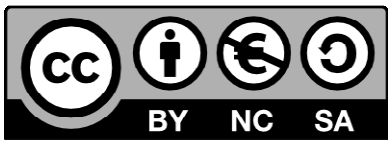
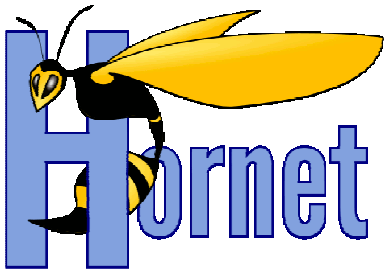


[Hornet]
Charte d'architecture



Charte d'architecture Hornet

Version : 1.0 du 07/12/2012 - **Etat** : Validé

SUIVI DES MODIFICATIONS

Version	Auteur	Description	Vérification	Date
1.0	S.Percier	Initialisation du document	S.Heutematte	07/12/2012

SOMMAIRE

TABLEAUX

FIGURES

1 Introduction

Ce document a pour vocation à lister les principes techniques fondateurs de cette architecture pour préciser sa modularité et les compétences nécessaires au développement des projets utilisant Hornet, sans décrire dans le détail l'ensemble des fonctionnalités techniques mises en œuvre.

1.1 Enjeux

Les enjeux d'Hornet comme solution industrielle des développements d'application sont :

- Fonctionnement sur les principaux navigateurs
- Permettre et faciliter l'accessibilité des applications réalisées et la conformité au RGAA
- Permettre une ergonomie standard pour les applications
- Permettre une ergonomie particulière pour certaines applications
- Faciliter le développement d'application

1.2 Architecture générale

L'architecture logicielle et technique multi-niveaux Hornet repose sur les standards Internet (W3C / applications Web Java).

Hornet fournit deux frameworks :

- côté serveur Web, hornetclient de technologie HTML, CSS et Javascript (basé sur « Yahoo ! User Interface Library »)
- côté serveur d'application, hornetsever de technologie JEE (fork de LISE 3.5 du projet Adullact Acube)

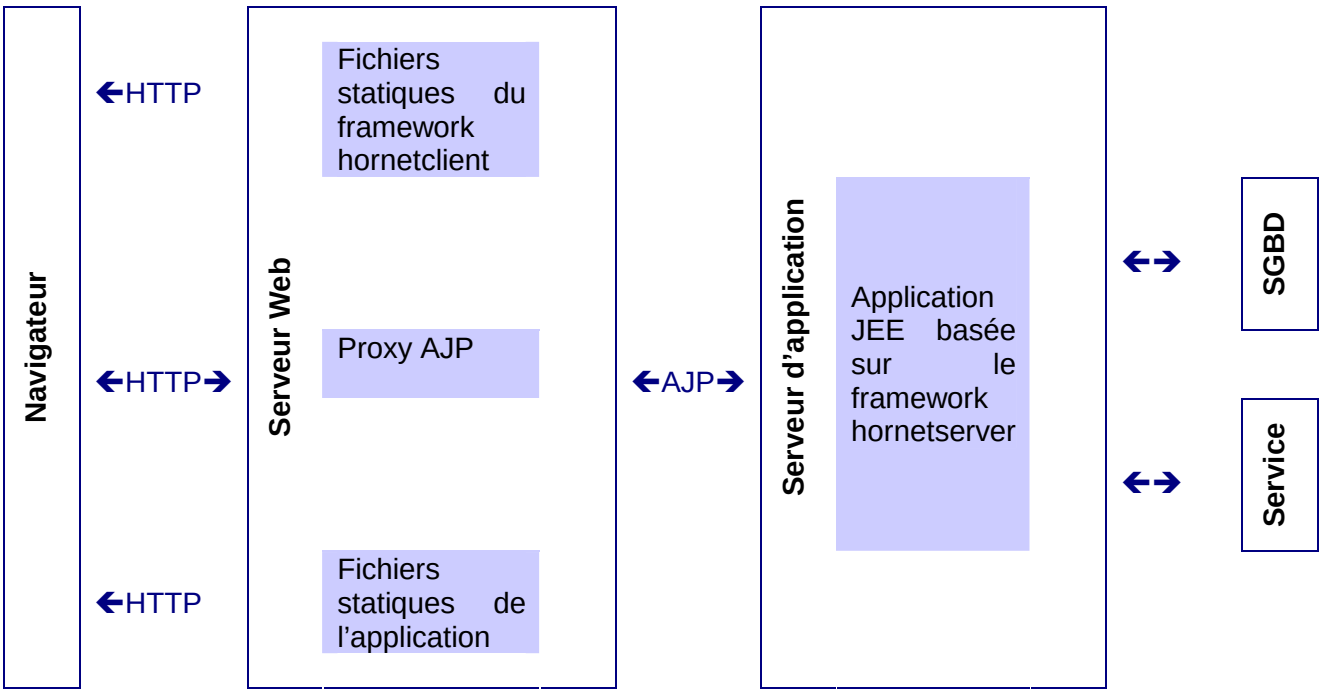


Figure 1 : Principe générale d'architecture des applications Hornet

Les fichiers statiques du framework hornetclient peuvent être hébergés sur un serveur web dédié indifféremment.

2 Principes techniques d'hornetclient

La partie cliente d'Hornet repose sur une implémentation du principe Hijax (hijack & ajax) avec « Yahoo ! User Interface Library » (ou YUI).

Les paragraphes suivants présentent les différents principes utilisés et composants mis en œuvre.

