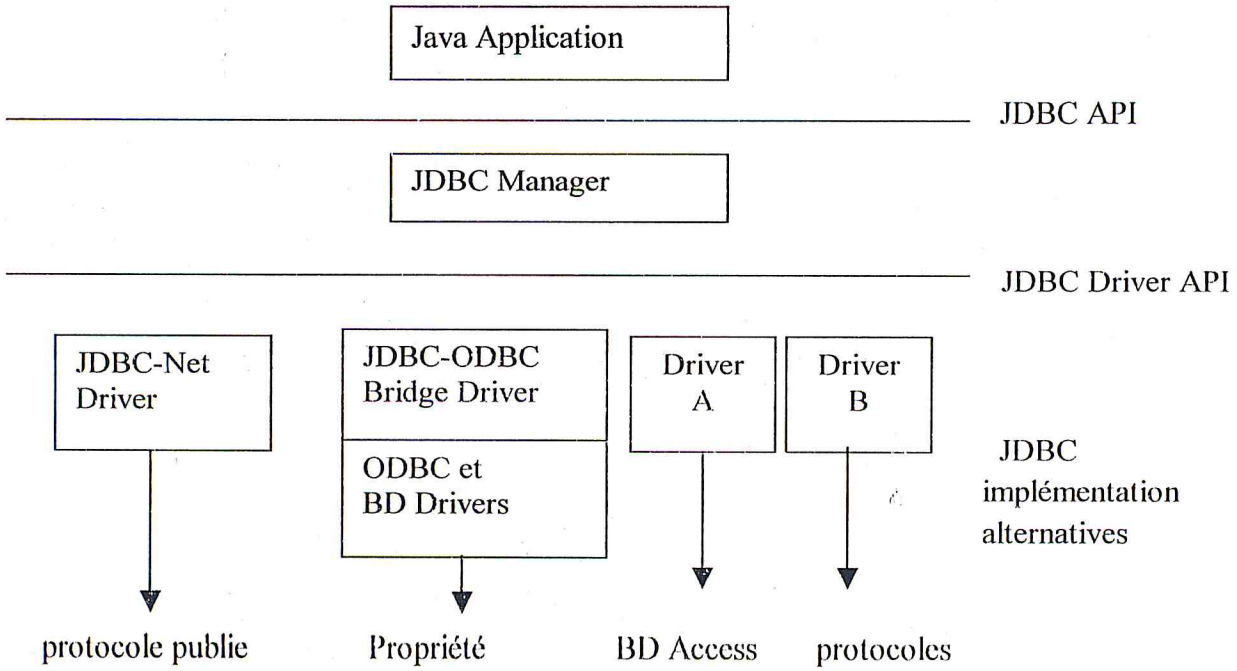


Figure 9 : Interface de JDBC

➢ **Les avantages JDBC**

- Une technologie simple, efficace, indépendante des plates-formes, orienté objet, extensible.
- Les appels sont potentiellement dangereux si elles sont diffusées sur un réseau largement ouvert au public comme Internet, par sécurité, Java impose des limitations draconiennes à ce que peuvent faire des appels. Cela est extrêmement gênant dans le cas d'Internet ou d'un extranet, dans ces cas, en effet, l'origine des appels et parfaitement connue, il n'y a donc pas à craindre les malversations des programmeurs malveillants.
- Interface utilisateur beaucoup plus riche.
- Capacité de maintien de la session, simplifiant la programmation d'application transactionnelle.
- Meilleure sécurité.

➢ **Les inconvénients de JDBC**

- Les appels java ne peuvent accéder directement aux fonctions matérielles.
- Les appels java sont difficilement interprétés par les navigateurs Web, à cause du non-compatibilité entre la version de la machine virtuelle du navigateur et la version de JDK *(Java Développement Kit)* avec lequel les appels ont été programmés et compilés.
- Les appels Java peuvent nécessiter plusieurs fichiers.

- Les appels résident dans le système de fichiers locaux uniquement pour la durée de la session courante du navigateur. En cas de visites fréquentes du même site, l'utilisateur devra à nouveau charger tous les fichiers associées aux appels.

➢ **Les Servlets**

- **Définition**

Les Servlets sont des petites applications également basées sur Java qui apportent une fonctionnalité dynamique aux serveurs Web. Elles possèdent un modèle de programmation similaire aux scripts CGI dans la mesure ou elles prennent en entrée une requête http fournie par le navigateur Web et ou elles sont supposées localiser ou construire le contenu approprie en réponse au serveur **[DAU00].**

➢ **Principe de Servlet**

Lorsque le serveur http reçoit une requête pour un URL, il la transmet en conteneur de servlets, qui à son tour la transmet à une instance de classe.

La transmission des requêtes est réalisée en regroupant toutes les données de la requête (l'URL, l'origine de la requête, les paramètres et leurs valeurs, ect...) dans un objet Java. Un objet Java semblable est développé pour contenir la réponse et pour avoir accès au flot de sortie qui va contenir les résultats do traitement de la requête.

Les classes de servlets sont responsables de la définition des méthodes qui servent à traiter les différents types de requêtes http, telles que les méthodes doGET() pour les requêtes GET, et doPOST() pour les requêtes POST. Les objets construits par le conteneur de servlets pour représenter une requête donnée et la réponse correspondante sont alors invoquées par le conteneur de servlets à chaque requête.

En utilisant les objets requête et réponse correspondants, la méthode de traitement accède aux propriétés de la requête et réalise les calculs qu'il convient de faire sur ces données pour construire sa réponse. Le code html de la réponse est inscrite dans le flot de sortie associé à l'objet réponse. Une fois que la méthode de traitement a achevé son exécution, le conteneur de servlets renvoie le contenu de l'objet réponse au serveur http qui, à son tour, le transmet au navigateur Web initiateur de la requête. Le traitement simultané de plusieurs requêtes par une servlet est réalisé en exécutant chaque appel à une méthode de traitement dans un thread séparé.

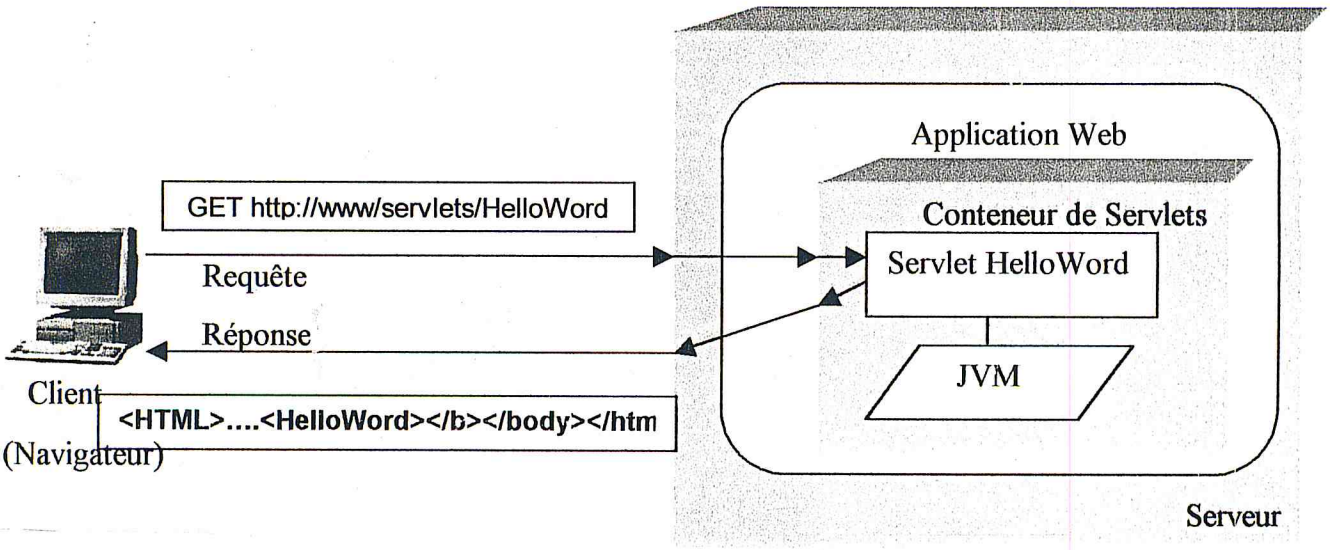


Figure 10 : Représentation de fonctionnement de Servlet

De plus, toutes les Servlets associées à un serveur Web s'exécutent dans un seul processus, ce processus exécute une JVM *(Machine Virtuelle Java)* qui crée un *thread* Java pour traiter chaque requête de type Servlet au lieu de crée un processus pour chacune des requêtes.

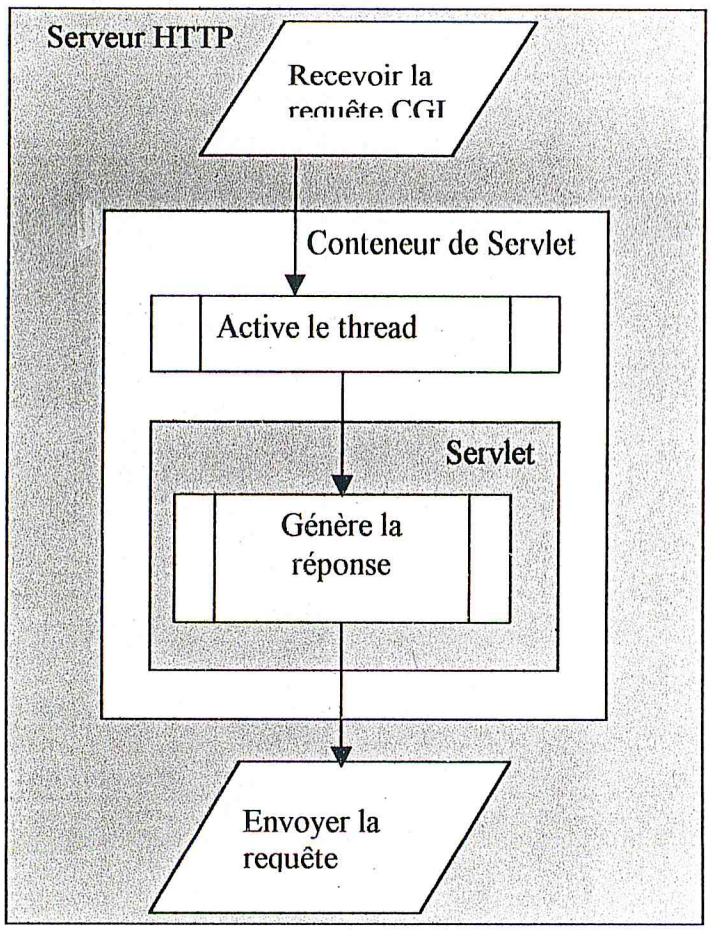


Figure 11 : processus serveur pour l'exécution des Servlets

➢ **Les avantages de Servlet**

- L'un des principaux atouts des Servlets est la réutilisation, permettant de créer des composants encapsulant des services similaires, afin de pouvoir les réutiliser dans des applications futures.
- Les Servlets fournissent un moyen d'améliorer des serveurs Web sur n'importe quelle plate-forme, d'autant plus que les Servlets sont indépendantes du serveur Web (contrairement aux modules Apache ou l'API Nescap Server) car il s'agit d'une technologie Java.
- Les Servlets étant une extension de la plate-forme Java, elles peuvent accéder à toutes les API Java. En effet, une Servlet Java peut envoyer et recevoir des messages électroniques, appeler des méthodes d'objets distants à l'aide de CORBA.
- Les Servlets exécutent dans un moteur de Servlet (parfois appelé conteneur de Servlets) utilisé pour établir le lien entre la Servlet et le serveur Web. Ainsi le programmeur n'a pas à se soucier du détail technique tel que la connexion au réseau, la mise en forme de la réponse à la norme HTTP, ...
- Les Servlets sont beaucoup plus performantes que les scripts, car il s'agit de pseudo-codes, chargés automatiquement lors de démarrage du serveur ou bien lors de la connexion du premier client. Les Servlets sont donc actives (résidente en mémoire) et prête à traiter les demandes des clients grâce à des *theards*. Cela permet donc une charge moins importante au navigateur du processeur du serveur (d'autant plus qu'un système de cache peut permettre de stocker les calculs déjà accomplis). Cela permet encore de prendre une place moins importante en mémoire, et d'éviter le blocage de la machine.

➢ **Les inconvénients de Servlet**

- L'inconvénient de taille à cette approche est que tout le contenu du document, aussi bien statique que dynamique, réside dans le code source du programme.
- Toute modification de document nécessite donc l'intervention d'un programmeur
- La source HTML a intégré aux programmes compiles.

➢ **JSP (Java Server Page)**

Introduire un contenu dynamique nécessite, le recoure à une certaine forme de programmation pour écrire comment ce contenu est généré. Le coût de la création du code et sa maintenance est plutôt élève et il faut donc minimiser autant que possible ce besoin en programmation. C'est précisément pour répondre à cette exigence, combinée à celle de Sun qui souhaitait obtenir un ensemble de fonctionnalités java complet et robuste pour le serveur, que ce système de patrons basés sur java - les JSP – a été engendré.

- **Définition**

Parmi les systèmes de patrons, les JSP se présentent comme un système hybride qui supporte deux styles d'ajout de contenu dynamique aux pages Web, des scripts peuvent être intègres dans les pages JSP contenant du code réel. Ce dernier est alors écrit en Java.

Les JSP supportent un ensemble de balises de style HTML qui interagissent avec des objets Java au niveau du serveur sans faire apparaître de code Java pur dans la page [DAU00].

➢ Principe de JSP

Le composant fondamental d'une application JSP basé sur les Servlets des JSP est une Servlet particulière souvent appelée << compilateur de pages >>. Le conteneur est configuré de façon à faire appel à cette Servlet pour tout requête avec des URLs correspondant à une extension des fichiers JSP. La tache de cette Servlet ne se limite pas à retrouver des pages JSP en réponse aux requêtes, Mais inclut également leur compilation. Ainsi, chaque page JSP est compilée en une Servlet dédiée à une page spécifique dont l'objectif et de générer un contenu dynamique spécifié dans le document JSP initial.

Par conséquent, chaque fois qu'un serveur http reçoit une requête pour une URL de type JSP celle-ci est envoyée au conteneur des JSP qui fait appel aux compilateurs de page pour la traiter. Si, pour un fichier JSP particulier, c'est la première fois que cette requête est reçue, cette Servlet compile le fichier en une Servlet.

Pour compiler une page, le compilateur de pages JSP analyse son contenu en recherchant des balises JSP. Pendant cette analyse, il transforme son contenu en un code source Java équivalent qui, lorsqu'il est exécuté généré les données en sortie spécifiées dans le contenu de fichier initial. Ainsi, le code HTML statique est transformé en du texte Java. Ce texte est inscrit sans modification et selon l'ordre initial dans le flot de sortie. Les balises JSP sont transformées en code Java pour générer le contenu dynamique ; les balises de type JavaBeans sont transformées en appels aux objets et propriétés correspondantes.

Ce code va être inséré dans la partie statique du code HTML initial aux emplacements spécifiés dans la page. En suite, ce code source est utilisé pour écrire les méthodes des Servlets. Lorsque le code de la Servlet a été entièrement construit, le compilateur de pages fait appel au compilateur Java pour compiler ce code source et ajoute la classe Java résultante au répertoire approprie dans les chemins des classes du conteneur de JSP.

Une fois que la Servlet de la page JSP compilée et en place, le compilateur de pages fait appel à cette Servlet pour générer la réponse à la requête initiale. Bien entendu, cette analyse, cette génération de code et cette compilation amènent une certaine surcharge. Fort heureusement, ces étapes ne sont nécessaires que la première fois que la requête pour une page JSP donnée est reçue.

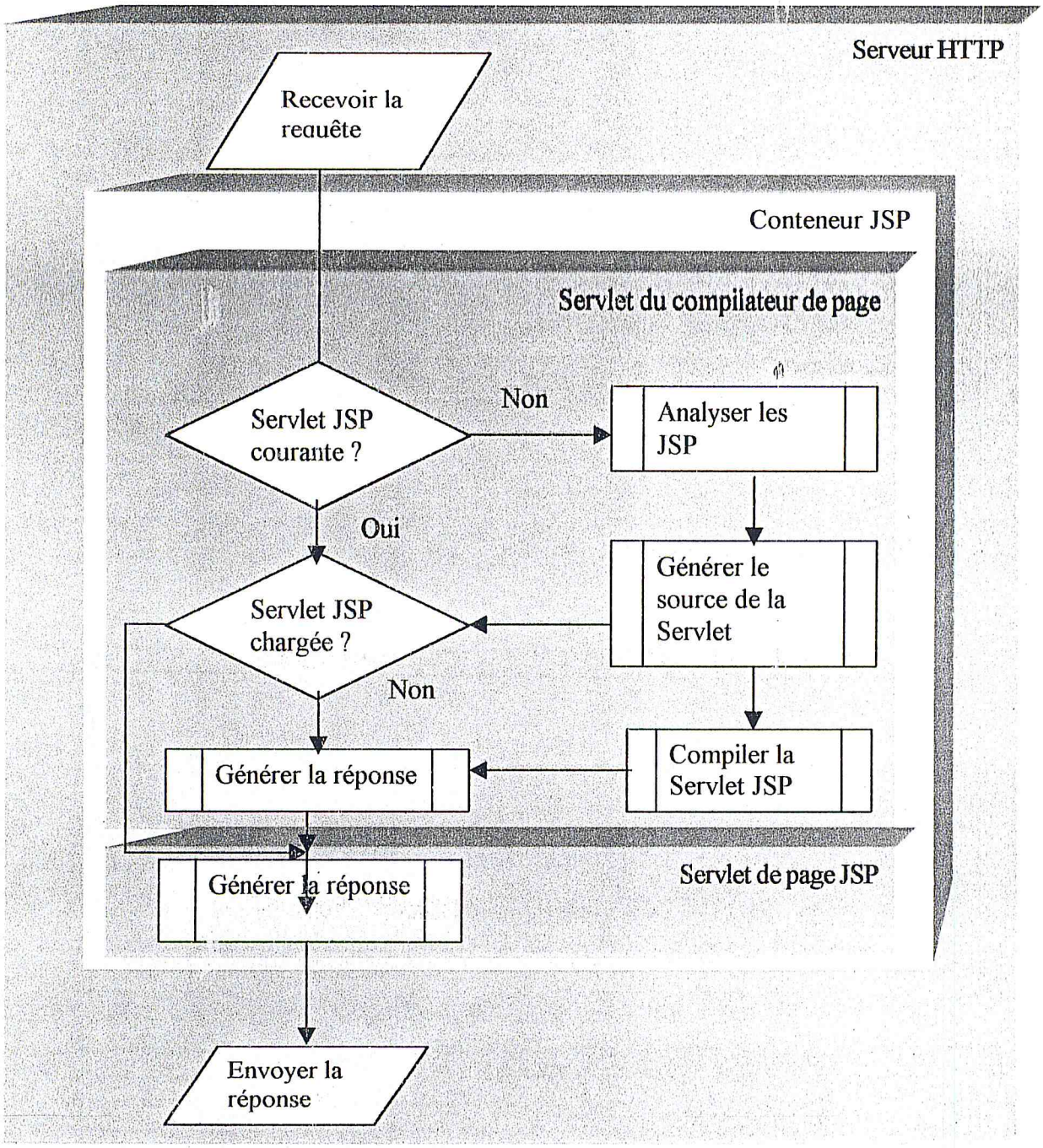


Figure 12 : Le processus serveur pour créer et exécuter des Servlets JSP

On observe que les requêtes du navigateur Web sont reçues par le serveur http. Ces requêtes JSP sont ensuite envoyées au compilateur de pages s'exécutant dans le conteneur de JSP. Ce dernier contrôle, alors si la Servlet de la page JSP demandée est à jour. Il s'agit de vérifier si une Servlet compilée existe pour cette page et si c'est le cas, si elle est plus récente que le contenu actuel de la page JSP. Sinon, le conteneur de JSP doit faire appel au processus d'analyse de la page, générer le code source et le compiler. La Servlet ainsi compiler et alors chargée dans le conteneur de Servlets. Si la Servlet de la page JSP est la Servlet courante, le conteneur doit s'assurer qu'elle

doit encore charger, car si l'on y recourt trop rarement elle peut avoir été déchargée après sa première création. Dans tous les cas le contrôle est ensuite transféré du compilateur de page à la Servlet de la page JSP, puis renvoyé au serveur http, pour retourner finalement au navigateur Web.

