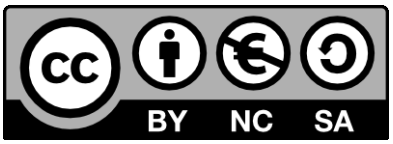
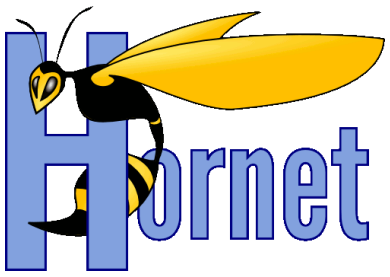


[HORNET]
Migration d'un projet Hornet 3.1 vers Hornet 3.4C



Développement Hornet 3.4C

Version : 1.1 du 13/06/2014 - **Etat** : Validé

SUIVI DES MODIFICATIONS

Version	Auteur	Description	Vérification	Date
1.0	O. Rousseil	Renommage fichier + numérotation Mise à jour pour Hornet 3.4		
1.1	S. Heurtematte	Mise à jour hornetclient	S. Heurtematte	13/06/2014

DOCUMENTS DE REFERENCE

Titre
HORNET_GUI_Création d'un projet Hornet_3.4
HORNET_GUI_Guide du développeur Hornet_3.4
HORNET_GUI_Guide de paramétrage_1.2
HORNET_NOR_Normes d'encodage_1.0

SOMMAIRE

TABLEAUX

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

FIGURES

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

1 Objectifs du document

Dans le cadre de la mise en place des évolutions au sein du Framework Hornet, ce document spécifie les actions à effectuer au sein d'un projet de type Hornet pour migrer vers la nouvelle version du Framework Hornet 3.4C à partir de la version précédente (3.1).

1.1 Principales évolutions

Hornet Sprint2 comporte des évolutions importantes :

- Mise en conformité de l'ensemble des sous-projets Hornet avec les règles et processus de construction, appliqués dans une construction via une Plateforme d'Intégration Continue (PIC)
- Externalisation des fichiers de propriété dans les applications Hornet
- Refactoring de hornetserver : réorganisation en multi-modules (core, web, clamav, clamavsimulateur, metrologiefilter, typemime, webservicehelper, httpparam)
- Nouvelle version de struts (2.3.15)

Hornet 3.4C comporte les évolutions et corrections :

- Reconfiguration log4j de la livraison
- Génération du template avec modification du nom de projet et de package
- Corrections de problème de compatibilité avec IE11 (et YUI)
    - ajout module « hornetoverride » dans « hornetclient »
- Nouvelle version de struts (2.3.16)

1.2 Versions cible des composants

Ce document est basé sur les versions de composants suivantes :

- hornetserver 3.4.0
- hornetclient 3.4.2
- hornettemplate 3.4.1

2 Mise à jour technique

2.1 Pré requis

2.1.1 Création d'un projet « modèle » à partir de hornettemplate

La migration vers Hornet 3.4C nécessite de récupérer de nouveaux fichiers et des fichiers mis à jour dans cette nouvelle version du framework. Ainsi, la source de ces fichiers doit être un projet « modèle » créé à partir de hornettemplate (cf. Guide de création d'un projet Hornet).

Dans la suite de ce document, sauf mention d'une autre source, les fichiers mentionnés sont donc ceux du projet créé à partir de hornettemplate **3.4.1.**

2.1.2 Thèmes

Les thèmes doivent être installés sur un serveur de framework Hornet.

2.2 Configuration du projet sous Eclipse

2.2.1 Pré-requis

Avant de commencer cette migration, il faut avoir installé et configuré l'environnement de développement conformément au guide de paramétrage, en particulier pour la configuration de Tomcat 6 et JDK 6 dans Eclipse.