2.1 Architecture

L'architecture retenue pour le fonctionnement dans le navigateur est basée sur un modèle en couche :

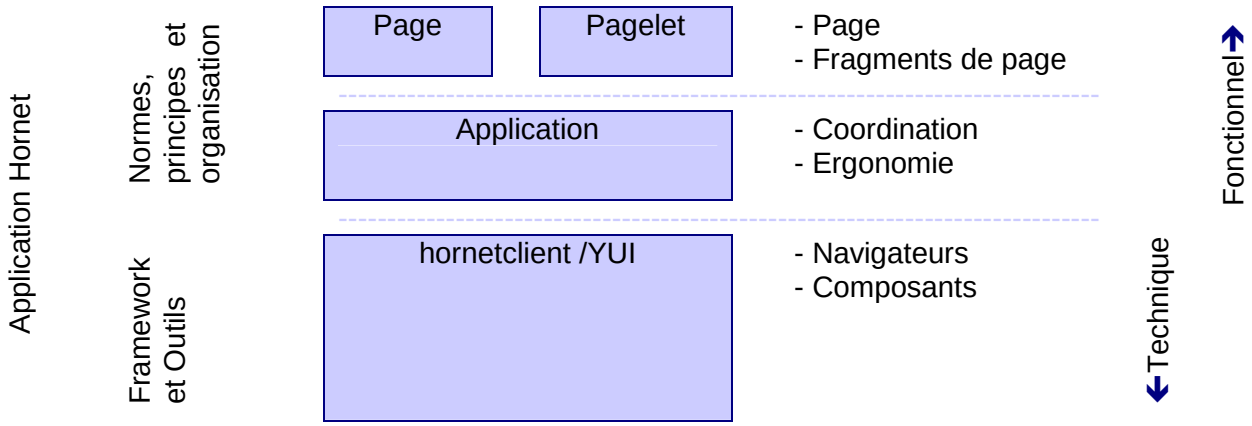


Figure 2 : Architecture en couche côté client

Les trois niveaux retenus sont :

- La couche hornetclient et YUI permet de s'abstraire du navigateur et propose de nombreux utilitaires et composants pour faciliter le développement d'une application : Manipulation du DOM, Ajax …,
- La couche application définit l'ergonomie standard, et coordonne les interactions entre composants et pages,
- La couche page/pagelet compose les composants Hornet afin d'offrir à l'utilisateur les fonctionnalités demandées.

2.2 Principes

2.2.1 Séparation des rôles

Les trois couches utilisées pour les pages web ont des rôles et usages différents :

Partie	Forme	Rôle
Sémantique	HTML	Cette partie est implémentée sous forme de document HTML ou XML conformément aux normes w3c et en respectant les règles d'accessibilité. Aucun élément de présentation ne doit être présent dans ces documents (hormis les liens d'identification et de classe CSS pour la présentation). Les éléments HTML doivent être utilisés conformément à leur usage prévu par le w3c, par exemple : - Les « tables » HTML doivent être utilisées essentiellement pour présenter des données sous forme de tableaux, et non pour structurer l'affichage.
Présentation	CSS	La partie présentation est réalisée via les styles CSS et les éléments HTML de type groupement (div, p, span …).

Interactive	Javascript	La partie interactive est réalisée en javascript à l'aide de « Yahoo ! User Interface Toolkit ».

Tableau 1 : Répartition des rôles des technologies côté client

2.2.2 « Progressive Enhancement » ou l'amélioration progressive

> « L' amélioration progressive est une manière de concevoir un site web qui prend très largement en compte l'accessibilité, la sémantique et le référencement. En séparant strictement le fond (le contenu proposé à l'utilisateur) et la forme (l'enjolivement et les interactions avancées), cette technique permet de présenter un contenu simple et de rendre un service minimum à tous les utilisateurs, quel que soit le débit de leur connexion ou leur navigateur, tout en améliorant progressivement l'affichage proposé en fonction de l'équipement de l'internaute. »
>
> Source : http://fr.wikipedia.org/wiki/Am%C3%A9lioration_progressive, le 10 décembre 2012.

Les pages web sont ainsi composées de trois parties :
- La partie sémantique en html ou xhtml (le fond),
- La partie présentation avec des styles CSS (la forme),
- La partie interactive, dynamique et événementielle avec du javascript (la forme).

Le service à l'utilisateur doit être « augmenté et enrichi » progressivement par chaque couche.

Généralement la partie interactive du « Progressive Enhancement » inclut des échanges de type AJAX, d'où l'appellation « hijax » fréquemment rencontrée pour la combinaison du Progressive Enhancement et d'AJAX.

2.2.3 Modes de fonctionnement

Le fonctionnement vu par l'utilisateur dans un navigateur web est similaire à celui d'un site web classique, cependant il y a un certain nombre de différence au niveau de la structure des pages web et de l'emploi des composants comme :

- Les pages web peuvent être utilisées sans Javascript, sans images (et autres informations visuelles) ou sans styles CSS,
- Chaque document HTML est conforme aux normes w3c et utilise les éléments et attributs HTML adéquat avec l'information véhiculée.