➢ **Les avantages de JSP**

- Une meilleure performance : comme les JSP sont implantées via des Servlets, ces derniers ont exécute dans un conteneur de Servlets, ce dernier est associe aux serveurs http classiques qui transmet toutes les requêtes relatives aux Servlets, de plusieurs requêtes pour une Servlet donnée où une JSP demeure incontournable, mais il est effectué dans des *threads* Java plutôt que dans véritables processus dédies (moins de surcharge et moins de ressources). Le conteneur de Servlets créant un réservoir de connexion lors de l'initialisation, il partage celle-ci entre différentes requêtes.
- La possibilité de réutilisation les composants : dans le cas des JSP, les composants écrits comme des objets JavaBeans, sont accessibles via une syntaxe de substitution (les balises de style HTML) qui se trouvent sur la page, ainsi que pour afficher et modifier leurs propriétés. L'avantage essentiel de la conception centrée sur les composants, c'est la réutilisabilité, telle que la réutilisabilité des composants est une propriété qui entraîne une amélioration directe de la productivité. De plus, les composants ne dépendent généralement pas de l'environnement d'exécution, par exemple, le même objet JavaBeans peut être utilisé dans une Servlet, une applique ou une page JSP. Multipliant ainsi les possibilités de réutilisabilité.
- La possibilité d'avoir une séparation de la présentation et l'implémentation : la prise en charge des objets JavaBeans dans les pages JSP permet de maintenir une séparation nette entre les données de présentation et l'implémentation des programmes. L'avantage de dissocier ces deux aspects tient à ce que les modifications apportées à l'un ne se répercutent pas sur l'autre.

  Les JSP s'appuient sur la syntaxe pour garantir cette séparation. La meilleure façon de maintenir la séparation entre la présentation et l'implémentation est de développer via HTML, un ensemble de balises personnalisées pour représenter les informations complexes. Ainsi, les balises personnalisées offrent une interface bien définie entre la présentation et l'implémentation sans contaminer les fichiers JSP avec du code d'implémentation ni les propriétés des JavaBeans avec le code de présentation.

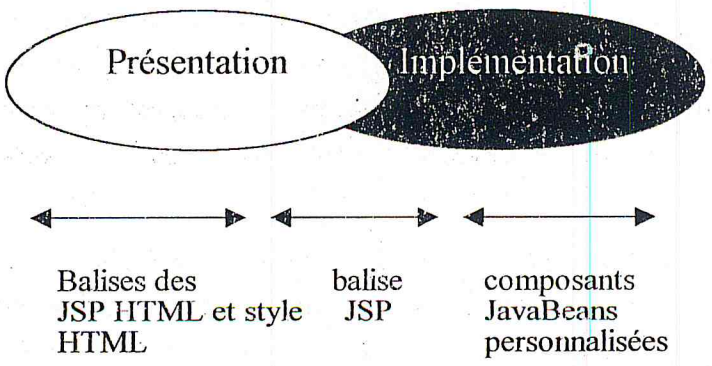


Figure 13 : Le support des propriétés des JSP pour la réalisation de la séparation de la présentation et de l'implémentation

- La répartition des tâches : la séparation par les JSP de la présentation et de l'implémentation à une conséquence importante, elle favorise une répartition claire des efforts de développement et de maintenance des applications pour la génération de contenu dynamique.

➢ **Les inconvénients de JSP**

- La maîtrise de la programmation devient nécessaire.
- Les fichiers des classes Java sont compilés en *bytecode* indépendant de la plate-forme plutôt qu'en un code assembleur natif. La JVM interprète le code et le transforme en instructions natives à l'exécution. Cette étape d'interprétation supplémentaire ralentit forcement l'exécution.

2.5 Conclusion

Après l'étude des avantages et des inconvénients de différentes technologies et d'architectures d'accès aux bases de données dans ce chapitre. On a choisi la technologie JSP sous une architecture trois tiers puisqu'elle présente de nombreux avantages par rapport aux autres langages de génération de contenu dynamique. Comme il s'agit d'une technologie basée sur Java, JSP profite de tous ce qu'offre ce langage pour le développement et le déploiement d'applications. En effet Java est un langage orienté objet fortement typé, permettant l'encapsulation, le traitement des exceptions et la gestion automatique de la mémoire.

D'autre part, grâce aux API normalisées pour les JSP et à la portabilité du *bytecode* compilé Java, on n'est pas limité à un seul type de plate-forme, de système d'exploitation. On peut changer les composants d'une page JSP comme on le souhaite.

De plus, JSP utilise l'ensemble de la plate forme JAVA sous-jacente ; par conséquent, les JSP peuvent directement tirer avantage de toutes les API Java.

La JSP a moins de surcharge et moins de ressources, lorsqu'elle exécute des requêtes dans des *threads* Java plutôt que dans des processus dédies, JSP est basée sur la conception centrée sur les composants comme JavaBeans qui donne un avantage essentiel, c'est la réutilisabilité. La prise en charge des objets JavaBeans permet de maintenir une séparation nette entre les données de présentation et l'implémentation des programmes, cette séparation à une conséquence importante entre toutes, elle favorise une réparation claire des efforts de développement et de maintenance des applications de la génération du contenu dynamique.

Les JSP offrent encore une réduction du temps de traitement puisque le code de la Servlet demeurera dans l'espace mémoire alloué à la JVM, par rapport aux systèmes de génération de contenus dynamique basés sur des lectures répétés de fichiers sur disque.

Chapitre 3 : Présentation du Java Server Page (JSP)

3.1 Introduction

Les JSP (*Java Server Pages*) sont un standard permettant de développer des applications Web interactives, c'est-à-dire dont le contenu est dynamique. C'est-à-dire qu'une page Web JSP (repérable par l'extension.*jsp*) aura un contenu pouvant être différent selon certains paramètres (des informations stockées dans une base de données, les préférences de l'utilisateur,...) tandis que page Web "classique" (dont l'extension est *.htm* ou *.html*) affichera continuellement la même information [DAU00].

Il s'agit d'un langage de script puissant exécuté du côté du serveur (au même titre que les scripts CGI, PHP, ASP,...) et non du côté client (les scripts écrits en JavaScript ou les applets Java s'exécutent dans le navigateur de la personne connectée à un site).

Les pages JSP sont intégrables au sein d'une page Web en HTML à l'aide de balises spéciales permettant au serveur Web de savoir que le code compris à l'intérieur de ces balises doit être interprété afin de renvoyer du code HTML au navigateur du client.

Ainsi, les Java Server Pages s'inscrivent dans une architecture 3-tiers, ce terme compliqué signifie qu'un serveur supportant les Java Server Pages peut servir d'intermédiaire (on parle généralement de serveur applicatif) entre le navigateur du client et une base de données (on parle généralement de serveur de données) en permettant un accès transparent à celle-ci. JSP fournit ainsi les éléments nécessaires à la connection au système de gestion de bases de données, à la manipulation des données grâce au langage SQL.

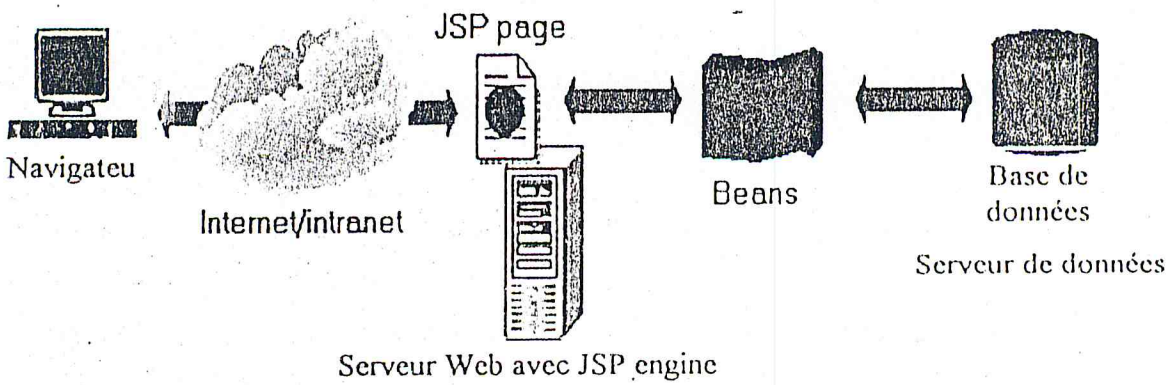


Figure 1 : Présentation de modèle JSP

3.2 Exécution de JSP

La condition nécessaire pour exécuter une JSP

Pour étudier JSP, il faut d'abord disposer d'un serveur Web, il faut donc une partie matérielle, à savoir un ordinateur accessible par l'Internet ou l'Internet d'entreprise, et la partie logicielle qui se présente sous la forme d'un serveur http exécutable sur le matériel. Les serveurs http les plus utilisés sont Apache (un logiciel libre), Netscape Entreprise Server et Microsoft Information Server.

Outre le serveur http, un logiciel implémentant le conteneur JSP est nécessaire. Par exemple Tomcat est une excellente plate-forme pour apprendre et expérimenter la technologie JSP, ainsi que pour développer et valider les applications.

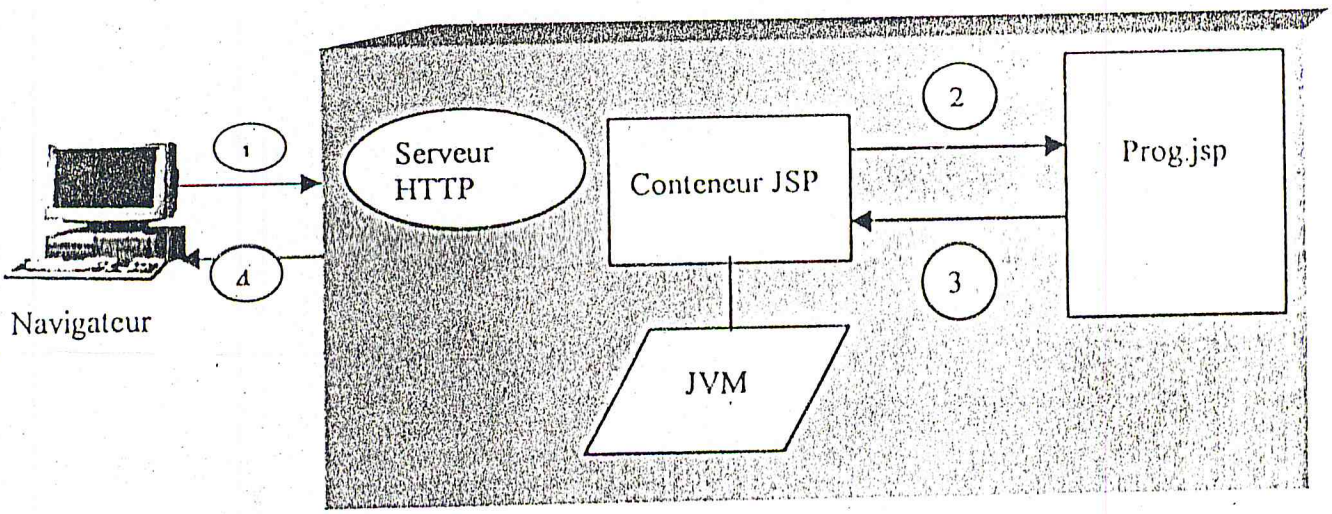
Bien sure, il faudra sur le serveur une Java Virtuel Machine tel que JDK (*Java Development Kit*) qui contient une JVM, un compilateur...

Le conteneur de JSP

Le conteneur de JSP est un processus séparé de serveur HTTP. La raison principale on est que le conteneur JSP est un processus Java qui exécute une JVM, alors que la plupart des serveurs HTTP sont d'écrits dans d'autres langages. Le point essentiel on l'occurrence, c'est que, pour les conteneurs de JSP associés aux serveurs HTTP classiques, il n'existe qu'un seul processus supplémentaire qui effectué le traitement de toutes les requêtes relatives aux JSP [LAU99].

Ce processus est lancé lorsque le serveur HTTP démarre et il s'exécute jusqu'à ce que le serveur s'arrête.

Exemple de déroulement de demande d'une page JSP



Etape 1 : client → serveur http +conteneur de JSP

L'exécution de toute page JSP commence par une requête réclamant une page JSP, il y a deux façons

- Le client charge une URL correspondant à une extension de fichier JSP. Et puis le navigateur construit une commande http GET

GET/ nom de page JSP http/1.1

- Le client saisit des informations dans un formulaire et clique sur envoi, en suite le navigateur construit une commande http POST avec les données du formulaire (chaîne de caractère).

POST/nom de page JSP http/1.1 Nom = nom saisit & Prénom =prénom saisit

Etape 2 : serveur http + conteneur de JSP → JSP

Le conteneur JSP

- A la 1ère invocation, crée une classe, la compile et l' instancie.
- Si la classe existe déjà et elle est récente que la page HTML, on l'utilise. Crée un thread
- Invoque une méthode de la classe contenant le code à exécuter.

La JSP

- S'exécute sur le serveur Web

Etape 3 : JSP → conteneur de JSP + serveur http

Une fois que la Servlet de la page JSP compilée et en place, le compilateur de pages fait appel à cette Servlet pour générer la réponse à la requête initiale.

Etape 4 : conteneur de JSP + serveur http → client

Le serveur propage le résultat au client dans une réponse http.

3.3 La syntaxe de JSP

JSP fournit quatre catégories de balises [DAU00]:

Les Directives

Il s'agit d'un ensemble de balises qui fournissent au conteneur de JSP des instructions spécifiques à une page pour lui indiquer comment doit être traité le document contenant les directives (par exemple elles peuvent être utilisées pour spécifier le langage de script de la page, insérer le conteneur d'une autre page ou indiquer la page utilise une bibliothèque de balises

personnalisées ..). Ces dernières n'affectent pas le traitement des requêtes isolées, mais plutôt des propriétés globales de la page JSP qui influent sur la transformation de la page en une Servlet.

Il y a trois directives :

- *La directive page :* <%@ page option...%>

La directive page est la directive JSP la plus complexe, principale élément parce qu'elle supporte une large variété d'attributs et de fonctionnalités associées.

Attribut	Valeur	Par défaut	Exemples
Info	Chaîne de caractères	Néant	Info= ''registration form.''
Language	Nom du langage de script	''Java''	Language=''Java''
ContentType	Type MIME ou un ensemble de caractères	Voir le premier exemple	ContentType=''text/html ; charset=ISO-8859-1'' ContentType=''text/xml''
Extends	Nom d'une class	Néant	Extends=''com.taglib.wdjsp.myJspPage''
Import	Noms de classe et/ou de paquetage	Néant	Import=''Java.net.URL'' Import=''Java.util.*,Java.text.*''
Session	Booléen	''true''	Session=''true''
Buffer	Taille du tampon ou false	''8KB''	Buffer=''12KB'' Buffer=''false''
AutoFlush	Booléen	''true''	Autoflush=''false''
IsThreadSafe	booléen	''true''	IsThreadSafe=''true''
ErrorPage	URL locale	Néant	ErrorPage=''results/failed.jsp''
IsErrorPage	Booléen	''false''	IsErrorPage=''false''

Tableau 1 : Les attributs de la directive *page*

- *La directive d'inclusion :* <%@ include file= '' '' %>

Cette directive permet aux auteurs des pages d'inclure le contenu d'un fichier. Le fichier à inclure est identifié par une URL locale et la directive est remplacée par le contenu du fichier spécifié.

- *La directive de la bibliothèque de balises :* <%@taglib uri=''...'' prefix=''tagPrefix'' %>

Cette directive est utilisée pour avertir le conteneur de JSP qu'une page se base sur une ou plusieurs bibliothèques de balises personnalisées. Une bibliothèque de balises est une collection de balises personnalisées qui peuvent être utilisées pour étendre les fonctionnalités de JSP au niveau d'une page.

Les éléments scripts

Alors que les directives JSP influencent le traitement de la page par le conteneur de JSP, les éléments de script permettrent aux développeurs d'intégrer directement du code dans la page JSP et, plus précisément, un code générant les sorties qui doivent apparaître dans le résultat renvoyé à l'utilisateur.

Les JSP fournissent trois types d'éléments de scripts :

- *Les déclarations :* <% ! Déclaration %>

Les déclarations sont utilisées pour définir des variables et des méthodes spécifiques à la page JSP, ces variables et ces méthodes peuvent être référencées par d'autres éléments des scripts dans la même page

- *Les expressions :* <%= expression %>

Les déclarations sont utilisées pour ajouter des variables et des méthodes à une page JSP, mais elles ne peuvent pas contribuer directement aux sorties de la page qui, en définitive, constituent l'objectif de la génération de contenus dynamique, cependant, l'élément expression des JSP est explicitement conçu pour la génération des sorties.

- *Les scriplets :* <%...code Java...%>

Les déclarations et les expressions supportent intentionnellement un nombre limite de types de script pour les scripts multi-usages, la construction JSP appropriée est la scriplet.

Les scriplets peuvent contenir des instructions d'un langage de script quelconque

Les Commentaires

Ils sont utilisés pour ajouter des textes explicatifs à une page JSP. Les JSP supportent différents styles de commentaire, en particulier celui qui permet de faire figurer ces commentaires sur la page généré. D'autres commentaires JSP ne sont visibles que sur le fichier JSP initiale ou dans le code source de la Servlet correspondante.

- *Commentaire de contenu* : < !-- comentaire -->

Ils sont appelés commentaire de contenu parce qu'ils utilisent la syntaxe de commentaire associer au type de contenu qui est généré par la page JSP. Pour écrire un commentaire qui sera inclus dans les sorties de la page JSP qui généré le contenu du web.

- *Commentaires JSP* : <% -- comentaire --%>

Les commentaires JSP ne dépendent pas du type de contenu qui doit être généré par la page. Ils sont également indépendants du langage de scripte utilise par la page. Ils ne sont visibles que dont le fichier JSP initiale.

- *Commentaire du langage de script* : <% /* comentaire */ %>

Enfin, les commentaires sont également introduit dans la page JSP à l'intérieur de scriptlets, en utilisant la syntaxe des commentaires native du langage de scripte.

Les Actions

Elles jouent trois rôles importants. D'abord les actions JSP interviennent dans le transfert des contrôles entre les pages. Ensuite, elles gèrent les appelets Java indépendamment du navigateur utilisé. Enfin elles permettent aux pages JSP d'interagir avec les composants JavaBeans résidant dans le serveur.

En autre, toutes les balises personnalisées définies en utilisant les bibliothèques deviennent des actions JSP.

- *Forwad* : L'action <jsp :forwad> est utilisé pour transférer d'une façon permanente le contrôle d'une page JSP vers un autre emplacement du serveur local. Le contenu généré par la page courante et supprime et le traitement de la requête est repris dans le nouvel emplacement.
- *Include* : L'action < jsp : include > permet aux autres de page d'inclure le contenu généré par un autre document local dans la sortie de la page courante à l'emplacement de la balise < jsp : include >. Contrairement à la balise < jsp : forword >, cette action est utilisée pour transférer temporairement le contrôle d'une page JSP à un autre emplacement du serveur local.
- *Plug- in* : L'action < jsp:plug-in > est utilisé pour générer de l'HTML spécifique au navigateur pour spécifier des applets Java se basant sur le plug-in Java de Sun Micro système.

3.4 Intégration des composants JavaBeans aux pages JSP

Les scriplets et les expressions JSP permettent aux développeurs d'ajouter des pages dynamiques aux pages web en intercalant du code java dans des pages HTML. On peut néanmoins y observer une absence de séparation claire entre la présentation et l'implémentation.

Les JSP, hormis les scripts, fournissent une autre approche centrée sur les composants pour la conception des pages dynamiques. Cette dernière permet aux développeurs d'interagir avec des composants Java, non seulement via du code Java, mais également en utilisant des balises de style HTML. Cette approche a pour autre avantage de faciliter le partage clair du travail entre les développeurs d'application et ceux qui s'occupent du contenu Web [FMC98].

3.4.1 Architecture à base des composants

Les composants sont des éléments indépendants et réutilisables qui encapsulent un comportement applicatif ou des données en un paquetage distinct. Ils effectuent des opérations sans révéler leurs mécanismes internes. L'abstraction et la réutilisation sont les deux principes essentiels de la possibilité de masquer la complexité d'un composant donné par l'adjonction d'une interface, à travers laquelle le composant interagit avec son environnement ou avec d'autres composants.

Les composants sont donc des éléments logiciels réutilisables qui peuvent être reliés pour construire une application. Un bon modèle de composants minimise le travail de codage des relations entre les différents éléments de l'application. Le principe des architectures de composants consiste à utiliser une interface commune pour manipuler des objets avec des outils de développement.

L'avantage de la réutilisation des composants se fait sentir, non seulement parce que différentes applications peuvent partager des composants, mais également parce qu'ils peuvent être réutilisés par la même application ou par des applications différentes.

Le modèle JSP est centré sur des composants Java appelés JavaBeans, qui sont un ensemble de classes, indépendant de toute architecture et de toute plate-forme, qui a permis de créer et d'utiliser des composants logiciels Java. Sa création se fait en créant une classe java qui implémente l'interface java.io.Sérializable et utilise les méthodes *get()* et *set()* pour exposer ses propriété.

3.4.2 Les balises des composants des JSP

Les JSP sont dotés d'un ensemble de balises Beans qui peuvent être utilisées pour placer les JavaBeans dans la page, et ensuite accéder à leurs propriétés. Contrairement aux scriptlets et aux expressions JSP, nul besoin d'être un programmeur Java pour concevoir des pages en utilisant des JavaBeans.

- **L'accès aux composants JSP**

Pour interagir avec un composant Java, il faut d'abord spécifier dans la page l'emplacement de la classe Java qui définit le composant et lui attribuer un nom.

