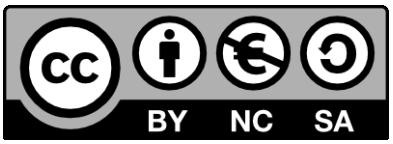
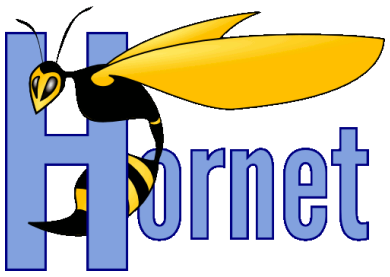


[Hornet]
Guide de migration d'un projet Acube vers Hornet



Développement Hornet 3.4C

Version : 1.0 du 25/04/2014 - **Etat** : Validé

SUIVI DES MODIFICATIONS

Version	Auteur	Description	Vérification	Date
1.0	O. Coatanlem F. Bernier-Malcoiffe A. Frigout	Initialisation du document	S. Heurtematte	07/11/2013

DOCUMENTS DE REFERENCE

Version	Titre
3.2	Guide de création d'un projet Hornet
3.4	Guide du développeur Hornet
1.0	Norme d'encodage

SOMMAIRE

SUIVI DES MODIFICATIONS **2**
DOCUMENTS DE REFERENCE **2**
SOMMAIRE **3**
TABLEAUX **3**
FIGURES **3**
1 INTRODUCTION **4**
2 PROCESSUS DE MIGRATION **5**
2.1 MOTIVATION ET PRINCIPE DE MIGRATION 5
2.2 PROCESSUS GENERAL 5
2.3 IMPACTS COTE SOCLE TECHNIQUE 5
2.4 RECENSEMENT DES IMPACTS ET CHANGEMENTS COTE CLIENT 5
2.4.1 Règles de gestion *6*
2.5 RECENSEMENT DES IMPACTS COTE SERVEUR 6
2.6 PRISE EN COMPTE DE LA NOUVELLE ARCHITECTURE 6
2.7 GESTION DE LA SECURITE 6
2.8 CREATION DE LA STRUCTURE ET DES MODELES DE PAGE CLIENTE 7
2.9 MISE A JOUR DE LA STRUCTURE DU PROJET SERVEUR 7
2.10 VERIFICATION DE FONCTIONNEMENT ET DE NON REGRESSION. 7
3 MIGRATION A PARTIR D'UN GABARIT HORNET **8**
3.1 GENERATION D'UN PROJET HORNET 8
3.2 GESTION DES DEPENDANCES 8
3.3 ARBORESCENCE PROJET 8
3.4 PRISE EN CHARGE DE L'ENCODAGE UTF-8 8
3.5 CONFIGURATION DU PROJET HORNET 8
3.5.1 Struts *8*
3.5.2 Tiles *9*
3.5.3 MyBatis *9*
3.5.4 Spring *9*
3.5.5 Spring Security *9*
3.6 REFONTE PARTIE CLIENTE 9
3.7 APPLICATION D'UN THEME 10

TABLEAUX

Aucune entrée de table d'illustration n'a été trouvée.

FIGURES

Aucune entrée de table d'illustration n'a été trouvée.

1 Introduction

Ce guide de migration présente les grandes étapes et principes d'une migration d'application ACube vers Hornet.

Hornet est un Framework, un outillage, une méthodologie qui permet de réaliser efficacement des applications Web accessibles (conformes au RGAA niveau AA) et performantes.

Hornet est composé de plusieurs parties :

- hornetclient pour les parties CSS et JavaScript sur les postes et navigateurs,
  - un thème par défaut
  - des composants (onglet, formulaire, menu horizontal, tableau, calendrier, …)

  hornetclient s'appuie sur la bibliothèque de composants éprouvés « Yahoo User Interface Toolkit » (YUI).
- hornetserver pour faciliter le développement java côté serveur,
- hornettemplate pour créer un gabarit de projet Hornet

Dans la suite de ce document, FRED 2 désigne les versions 2.x du Framework ergonomique FRED ACube, et LISE 3 désigne les versions 3.x du Framework LISE ACube.

Le remplacement de ACube par Hornet apporte des solutions et des améliorations sur des points tels que l'accessibilité, l'ergonomie ou encore la facilité de développement.

2 Processus de migration

2.1 Motivation et principe de migration

Il y a plusieurs motivations possibles pour une migration vers Hornet comme :

- Disposer d'une application avec une nouvelle IHM moderne,
- Satisfaire des exigences d'accessibilité,
- Ajouter des fonctionnalités à des écrans ACube existants.