Trois grands modes d'utilisation sont possibles :

Mode	Description
« Web 1.0 »	L'utilisateur accède à des pages utilisant des styles, images et javascripts et chaque clic et appel entraine le chargement complet d'une nouvelle page dans le navigateur,
Hijax	Ce mode complète le précédent en ajoutant une couche interactive javascript sans perturber le fonctionnement « Web 1.0 » de la page. Le code javascript intercepte certaines actions utilisateurs et remplace le chargement de nouvelle page par des appels Ajax et des manipulations du DOM pour mettre à jour la page avec les informations provenant des requêtes Ajax. Par exemple, dans le cas d'une liste de 50 items affichée dans un tableau de 25 items sur deux pages, la pagination est implémentée en AJAX et seul le contenu du tableau est modifié lors de la pagination sans entrainer de chargement complet du document HTML.
Single Page Application (SPA)	Le mode SPA étend le fonctionnement hijax à toute une

	application web. Dans le mode hijax, les actions interceptées correspondent généralement aux fonctionnalités métiers représentées par la page elle-même, les changements de fonctionnalités entrainant le chargement complet d'une nouvelle page. Par exemple, dans le cas d'une liste de 50 items affichée dans un tableau de 25 items sur deux pages, la pagination est implémentée en ajax et seulement le contenu du tableau est modifié lors de la pagination alors que l'accès à une autre fonctionnalité via le menu entraine le chargement complet de la page. Dans le mode SPA, toutes les actions sont interceptées et remplacées par des appels ajax modifiant partiellement le DOM de la page, il n'y a alors plus de chargement complet de page. Ce mode permet une grande richesse et interactivité des applications web, mais il nécessite un cadre de conception et développement uniformisé entre les pages et fonctions.

Tableau 2 : Trois modes de fonctionnement

Les paragraphes suivants présentent de façon plus détaillées les caractéristiques essentielles de la solution pour les modes Hijax et SPA.

2.2.3.1 Mode Hijax

Le mode hijax ajoute du code javascript modifiant le comportement d'une page et remplaçant les actions utilisateurs par des requêtes ajax et ceci sans perturbation de manière à ce que le service soit disponible indépendamment du support ou non du javascript au sein du navigateur.

Ceci implique que le code javascript des requêtes ajax doit être conçu pour interagir avec la partie serveur et interpréter les réponses aux actions utilisateurs.

Ceci implique également que la partie serveur doit être conçue pour servir à la fois les requêtes http issues des appels classiques « web 1.0 » et des appels ajax et retourner les résultats appropriés à chaque mode.

La collaboration entre les parties serveurs et clientes se fait par le biais de contrat d'interface et de collaboration :

- la distinction entre les deux types de requêtes est faite soit en ajoutant un paramètre (mode=hijax) ou en modifiant l'url (/monaction -> /monaction-hijax),
- la partie serveur publie la syntaxe, le format et les schémas des requêtes et réponses métiers,
- la partie cliente javascript interprète les réponses métiers du serveur et sait également traiter les cas particuliers issus des modules techniques côté serveurs tels que les demandes d'authentification ou les redirections http.

L'exemple suivant illustre la différence de fonctionnement entre les deux modes dans le cas d'une liste affichée sous la forme d'un tableau paginable :

Sans hijax (mode « Web 1.0 »	Avec hijax
1. L'utilisateur demande la première page de la liste,	
2. L'application retourne un document HTML complet (entête, menu, corps, bas de page) avec un tableau des 25 premiers éléments et des liens hypertextes pour la pagination,	
	2.a. Le mode hijax s'initialise dans le navigateur et les liens hypertextes pour la pagination sont court-circuités par le code javascript contenu dans la page et prévu pour l'interactivité et les échanges ajax,
3. L'utilisateur demande à accéder à la page suivante en cliquant sur le lien hypertexte correspondant,	
3.a. Le navigateur émet une requête http pour afficher la page correspondante,	3.a. Le clic ayant été détourné par le mode hijax, le code javascript émet une requête ajax à la place

	de l'appel direct du navigateur en ajoutant un code permettant de reconnaitre la requête ajax,
4. L'application retourne un document HTML complet (entête, menu, corps, bas de page) avec un tableau des 25 éléments suivants et des liens hypertextes pour la pagination.	4. L'application reconnait la requête ajax et retourne un document partiel (sans entête, menu, corps, bas de page) contenant les 25 éléments suivants
	4.a. Le code javascript réceptionne la réponse à la requête ajax et met à jour le tableau.

Tableau 3 : Cinématique avec ou sans hijax

2.2.3.2 Mode Single Page Application (SPA)

Le mode SPA étend le mode hijax à toute une application (ou une partie) ; seul le corps principal est rechargé lors d'un changement de fonction, les parties comme l'entête ou le menu restent inchangées.

En plus des du mode « web 1.0 » et hijax, la partie serveur doit également être conçue pour renvoyer des réponses HTML constituées uniquement du corps principal.

Les contrats d'interface et de collaboration sont ainsi étendus entre les différentes parties :

- Il doit être possible de distinguer les trois types de requêtes : html complet, html partiel, xml ajax,
- les corps de page sont contraints de manière à pouvoir s'exécuter l'un après l'autre sans effet de bord,
- les identifiants DOM doivent être uniques dans l'ensemble des pages concernés par le mode SPA.
-

L'exemple suivant illustre le fonctionnement SPA par rapport au mode « web 1.0 » dans le cas d'une liste des pays et d'une liste des organisations.

Mode « Web 1.0 »	Mode SPA
1. L'utilisateur demande la première page de la liste des pays,	
2. L'application retourne un document HTML complet (avec entête, menu, corps, bas de page) avec un tableau des 25 premiers pays et des liens hypertextes pour la pagination,	
	2.a. Le mode hijax s'initialise dans le navigateur et les liens hypertextes pour la pagination sont court-circuités par le code javascript contenu dans la page et prévu pour l'interactivité et les échanges ajax,
	2.b. Les liens concernés par le mode spa, dont l'accès à la liste des organisations, sont court-circuités et remplacé par du code javascript,
3. L'utilisateur demande à accéder à la page suivante en cliquant sur le lien hypertexte correspondant,	
3.a. Le navigateur émet une requête http pour afficher la page correspondante,	3.a. Le clic ayant été détourné par le mode hijax, le code javascript émet une requête ajax à la place de l'appel direct du navigateur en ajoutant un code permettant de reconnaitre la requête ajax
4. L'application retourne un document HTML complet (avec entête, menu, corps, bas de page) avec un tableau des 25 pays suivants et des liens hypertextes pour la pagination,	4. L'application reconnait la requête ajax et retourne un document partiel (sans entête, menu, corps, bas de page) contenant les 25 pays suivants
	4.a. Le code javascript réceptionne la réponse à la requête ajax et met à jour le tableau.
5. L'utilisateur demande à accéder à la liste des organisations en cliquant sur le lien hypertexte correspondant dans le menu,	
5.a. Le navigateur émet la requête http	5.a. Le clic ayant été détourné par le mode hijax,