Il suffit de maîtriser trois balises JSP simples :

- *La balise <jsp :useBean>*

<jsp :useBean> informe la page quand on le souhaite mettre à sa disposition un JavaBeans. Cette balise est utilisée pour créer un composant ou pour en obtenir un du serveur. Elle nécessite que des attributs, id et class.

La balise <jsp :useBean> se présente sous deux formes :

<jsp :useBean id=''bean name'' class =''classe name'' />

<jsp :useBean id=''bean name'' class =''classe name'' >

code d'initialisation

</jsp :useBean>

Toutes les valeurs d'attribut possibles que supporte la balise <jsp :useBean> sont présentés dans le tableau suivant :

Attribut	Valeur	Entité par défaut	Exemple de valeur
Id	Identificateur Java	Néant	My bean
Scope	Page, request, session, application	Page	Session
Class	Nom de la classe Java	Néant	Java.util.date
Type	Nom de la classe Java	Idem de classe	Com.manning.jsp.abstractPerson
BeanName	Une classe Java ou un beans serialisé	Néant	Com.manning.jsp.uscurrency.ser

Tableau 2 : Tableau de balise <jsp :useBean >

La balise <jsp :useBean> crée une instance du composant et lui affecte l'ID tel qu'il a été précisé dans l'attribut id. Lorsque le nouvel objet est créé, il effectue les taches ou les traitements de données que son concepteur lui a attribués. Tout cela est parti intégrante du processus ordinaire d'instanciation de JavaBeans et que ce dernier est devenu accessible à la page, on peut commencer à utiliser ses propriétés. En fonction de sa conception. Les propriétés peuvent soit tout simplement fournir des informations directes, soit exécuter des transactions complexes ou rechercher des informations dans une base de données. Quoi qu'il en soit, les résultats sont accessibles depuis les propriétés du JavaBeans.

- *La balise <jsp :getProperty>*

Pour accéder aux propriétés d'un JavaBeans depuis les JSP, on peut tout d'abord utiliser la balise <jsp :getProperty>. Contrairement à la balise <jsp :useBean> qui effectue une tâche en arrière-plan mais ne produit aucune sortie, la balise <jsp :getProperty> est vide et en attente de deux attributs, name et property.

Sa syntaxe est la suivante :

<jsp : getProperty name =''nm du bean '' property =''nom de la propriete'' />

Dans le code HTML résultant qui est affiché à l'exécution, la balise est remplacée par la valeur de la propriété du JavaBeans demandé. La propriété est d'abord convertie en texte par le conteneur de JSP.

- *La balise <jsp :setProperty>*

Cette balise permet de modifier les propriétés d'un composant. Elle peut être utilisée dans n'importe ou dans la page. A condition que le développeur du composant ait autorise l'accès en écriture à cette propriété.

La balise <jsp :setProperty> est l'usage simple. elle nécessite trois attributs :

L'attribut name designe le JavaBeans concerné.

L'attribut property indique les propriétés que l'on souhaite initialisé

L'attribut value est le texte que l'on veut affecter à la propriété

<jsp :setproperty name=''nom du bean'' property=''non de la propriete'' value=''la valeur de propriete'' />.

A l'exécution, la page JSP évalue les balises dans leur ordre d'apparition. Toute valeur de propriété que l'on initialise n'a prise en compte que dans les balises qui suivent la balise <jsp :set Property>. L'attribut value peut être spécifié sous la forme d'un texte ou calculé à l 'exécution au moyen d'expression JSP.

3.4.3 Contrôles de la portée d'un JavaBeans

Chaque fois que la page est demandée, une nouvelle instance de JavaBeans est créée et éventuellement modifiée. De tels JavaBeans sont mémorisés au niveau du serveur et réutilisés dans plusieurs pages ou dans plusieurs requêtes pour la même page. Cette possibilité permet de créer un JavaBeans une seule fois et puis d'y accéder pendant toute la duré de la visite du site par un utilisateur.

- **Accessibilité d'un JavaBeans et durée de vie**

L'accessibilité et la duré de vie d'un JavaBeans sont consolées via l'attribut scope de la balise <jsp :usebean >. Cet attribut peut avoir page, request, session ou application comme valeurs.

Un récapitulatif des effets de la portée d'une valeur sur l'accessibilité et la durée de vie d'un JavaBeans est présenté dans le tableau suivant :

Portée	accessibilité	Durée de vie
Page	Page courante unique	Jusqu'a ce que la page soit affichée ou le contrôle soit transféré à une nouvelle page
Request	Page courante et toute page incluse ou transférée	Jusqu'a la fin de traitement de la requête et la réponse et renvoyée à l'utilisateur
Session	La requête courante et toute requête ultérieure apparaissant dans la même fenêtre de navigateur	La durée de vie de la session de l'utilisateur
Application	La requête courante et toutes celle à venir qui font partie de la même application Web	La durée de vie de l'application

Tableau 3 : Les portées possibles d'un JavaBeans

La syntaxe de l'attribut scope est présentée ci-après. Un JavaBeans ne peut avoir qu'une seule valeur de portées. On ne peut pas les contribuer dans la mesure où elles sont par définition naturellement exclusives.

<jsp : useBean id=''beanName'' class=''classe '' scope= ''page/requeste/session/applicaation'' />

- **Les JavaBeans de portée page**

Un composant JSP ayant cette portée est le moins accessible et cela correspond à la durée de vie la plus courante. Chaque fois qu'une page et demandée par un visiteur, une instance de JavaBeans est créée. Si des balises d'initialisation ou des scriptlets sont présentées dans la balise, elles seront exécutées à chaque fois, il ne persiste pas entre les requêtes.

- **Les JavaBean de request**

Si l'on spécifie la valeur request pour l 'attribut de portée de la balise <jsp :usebean>, le conteneur de JSP va essayer de récupérer le JavaBeans de la requête elle-même. La durée de vie d'un JavaBeans qui a une portée de requête est la même que s'il avait une portée de page à ceci prés que son accessibilité étendue aux pages référencées dans les balises <jsp :include> et <jsp :forwad>. De cette façon l'utilité de la portée est double.

- **Les JavaBeans de session**

La portée de niveau session introduit la persistance des composants dans les JSP, est constitue l'une de ses constructions les plus puissantes. Contrairement reporté de requêtes et de pages, un JavaBeans ayant session comme valeur de l'attribut de porté doit perdurer au-delà de la durée de vie d'une seule requête, parce qu'il est place dans l'objet de session de l'utilisateur.

- **Les JavaBeans d'application**

Un JavaBeans de portée application possède un cycle de vie et une accessibilité plus importante qu'un JavaBeans de session. La durée de vie d'un Javabean d'application perdure au long de l'existence du conteneur de JSP, ce qui veut dire qu'ils ne sont pas récupérés jusqu'a l'arrêt du serveur.

La portée application est utilisée pour enregistrer des informations utiles tout au long de l'application et qui ne sont pas spécifiques à des pages précises ayant demandé l'accès au JavaBeans.

3.5 Intégration des pages JSP avec les Entreprises Java Beans (EJB)

3.5.1 Définitions des composants Entrepris Java Beans

Les EJB sont des composants réutilisables qui sont utilisés dans des architectures multi-tiers distribuées. Le développement d'applications devient facile, grâce à la réutilisation d'EJB que l'on a créés ou de composants prêts à l'emploi. Les EJB offrent des fonctions qui, jusqu'ici, étaient censées représenter la principale difficulté du développement d'application Web [DAU00].

3.5.2 Architecture

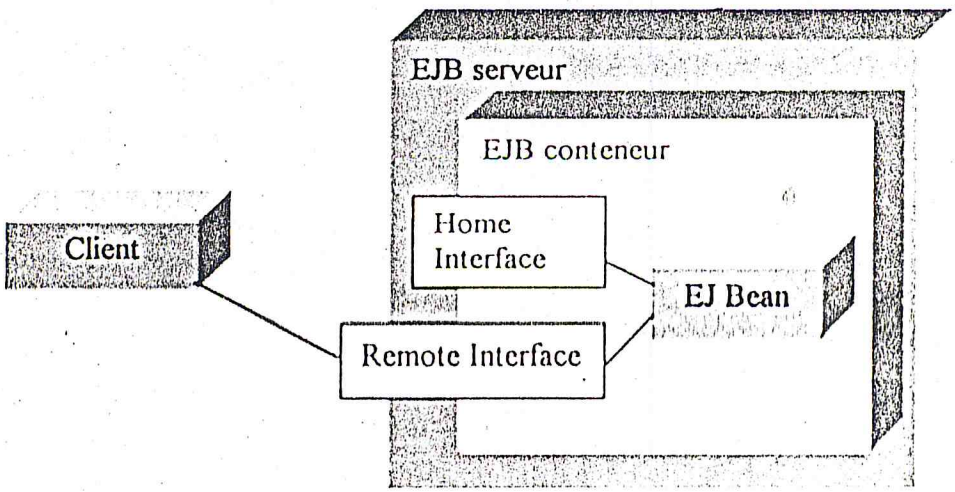


Figure 2 : Architecture d'EJB

- **Conteneur de EJB (JServer)**

C'est un petit serveur d'application incorporé à Oracle8i

- **Remote Interface**

Remote Interface est la vue du client de l'Entreprise JavaBeans, et la tâche de l'Entreprise JavaBeans developer est de déclarer cette interface en utilisant la syntaxe de Java RMI. il est de la responsabilité du fournisseur d'outils de récipient de l'Entreprise JavaBeans on lance la base de données par les commandes : de produire du code de cette interface.

- **Home Interface**

Home Interface est pour un Bean de session fournit le mécanisme par lequel le récipient crée de nouveaux Beans de session au nom du client. Home Interface, juste comme Remote Interface, est déclaré par le Bean developer en syntaxe de RMI, et encore, il est mis en application par les outils du fournisseur de récipient.

- **Bean de EJB**

Les Beans de l'Entreprise JavaBean sont schématisées comme suit :

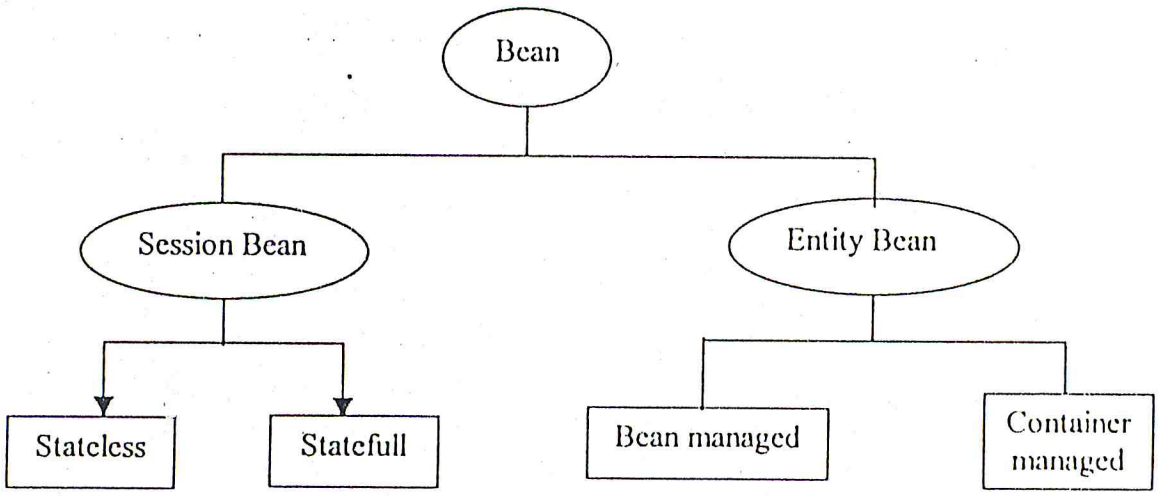


Figure 3 : Présentation des EJ Beans

Session Bean

1. Stateless session bean : c'est un bean "léger", et il ne conserve pas d'information entre deux appels successifs
2. Statefull session bean : c'est un bean "lourd".

Entity Bean

1. Bean managed persistence : La gestion de la persistance est à la charge du client
2. Container managed persistance : La persistance des données est gérée automatiquement par le conteneur.

- **Client**

Le client de l'Entreprise JavaBeans peut être une variété de choses : Par exemple, un Servlet, un applet ou peut-être un programme de C/C++.

Les importances de programme client sont montrées dans les points suivants:

- L'établissement du contexte initial de JNDI.
- Localisation du Home Interface de l'Entreprise JavaBean en utilisant JNDI.
- L'employement du Home Interface pour créer une instance de l'Entreprise JavaBean.
- L'utilisation de Remote Interface pour exécuter les méthodes de l'Entreprise JavaBeans

3.5.3 La différence entre un composant EJB et un composant JavaBeans

Ils n'ont techniquement pas grand chose en commun, même s'ils tentent d'appliquer le même principe de réutilisation des composants. Les composants EJB, à l'instar des composants JavaBeans déjà étudies, sont dans composants logiciels basés sur le langage Java.

Toutefois, ils suivent et implémentent un ensemble complètement différent de conventions et d'interfaces. Surtout, les EJB ne sont pas accessibles directement via des conteneurs de JavaBeans ni par l'intermédiaire de balises JSP.

L'intérêt des EJB est qu'ils, permettent d'encapsuler dans des composants réutilisables implémentent cote serveur une logique métier.

3.6 Les serveurs d'application

3.6.1 Le serveur JSERV

le " JServ " est le produit du projet d'Apache, il est installé dans le serveur de HTTP d'Apache. Le module de JServ étant installé est configuré avec le même principe que Apache.

3.6.2 Le serveur OSE (Oracle Servlet Engine)

Le moteur de Servlet d'oracle (OSE) fonctionne comme serveur Web spécialisé, conçu comme a serveur *scalable* de servlet à l'intérieur de la base de données *d'Oracle8* i.

OSE fonctionne à partir des services qui sont configurés pour écouter des end-point. Dans un service il y a un ou plusieurs domaines. Les domaines contiennent un ou plusieurs contextes de Servlet. Les contextes de Servlet représentent la couche Web application, contenant des Servlets et de diverses entrés de soutien.

Le domaine est configuré pour tracer les chemins virtuels aux contextes de Servlet. Des contextes de Servlet sont configurés pour tracer les chemins virtuels aux Servlets.

➢ **Créez Les Services**

On définissant le niveau de base d'un service, du nom de service, des groupes de propriété et de root, avec la commande de *createwebservice*. Si on crée un service de multi-domaine, cette commande définit finalement l'adresse IP et le nom d'host virtuel. On consulte n'importe quelle structure configurée de service avec la commande de *getproperties*.

getproperties /service/<service_name>

Avec la commande *addendpoint*, on peut ajouter un nouveau end-point dynamiquement avec un listener existant de base de données. On peut finalement créer un nouvel end-point statiquement.

➢ **Créez Les Domaines**

On crée un Web domaine avec la commande *createwebdomain*. Un nouveau Web domaine est possède le schéma courant qui exécutée la commande. Les Servlets contenus dans ce domaine, sont exécutés en tant que propriétaire de domaine. Chaque Web domaine est initialise avec le contexte de groupe.

- **Groupe De Contextes**

On peut voir les résultats de la structure configurée de domaine par la dactylographie:

getproperties<domainroot>/config

Le groupe de contexte énumère les tracés entre les chemins virtuels et les contextes qui manipulent des demandes de HTTP. C'est une liste de paires de nom-valeur.

- La pièce de nom est un chemin virtuel.
- La pièce de valeur est le nom d'un contexte, relativement au sous-répertoire de contextes du Web domaine.

C'est une liste de chemins virtuels tracés aux contextes appropriés de Servlet qui service des demandes de HTTP.

- **Groupe De MIME**

Le groupe de MIME énumère la prolongation aux tracés de type de MIME que le Web domaine soutient.

➢ **Créez Les Contextes de Servlet**

On crée les contextes de Servlet avec la commande *createcontext*. Un contexte de Servlet est contenu dans l'annuaire de contextes dans le root de domaine. On peut configurer le texte de contexte de Servlet pour soutenir des Stateless ou Stateful Servlets.

On peut voir les résultats de la structure configurée de contexte de Servlet par la dactylographie:

getproperties<domainroot>/contexts/<servlet_context>/config

On peut voir tracer virtuel dans l'objet de config de contexte de Servlet avec:

getproperties <domainroot>/config

➢ **Ajoutez Servlets**

Publié la Servlet de nom dans le contexte de Servlet avec la commande de *publishservlet*. Cette commande peut finalement associer un chemin virtuel au Servlet appelé

On peut voir les résultats de la structure configurée de service par la dactylographie:

getproperties <domainroot>/contexts/<servlet_context>/named_servlets/<servletA>

On peut voir tracer virtuel dans l'objet de configurer de contexte de Servlet avec:

getproperties <domainroot>/contexts/<servlet_context>/config

3.7 Intégration aux bases de données par JSP et JDBC

Les bases de données sont aujourd'hui intégrées à la plupart des sites Web, qu'elles alimentent dynamiquement en données. On peut citer comme exemple d'utilisation de base de donnée sur le Web la gestion des bandeaux publicitaires, des informations, des listes de contacts, etc.

Pour de tels besoins, les bases de données et les pages JSP forment une bonne combinaison : les bases de données relationnelles prennent en charge l'organisation à moindre coût de grands ensembles de données dynamiques, tandis que les pages JSP offrent un moyen simple de présenter ces données. On obtient ainsi des applications Web dynamiques bénéficiant à la fois de la puissance des bases de données relationnelles et de la souplesse de présentation des JSP.

JSP ne propose pas de balises prédéfinies pour la connectivité à des bases de données. Les concepteurs JSP ont en effet préféré tirer avantage de JDBC, une norme d'interface Java puissante et populaire pour les bases de données.

La communication avec des bases de données depuis une application JSP nécessite la présence d'un pilote (driver) propriétaire écrit pour l'API JDBC. En pratique, on masque l'accès à la base de donnée dans une Servlet ou un composant Java pour l'isoler des détails de présentation de la page JSP. Deux approches différentes sont donc possibles.

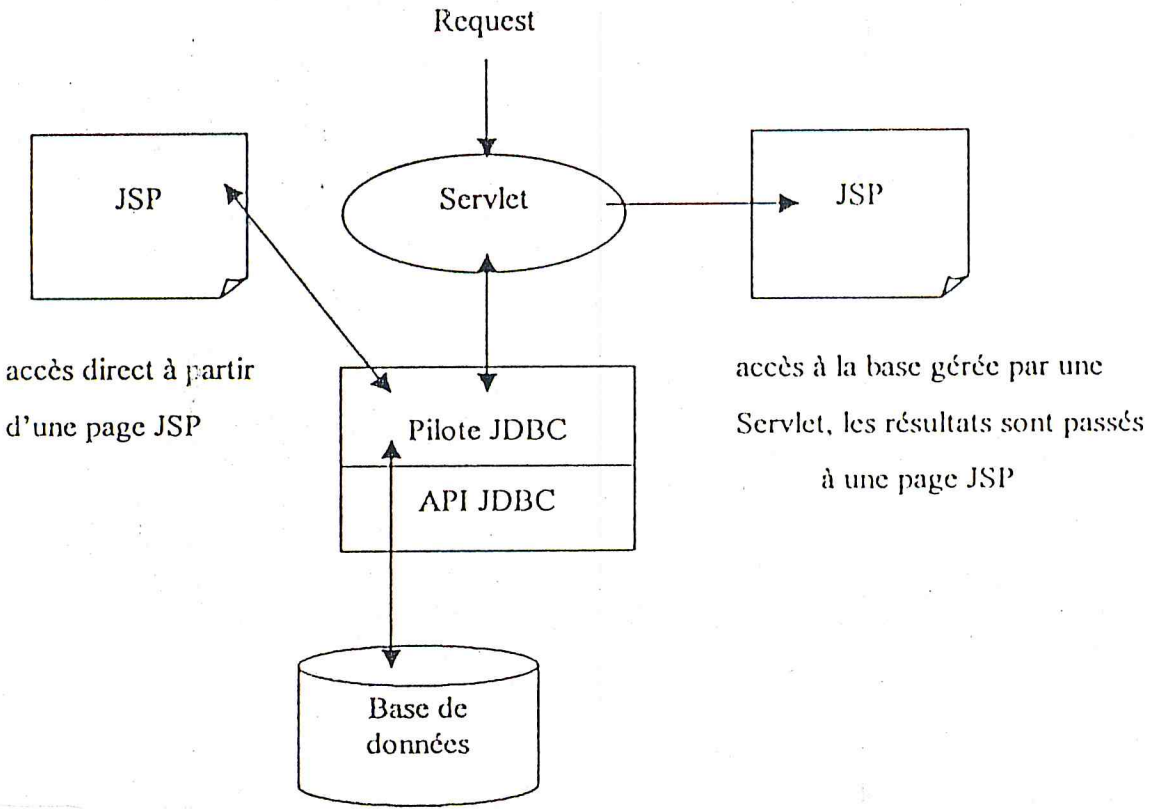


Figure 4 : Type d'accès possibles à des bases de données avec JSP

3.7.1 Intégration des données d'une table dans une page JSP

L'intégration d'une base de donnée à une application JSP repose principalement sur l'utilisation de composants JavaBeans, servant à faire le lien entre la base et les pages JSP. Chaque composant permet de récupérer ligne par ligne des données contenues dans une table. Grâce à l'analogie qui peut être établie entre une table de base de donnée et un composant : la structure d'une table, telle la classe d'un composant spécifie le type et le nom des données qu'elle contient ; les colonnes d'une table, telles les propriétés d'un composant permettre de stocker les valeurs de chaque instance du type contenu.