La migration d'une application ACube FRED 2 et LISE 3 vers Hornet peut se faire partiellement ou totalement.

Cependant vu les exigences d'accessibilité et la structure des applications FRED 2, la coexistence d'Hornet et FRED 2 au sein d'une application n'est pas souhaitable.

Dans la plupart des cas, la migration d'Hornet implique une refonte totale de la partie présentation d'une application.

Il est toujours possible de migrer partiellement une application ou d'utiliser Hornet et FRED2 dans une même page, mais ces cas doivent faire l'objet d'une étude spécifique.

2.2 Processus général

Le processus général d'une migration est :

- Impacts côté socle technique
- Recensement des impacts et des changements côté client,
- Recensement des impacts côté serveur,
- Prise en compte de la nouvelle architecture,
- Création de la structure et des modèles des pages clientes,
- Mise à jour de la structure de la partie serveur,
- Vérification de fonctionnement et de non régression.

2.3 Impacts côté socle technique

ACube (avec LISE jusqu'à 3.6.0) est basé sur le socle technique Debian 5/Java 5/Tomcat 5.
Hornet est basé sur le socle technique Debian 6/Java 6/Tomcat 6.

La migration de Tomcat 5 à 6 et de Java 1.5 à 1.6 concerne surtout les environnements d'exécution et de développement à tous les stades du cycle de vie du projet.

Pour rappel, il ne faut plus utiliser les packages sun.* puisqu'ils ne sont plus maintenus par Oracle et ne sont pas portables. Dans la plupart des cas, il existe des équivalents dans les packages java.*.

2.4 Recensement des impacts et changements côté client

Ce recensement consiste à lister les impacts de l'utilisation d'Hornet à la place de FRED.
Hornet permet plus de souplesse sur les ergonomies des applications, par exemple :

- Les menus ne sont plus obligatoirement à gauche de la page, et peuvent être situés à l'horizontal sous l'entête,
- Il est possible d'afficher des fenêtres de dialogue modales,
- Le tableau éditable à la FRED 2 est remplacé au choix par :
  - deux pages, le tableau et le formulaire d'édition.
  - le formulaire d'édition en pop-in

2.4.1 Règles de gestion

Il faut également s'assurer qu'aucune règle de gestion décrite dans les spécifications ne soit effectuée uniquement au sein de la partie cliente. Si le cas s'avère être présent, le plan de test de l'application doit impérativement contenir les cas permettant de tester ces règles de gestion avec la version Hornet.

2.5 Recensement des impacts côté serveur

Côté serveur, les impacts concernent essentiellement la construction des pages HTML et les flux de données.

Avec Hornet, les pages HTML sont générées côté serveur (JSP) et contiennent les données. La présentation et l'affichage sont contrôlés par des feuilles de styles CSS.

Avec FRED2, les pages HTML contiennent uniquement la présentation et les données sont échangées avec le serveur via des flux XML et des requêtes Ajax.

Dans le cas où votre application est basée sur LISE 2, la mise en œuvre d'Hornet client nécessitera probablement une réécriture complète des classes actions et des flux associés et la migration du code serveur vers Hornet serveur.

2.6 Prise en compte de la nouvelle architecture

L'architecture Hornet diffère de manière importante de celle d'ACube; plus particulièrement s'agissant de la partie présentation.

Fondamentalement, avec ACube, les pages HTML ne contiennent que des conteneurs HTML. C'est le JavaScript (via FRED) qui enrichit dynamiquement ces conteneurs. Il crée des fragments de HTML (composants formulaires, tableaux, …) et les initialise avec les données applicatives récupérées de manière asynchrone sur le serveur. Avec un navigateur ne supportant pas le JavaScript, les pages sont, pour ainsi dire, vides.

Hornet a pour principe majeur le « *Progressive Enhancement* » (ou l'amélioration progressive).

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.

De plus, dans Hornet, les scripts client (YUI, hornetclient, thèmes) sont externes à l'application, sur un serveur distant.

L'architecture Hornet est détaillée dans les deux premiers chapitres du « Guide du développeur Hornet ».

2.7 Gestion de la sécurité

Depuis Hornet 3.1, la gestion de la sécurité des applications est assurée par Spring Security.

Ses principales fonctionnalités couvrent à la fois l'authentification et le contrôle d'accès basé sur la notion de rôle, au niveau, de l'IHM (par le biais d'une librairie de balise[1] dédiée), des URL et du code Java (par annotation).