correspondante	le code javascript émet une requête ajax à la place de l'appel direct du navigateur en ajoutant un code permettant de reconnaitre la requête spa
6. L'application retourne un document HTML complet (avec entête, menu, corps, bas de page) contenant la liste des organisations.	6. L'application retourne un document partiel correspondant au document HTML de la liste des organisations sans l'entête, menu et bas de page.
	6.a. Le code javascript remplace le corps de page de la liste des pays par celui de la liste des organisations.

Tableau 4 : Cinématique « Web 1.0 » vs SPA

Chaque fragment HTML correspondant au corps de page est conçu pour être indépendant de son conteneur qui peut-être :

- soit la page complète obtenue lors d'un premier accès à la page liée au fragment HTML (ex. la liste des pays dans la page correspondante),
- soit la page complète obtenue lors d'un premier accès à une autre la page (ex. la liste des organisations accédée depuis la page de la liste des pays),
- soit un autre fragment HTML contenu lui-même dans un autre conteneur (ex. onglets)

Le concept de fragment HTML réutilisable dans différents conteneurs est appelé « pagelet » et est similaire aux concepts de porlets dans les spécifications JEE portail, avec l'avantage supplémentaire que les pagelets sont utilisables dans toutes pages HTML.

Le composant issu de la communauté YUI « gallery-dispatcher » est utilisé pour faciliter le développement SPA, il permet notamment d'exécuter les scripts contenus dans les fragments de document échangés en ajax.

2.3 Composants

2.3.1 Yahoo ! User Interface Library

2.3.1.1 Généralités

« The Yahoo! User Interface Library (YUI) is an open-source JavaScript library for building richly interactive web applications using techniques such as Ajax, DHTML and DOM scripting. YUI includes several core CSS resources. It is available under a BSD License.[1] Development on YUI began in 2005 and Yahoo! properties such as My Yahoo! and the Yahoo! front page began using YUI in the summer of that year. YUI was released for public use in February 2006.[2] It is actively developed by a core team of Yahoo! engineers.

In September 2009, Yahoo! released YUI 3, a new version of YUI rebuilt from the ground up to modernize the library and incorporate lessons learned from YUI 2. Among the enhancements are a CSS selector driven engine for retrieving DOM elements, a greater emphasis on granularity of modules, a smaller seed file that loads other modules when necessary, and a variety of syntactic changes intended to make writing code faster and easier.[3]

The YUI Library project at Yahoo! was founded by Thomas Sha and sponsored internally by Yahoo! co-founder Jerry Yang; its principal architects have been Sha, Adam Moore, and Matt Sweeney. The library's developers maintain the YUIBlog; the YUI community discusses the library and implementations in its community forum. »

Source: http://en.wikipedia.org/wiki/YUI_Library, le 10 décembre 2012.

Il existe actuellement deux versions de YUI : YUI2 et YUI3.

2.3.1.2 Graded Browser Support

YUI fournit un support des navigateurs et plateformes en trois niveaux (cf http://yuilibrary.com/yui/docs/tutorials/gbs/) :

Niveau du navigateur	Description
C-grade	Support de base sur la partie HTML uniquement. Une « liste noire » référence navigateurs et versions. « C-grade browsers are identified, incapable, antiquated and rare. QA tests a sampling of C-grade browsers, and bugs are addressed with high priority. »
A-grade	Niveau de support le plus élevé. 96 % des visiteurs des sites Yahoo ! utilise un navigateur A-grade. Une “liste blanche ” référence ces navigateurs et versions. « A-grade browsers are identified, capable, modern and common. QA tests all A-grade browsers, and bugs are addressed with high priority. »
X-grade	Niveau de support supposé identique au A (navigateurs en version plus récente que les navigateurs A-grade). Si des incompatibilités sont identifiées sur un navigateur X-grade, il est alors transféré dans la catégorie « C-grade ». « X-grade browsers are assumed to be capable and modern. QA does not test, and bugs are not opened against X-grade browsers. »

Tableau 5 : Niveaux de support YUI pour les navigateurs

Les tests d’assurance qualité sont passés par YUI uniquement sur les navigateurs en A-grade (intégralement) et en C-grade (sur la partie HTML uniquement).

Internet Explorer	6.0	7.0	8.0	9.0
Chrome †	Latest stable			
Firefox †	Latest stable			
Safari	Latest stable (desktop)		iOS 4.†	iOS 5.†
WebKit	Android 2.2.†	Android 2.3.†	Android 4.†	

Tableau 6 : Support des navigateurs A-grade au troisième trimestre 2012

(http://www.yuiblog.com/blog/2012/07/26/gbs-update-2012q3/)

Le symbole en croix (comme dans “iOS 5.†”) indique que la plus courante version non-beta de cette branche est prise en compte par le support.

2.3.1.3 Cohabitation de YUI2 et YUI3 via 2in3

Le projet 2in3 (https://github.com/yui/2in3) permet d’utiliser les composants YUI2 comme des composants natifs YUI3.

2in3 est embarqué dans YUI3, sans que les ressources js & css ne soient livrés avec YUI3.

Ce projet est géré par Dav Glass, un membre de l’équipe YUI.