Les bases de données ont généralement pour rôle d'initialiser dynamiquement un composant Java à partir des informations d'une table.

3.7.2 Utilisation de JDBC

Les accès à une base de données doivent être effectue en sept étapes consécutives :

- Chargement d'un pilote JDBC.
- Définition de l'URL de connexion.
- Etablissement de la connexion.
- Création d'une instruction.

- Exécution de la requête.
- Traitement des résultats.
- Fermeture de la connexion.

Etape 1 : Chargement d'un pilote JDBC

Le pilote JDBC est un « logiciel » qui a connaissance des méthodes d'accès à une base de données. Ce pilote est disponible sous forme de classe Java et généralement dans un pacquage JAR. La première méthode consiste à charger cette classe. Pour cela, l'utilisation de la méthode statique Class.forName() est d'une aide précieuse. En effet, grâce à l'utilisation de cette classe, le programme a la possibilité de rester totalement indépendant de la base de données utilisée en conservant le nom du pilote dans un fichier de propriétés.

Exemple

```
Try {
    Class.forName(«'' oracle.jdbc.driver.oracleDriver") ;
}
Catch(ClassNotFoundException e ){
System.err.println( erreur de chargement du driver: + e );
}
```

Etape2 : Définition de l'URL de connexion

Afin de localiser le serveur ou la base de donnée, il est indispensable de spécifier une adresse sous forme d'URL de type « jdbc : » , le format exact de cette URL est dépendant du pilote JDBC utilisé.

Etape 3 : Etablissement de la connexion

Lors de la connexion, il est fort probable de spécifie un certain nombre de paramètres tels que le nom de l'utilisateur et un mot de passe.

Pour l'établissement de la connection, on utilise une classe de pacquage Java.sql, la classe DriverManager. Celle-ci disponible d'une méthode statique permettant d'obtenir une connexion à l'URL, la méthode getConnection() qui retourne un objet de type connexion.

Exemple

```
Import     Java.sql .* ;
.....
try { connection con=DriverManager.getConnection(url, userId, password) ;
```

}

catch (SQLException sqle) {system.err.println("erreur lors de la connexion: "+ sqle) ;}

Etape4 : Création d'une instruction

Afin d'accéder ou de modifier les informations contenues dans la base de données, il convient d'utiliser un objet de type Statement. Une instance de cet objet est retournée par la méthode Connexion.createStatement() comme ceci :

Exemple

Statement statement= con.createStatement () ;

Etape 5 : Exécution d'une requête

Pour effectuer une requête sur la base de données. Les requêtes de type sélection doivent utiliser un objet de type ResultSet afin de pouvoir en traiter le résultat.

Exemple

String query = " SELECT * FROM Employés ";

ResultSet resultset = statement.executeQuery (query);

Etape 6 : Traitement du résultat

L'objet précédemment utilisé, permet d'avoir un accès aux données résultantes de la requête en mode enregistrement par enregistrement. Un certain nombre d'accesseurs ont été défini de récupérer

unitairement les données de chacune des colonnes de notre enregistrement. Chaque accesseure permet de récupérer un résultat en spécifiant le numéro de la colonne désirée ou bien son nom.

L'objet ResultSet dispose aussi d'un certain nombre de méthodes permettant de naviguer d'un enregistrement à un autre. Ainsi, la méthode ResultSet.next() positionne sur l'enregistrement suivant et qui indique s'il existe un autre enregistrement à suivre ou bien si on positionne sur le dernier.

Exemple

While (resultset.next()) {

System.out.println (resultset.getString(1))};

Etape 7 : Fermeture de la connexion

Dernier étape enfin, la fermeture de la connexion de la base de données. Sans ce la, en risque de maintenir inutilement des ressources dont on peut plus utiliser. Pour fermer explicitement une connections, on utilise la méthode Connection.close().

Chapitre 4 : Etude conceptuelle et implémentation

4.1 Introduction

Après l'étude théorique des éléments qui semble contribuer dans l'élaboration de notre système, nous allons aborder la partie réalisation.

Ce chapitre qui a pour but d'analyser les besoins et de définir les objectifs à atteindre. On commence tout d'abord par une analyse du problème, ensuite on vous présente la solution retenue pour l'élaboration de la nouvelle architecture, finalement les étapes de réalisation.

L'une des principales missions de la CTI (Centre de Traitement Informatique) est d'exploiter les systèmes informatiques opérationnels, il existe à leurs niveaux une application Ressources Humaines qui représente comme on l'a vue précédemment les informations de chaque agent de la SONATRACH. Le département administration veut permettre aux personnels du département d'interroger la base de données Ressources Humaines à travers l'INTRANET en utilisant un navigateur Internet (browser) ce que le système actuel ne permet pas.

4.2 L'analyse du système actuel

L'application Ressources Humaines à été élaborée sous une architecture client / serveur dite classique, c'est une architecture à deux niveaux : Le premier niveau est le client qui comporte l'application (la logique applicative, et l'interface graphique utilisateur) et le deuxième niveau le serveur qui se charge de la gestion des données (BD).

4.3 Les failles du système actuel

- La non portabilité de l'application conçue sous cette architecture (l'application n'est pas multi-plateforme).
- Un client lourd implique une machine cliente puissante.
- Un accès peu optimisé à la base de données engendre un trafic plus important dans le réseau
- Maintenance difficile de l'application due à l'indépendance de l'interface graphique et des traitements
- Une formation des agents est nécessaire pour une bonne utilisation de l'application.
- Difficulté de mise à jour des postes clients mesurée en temps et en coût.
- La sécurité d'un système en architecture client-serveur gérée au niveau du SGBDR est une tache lourde.
- Le système organisationnel au niveau de la SONATRACH stipule que le traitement de la paye est centralisé, donc les agents des autres structure de la SONATRACH doivent se déplacer ver la direction générale pour apporter les imprimés des masques de saisies

comportant les Informations sur les agents : le traitement de la paye subi un retard (La diménuation de la mobilité).

4.4 La solution proposée

Prévoire une solution Intranet / base de données en implémentant l'application Ressources Humaines sous une architecture trois tiers, pour permettre aux utilisateurs d'accéder à la base de données en consultation qu'en mise à jours à partir d'un browser Web.

4.5 Les objectifs à atteindre

Afin de pouvoir réaliser une architecture qui réponde aux souhaits de la SONATRACH, et donc de résoudre les problèmes rencontrés face au système actuel, on fixant les objectifs suivants :

- Réaliser l'architecture de trois niveaux, dans un environnement fiable.
- Alléger le plus possible le client.
- Offrir une interface conviviale et interactive permettant d'assister les clients à la formulation de leurs requêtes via l'intranet.
- Evolutivité du système.
- Sécuriser l'accès à la base de données (la sécurité au niveau du serveur application : confidentialité, authentification).
- Réaliser une application 100% portable.
- Permettre aux structures de la SONATRACH d'accéder directement à la base de données à travers l'intranet et d'effectuer leurs mise à jours.
- Permettre une meilleure exploitation du SGBD Oracle8i.

4.6 Les outils retenus

Le principe d'une architecture trois-tiers est relativement simple: il consiste à séparer la réalisation des trois parties vues précédemment (stockage des données, logique applicative, présentation). Ici il s'agit de séparer leur implantation, cette séparation signifie qu'il est possible de déployer chaque partie sur un serveur indépendant, toutefois cela n'est pas obligatoire. La mise en place de ce type d'architecture permet dans tous les cas une plus grande évolutivité du système. Il est ainsi possible de commencer par déployer les deux serveurs sur la même machine, puis de déplacer le serveur applicatif sur une autre machine lorsque la charge devient excessive. Les éléments choisis pour la réalisation du système en architecture trois-tiers sont les suivants :

- Plate-forme : le système d'exploitation Linux.
- Le langage de programmation : Java est un langage orienté objet portable sécurisé…
- SGBD : Oracle8i est en fait la référence en matière de base de données relationnelle. Les possibilités d'administrations sont importantes. Il a était acquis par la SONATRACH sous

licence, et donc il est important de savoir l'exploiter, non seulement par l'hébergement de la base de données, mais à travers ses autres objets notamment OSE.

- Serveur WEB : Apache est le plus fiable des serveurs Web (apache avec l'extension mod-ssl).
- Le navigateur Web : Internet explorer ou Netscape fait donc office de client universel grâce à sa capacité à traiter le HTML et le protocole HTTP.
- Langage de conception de page Web : HTML
- Les JSP peuvent faire appel à des objets java réutilisable comme les Java Beans de type session statefull et les Entreprises Java Beans.
- L'environnement d'exécution des JSPs: notre application est développée dans le serveur d'application Jserv qu'est propriétaire à apache, après on transfère cette application dans le serveur le plus permanent Oracle Servlet Engine. OSE est un serveur de servlet intérieur à la base de données Oracle 8i, l'accès à la base de données est par conséquence optimisé, et il n'est donc pas nécessaire d'installer ce serveur, l'accès à la base de données est fait par le pilote JDBC, une API fourni avec Oracle8i. OSE permet une administration de la sécurité. On reste donc dans la famille Oracle.

4.7 Les étapes de réalisation

- **L'installation de la plate forme**

Lors de l'installation de Linux version7, en plus de la partition swap, et native, on créant une autre partition spécialement pour l'installation d'oracle8i sous Linux, qu'on appellera /u01.pour plus des détails voir [annexe A].

Les étapes d'installation d'Oracle8i sous Linux

Pour installer Oracle8i version 8.1.7 sous Linux on a besoin du jdk1.2 sous Linux, il faut donc copier le jdk1.2 dans le répertoire /usr/local

Avant de commencer l'installation d'oracle8i il faut :

1. Création de deux groupes oinstall et dba à partir du shell

   Commande : groupadd dba

   groupadd oinstall

2. Création d'un utilisateur Oracle qui aura comme groupe primaire oinstall, et groupe secondaire dba :

   Commande useradd –g oinstall -G oracle

3. Donner un mot de passe à Oracle

   Commande : passwd oracle

4. Se déconnecter en tant que root.
5. Se connecter avec le compte oracle

6. Copier l'image de cdrom dans un répertoire ou une partition crée lors du partitionnement.
7. Décompresser le fichier (création du répertoire Disk1)
8. Créer les premières variables d'environnement ORACLE_BASE et ORACLE_HOME

   Commande : ORACLE_BASE=/u01/aoo/oracle ensuite taperla commande : export

   Commande : ORACLE_HOME=$ ORACLE_BASE/product/8.1.7

   Ensuite taper la commande export

   La commande env affiche toutes les variables d'environnement existante
9. Entrer dans le répertoire et exécuter la commande./runInstaller

Durant l'installation d'Oracle8i sous Linux :

- Un lien vers le jdk1.2 est créé.
- Une base de données par défaut est crée, on a alors a lui donné un nom, ou ce qu'on appelle un SID.
- Le listener est configuré à écouter les requêtes vers la base.
- Le serveur Apache est configuré à écouter sur le port 7777 par défaut.

10. Installation terminée.

4.8 Le serveur de base de données Oracle8i

Nous avons conçu la base de données sous un serveur Oracle8i, c'est à ce niveau que sont stockées les données de l'entreprise.

Le noyau Oracle, étant un SGBD relationnel, sa fonction est de gérer les données de l'entreprise de façon à permettre à tous les utilisateurs d'accéder à leurs informations en toute sécurité et de garantir leur intégrité et leur cohérence. Comme les données de la base de données Ressources Humaines sont strictement confidentiel, on peut pas travailler sur cette base, on a créée une petite propre base de données, et Pour tester notre application on a créant un compte dans la base de donnée Ressources Humaines avec un nom « blida » et mot de passe « blida » par les étapes suivantes :

1) On lance le listener par la commande : *lsnrctl start*

2) On lance la base de données par la commande : *dbstart*

3) On commence SQL*Plus par la commande : *sqlplus*

4) Ouverture au compte d'administrateur de la base de données. :

Enter user-name : system

Enter password : manager

Par défaut, system et manager sont le user-name et le password pour l'administrateur de base de données. Nous avons créées un propre compte utilisateur « blida » par les étapes suivantes :

Dans SQL*Plus :

SQL>create user blida identified by blida;

Si l'exécution réussie, SQL*Plus répondra avec "User Created."
puis on entrer

SQL> grant resource,connect to blida;

Si l'exécution réussie, SQL*Plus répondra avec "Grant succeeded."

4. Enfin, pour sortir de SQL*Plus : *exit*

4.9 Le serveur Web Apache

Les serveurs Web sont des processus spécifiques (httpd), installés sur un ordinateur. Ils gèrent des informations présentées dans un format spécifique sous la forme de pages Web et stockées sous le nom de documents HTML.

Les serveurs Web traitent les requêtes adressées par le client Web (browser) et mettent à sa disposition les pages Web réclamées. Les pages Web demandées par le navigateur sont transférées à l'ordinateur local sur lequel il est installé, et d'où sont parties les requêtes.

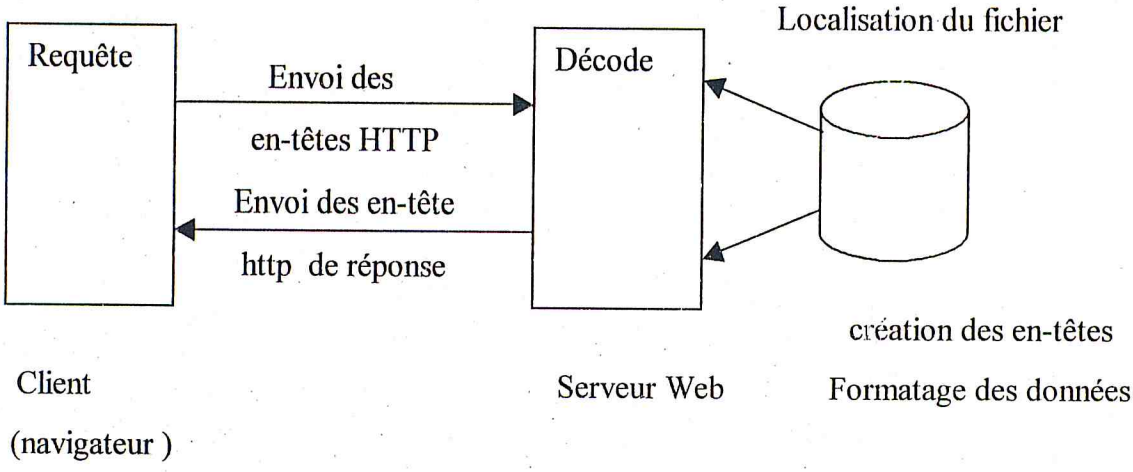


Figure 1 : Présentation de fonctionnement http

Nous avons implémenté un serveur Apache qui, en tant que logiciel libre, est gratuit, d'autre part Apache est un système performant, fiable et en perpétuelle évolution. Développé par un groupe de travail indépendant. Ce serveur devrait être installé et configuré automatiquement lors de l'installation sous Linux.

4.10 Les configurations nécessaires

On peut des a présent démarré le listener par la commande : lsnrctl start, démarrer la base de données par la commande : dbstart et le serveur Web Apache par la commande : apachectl start

- **La configuration du serveur Apache**

il est nécessaire de spécifier l'adresse de bouclage127.1.1.0, et le port d'écoute à 7777

Remarque :

Les endpoints entre 0 et 1024 sont accessible uniquement par le compte root, et de 1024 à plus sont accessible par les comptes utilisateurs.

Il est possible de déployer notre application sur le serveur Web apache et le conteneur de servlet Jserv fourni par oracle.

L'index serra enregistrée dans un répertoire htdocs sous Apache, et accédée a partit du navigateur par l'adresse http: //localhost : 7777, où localhost est le nom de la machine et 7777 et le port d'écoute.

4.11 Ecriture d'une page JSP

Après avoir installé et configuré les packages nécessaire pour notre application, il nous reste à concevoir des pages JSPs.

- **Page JSP de consultation d'une table**

Pour consulter une table d'une base de données il faut faire la connection à cette base.

```
Connection  conn = DriverManager.getConnection(connStr,
                   "blida", "blida");
```

« ConnStr » est l'URL qu'on a programmé comme suit :

```
String connStr=request.getParameter("connStr");
if (connStr==null) {
connStr=(String)session.getValue("connStr");
} else {
session.putValue("connStr",connStr);
 }
if (connStr==null) {%>
<jsp:forward page="setconn.jsp" />

session.putValue("connStr",connStr);
 }
```

Maintenant on consulte les tables de notre base de données, il convient d'utiliser un objet de type statement. Une instance de cet objet est retournée par la méthode Connexion.createStatement() comme ceci :

```
Statement stmt = conn.createStatement ();
```

Pour effectuer une requête sur la base de données. Les requêtes de type sélection doivent utiliser un objet de type resultSet afin de pouvoir traiter le résultat.

```
ResultSet rset = stmt.executeQuery ("SELECT * " +
                                    "FROM blida.AGENT ORDER BY NOM");
```

pour fermer la connection on utilise les méthodes suivantes :

```
rset.close();
stmt.close();
conn.close();
```

On regroupe tous cela dans la page JSP suivante :

La page JSP : « consult.jsp »

```
<%@ page import="java.sql.*" %> // le package d'etablissement de la connection

 <% --Définition de l'URL de connexion  --%>
<%
String connStr=request.getParameter("connStr");
if (connStr==null) {
connStr=(String)session.getValue("connStr");
} else {
session.putValue("connStr",connStr);
 }
if (connStr==null) { %>
<jsp:forward page="setconn.jsp" />
<%
}
%>
<HTML>
    <HEAD>
          <TITLE>
  Consultation des agents
          </TITLE>
    </HEAD>
<BODY BGCOLOR=EOFFFO>
<H1> <B>
    Consultation des agents
</H1></B>
<HR>
 <B> je veut consulter la table agent de schema blida .</B>
 <P>
 <% --Etablissement de la connexion  --%>

<%
try {
   Connection  conn = DriverManager.getConnection(connStr,
                      "blida", "blida");
  Statement stmt = conn.createStatement ();
  ResultSet rset = stmt.executeQuery ("SELECT * " +
                     "FROM blida.AGENT ORDER BY NOM");
if (rset.next()) { %> // pour traiter les resultas
```

```
<TABLE BORDER=1 BGCOLOR="C0C0C0">
<TH WIDTH=100 BGCOLOR="white"> <I>Matricule</I> </TH>
<TH WIDTH=200 BGCOLOR="white"> <I>Nom</I> </TH>
<TH WIDTH=300 BGCOLOR="white"> <I>Prénom</I> </TH>
<TR> <TD ALIGN=CENTER> <%= rset.getString(1) %> </TD>
<TD ALIGN=CENTER> <%= rset.getString(2) %> </TD>
<TD ALIGN=CENTER> <%= rset.getString(3) %> </TD>
</TR>
<%
while (rset.next()) {
%>
<TR> <TD ALIGN=CENTER> <%= rset.getString(1) %> </TD>
<TD ALIGN=CENTER> <%= rset.getString(2) %> </TD>
<TD ALIGN=CENTER> <%= rset.getString(3) %> </TD>
</TR>

<% }
%>
</TABLE>
<% }
else {
%>
<P>désolé, il n'y a pas des enregistrements </P>
<%
    }
rset.close();
stmt.close();
} catch (SQLException e) {
out.println("<P>" + "il y a un erreur durant l'exécution d'une requête :");
out.println ("<PRE>" + e + "</PRE> \n <P>");
}
%>
</BODY>
</HTML>
```

Voici le résultat de cette page :

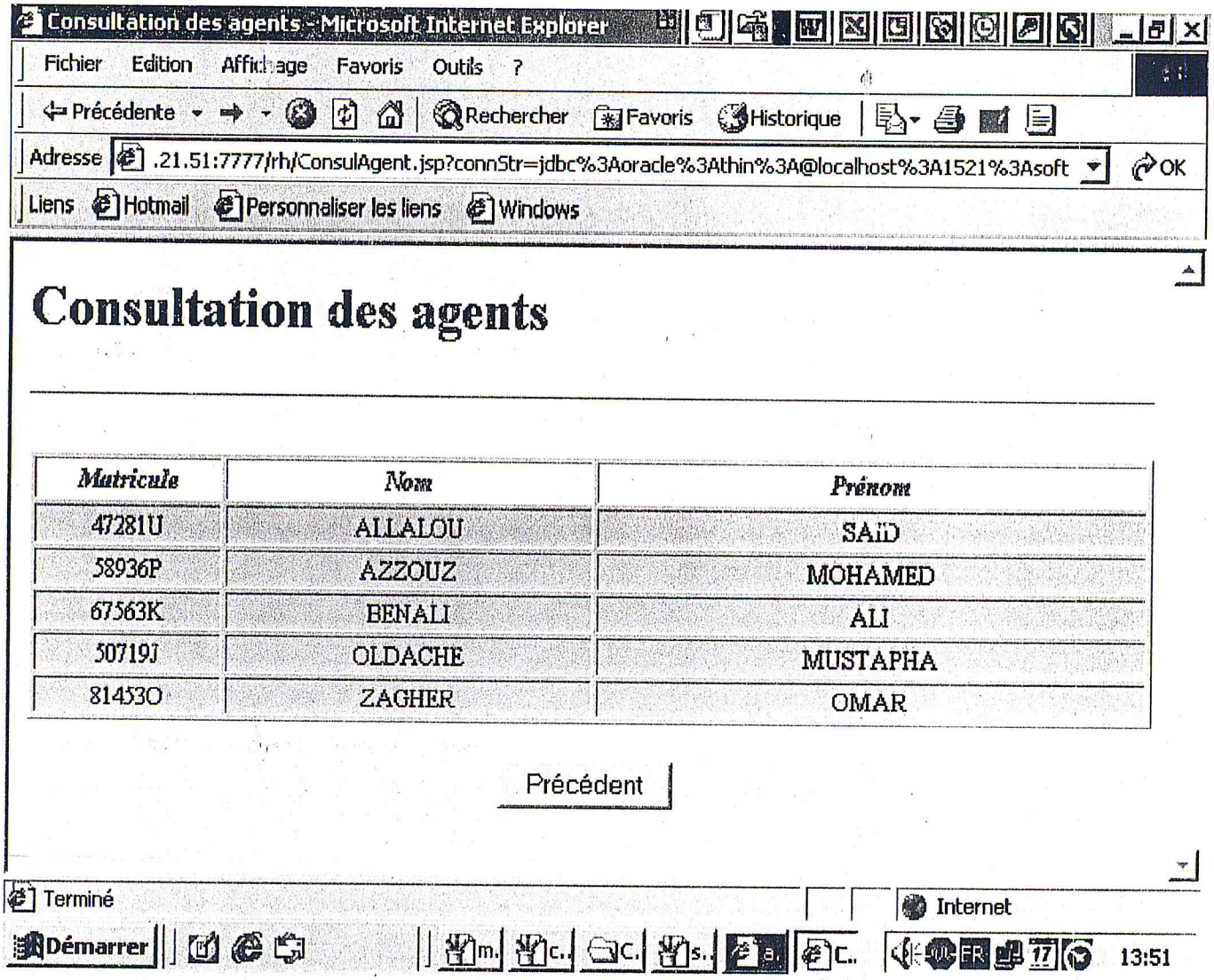


Figure 2 : Résultat de la page JSP « consult.jsp »

- **Page JSP de mise à jour**

Dans notre application, la mise à jour consiste en trois opérations : l'insertion, la modification, et la suppression. L'appel des différents pages à partir de la page MAJ.jsp peut être représenté par le même principe de consultation.