2.3 Configuration de l'application

2.3.1 Tâche Ant

- **Supprimer**, s'ils existent, les fichiers : « **buildTemplate.xml** » et « **buildTemplate.properties** ».
- **Remplacer** le fichier de construction du projet « **build.xml** » et build.properties.
- **Modifier** le **« build.properties »** avec les données du projet

```
manifest.title                          = Projet MONPROJET
manifest.symbolicname                   = net.adullact.hornet.MONPROJET
manifest.version                        = 1.0
manifest.licence                        = http://www.cecill.info/licences/Licence_CeCILL_V2-fr.txt
manifest.implementation.vendor          =
manifest.specification.vendor           =
manifest.download.url                   =
manifest.doc.url                        =
```

L'externalisation des fichiers de paramétrage Ivy est également incluse dans cette nouvelle version. Ceux-ci doivent être présents dans le répertoire : « **C:\Users\NomUtilisateur\.ivy2 ».**

Il faut donc supprimer les fichiers ivysettings.xml et ivysettings.properties du projet.

Les cibles de construction globale des projets dépendent des types de déploiements : intégration ou release.

- Livraison en intégration : cible « **publish-project-all-integration** » du fichier « build.xml »
- Livraison en release : cible « **publish-project-all-release** » du fichier « build.xml »

2.3.2 Fichiers de propriétés externalisés et contexte

Depuis Hornet 3.1, les fichiers de propriétés sont maintenant externalisés. Aucune configuration propre à l'environnement ne doit figurer dans le livrable de l'application web après construction.

Le war produit doit être générique et la phase de reconstruction du livrable, prise en charge par le fichier dynamique.xml, n'est plus nécessaire.

2.3.2.1 Source de données

La source de donnée est définie au niveau d'un fichier Spring (spring-appContext-datasource.xml ou spring-appContext-dao.xml le plus souvent), injectant dans ce dernier le paramétrage externalisé nécessaire pour la connexion à la base de données (url, pilote, identifiants…).

Exemple de référencement dans un fichier de configuration Spring :

```
    <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="$jdbc{jdbc.driver}" />
        <property name="url" value="$jdbc{jdbc.url}" />
        <property name="username" value="$jdbc{jdbc.username}"/>
        <property name="password" value="$jdbc{jdbc.password}"/>
    </bean>
```

Fichier de contexte pour une base HSQLDB :

```
jdbc.driver = org.hsqldb.jdbcDriver
jdbc.url = jdbc:hsqldb:mem:EXEMPLE DB
jdbc.username = sa
jdbc.password =
```

2.3.2.2 Propriétés

Les fichiers de propriétés sont maintenant stockés dans un répertoire indépendant de l'application déployée dont l'emplacement est défini dans le fichier de description du contexte de l'application dans un tag `Environment`. Exemple :

```
    <Environment name="conf/applitutorielprop" value="D:\\devjava\\workspace\\applitutoriel/envconfig"
type="java.lang.String" />
```

Pour le développement, ce chemin est généré par un script Ant (initDev) et les fichiers sont stockés dans un répertoire « **envconfig** » du projet.

- Recopier le répertoire « **envconfig** » depuis le template puis supprimer les fichiers properties qu'il contient à l'exception du fichier « **hornet.properties** ».
- Déplacer tous les fichiers de configuration contenant des paramètres liés à l'environnement d'exécution dans ce répertoire (ex : serveur de mail, adresse serveur CAS, url framework client, emplacement des fichiers de log …).
- Supprimer le répertoire « **livrable/CONFIG-XXX** ».
- Recopier le répertoire « **livraison/webapp** » depuis le template et adapter son contenu (fichiers properties et context.xml) à partir du fichier « **dynamique.properties** » et des fichiers déjà déplacés dans « **envconfig** ».
- Supprimer le répertoire « **livraison/CONFIG-XXX** ».