Les modifications entre la version « normale » de YUI2 et la version pour YUI3 sont légères et réalisées par script. Elles consistent essentiellement à changer les noms de modules, et à enrober le script YUI2 dans une fonction « sandbox ».

2.3.1.4 Outillage pour le développement

YUI propose de nombreux outils et utilitaires pour les développements comme :

- Un logger à niveau « façon log4j »,
- Un compresseur de javascript et css,

- Un système de build basé sur ant pour faciliter le développement et l'intégration d'un composant,
- Un framework de tests unitaire « façon junit »,
- Profiler Javascript pour YUI 2 et 3,
- Plugin Firebug Yahoo ! YSlow pour les performances.

2.3.1.5 Agrégation dynamique de flux

Le système de dépendances entre les composants YUI permet d'utiliser des fichiers scripts agrégeant les sources de plusieurs composants (rollup).

Il est ainsi possible de réduire le nombre de requêtes http et de limiter les échanges pour télécharger le Javascript associé à une page.

2.3.1.6 Accessibilité

L'équipe YUI porte une attention particulière à l'accessibilité, il existe de nombreuses publications et études à ce sujet sur le site de YUI.

YUI dispose d'une équipe dédiée accessibilité (cf. http://yaccessibilityblog.com/library/).

2.3.1.7 Documentation

La documentation YUI est riche et (relativement) complète :

- Documentation des API sous un équivalent jsdoc,
- Tutoriaux et exemples d'implémentation,
- Nombreuses vidéos d'explication
- …

2.3.2 Adaptations Hornet de YUI

Les adaptations Hornet sont de trois ordres :

- Des normes et des documents d'accompagnement:
  - sélection de « comment faire » et « quoi utiliser » avec YUI,
- Des composants & des surcouches dans hornetclient:
  - Création de composants,
  - Extension de composants existants,
- Pour l'ergonomie et Accessibilité :
  - Des normes et « comment faire » spécialisés autour de l'accessibilité

2.4 Thèmes

Hornetclient est fourni avec un thème par défaut intégré. Hornet permet de mettre en œuvre et d'utiliser des thèmes autres en fonction de l'arborescence suivante :

[…]/hornetclient/X.Y.Z/themes

hornet-skin-themeSpec-2.0.0
hornet-skin- themeSpec -2.0.1
hornet-skin- themeSpec -2.0.2
hornet-skin- themeSpec -2.0.0

3 Principes techniques d'hornetserver

Hornetserver est indépendant de la partie cliente (hornetclient). L'architecture applicative JEE hornetserver repose sur plusieurs design patterns (ou paradigme de programmation) reconnus permettant une meilleure maintenance des applications et une flexibilité vers de nouvelles technologies.