4.12 Création d'une page HTML pour invoquer la page

Après avoir créé les pages nécessaires à la consultation et la manipulation des informations de la base, on doit créer une ou plusieurs pages HTML pour référencer la page en question.

La page HTML doit contenir l'URL suivante :

« http://www.rh.sonatrach.dz: 7777/consulAgent.jsp »

Domaine numéro de port Nom de page

Une fois les URLs définies l'utilisateur peut ainsi voir de son navigateur le résultat de la page invoquée.

4.13 Utilisation des JavaBeans dans une page JSP

Pour la séparation nette entre l'implémentation et la présentation dans la page JSP, on intégrant des composants JavaBeans. Les Beans utilisent des méthodes setXXX () et get XXX() pour chacun de ses attributs XXX.

- **Un JavaBean de consultation d'une table**

Page JSP : « empBean.java »

```
import java.sql.*;
public class agentBean {
private String MAT;
private String NOM;
private String PREN;
public agentBean(){
   this.MAT=" ";
   this.NOM=" ";
   this.PREN=" ";
.....
}
public void setMAT(String mat){// maître des valeurs à l'attribut MAT
  this.MAT=mat;}
public String getMAT(){// récupérer les valeurs de l'attribut MAT
  return this.MAT;}
public void setNOM(String nom){
  this.NOM=nom;}
public String getNOM(){
  return this.NOM;}
public void setPREN(String pren){
  this.PREN=pren;}
public String getPREN(){
  return this.PREN;}
}.....
```

4.14 Le mécanisme d'exécution des JSPs avec les JavaBeans

Les Java Beans seront stockés dans un répertoire en forme de fichiers.class dans un emplacement spécifié dans un fichier.jar. Le chemin de ce fichier.jar doit figuré dans le fichier de configuration d'apache (jserv/etc/jser.properties).

Exemple :

Nous avant à écrire le bean qui consulté une table, on aura empBean.java, après compilation, on aura empBean.class. Le Bean est instancié par une page JSP de notre application : Il faut importer le package, et instancier le Bean avec un id, et spécifier la classe d'instanciation. Lors de la compilation le package est recherché dans le fichier /etc de Jser exactement dans jserv.properties.

A l'exécution deux objets implicites sont crées : requête et session. Ces deux objets possèdent donc un paramètre de connexion, c'est la chaîne de caractère de connexion à la base de données.

Pour appelé ce Bean dans la page de consultation en utilise la syntaxe suivante :

```
<jsp:useBean id="empB" class="empBean" scope="session">
<jsp :setproperty name=''empB'' property=''MAT'' value='rset.getString(1)' >
<jsp :setproperty name=''empB'' property=''NOM'' value='rset.getString(2)' >
<jsp :setproperty name=''empB'' property=''PREN'' value='rset.getString(3)' >
……
</jsp:useBean>
<jsp :getproperty name=''empB'' property=''PREN'' />
……
```

4.15 Transfert l'application au serveur OSE

- **Création d'un service**

```
l/ade/jofernan_adejis/oracle/bin/sess_sh @
# USAGE: @blidaService.ssh
echo "Creating HTTP Service ..."
createwebservice -root /blidaRoot blidaService
echo "Adding endpoints"
addendpoint -port 8088 -register blidaService blidaPublic
 addendpoint -port 9088 -register -ssl blidaService blidaSSL
addendpoint -net8 -register blidaService blidaNet8
echo "granting ownership to BLIDA"
chown -R BLIDA /blidaRoot
echo "Service creation complete"
```

- **Création des Contextes de Servlet**

```
 echo "Creating webdomain and default context ..."
createwebdomain -docroot &1 /blidaRoot
echo Binding service event log...
bind /blidaRoot/servicelogs/event -rebind \
-c SYS:oracle.aurora.namespace.rdbms.TableStream \
-f oracle.aurora.namespace.PublishedObjectFactory \
-string table.name blida.EVENT$LOG
echo and error log
bind /blidaRoot/servicelogs/error -rebind \
-c SYS:oracle.aurora.namespace.rdbms.TableStream \
-f oracle.aurora.namespace.PublishedObjectFactory \
-string table.name blida.ERROR$LOG
```

```
echo "Creating additional servlet context ..."
createcontext -virtualpath /groove -docroot &2 /blidaRoot blidaContext
cd /blidaRoot/contexts/blidaContext
echo "setup default welcome page"
addgroupentry config context.properties context.welcome.names welcome.html
echo create HTTP access servlet context logs
cd /blidaRoot/contexts
accesslog -table BLIDA.HTTP$LOG$ blidaContext
accesslog -table BLIDA.HTTP$LOG$ default
```

- **Publication du Servlets**

```
    echo publish serlvet to view access log
publishservlet -virtualpath /http_log blidaContext httpLog_viewer
SYS:oracle.aurora.mts.http.servlet.HttpRdbmsLogServlet -properties
table.name=BLIDA.HTTP$LOG$
echo publish serlvet to view error log
publishservlet -virtualpath /error_log blidaContext error_log_viewer
SYS:oracle.aurora.mts.http.servlet.TableReaderServlet -properties
table.name=BLIDA.ERROR$LOG
echo publish serlvet to view event log
publishservlet -virtualpath /event_log blidaContext event_log_viewer
SYS:oracle.aurora.mts.http.servlet.TableReaderServlet -properties
table.name=BLIDA.EVENT$LOG
```

4.16 Sécurité de l'application

Notre application comporte trois niveaux de sécurité :

- **Sécurité du serveur**

Notre serveur d'application est sécurisé par :

- FIRWAL : est un matériel spécial installé dans le réseau.
- Fermeture des port inutile et ouverture du port 80, 443, et les ports de listener si Oracle est installé dans une machine appart que le serveur Apache.
- TRIPWIRE : créée une base de données de fichier système thermite, qui peut renseigner à tout moment un quelque modification de ces fichiers.
- SXID : il sécurise les programmes qui permet devenir momentanément root et avoir les même droit d'accès que root.

➢ **Sécurité des transactions**

Les échanges des informations sont sécurisés par :

- **SSL (Secure Socket Layer)**

La technique de SSL suite les étapes suivantes:

1. Le message à signer est passé dans une fonction *hash* à sens unique pour transformer l'ensemble du message (longueur variable) en une chaîne de caractère fixe (longueur fixe et de petite taille) : le *message digest*. La fonction ne garantit que messages différents ne peuvent pas produire le même résultat.
2. Le message digest est chiffré avec la clé privée de l'envoyeur et ajouté à la fin du message. On dit alors que le message est signé.
3. Lorsque le message est reçu par le récepteur, celui-ci applique alors la même fonction hash sur le message reçu et obtient un message digest.
4. Le récepteur déchiffre la signature avec la clé publique de l'émetteur et compare le résultat avec le message digest obtenu dans l'étape 3. si les valeurs sont identiques, l'intégrité du message est garantie et le récepteur est sur que le message provient de celui qui a fourni sa clé publique.

- **Les certificats**

Le mécanisme d'obtention d'un certificat est le suivant :

1. Le client s'adresse à un organisme de certificat en lui fournissant sa clé publique et les informations concernant (nom, société, adresse messagerie....)
2. L'organisme de certificat vérifie les renseignements fournis par le client quant à son identité.
3. L'organisme de certificat établit un certificat sous forme de fichier.

➢ **Sécurité des clients**

Comme les informations de l'application Ressources Humaines est strictement confidentielles, il s'agit de la sécurité d'Oracle pour l'accès à la base de données et au serveur d'application. Ceci on a le fait par création d'un mot de passe au niveau du serveur Apache.

Chapitre 5 : Présentation du logiciel

5.1 Introduction

Les applications implémentées sous la plate-forme Intranet fournissent des interfaces graphiques pour les différents utilisateurs de l'entreprise. Ces interfaces permettent aux utilisateurs de consulter et de manipuler facilement les données à distance.

Dans ce chapitre, nous allons présenter les interfaces graphiques(fenêtres, boite de dialogue) et ses différentes fonctionnalités.

5.2 Présentation du site Web

Une page Web intitulée « page d'accueil Ressources Humaine » est la page HTML téléchargée aux utilisateurs qui se connectent au site Web de la division Ressources Humaines à l'adresse suivant :

http://Sonatrach-RH.dz.

Lorsque l'utilisateur tape cette adresse, la fenêtre de mot passe du serveur est automatiquement générée :

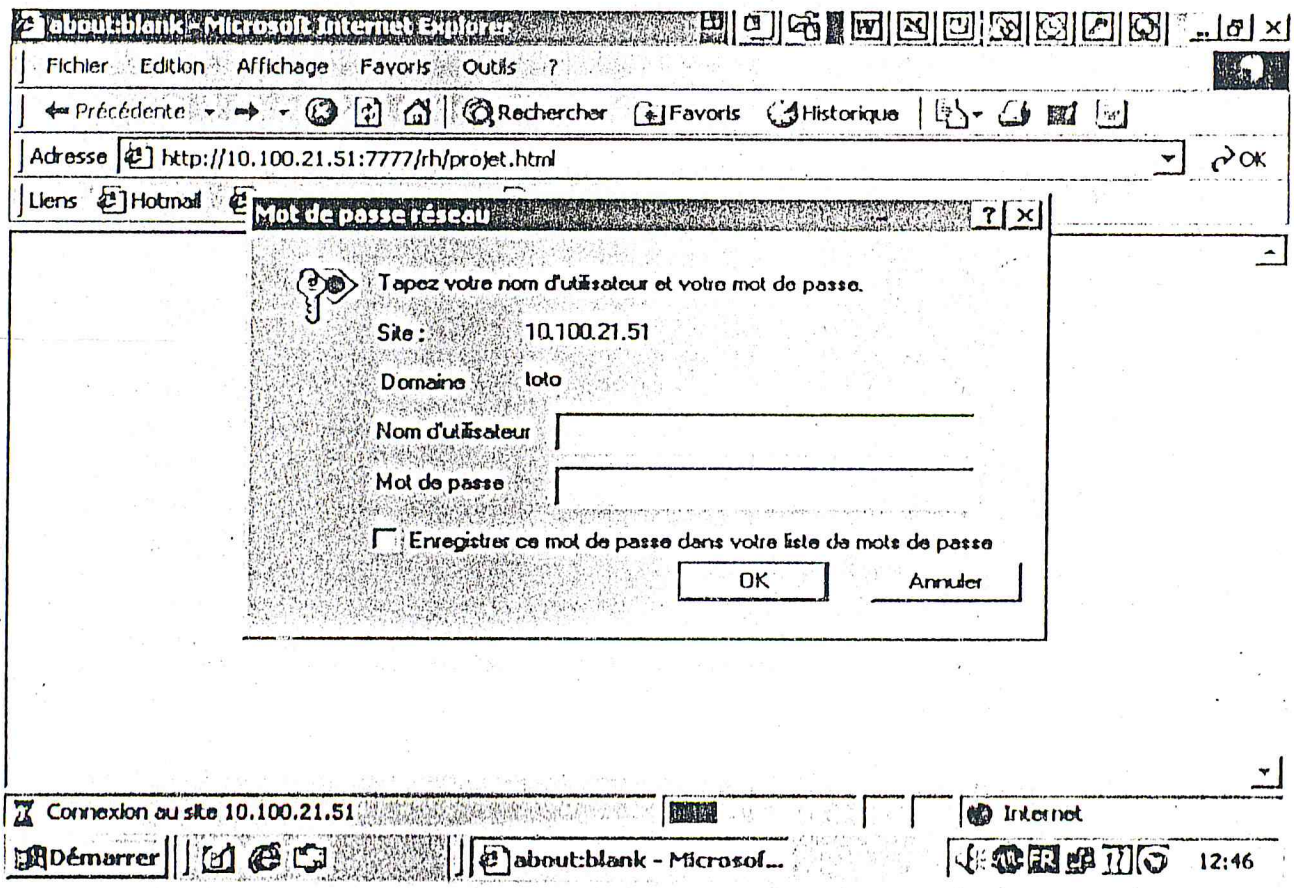


Figure 1 : La fenêtre de mot de passe .

Après la vérification du nom et du mot de passe de l'utilisateur sur le réseau, la page qui contient des liens hypertextes vers l'application de consultation et de mise à jour est affichée.

la figure ci – dessous représente la page d'accueil de division Ressources Humaines :

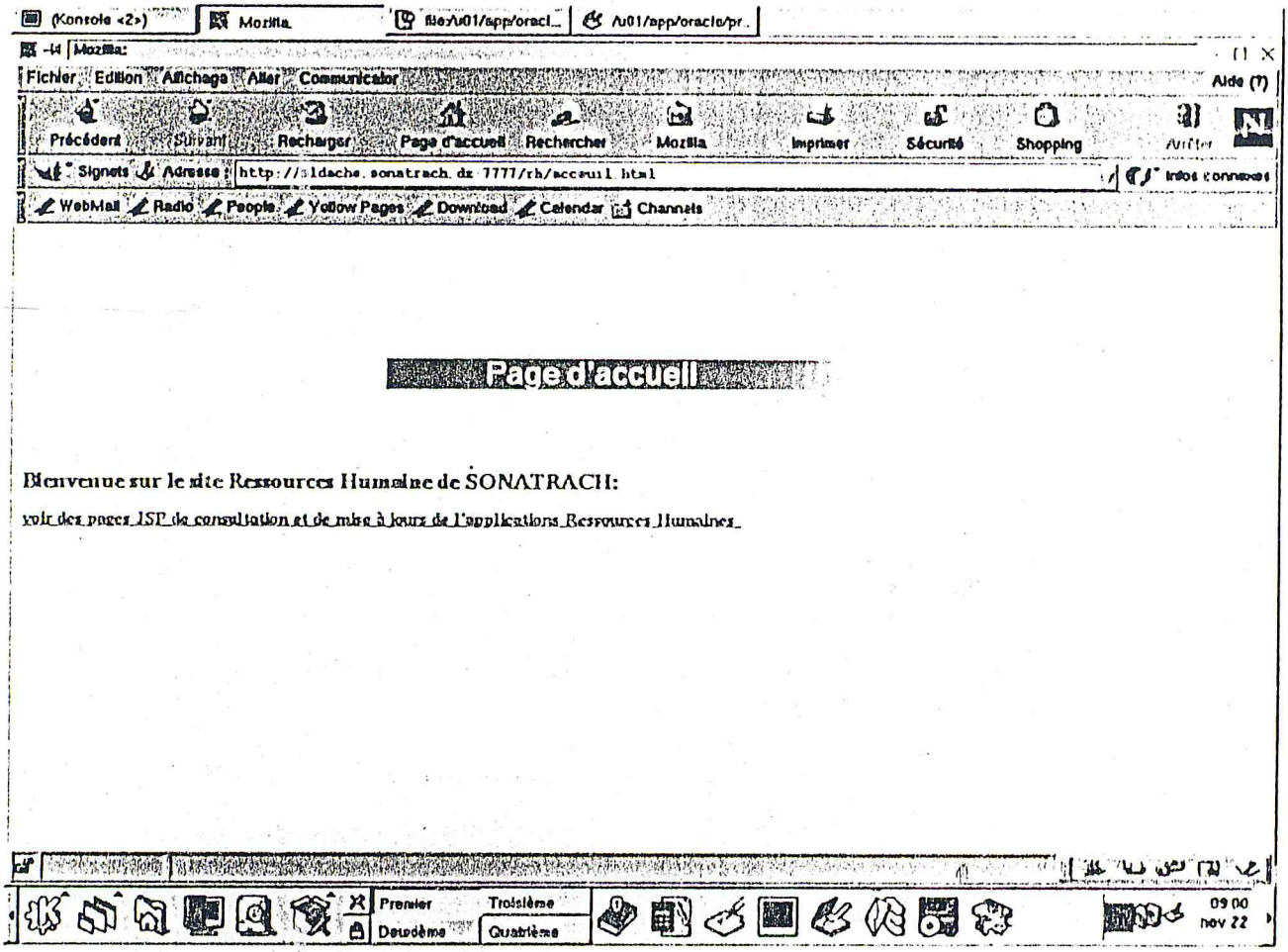


Figure 2 : La page d'accueil de division Ressources Humaines.

2. Application Intranet de consultation et de mise à jour

Après avoir cliqué sur le lien hypertexte de la page d'accueil, une autre page HTML est référencée.

la figure ci – dessous représente la page Web contenant deux liens, l'un pour la consultation, l'autre pour la mise à jour :

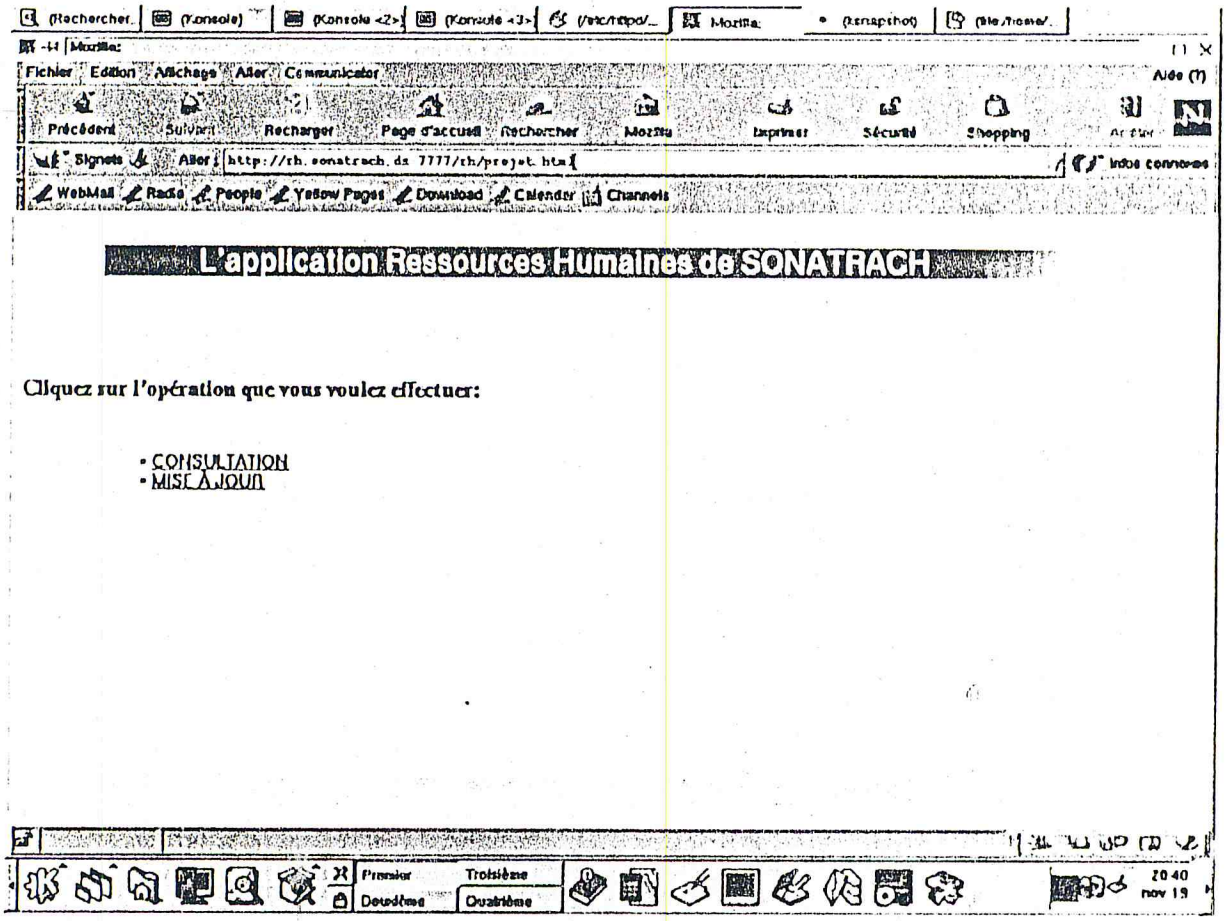


Figure 3 : Page Web représentant notre application.