Exemple de répertoire envconfig après création :

```
├──envconfig
│   │   cas.properties
│   │   hornet.properties
│   │   log4j.properties
│   │   mail.properties
│   │
│   └──template
│           context.xml
```

Exemple d'arborescence livraison + livrable **avant** modification :

```
├──livrable
│   ├──CONFIG-WEB
│   └──CONFIG-WEBAPP
│           dynamique.properties
│           dynamique.xml
│
├──livraison
│   └──CONFIG-WEBAPP
│       ├──src
```

```
|         |       cas.properties
|         |       log4j.properties
|         |       mail.properties
|         |
|         └──WebContent
|             ├──META-INF
|             |       context.xml
|             |
|             └──WEB-INF
|                     web.xml
```

Exemple d'arborescence livraison + livrable **après** modification :

```
├──livraison
|   └──webapp
|       ├──config
|       |       cas.properties
|       |       hornet.properties
|       |       log4j.properties
|       |       mail.properties
|       |
|       └──context
|               context.xml
```

2.3.2.3 *Références au ressources statiques Hornet*

Les URL vers les frameworks hornetclient et YUI étaient définis directement dans le descripteur de déploiement (web.xml). Deux filtres ont également été ajoutés pour prendre en compte des paramétrages externalisés.

- Mettre à jour le nouveau fichier « **hornet.properties** » avec les valeurs présentes dans le web.xml (URL framework client, URL thèmes …).
- Recopier le fichier web.xml du template en adaptant son contenu :
  - Id et « display-name » de l'application web.
  - nom du paramètre « **appConfigLocation** ».
  - liste des fichiers spring à charger.
  - suppression des références à HSQLDB si besoin.

2.3.2.4 *Préparation de l'application pour le développement*

Le contexte doit être initialisé pour le développement.

- Lancer la tâche « **init-dev** »
- Vérifier la présence du fichier « **WebContent/META-INF/context.xml** »
- Vérifier son contenu

2.3.3 hornetserver

2.3.3.1 *Gestion des dépendances*

- Faire pointer le projet vers la version « **hornetserver-web 3.4.0**».

Exemple :

```
<dependency org="fr.gouv.diplomatie.hornet" name="hornetserver-web" rev="3.3.0"
    conf="compile->master,libraries;provided;runtime->master,libraries;test"
    transitive="true">
        <artifact name="hornetserver-web" type="jar"></artifact>
    <exclude name="avalon-framework" />
    <exclude name="cglib-nodep" />
    <exclude name="dom4j" />
    <exclude name="velocity" />
</dependency>
```

La liste des librairies à exclure est à mettre à jour vis-à-vis de la configuration « core » de Hornet.

2.3.3.2 *Autres dépendances et update des librairies*

- Lancer la tâche Ant « **init-dev** » afin de récupérer les nouvelles librairies dans le répertoire : « **WEB-INF/lib** ».
- Vérifier la présence et la mise à jour des nouvelles dépendances :

- **hornetserver-core-3.4.0.jar**
- **hornetserver-web-3.4.0.jar**
- **struts2-core-2.3.16.jar**

- Puis celles du répertoire « **/lib** »
  - **lib/junit-4.8.2.jar**
  - **lib/spring-test-3.0.7.RELEASE.jar**

2.3.4 hornetclient

2.3.4.1 Ressources statiques

Modifier le fichier « **envconfig/hornet.properties** », afin de pointer la variable « **fwkRoot** » vers la nouvelle version du framework.

```
fwkRoot=http://<URL SERVEUR>/hornetclient/3.4.0/fwk
yui3Root=http://<URL SERVEUR>/yui/yui/3.8.1
```

2.3.4.2 Thèmes

- Modifier le fichier « **envconfig/hornet.properties** », afin de définir le thème et sa version :
  - **themeName**
  - **themeVersion**

```
themeName=diplonet
themeVersion=<VERSION>
```

2.3.5 YUI

La référence au framework Yahoo UI a changé. Vérifier que l'url **yui3Root** pointe bien vers la version **3.8.1** dans les fichiers « **envconfig/hornet.properties** » et « **livraison/webapp/config/hornet.properties** » issus du template.