3.1 Architecture

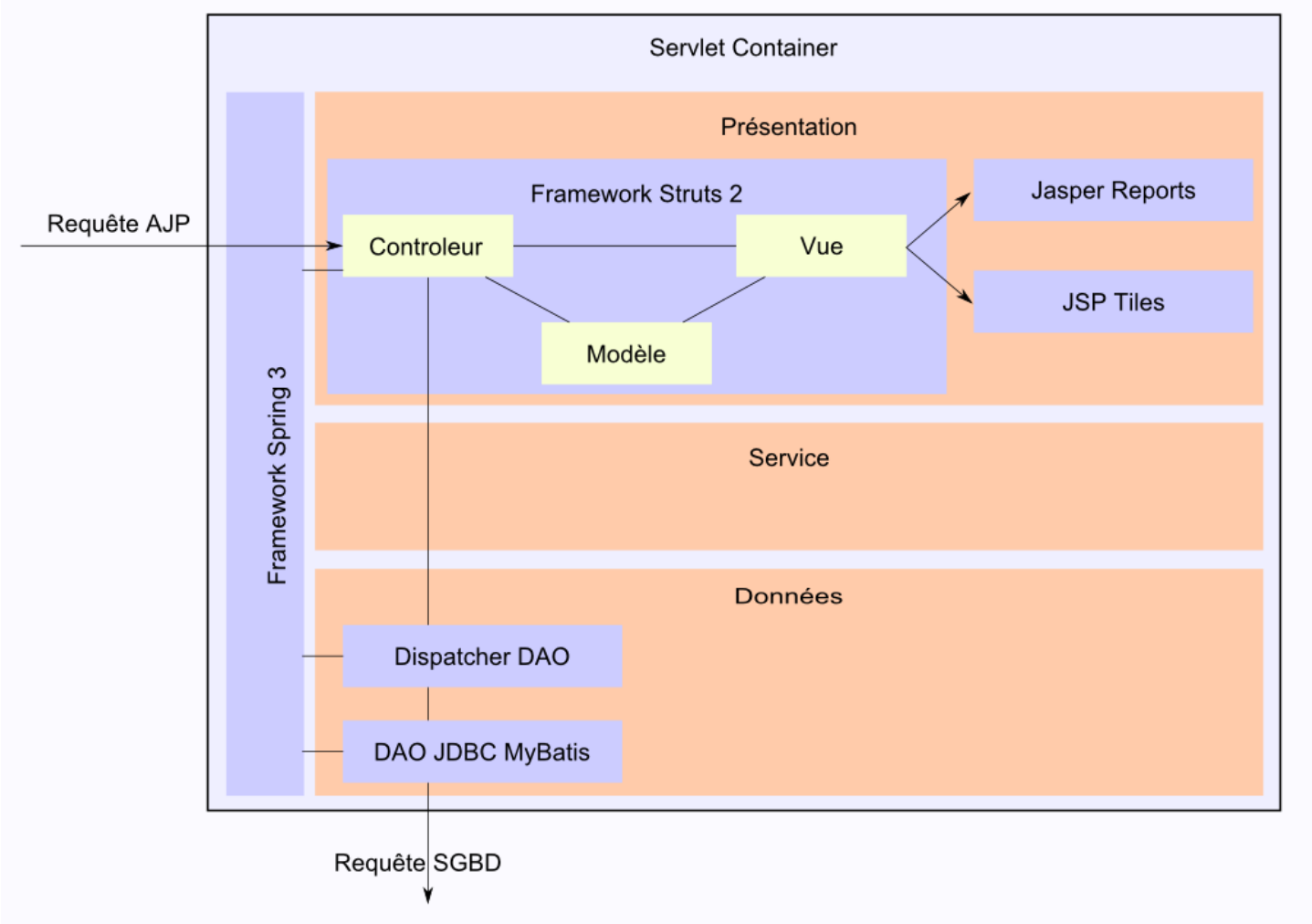


Figure 3 : Architecture hornetserver

3.2 Principes

3.2.1 Paradigme de programmation DAO

Le paradigme de programmation « Data Access Object » ou DAO permet de résoudre la problématique de séparation de la logique métier et de la logique rapatriement de données. Ainsi, l'accès à plusieurs accesseurs (exemple d'accesseur : JDBC, WebServices, JNDI, EJB, JavaMail, LDAP…) pour un même objet métier est entièrement transparent, permettant ainsi de s'affranchir du support de données et de s'ouvrir sur toute évolution possible.

L'accès à ces différents accesseurs s'effectue à travers des APIs qui sont spécifiques à l'accesseur. L'implémentation change donc en fonction de l'accesseur pour un même objet métier.

L'utilisation du design pattern DAO permet de faire abstraction de l'accesseur et de l'encapsuler au sein d'une interface indépendante de la méthode d'accès aux données. L'accesseur gère alors la connexion et la communication avec la source de données.

Dans cette structure, l'objet métier est décomposé en :

- un « Value Object » ou VO qui représente un objet « serializable » manipulant les données issues des différentes sources. Ce dernier est créé au sein d'un DAO
- un « Objet Delegate » qui encapsule les accesseurs au travers d'une interface standard (DAO) qui est déclinée en fonction des méthodes d'accès (DAO JDBC, DAO LDAP, …)

Le DAO représente l'interface aux méthodes statiques qui manipulent les VO. Il est implémenté par les différents accesseurs spécifiques (JNDI, EJB, JDBC…) nécessaires à l'objet.

Le DAO permet de cacher la complexité d'accès aux données tandis que le delegate permet de « typer » cet accès en sélectionnant le bon accesseur.

La structure simplifiée de ce design pattern est représentée ci-dessous :

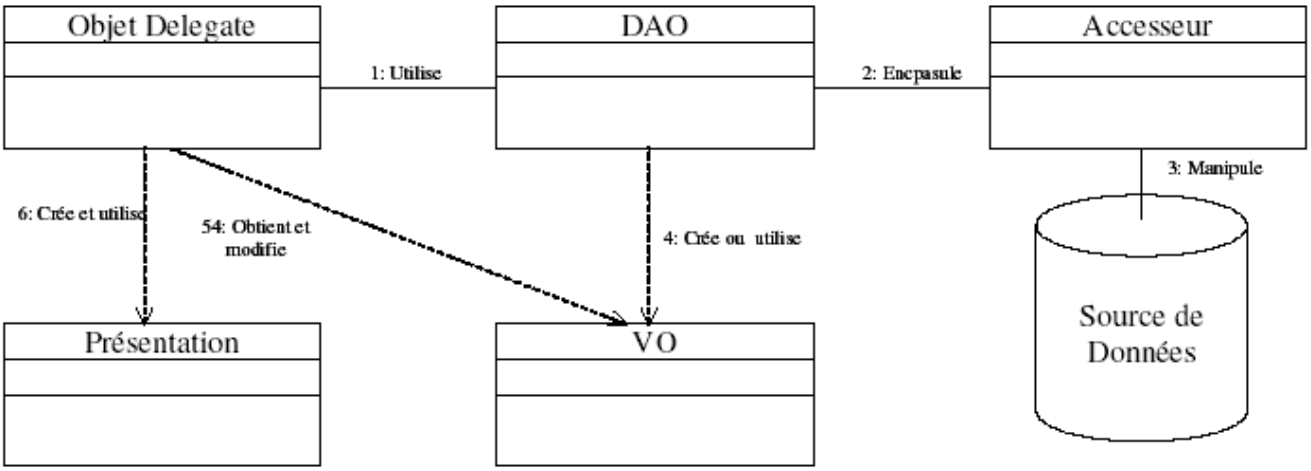


Figure 4 : Design Pattern DAO

Ce paradigme conseillé par Sun fournit une architecture ouverte vers de futures évolutions et favorise la mutualisation du code nécessaire aux accesseurs propres aux besoins du projet.

3.2.2 Paradigme de programmation MVC

Le modèle MVC (Model / View / Controller) en trois couches prend en charge les interactions avec les utilisateurs, les traitements métiers de l'application et l'accès aux sources de données auxquelles est connectée l'application. Ces couches sont généralement complétées par un socle technique en charge de services transversaux partagés (journalisation, configuration, …).