- **Application de consultation :**

En cliquant sur le lient « consultation » une autre page HTML est référencée, cette dernière contient les différents liens qui font référence à notre page JSP de consultation.

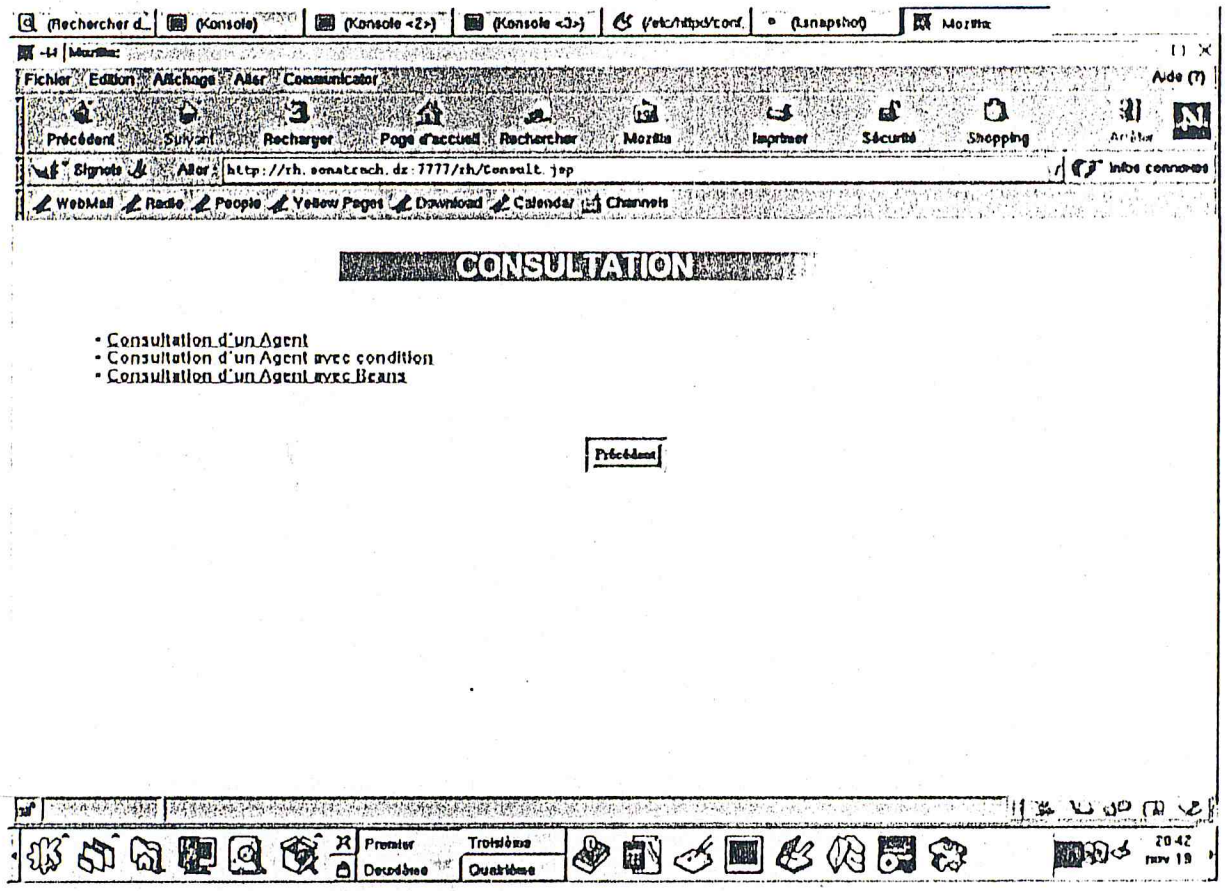


Figure 4 : Page HTML représentant les liens faisant référence aux pages JSPs de consultation.

En cliquant sur le lien « consultation des agents » la page JSP qui affiche les éléments d'une table est automatiquement généré. Donc l'utilisateur n'est pas obligé d'écrire l'URL référençant la page JSP en question.

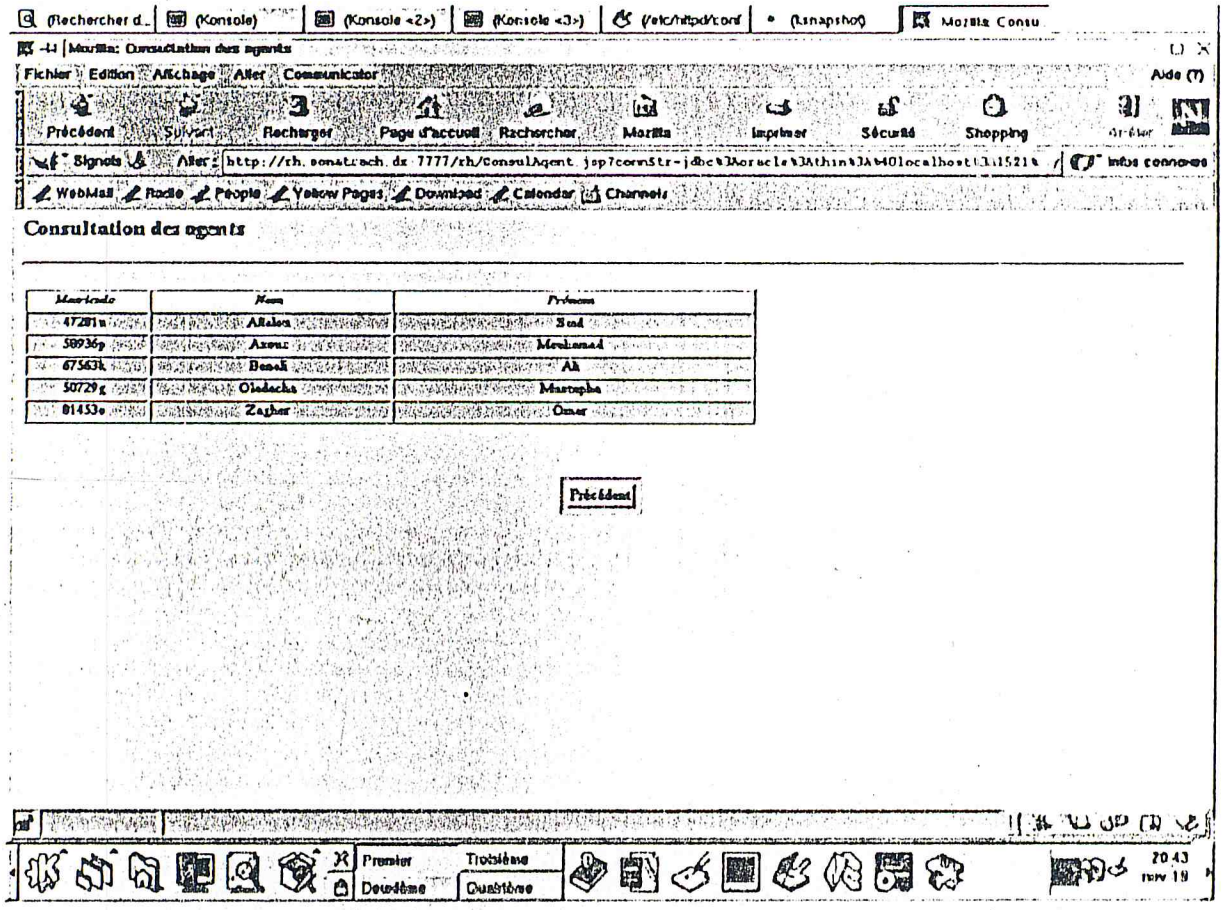


Figure5: Résultat de la page Web représentant notre application.

L'utilisateur peut taper l'URL sur son navigateur Web, et invoquer la page JSP contenant les informations qui l'intéressent. Il peut donc taper l'URL suivant :

« http: /rh.sonatrach.dz/rh/condition1.jsp » pour que cette page soit affichée :

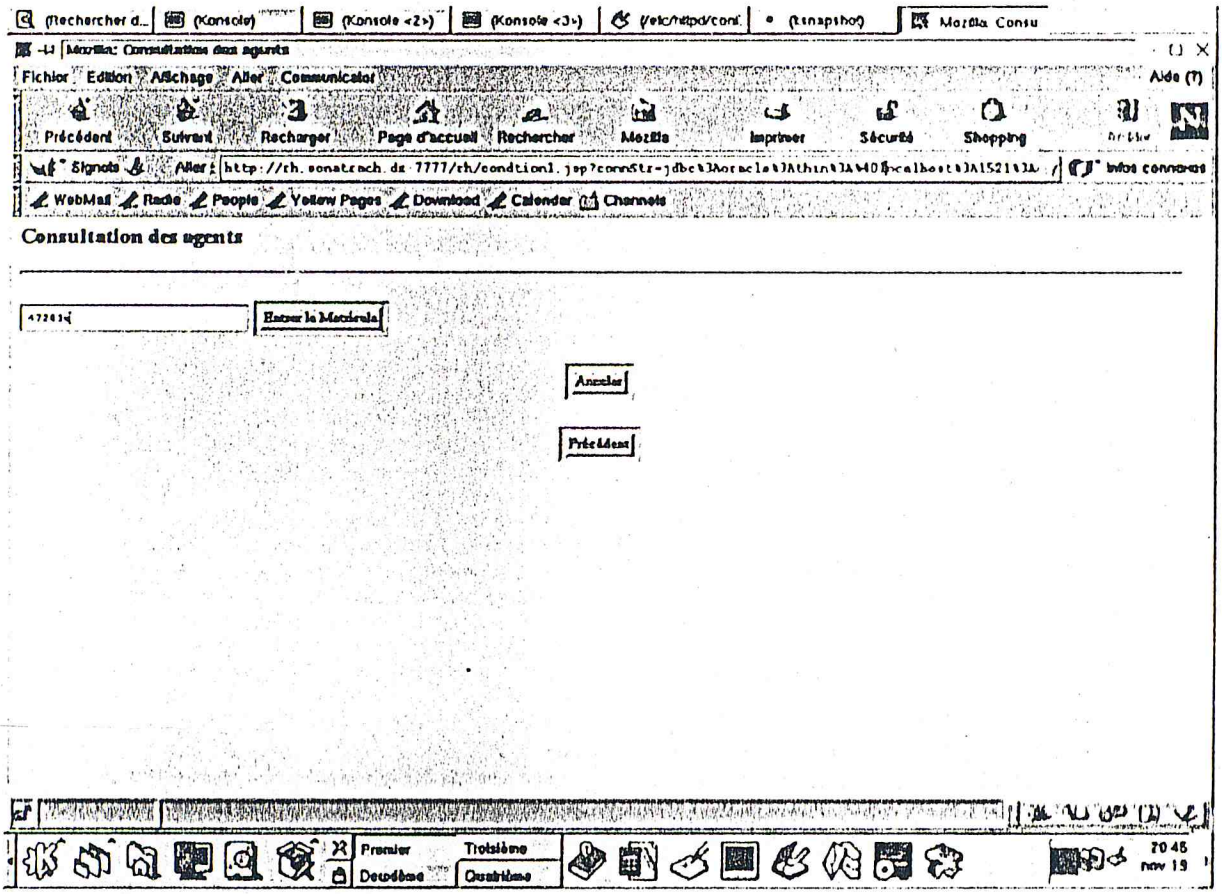


Figure 6: Présentation des conditions de consultation.

L'utilisateur tape le matricule de l'agent et clique sur le bouton « Entrer le matricule » qui fait appel à la page JSP qui affiche les éléments de cet agent.

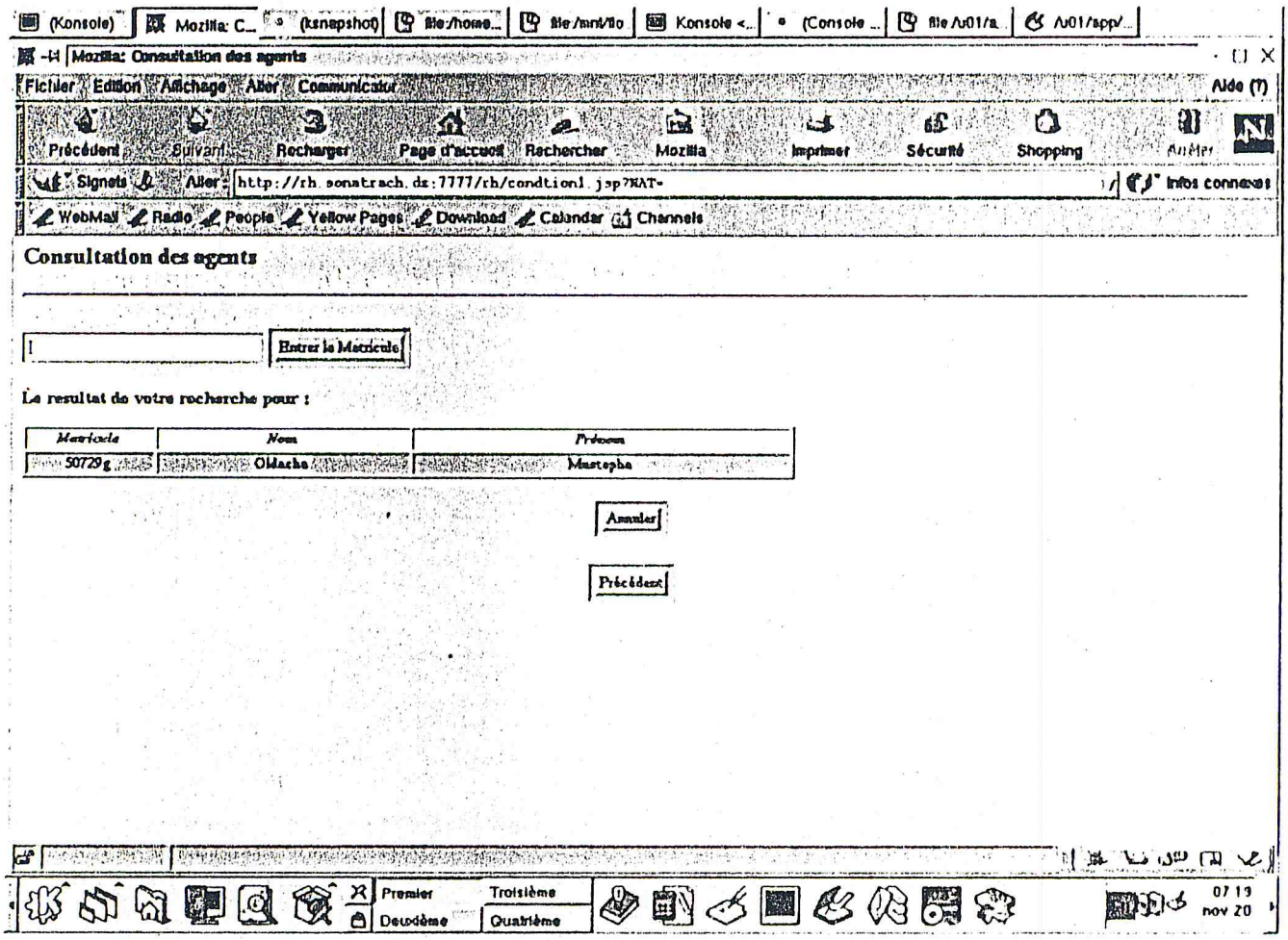


Figure 7: Résultat de consultation avec condition.

- **Application de mise à jour**

En cliquant sur le lien « Mise à jour » la page HTML qui contient les liens hypertexte vers les trois opérations de mise à jour :

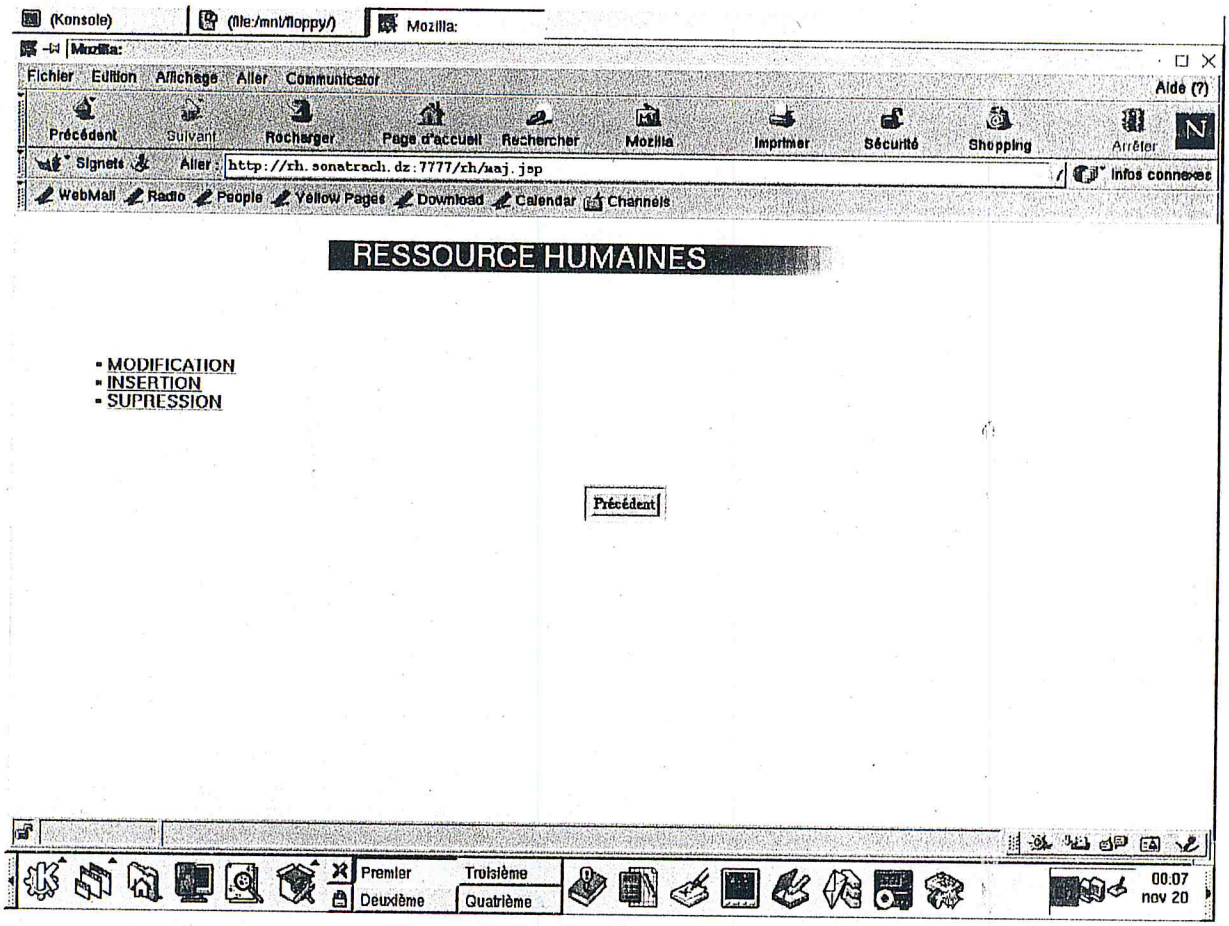


Figure 8 : Page JSP qui contenant les liens faisant référence aux page JSP mise à jour .