La configuration de ce composant s'effectue principalement dans un fichier xml acceptant les déclarations de bean grâce à l'injection de dépendance de Spring et offrant un namespace[2] spécifique facilitant les paramétrages les plus basiques.

Pour plus de détails, se référer au Guide du développeur Hornet.

2.8 Création de la structure et des modèles de page cliente

Avec ACube, le « layout » des pages de l'application est effectué à l'aide du « frameset » de FRED 2.
Avec Hornet, il est réalisé à l'aide de JSP et Apache Tiles.
Cette étape consiste à supprimer l'ensemble de la partie cliente liée à FRED 2 (soit le contenu du répertoire « Client ») et créer les modèles Tiles équivalents.
Hornet Server inclut des taglib Struts2 pour faciliter la création de page fonctionnant avec Hornet client, ces taglib sont à privilégier dans les JSP.

2.9 Mise à jour de la structure du projet serveur

Côté serveur, les mises à jour concernent :

- L'arborescence globale
- Les tâches de construction (build.xml)
- La gestion des dépendances avec Ivy :
  Il faut dans un premier temps, supprimer le contenu du répertoire « lib » puis ajouter la librairie propre à Ivy.
  De même, il faut supprimer le répertoire « WEB-INF/lib », les librairies doivent être récupérées par les tâches Ivy.
  Pour la mise en place et l'utilisation d'Ivy, il faut se référer au chapitre dédié à la gestion des dépendances dans le « Guide du développeur Hornet ».
- L'ajout de Tiles comme plugin Struts pour la factorisation des pages HTML.
  Une modification de la configuration Struts est requise afin d'y inclure Tiles. De plus, d'autres fichiers de configuration tel que « tiles.xml » seront nécessaires.
- Les pages et scripts de référence
- Les fichiers de propriétés doivent être également externalisés du répertoire de source de l'application.

Pour ces changements, le plus simple est de créer un nouveau projet Hornet server à partir du template et de comparer avec le projet à migrer.

2.10 Vérification de fonctionnement et de non régression.

Comme montré précédemment, la migration d'une application d'ACube vers Hornet représente un effort important dont le résultat doit faire l'objet d'une vérification de fonctionnement et de non régression complète.

Une vigilance particulière doit être portée sur le point soulevé en §2.4.1 Règles de gestion.

[1] Taglib
[2] Espace de nommage

3 Migration à partir d'un gabarit Hornet

3.1 Génération d'un projet Hornet

Le delta entre une arborescence de type ACube et une arborescence Hornet est très important. Aussi, il est nécessaire de partir d'un projet Hornet bien organisé. Pour cela, on utilise un gabarit Hornet permettant de créer un nouveau projet. La création de ce projet doit être effectuée en suivant le « Guide de création d'un projet Hornet » basé sur l'utilisation de hornettemplate.

3.2 Gestion des dépendances

Dans Hornet, la gestion de configuration est assurée par Ivy. Il faut donc utiliser un fichier de configuration listant les dépendances plutôt que d'embarquer les librairies dans le répertoire « lib » du projet.

Un projet Hornet généré par hornettemplate contient déjà un ensemble de librairies de référence (Spring, Struts, Apache Commons, jTDS, …). Chaque projet utilise ensuite, si besoin, ses propres librairies qui sont renseignées dans le fichier de configuration d'Ivy.

3.3 Arborescence projet

Côté serveur, lorsqu'on a une arborescence de projet ACube ainsi qu'une bonne organisation des packages Java, la correspondance avec la nomenclature Hornet est assez simple. On peut facilement transposer les classes d'un package ACube vers un package Hornet.

3.4 Prise en charge de l'encodage UTF-8

Hornet, à partir de la version 3.0, gère l'UTF-8 sur l'ensemble de la chaîne. Cela implique :

- La configuration du projet et du workspace[3] en UTF-8
- La mise à niveau des scripts de construction des livrables à partir de ceux de hornettemplate
- L'ajout d'en-têtes aux fichiers XML et JSP
- La conversion de l'ensemble des ressources du projet en appliquant l'encodage UTF-8 sans BOM à l'exception des fichiers properties

Se référer aux règles décrites dans le document « Norme d'encodage » (§ « Composants applicatifs concernés par l'encodage »).

3.5 Configuration du projet Hornet

Dans un projet Hornet, les fichiers de configuration doivent être placés dans /src/config pour les fichiers constants à l'application : les fichiers de paramétrage (fichiers de propriétés) doivent eux être externalisés dans le répertoire « envconfig ».