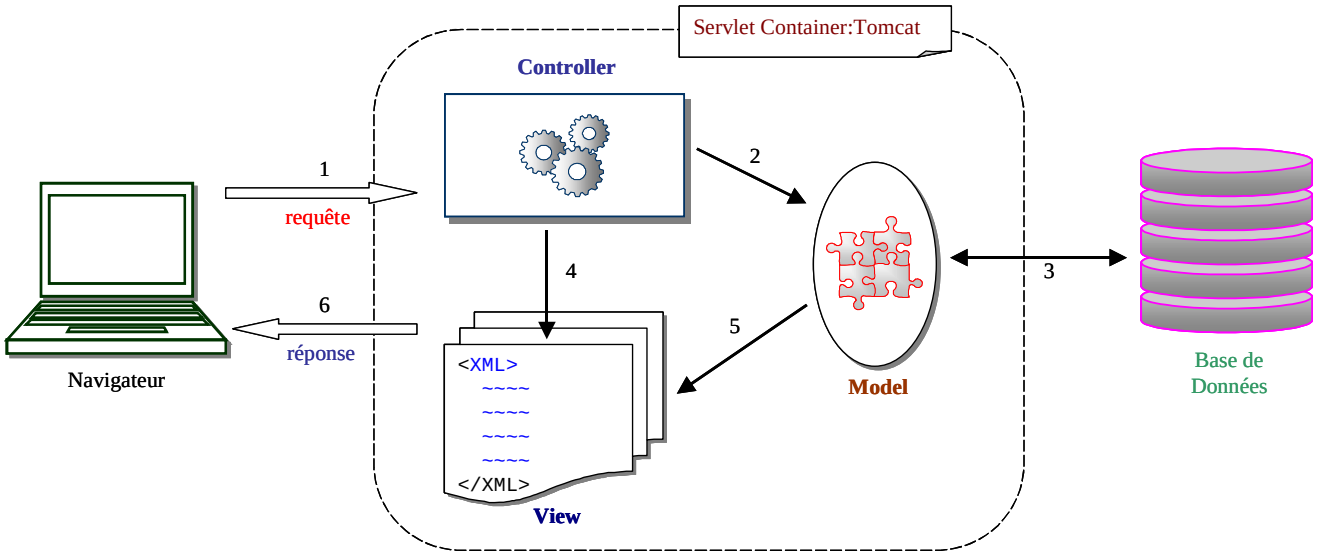


Figure 5 : Design Pattern MVC

Couche	Description
Le contrôleur	une servlet unique directement invoquée récupère les informations provenant de l'utilisateur et contrôle la validité des actions demandées par le client. Le contrôleur appelle le bon traitement applicatif dans le modèle, et renvoie vers la vue appropriée. C'est donc à ce niveau que la logique de navigation et la vérification de la validité de la session utilisateur (timeout…) est mutualisée.
Le modèle	couche principale en charge des traitements applicatifs et des objets métiers. Le modèle ne doit pas manipuler d'éléments liés à l'interface graphique ou faire référence à des servlets ou des vues. Les objets modélisant le métier proviennent directement de l'analyse fonctionnelle. L'intérêt de ces objets est de cacher la complexité de l'accès aux données et d'offrir une interface simple et proche du domaine que l'on désire représenter.
La vue	génération des formats de sortie attendue pour affichage résultant de l'action de l'utilisateur. Ce code devra être aussi court que possible (les traitements se déroulent dans le modèle et les mécanismes d'accès aux données sont encapsulés dans les objets métiers).

Tableau 7 : Les couches MVC

Dans ce modèle, le cycle de vie d'une requête est alors le suivant :

1. Le client envoie une requête à l'application. La requête est prise en charge par le contrôleur (étape 1).
2. Le contrôleur analyse la requête et réoriente celle-ci de manière à exécuter les traitements nécessaires à la satisfaction de la requête. Il sollicite pour cela les objets métiers (étape 2).
3. Les objets métiers fournissent des données (gestion des requêtes SQL) au contrôleur (étape 3).
4. Le contrôleur encapsule les données métiers dans des JavaBeans, sélectionne la vue qui sera en charge de la construction de la réponse et lui transmet les JavaBeans contenant les données métier (étape 4).
5. La vue construit la réponse (étape 5) en faisant appel aux JavaBeans qui lui ont été transmis et l'envoie au navigateur (étape 6).
6. Lorsque nécessaire, pour le traitement d'erreurs par exemple, le contrôleur traite directement la requête, sélectionne la vue de sortie et lui transmet les informations (étape 4) dont elle a besoin afin d'afficher un message approprié (étape 6).

Ce modèle en couches présente de nombreux avantages parmi lesquels :

- il facilite le découpage du travail et les responsabilités des développeurs en définissant plusieurs profils types d'intervenants,
- il améliore la réutilisation et la mutualisation du code dans une optique de qualité des développements,
- il facilite la maintenance. En cas de bogue ou de problème de performance, ce découpage permet de détecter plus rapidement la source du problème et de corriger la partie défaillante sans toucher aux autres éléments,
- il a été totalement implémenté à travers plusieurs solutions libres.

3.2.3 Paradigme Ioc / injection de dépendances

L'inversion de contrôle (Inversion of Control, IoC) est un patron d'architecture commun à tous les frameworks. Il fonctionne selon le principe que le flot d'exécution d'un logiciel n'est plus sous le contrôle direct de l'application elle-même mais du framework ou de la couche logicielle sous-jacente.

L'inversion de contrôle est un terme générique. Selon la problématique, il existe différentes formes, ou représentation d'IoC, le plus connu étant l'injection de dépendances qui est un patron de conception permettant, en programmation orientée objet, de découpler les dépendances entre objets.

L'injection de dépendances (Dependency Injection) est un mécanisme qui permet d'implanter le principe de l'inversion de contrôle. Il consiste à créer dynamiquement (injecter) les dépendances entre les différentes classes en s'appuyant généralement sur une description (fichier de configuration). Ainsi les dépendances entre composants logiciels ne sont plus exprimées dans le code de manière statique mais déterminées dynamiquement à l'exécution.