L'utilisateur a le choix d' effectuer une modification, une insertion ou une suppression. Si par exemple il cliquant sur le lien suppression, la page JSP suivante est affichée :

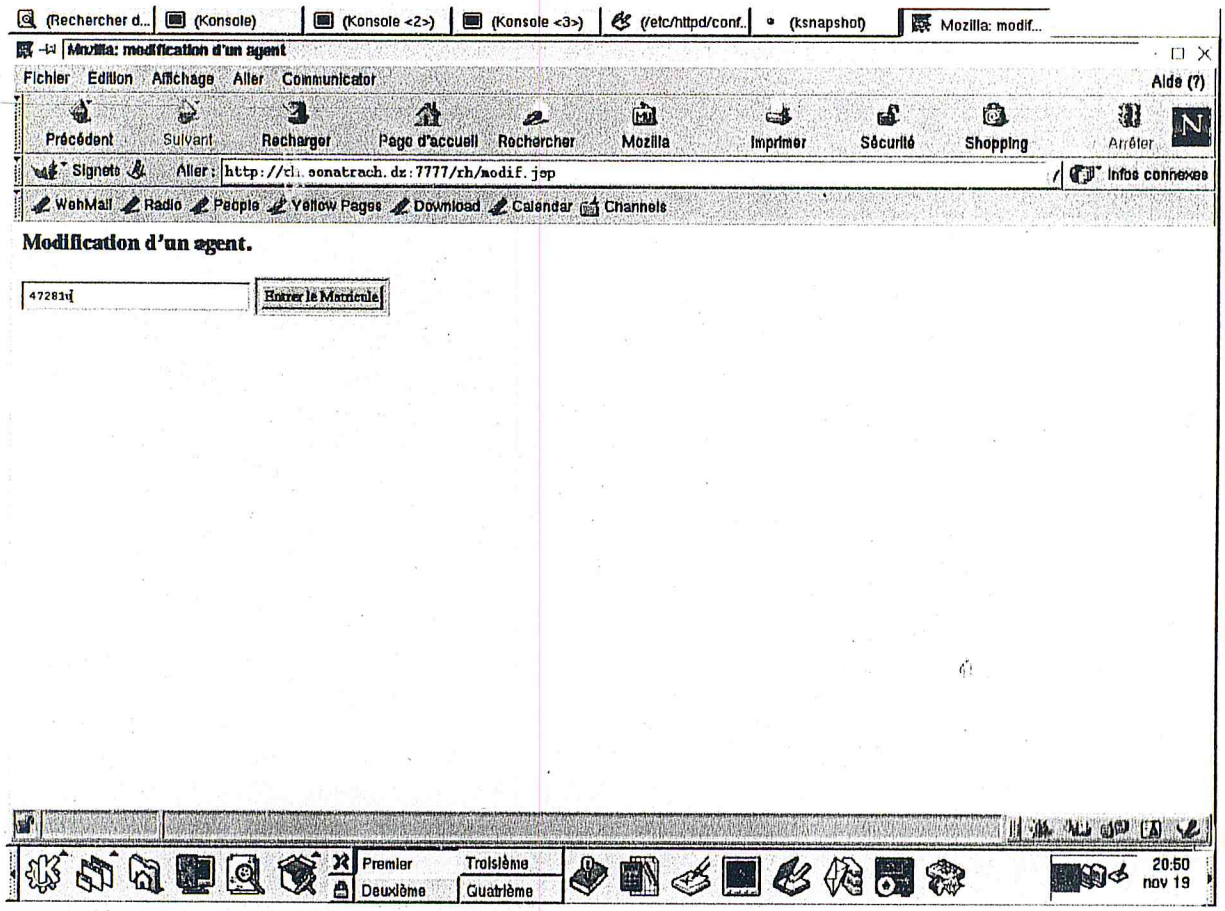


Figure 9 : Présentation de la condition de modification.

Lorsque l'utilisateur tape le matricule de l'agent et clique sur le bouton « Entrer le matricule », la page JSP « modif.jsp » est directement exécutée. Le même raisonnement pour l'insertion et la suppression.

L'exécution de la page JSP de modification génère un formulaire comme la montre la figure ci-dessous. L'utilisateur choisit les éléments à modifier dans les différents champs du formulaire et clique sur le bouton « modifier ».

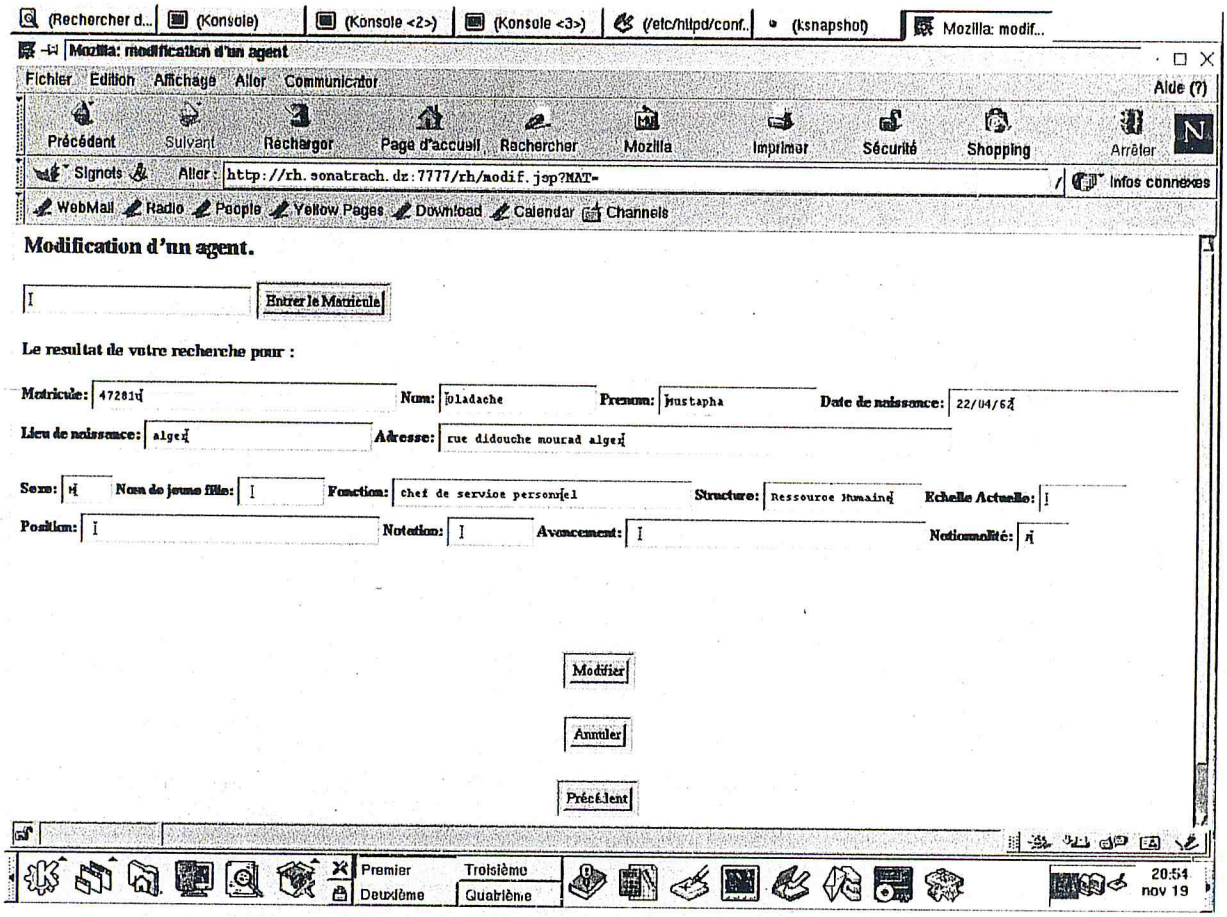


Figure 10 : Résultats de modification.

Lorsque l'utilisateur aura cliqué sur le bouton « modifier », une autre page JSP nommé « modif1.jsp » sera exécutée, a ce moment, deux cas peuvent se présenter :

Dans le cas ou l'utilisateur aura modifié ses informations sans erreurs, la page JSP retounera un message de succès dans la page HTML suivante :

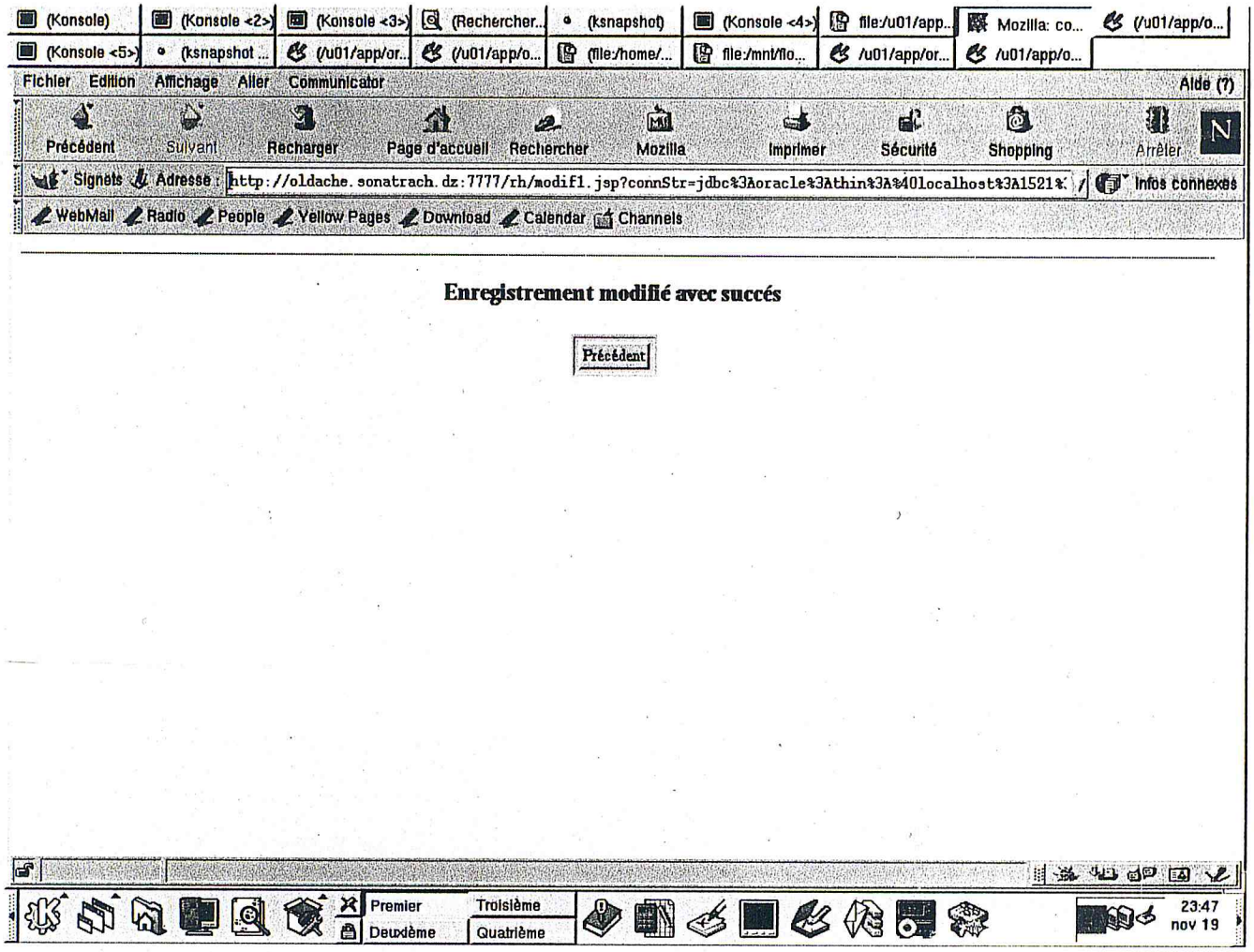


Figure 11 : Résultat de la page JSP « modif.jsp ».

Dans le cas contraire la page JSP retounera un message d'erreur dans la page HTML suivante :

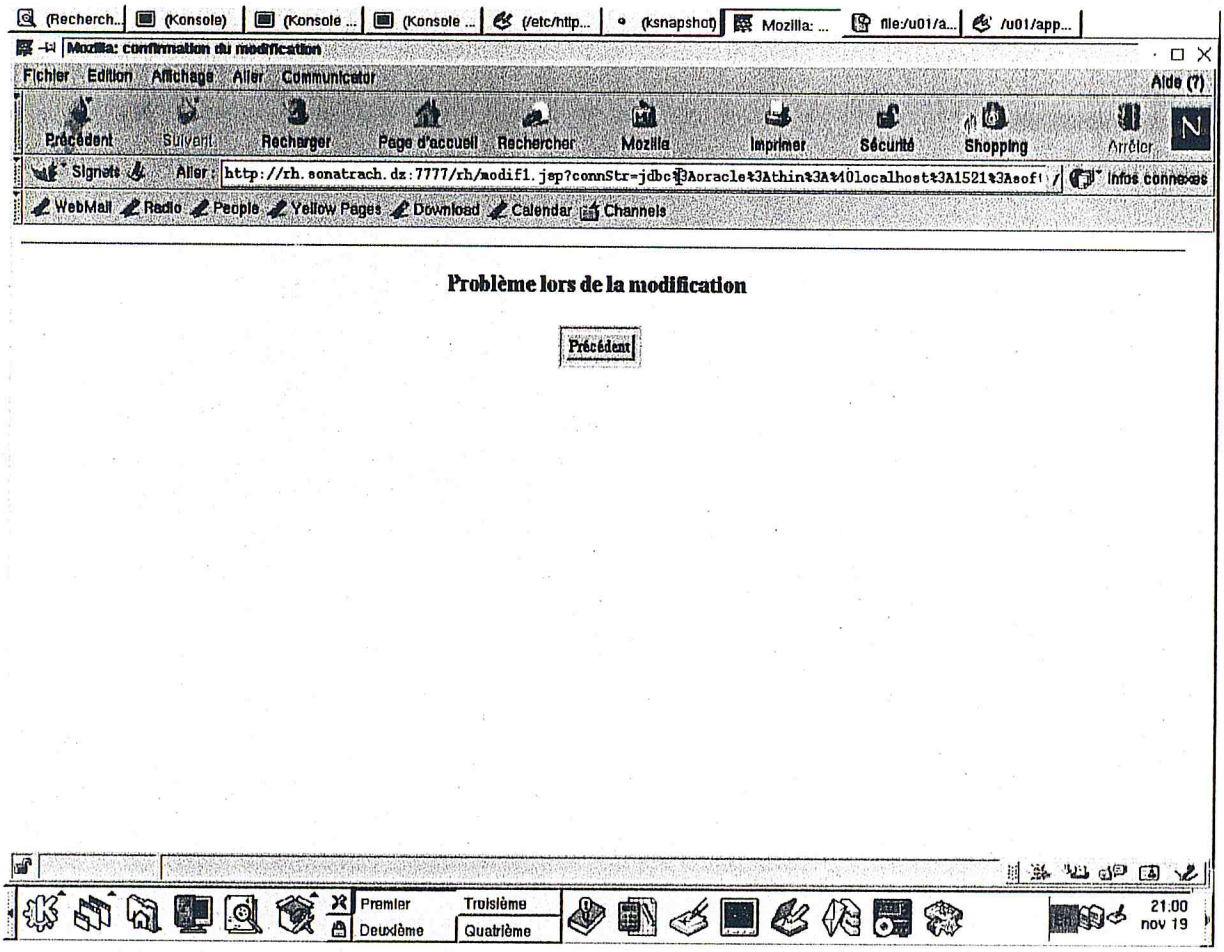


Figure 12 : Résultat de la même page JSP « modif.jsp ».

Conclusion générale

Le projet qui nous à été confié porte sur l'étude, la conception et l'implémentation d'un système de consultation et de mise à jour d'une base de données via un serveur Intranet/Internet, et cela en utilisant toute la technologie Oracle, autant pour la base que pour le serveur Web sous l'environnement LINUX.

L'étude des différentes architectures, nous a permis de bien comprendre les mécanismes d'accès au base de données à l'aide du Web, et en quel circonstances on doit les utiliser.

Les résultats des études présentées permettent d'entrevoir des perspectives d'extension vers l'orienté objet car la technologie Oracle fait partie de l'architecture NCA (Networking Computing Architecture), une architecture distribuée, évolutive et orientée objet.

Les applications conçues ne représentent qu'une modeste étude dans le domaine proposé. Pour cela on compte énormément sur les ingénieurs du CTI de profiter de cette étude et de l'enrichir de leurs savoir.

BiBLIOGRAPHIE

[DAU00] : DuameK.Fields, MarkA.KolB, 'JSP Java Server Page ', édition Eyrolles novembre 2000.

[LAU99] : Laura Lemay & Rogers Cadenhead, 'Java2 Plate-Fome', édition CampusPress septembre 1999.

[DAV00] : David Egan – Paul Zikopoulos, 'Base de données sous Linux ', édition Eyrolles septembre 2000.

[GEO02] : Georges Gardarin, 'Bases de données ', édition Eyrolles avril 2002.

[JOH00] : Johann Christiane Hanke, 'HTML 4 XML ', edition Micro Application 2000.

[LAR00]: Larne Pekowsky, 'l'intro JSP', edition CampusPress août 2000.

[MIC00]: Michel, ' LINUX Mandrake 7.0 ', édition CampusPress septembre avril 2000.

[KNO98] : Knowledge,' Le Langage SQL et l'Oulil SQL*Plus', édition juillet 1998.

[FMC98] : F.Macary, 'Java client-serveur, JavaBeans', JDBC, édition Eyrolles septembre 1998.

[SCO99] : Scot Oaks,'Sécurité en Java', édition française novembre 1999.

[DLL99] : Didier Deleglise,'Oracle et le Web', édition EYROLLES 1999.

[NYM02] : NymyJ. Yearger etRobertE. McGrath., 'Technologie des serveurs Web', édition International Thampson Publishing 2002.

[GRE97] : George Réese,'JDBC et JAVA', édition EYROLLES 1997.

➢ Ressources Internet :

ORACLE :

« http:// www.oracle.com »

« http:// www.olab.com »

« http:// technet.oracle.com »

JAVA :

« http:// www.javasoft.com »

« http:// www.java.sun.com »

« http:// www.club-java.com »

« http://www.javaworld.com/ »

HTML :

« http:// www.microapp.com »

JSP :

« http:// www.developpez.com »

« http:// www. Sun-JSP.com »

« http:// www.Apche.com »

« http://java.sun.com/products/jsp/ »

« http://www.servlets.com »

« http://www.jspinsider.com/index.jsp »

« http://www.jspin.com/ »

LINUX :

« http://www.linux.com/ »

« http://www.kernel.org/ »

« http://www.li.org/ »

« http:// Java.apache.org/Jserv »

« http:// www.Apache.org»

Annexe A : Aperçu sur l'environnement Linux

Le système d'exploitation utilisé pour ce projet est Linux, c'est la raison pour laquelle on donne un aperçu sur ce système en montrant son architecture, ses caractéristiques et ses points forts.

1. Le système LINUX

LINUX est un système d'exploitation multi-tâches et multi-utilisateurs offrant aux utilisateurs de nombreux utilitaires interactifs. En tant que système d'exploitation, son rôle principal est d'assurer aux différentes taches et aux différents utilisateurs une bonne répartition des ressources de l'ordinateur (mémoire, processeur(s), espace disque, imprimantes,.....) et cela sans intervention des utilisateurs.

2. Les composants du système

LINUX est un système d'exploitation composé de

- Un noyau assurant la gestion de la mémoire et des entrées sorties de bas niveau et l'enchaînement des différentes taches ;
- Un ensemble d'outils de base parmi lesquels :
  - Les commandes permettant la manipulation des fichiers ;
  - Les commandes permettant la gestion d'activités du système ;
  - Les commandes permettant la communication entre utilisateurs ou entre système ;
  - Les éditeurs de texte ;
  - Les outils de traitement de texte ;
  - Des compilateurs de langage (C ou Fortran par exemple) et un éditeur de lien ;
  - Des outils généraux de développement : gestionnaires de source, débogueurs, archiveurs, constructeurs d'analyseurs lexicaux et syntaxiques, etc.

3. Principales caractéristiques du système

Les principales caractéristiques de LINUX sont :

- Un système hiérarchisé de processus d'où un processus hérite de son parent lors de sa création ;
- Un ensemble de point d'accès par le noyau dans les applications développées dans langage C et appelé **appel système** ;
- Un aspect multitâches du système pour chaque utilisateur par la possibilité pour un utilisateur de lancer depuis un interpréteur shell des processus tout en « conservant la main » ;
- Un système de fichier hiérarchique ;
- Une vision unique des différents types d'entrées sorties ;

4. Les fichiers et les répertoires sous LINUX

Dans LINUX tous est fichier, qu'il s'agisse des périphériques tels que les lecteurs de disquettes, les commandes simples comme ls, les fichiers de configurations ou bien encore les répertoires. Tous cela est représenté sous la forme d'un vaste système de fichier.