3.5.1 Struts

La principale difficulté réside dans l'utilisation d'un fichier Struts fortement lié au Framework client FRED. La différence la plus importante concerne la gestion d'erreurs. Dans un projet Hornet, on emploie un interceptor alors que dans un projet ACube, on utilise un résultat global. Utilisé conjointement à Tiles, le mécanisme Hornet permet d'afficher les erreurs métier et technique directement dans la page courante sans redirection, simplifiant ainsi leur prise en charge.

[3] Espace de travail sous Eclipse

HORNET s'appuie sur une version récente de Struts (2.3.16) et de ses dépendances. Cela implique de reporter les nouveautés présentes dans hornettemplate sur le projet à migrer, à savoir:

- Ajout de l'en-tête xml UTF-8 dans le fichier de configuration Struts
- Remplacement des noms de ResultTypes et d'Interceptors dépréciés (renommé de la forme « spam-ham » à la forme CamelCase « spamHam »)
- Remplacement du FilterDispatcher de Struts 2.0 par son équivalent 2.3 dans le fichier web.xml

La solution la plus simple consiste sans doute à partir des fichiers de configuration de hornettemplate et d'y ajouter les actions du projet.

3.5.2 Tiles

Tiles n'est pas utilisé dans les projets ACube. La configuration présente dans le fichier *tiles.xml* du projet Hornet est le socle de base et sert de modèle afin de configurer l'ensemble des pages.

3.5.3 MyBatis

Le fichier référençant les configurations de mapping MyBatis est renommé en ibatis.xml et placé dans le répertoire /src/config. La nomenclature Hornet préconise de créer un fichier de mapping[4] par table en base de données et de le suffixer par « _SqlMap.xml ».

3.5.4 Spring

HORNET s'appuie sur une version récente de Spring (3.0.7.RELEASE) et des ses dépendances.

La migration de Spring de la version 2.5 à la version 3.0 implique des changements au niveau des fichiers de configuration de Spring :

- Mise à jour des références aux XSD (2.5 → 3.0)
- Remplacement des attributs « autowire » à « autodetect » (déprécié) par une construction des beans reposant sur l'emploi des tags « constructor-args ».

Là encore, hornettemplate permet de récupérer ces éléments pour les appliquer au projet à migrer.

3.5.5 Spring Security

Depuis la version 3.1 de Hornet, la sécurité des applications est gérée par Spring Security. Ce nouveau composant permet la gestion de l'authentification via des mécanismes type CAS, ou encore vis-à-vis d'une base de données contenant les informations utilisateurs ou d'un fichier déclaratif d'utilisateurs, ainsi que la gestion des droits au niveau de l'IHM et de la couche service.

La configuration s'effectue principalement dans un fichier XML comprenant entre autre

- Une liste d'associations pattern d'URL / liste de rôles autorisés à y accéder (pour gérer la restriction d'accès aux pages de l'application)
- La configuration du gestionnaire d'authentification (CAS, base de données …)
- La configuration du service chargé de récupérer les informations utilisateur (identité, profils)
- L'activation des annotations de sécurité pour la couche service

3.6 Refonte partie cliente

La modification la plus importante concerne la partie cliente. Le passage à Hornet implique un changement profond de fonctionnement de l'IHM. De ce fait, une refonte complète est nécessaire.

[4] Correspondance

Dans un projet ACube, la partie commune à toutes les pages de l'application (entête et pied de page) est dupliquée dans chaque page. La migration est l'occasion de mettre en commun ces éléments grâce à l'utilisation de Tiles et des layouts de base fournis par Hornet.

Dans tous les cas, les anciennes pages HTML sont entièrement réécrites sous forme de JSP éventuellement complétées par les composants YahooUI et hornetclient.

La structure des pages est revue. Dans un projet ACube, celle-ci est réalisée à l'aide de tableaux HTML. Dans un projet Hornet, il est préconisé d'utiliser YUI Grids CSS (basé sur des blocs ( ) et des styles CSS pour positionner les éléments de la page).

3.7 Application d'un thème

Un projet créé à l'aide de hornettemplate utilise le thème par défaut (cf. Guide de création d'un projet Hornet).

Si un autre thème est nécessaire, il est tout à fait possible de le développer. Il faut alors faire attention à ce qu'il corresponde au layout fourni par hornettemplate.

FIN DU DOCUMENT