3.3 Principaux composants

3.3.1 Struts 2

Struts 2, framework de classes Java créé en open source dans le cadre de l'ASF (Apache Software Foundation), implémente complètement le modèle MVC2 en proposant un mécanisme d'automatisation des relations entre servlets, JSP et objets métiers. Cette intégration se fait par le biais des classes ActionForms.

La manipulation des classes ActionForms au sein des JSP est grandement facilitée par l'emploi des taglibs essentiellement dédiées à cet usage. Struts 2 fournit donc un peu plus d'une cinquantaine de tags personnalisés regroupés au sein de quatre librairies (taglibs) :

- La librairie des tags bean pour la manipulation pure de JavaBeans,
- La librairie des tags HTML pour la manipulation des formulaires HTML,
- La librairie des tags logiques pour la mise en place de traitements conditionnels et/ou itératifs,
- La libraire des tags template qui propose un mécanisme de gestion de modèles de JSP (les templates).

En synthèse, Struts 2 propose de construire les applications Java autour d'une organisation de l'Interface Homme Machine en implémentant le modèle MVC2.

Cette organisation apporte un respect net de la séparation entre présentation (JSP) et contrôle du dialogue (Actions) en excluant autant que faire se peut tout code Java des JSP (utilisation des taglibs).

Par ailleurs, la centralisation des accès à l'application via le contrôleur permet un contrôle fin et personnalisé des traitements et offre une grande marge de manœuvre pour la gestion des profils ou des rôles utilisateurs.

Struts 2 implémente le modèle MVC2, tout en offrant une architecture beaucoup plus souple que Struts 1.x, notamment par l'intégration des éléments suivants :

- Support de l'injection de dépendances au niveau des actions (via Spring)
- Utilisation d'OGNL pour le mapping des valeurs de formulaires, et possibilité d'utiliser n'importe quelle classe comme une action, n'obligeant plus d'étendre les classes Action et ActionForm et rendant possible l'utilisation de simple POJO pour coder les actions
- Mécanisme d'intercepteurs pré et post traitement permettant la factorisation de traitements communs

Struts 2 offre en particulier aux applications Hornet une bonne testabilité, sous forme de tests unitaires des classes actions aidée par l'indépendance de Struts 2 de l'API Servlet.

3.3.2 Spring 3

Le framework Spring a été développé dans l'esprit d'offrir une infrastructure permettant le développement d'application Java d'entreprise, en s'affranchissant des lourdeurs de la norme J2EE, et notamment des EJB. Pour cela, Spring propose un conteneur léger de Java Beans, gérant le cycle de vie des objets, et reposant sur le pattern de l'injection de dépendance.

Habituellement, pour initialiser un objet qui dépend d'un autre objet, deux stratégies pouvaient être envisagées : l'utilisation de l'opérateur new, ou la mise en œuvre d'une factory, souvent codée par le développeur.

Spring offre une factory générique paramétrable via des fichiers XML décrivant les dépendances entre les classes de l'application. Une fois les dépendances décrites, Spring gère l'instanciation des objets et leur injection dans les objets dont ils dépendent.

Cette rupture de dépendances dures entre les objets offre plusieurs avantages :

- Modularité des composants développés
- Testabilité accrue

Dans le domaine des tests, Spring offre également un plus value importante en offrant une intégration poussée avec JUnit pour permettre de tester rapidement, et dans l'IDE, l'intégration des différents composants.

3.3.3 MyBatis

MyBatis, issu d'iBatis, ancien projet de l'ASF (Apache Software Foundation), est un framework de mapping objet-relationnel, permettant la mise en place d'une correspondance entre des requêtes SQL et des instances d'objets, en se basant sur des fichiers de description au format XML.

Associé au framework Spring (en particulier son module Spring DAO), et par l'intermédiaire de son outil iBator, MyBatis permet la génération automatique des VO et des DAO en se basant sur le schéma de la base de données, en proposant un mapping automatique des types JDBC sur les types de base du langage Java. L'association avec Spring permet également à MyBatis de s'appuyer sur le support poussé des transactions offert par celui-ci.

Utilisant JDBC, MyBatis est compatible avec toutes les bases de données disposant d'un driver JDBC.

3.3.4 Tiles : Gestion facilitée des différents modes

Le framework hornetserver est mis en œuvre pour faciliter le rendu selon les différents modes (html complet, partiel ou XML).

Le composant Tiles sert à gérer les compositions de page en implémentant le pattern « composite view » (cf. http://java.sun.com/blueprints/patterns/CompositeView.html) :

Des modèles de page extensibles et génériques permettent de factoriser le rendu HTML (ou autre), chaque page étends le modèle voulu et seule la partie variante (généralement le body) est à écrire.

3.3.5 JasperReports

JasperReports est un outil de Reporting OpenSource, offert sous forme d'une bibliothèque qui peut être embarquée dans tous types d'applications Java.

JasperReports se base sur des fichiers XML pour la présentation des états.

3.3.6 Patterns ergonomiques paramétrables et extensibles

Des patterns d'ergonomie sont définis et matérialisés par des classes/packages/jsp et autres artefacts jee :

- Cadre d'application avec menu, entête et corps de page,
- Interaction tableau éditable et paginable,
- Formulaire de saisie avec contrôle de format,
- …

Ces patterns correspondent aux interactions classiquement rencontrées au sein des applications web.

Chaque pattern propose un paramétrage standard et des points d'extension, par exemple le paramétrage pour le pattern « tableau éditable » correspond :

- aux différentes colonnes à afficher et leurs caractéristiques (format, triable …)
- au formulaire d'édition lié au tableau.

Certains paramétrages se font dans des descripteurs XML ou bien dans les jsp.