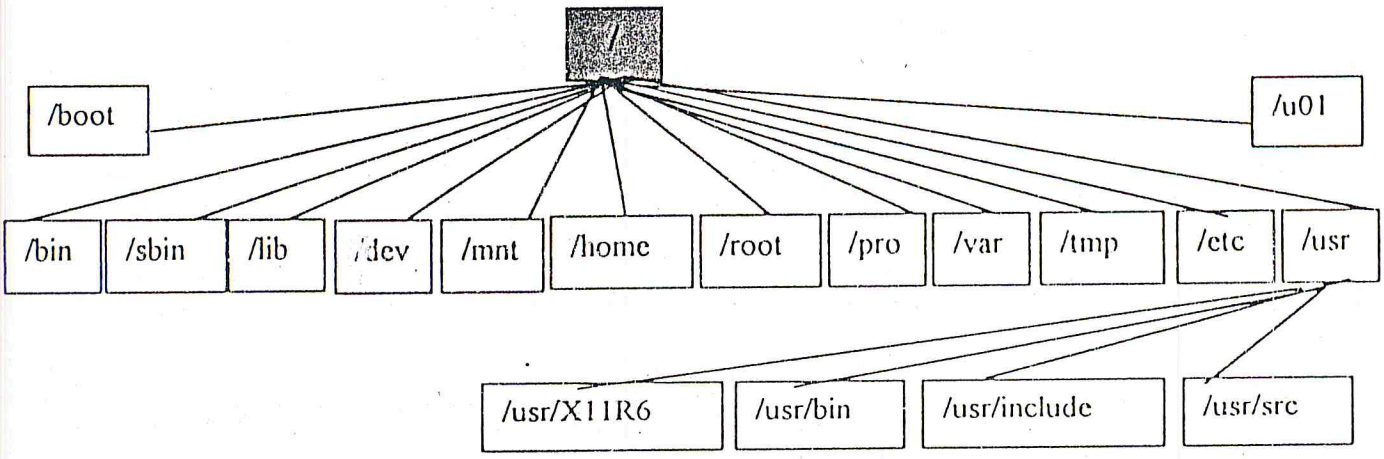


Schéma : Un exemple d'arborescence sous LINUX.

Répertoire	Contenu
/	La racine
/boot	Le noyau et le fichier de démarrage
/bin	Les commandes (en binaires)
/sbin	Des commandes supplémentaires(en binaires)
/lib	Les librairies communes
/dev	Les périphériques
/mnt	Les répartitions et lecteurs
/home	Les répertoires des utilisateurs
/root	Le répertoire « home » du root
/proc	Les processus
/var	Les fichiers dont le contenu change souvent
/tmp	Les fichiers temporaires
/etc	Les fichiers de configuration générale avec sous répertoires à théme
/usr/X11R6	Les principaux fichiers et commandes concernant X11
/usr/bin	Commandes des applications
/usr/include	En-tete communes
/usr/src	Les sources du noyau

Annexe B : Package Java.sql

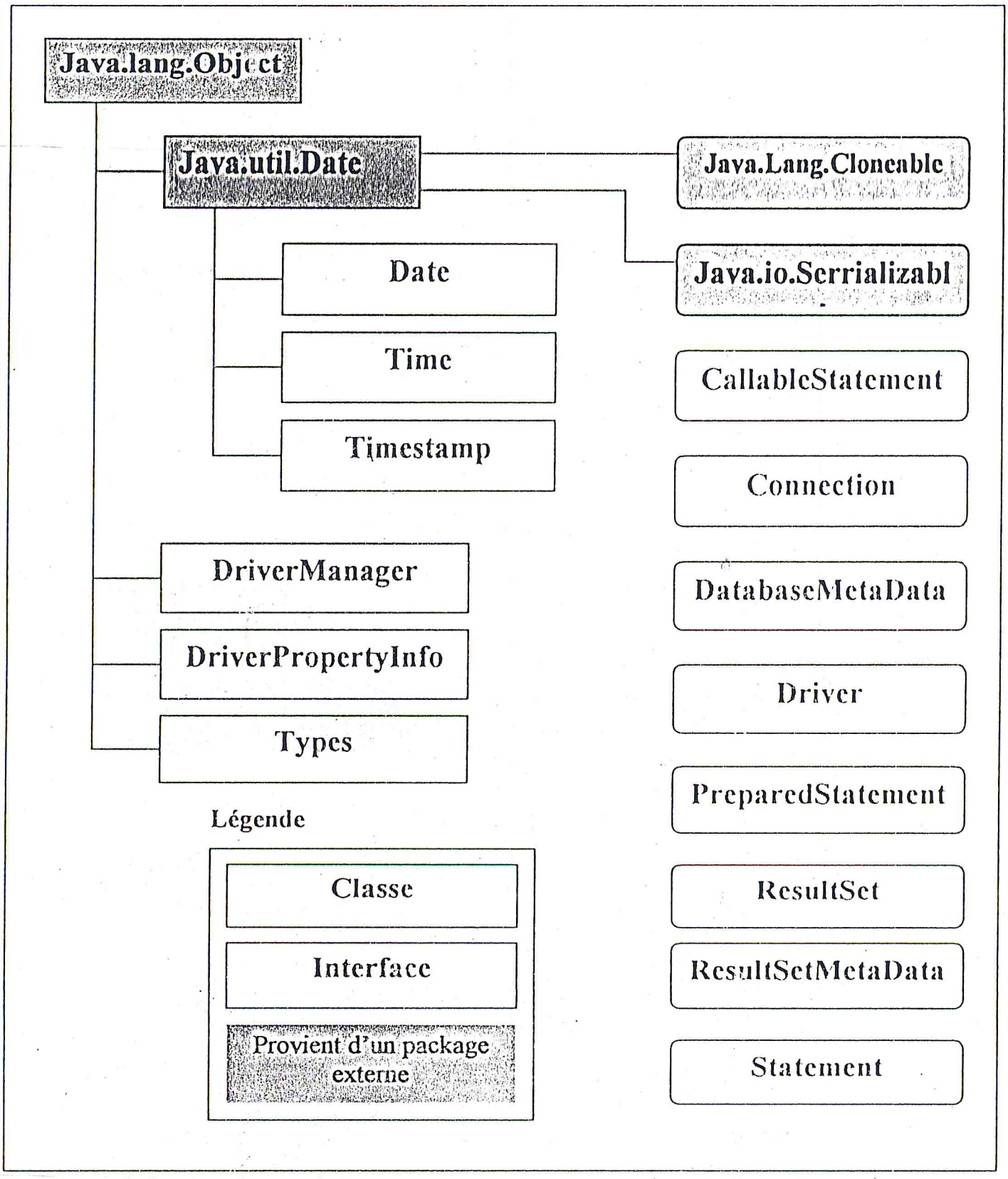


Figure 1 : le package Java.sql

Interface CallableStatement

Cette interface permet d'exécuter des procédures stockées dans la base de données.

Interface Connection

L'interface *Connection* représente une session sur une base de données. Elle permet d'accéder à des informations sur les tables, les possibilités du pilote, son état courant, ainsi que le niveau.

Un objet *connection* sert à créer des objets permettant d'effectuer des requêtes (CreateStatement, prepareCall, PrepareStatement).

Des informations sur la base de données sont disponibles via la méthode *getMetadata*..

Interface DatabaseMetaData

Cet ensemble de méthodes donne un accès à des informations de haut niveau sur les possibilités de base de données et le schéma courant.

Interface Driver

Cette interface permet de faire le lien avec le pilote réel utilisé les requêtes. C'est cet objet qui doit être instancié en premier car il fournit un objet Connection grâce à la méthode connect (). Les autres méthodes fournissent des informations sur le pilote lui-même.

Interface PreparedStatement

Cette interface permet de définire des requêtes précompilées, par exemple lorsqu'un ordre SQL doit être exécuté souvent mais avec des paramètres différents.

Interface ResultSet

Un ResultSet est un objet représentant le résultat d'une requête. On parcourt le ResultSet de ligne en ligne avec la méthode next() (qui renvoie une valeur vraie tant qu'il reste des lignes).

Interface ResultSetMetaData

Cette classe permet d'obtenir des informations supplémentaires sur la structure d'un resultset.

Interface Statement

Cette interface est la plus utilisée en pratique puisqu'elle permet d'exécuter des ordres SQL dits statiques (ne contenant pas de paramètres d'entrée et de sortie).

Classe Date

Un objet sql.Date représente une date (années, mois, jour) avec une conversion standard pour les formats courants SQL (yyy-mm-dd). Cette classe est destinée principalement à fournir au package java.sql un type SQL Date, les principales méthodes sont définies dans java.util.Date.

Classe Time

Un objet sql.Time représente un temps (heures, minutes, secondes) avec une conversion standard pour les formats courants SQL (hh :mm :ss). Cette classe est destinée principalement à fournir au package java.sql un type SQL Time, les principales méthodes sont définies dans java.util.Date.

Classe Timestamp

La classe Timestamp représente le type date /heure TIMESTAMP SQL. Elle ajoute à la classe java.util.date la notion de nanoseconde, absante dans le type de date, ainsi que les méthodes pour comparer et convertir des objets Timestamp en String.

Classe DriverManager

Le gestionnaire des pilotes est chargé de mettre en relation un pilote JDBC avec la source de données demandée. La discrimination se fait sur le sous-protocole défini dans l'URL des méthodes getConnection(...).

Classe DriverPropertyInfo

Cette classe définit la structure des propriétés que l'on peut associer à un pilote JDBC.

Classe Types

Cette classe ne contient que les définitions de constantes. Ces constantes permettent d'indiquer aux méthodes qui ne manipule pas des types Object.

Annexe C : Java Server Page (JSP)

Cette Annexe résume les différentes syntaxes de balises JSP

Commentaires

< % -- comentaire -- %>

Les commentaires JSP sont supprimés par le moteur JSP au moment de la traduction. Par conséquent, ils n'apparaissent jamais dans le servlet final et ne sont jamais envoyés à l'utilisateur. Les commentaires permettent d'indiquer aux personnes chargées de la mise à jour d'une page le rôle de chaque portion de code HTML.

Déclarations

<% ! type varname ; %>

<% ! returntype methodname(argument1, argument2...) {} %>

Les déclarations peuvent être utilisées pour ajouter des variables ou des méthodes dans une JSP, tout ce qui se trouve dans une déclaration est ajouté dans le servlet produit au niveau de la classe. Pour les variables, cela signifie qu'il existera typiquement une instance de la variable, qui sera partagée entre toutes les requêtes.

Expressions

<%= expression %>

Les expressions peuvent correspondre à une simple variable ou à un appel de méthode ou à une forme mathématique complexe contenant plusieurs termes.

Scriptlets

<%...code Java...%>

Les scriptlets permettent à du code Java quelconque, et éventuellement à d'autres pages, d'être placé dans une JSP. La plupart du temps, ce code doit se trouver dans des beans et dans d'autres classes, mais il existe des cas où cela n'est pas recommandé. Une logique complexe peut être ajoutée dans les pages en entourant des zones de texte avec des scriptlets. Le premier scriptlets doit se terminer par une accolade ouvrante et le second doit contenir une accolade fermante correspondante.

Directive Include

<%@ include file= '' '' %>

La directive d'inclusion ajoute le texte d'une JSP ou d'un fichier dans un autre fichier.

Directive Page

<%@ page option...%>

La directive de page spécifie un grand nombre d'option qui affectent une page entière. Chacune de ces options peut être employée indépendamment des autres. Elles peuvent toutes apparaître dans la même directive de page ou chacune peut être placée dans une directive indépendante.

Language=''Java''

Le code spécifie le langage à utiliser dans les scriptlets. A l'heure actuelle, seul Java est supporté.

extends=''base class''

Ce code peut obliger le servlet généré à étendre une classe spécifique. Il doit être utilisé très rarement,

import=''package.class''

import=''package.''*

Les paramètres d'importation d'une directive de page sont transformés en instructions d'importation dans le fichier Java généré.

session=''true | false''

Par défaut, la première fois qu'un utilisateur accède à une JSP d'un site, une session est ouverte pour cet utilisateur et ce dernier reçoit un cookie. Ce comportement peut être désactivé en mettant le drapeau de session à false.

buffer=''none| sizekb''

Ce drapeau définit la quantité de donnée qui seront mises en mémoire avant d'être envoyées à l'utilisateur.

autoFlash =''true | false''

Si la mise en mémoire est activée et que cette valeur vaille true, les données sont automatiquement envoyées à l'utilisateur lorsque la mémoire tampon est pleine. Si la mise en mémoire est activée et que cette valeur vaille false, une exception est déclenchée lorsque la mémoire tampon est pleine. Si la mise en mémoire est désactivée, cette valeur n'a aucun effet. Sa valeur par défaut est true.

isThreadSafe=''true | false''

Cette valeur indique au moteur JSP s'il peut envoyer simultanément plusieurs requêtes sur une même page, sans provoquer d'erreur. Par défaut, elle vaut true. Lorsque cette valeur vaut false, cela revient à déclarer que le servlet généré implémente SingLeThreadModel.

info=''text ''

Cette valeur définit une description du but de la page, de son auteur, ainsi que d'autres informations similaires. Tout ce qui défini dans cette chaîne peut être récupéré par la méthode getServletInfo() du servlet généré.

errorPage=''pathToErrorPage''

Cette valeur permet de personnaliser la page envoyée à l'utilisateur lorsqu'une erreur se produit au moment de l'exécution. Par défaut, dans ce cas, l'utilisateur voit la page d'erreur standard, error

500. Mais si cette valeur contient le chemin d'une autre page JSP, le contenu de cette page est affiché à la place du message d'erreur.

isErrorPage=''true | false ''

Cette valeur indique que la JSP sera utilisée comme une page d'erreur. Lorsqu'elle vaut true, la JSP a accès à une valeur implicite supplémentaire appelée exception, qui sera une exception ou un Throwable représentant l'erreur qui s'est produite. Sa valeur par défaut est false.

contentType=''mime-type''

Cette valeur modifie le type de contenu de la page. Dans les implémentations actuelles, elle est équivalente à request.set(''mime-type''). Sa valeur par défaut est text.html.

Directive Taglib

<%@taglib uri=''...'' prefix=''tagPrefix'' %>

Cette directive charge un ensemble d'étiquettes personnalisées.

Transmission de requête

Cette étiquette peut servir à envoyer la requête à une autre page.

<jsp :forward page=''target''>

<jsp:forward page=''target''>

<jsp:param name=''name'' value=''value''/>

...

</jsp:forward>

Inclusion au moment de la requête

Cette étiquette peut être utilisée pour inclure une JSP ou une page HTML dans une autre page.

<jsp :include page=''target'' flush=''true''>

<jsp:include page=''target'' flush=''true''>

<jsp:param name=''name'' value='value''/>

...

</jsp:include>

Plug-in

Cette étiquette génère le code d'une applet ou d'un objet intégré, en fonction de ce que le navigateur peut gérer.

useBeans

<jsp :useBean name=''id'' parameters.../>[first form]

<jsp :useBean name=''id'' parameters>[second form]

.... JSP code.....

</jsp:useBean>

- Cette étiquette rend un Bean disponible pour une utilisation ultérieure dans la page.
- Sous sa seconde forme, le code compris entre les étiquettes ouvrantes et fermentes est exécuté lors de la première construction de Beans, mais pas pendant les utilisations de ce Bean.

Plusieurs paramètres peuvent être employés dans les deux formes pour modifier la manière dont le Bean est créé ou d'autres aspects de ce Bean. Ces paramètres sont décrit dans la page suivante :

scope=''page | request | session | application '' : Ce paramètre place le Bean dans le champ spécifié ou y accède à partire de ce champ s'il existe déjà. Sa valeur par défaut est page.

beanName = ''name'' : Ce paramètre demande à l'étiquette useBean de charger un bean mis en séric, appelé name.

classe=''package.class '' : Ce paramètre indique à l'étiquette useBean la classe à employer lors de la construction du bean.

type = ''paquage.class'' : Ce paramètre indique le type correspondant au bean.

Setproperty

<jsp :setproperty name= ''name'' property=''propName'' value=''value''/>_[first vesion]

<jsp :setproperty name= ''name'' property=''propName''/>_[second vesion]

<jsp :setproperty name= ''name'' property="propName"_ param=''param_name"/> [third version]

<jsp :setproperty name= ''name'' property=''*''/>_[fourth vesion]

Cette étiquette définit une ou plusieurs propriétés d'un Bean.

- La première version place une valeur particulière dans la propriété spécifie.
- La deuxième version définit la propriété à partir du paramètre de formulaire ayant le même nom.
- La troisième version définit la propriété à partir d'un paramètre du formulaire possédant éventuellement un nom différent.
- La dernière version définit toutes les propriétés pour lesquelles un paramètre de formulaire correspondant est fourni.

Getproperty

<jsp:getproperty name=''id'' property=" propName"/>

Cette étiquette place la valeur actuelle d'une propriété dans la page résultante. Avec les implémentations actuelles, elle est équivalente à :

<%=beanName.getpropName () %>

Annexe D: Les tables de la base de donnée Ressources Humaines

La base de données Ressources Humaines contient des références (données) associées aux différentes entités. Ces entités sont représentées sous forme de tables

- **Table Agent :** regroupe toutes les informations concernant les agents.

La table est structurée de la manière suivant :

Attribut	type	Taille(octets)
Matricule	A	6
Photo	BMP	
Nom	A	15
Prénom	A	15
Date de naissance	D	
Lieu de naissance	A	20
Adresse	A	40
Sexe	A	1
Nom de jeune fille	A	15
Téléphone de l'agent	N	8
Date de confirmation	D	
Situation familière	A	1
Situation vis à vis du sonatrach	A	1
Numé SS	N	12
Groupe sangun	A	3
Cotisation mutuelle	BL	

- **Table Etablissement :**

Attribut	type	Taille(octets)
Code établissement	A	1
Désignation établissement	A	30
Adresse établissement	A	40
Ville établissement	A	20
Code postale	N	5
Pays établissement	A	10
Téléphone établissement	N	10
Nom personne contacter	A	15
Fonction personne	A	20
Secteur	BL	

- **Table Diplôme :** contient toutes les informations sur le diplôme de l'agent

Attribut	type	Taille(octets)
Code diplôme	A	5
Désignation diplôme	A	15
Valeur diplôme	N	3

- **Table Spécialité :** regroupe toutes les informations concernant la spécialité de l'agent.

Attribut	type	Taille(octets)
Code spécialité	A	2
Libelle spécialité	A	20

- **Table Mode arrivée :**

Attribut	type	Taille(octets)
Code arrive	A	1
Libelle arrive	A	60

- **Table Fonction :** regroupe toutes les informations concernant la fonction de l'agent.

Attribut	type	Taille(octets)
Code fonction	N	5
Libelle fonction	A	30

- **Table Section :**

Attribut	type	Taille(octets)
Code section	A	2
type section	A	15
Libellé section	A	20

- **Table Conjoint :** regroupe toutes les informations concernant conjoint de l'agent.

Attribut	type	Taille(octets)
Numéro Conjoint	N	1
Nom conjoint	A	15
Prénom conjoint	A	15
Date naissance conjointe	D	
Lieu de naissance conjoint	A	20
Adresse conjoint	A	20
Situation conjointe	A	15

- **Table Enfant :** regroupe toutes les informations concernant l'enfant de l'agent.

Attribut	type	Taille(octets)
Numéro enfant	N	2
Nom	A	15
Prénom	A	15
Date naissance	D	
Lieu de naissance	A	20
Adresse enfant	A	40
Situation enfant	A	15

- **Table Nationalité :** regroupe toutes les informations concernant la nationalité de l'agent.

Attribut	type	Taille(octets)
Code nationalité	A	3
Libelle nationalité	A	20

- **Table Position :** regroupe toutes les informations concernant la position de l'agent.

Attribut	type	Taille(octets)
Code position	A	2
Libelle position	A	20

- **Table Organisme d'assurance :**

Attribut	type	Taille(octets)
Code organisme social	A	1
Adresse organisme social	A	10
Libellé organisme social	A	40

- **Table Nature acte :**

Attribut	type	Taille(octets)
Code nature	A	6
Horaire	N	6

- **Table Acte médical :**

Attribut	type	Taille(octets)
Code acte	A	4
Libellé acte	A	25
Horaire	MN	5

- **Table Acte optic :**

Attribut	type	Taille(octets)
Numéro verre	A	4
Nature verre	A	20
Montant verre	MN	5

- **Table Cause accident** : regroupe toutes les informations concernant la cause d'accident de travaille de l'agent.

Attribut	type	Taille(octets)
Code cause	A	2
Libellé cause	A	30

- **Table Nature lésion :**

Attribut	type	Taille(octets)
Code nature lésion	A	1
Libellé nature lesion	A	20

- **Table Lésion :**

Attribut	type	Taille(octets)
Code lésion	A	1
Libellé lésion	A	20

- **Table Date :**

Attribut	type	Taille(octets)
Date	D	8

1.5.2 Les relations entre les tables

- **Relation Possède :**

Attribut	type	Taille(octets)
Data d'obtention	D	
Mention	A	20

- **Relation Est en position :**

Attribut	type	Taille(octets)
Date fin position	D	
Lieu position	A	40

- **Relation Congé :**

Attribut	type	Taille(octets)
Date fin congé	D	
Motif congé	A	40

- **Relation A un :**

Attribut	type	Taille(octets)
Date fin mariage	D	

- **Relation A pour :**

Attribut	type	Taille(octets)
Lieu de parenté	A	10

- **Relation Avoir accident travail :**

Attribut	type	Taille(octets)
Lieu accident	A	40
Heure accident	T	

- **Relation Remboursement :**

Attribut	type	Taille(octets)
Date début soins	D	
Date fin soins	D	
ML ordonnances	MN	7

- **Relation Est remboursée :**

Attribut	type	Taille(octets)
Date de soins	D	
Montant remboursé	N	7