2.4 Mise à jour struts

Depuis le passage à Struts 2.3.16, les tags « s:form » doivent avoir une action définie.

```
<s:form action="" id="idform" >

   <s:label for="id1" value="%{getText('champs1.libelle')}" requiredLabel="true"/>
   <s:textfield id="id1" name="champs1" />

   <s:label for="id2" value="%{'champs2'}" requiredLabel="true" />
   <s:datetimepicker id="id2" name="champs2" displayFormat="dd/MM/yyyy" />

   <s:submit value="Rechercher" />
   <s:reset value="Reinitialiser" />
</s:form>
```

Un bouton submit non visible peut être placé au début du formulaire. Il permet de s'assurer que l'action appelée en cliquant sur « enter » dans un champ est bien l'action par défaut. (D'autres boutons submit associés à d'autres actions peuvent en effet être présents dans le formulaire).

```
 <s:submit id="btnValiderDefault" cssClass="none" value="Valider" name="btnValider" />
```

3 Mise à jour structurelle

3.1 Pages dynamiques

3.1.1 Pages JSP

- Remplacer les JSP suivantes dans le répertoire /Webcontent/WEB-INF/tiles-jsp
  - **layout/baseLayout.jsp**
  - **frameset/nav.jsp**
  - **contact/contact.jsp**

> ➲ **Si des développements spécifiques ont été réalisés sur ces pages, ils seront écrasés. Les spécificités seront éventuellement à reporter dans les nouvelles pages.**

3.2 Ressources statiques

3.2.1 Scripts

Mettre à jour les ressources statiques dans le répertoire : **/WebContent/static/js/**
Modifier :

- **base.js**
- **form.js**

> ➲ **Si des développements spécifiques ont été réalisés sur ce fichier, ils seront écrasés. Les spécificités seront éventuellement à reporter dans les nouvelles pages.**
> ➲ **Attention : la gestion des paramètres de notification a évolué dans la méthode « createForm » (voir la signature de la méthode dans le fichier form.js). Il peut être nécessaire de mettre à jour les références vers ces paramètres.**

3.2.2 CSS

La modification du « baseLayout.jsp » implique de mettre à jour les classes CSS gérant l'affichage des éléments HTML.
Mettre à jour le fichier **/WebContent/static/css/global.css**.

Remplacer :

```
/* Element a masquer le temps du chargement */
.hornet-loading-page,
.hornet-loading-menu .menu,
.hornet-loading-table .table,
.hornet-loading-tabview .tabview,
.hornet-loading-form
{
    display: none;
}
```

Par :

```
/* Style a appliquer le temps du chargement de la page */
.hornet-page-loading
{

}
/* Elements a masquer le temps du chargement des composants yui */
```

```
.yui3-js-enabled .hornet-menu-loading,
.yui3-js-enabled .hornet-table-loading,
.yui3-js-enabled .yui3-tabview-loading,
.yui3-js-enabled .yui3-hornetautocomplete-loading
{
    display: none;
}

/* Element a masquer tout le temps */
.hornet-hidden
{
    display: none;
}
```

3.2.3 Utilisation des CSS pour le masquage des éléments HTML

3.2.3.1 Masquage des formulaires

Remplacer la classe « **hornet-loading-form** » par « **hornet-hidden** ».

3.2.3.2 Masquage des onglets

Ajouter la classe « **yui3-tabview-loading** » sur les éléments HTML conteneurs des onglets (ayant la classe « **tabview** ») :

```
<div class="tabview yui3-tabview-loading">
```

Supprimer le code :

```
Y.one(Y.config.doc.documentElement).removeClass("hornet-loading-tabview");
```

3.2.3.3 Masquage des tableaux

Ajouter la classe « **hornet-table-loading** » sur les zones HTML conteneurs des tableaux (ayant la classe « table ») :

```
<div class="table hornet-table-loading">
```

Remplacer les codes :

```
Y.one(Y.config.doc.documentElement).removeClass("hornet-loading-table");
Y.one(Y.config.doc.documentElement).addClass("hornet-loading-table");
```

Respectivement par :

```
Y.all(".table").removeClass("hornet-table-loading");
Y.all(".table").addClass("hornet-table-loading");
```

Pour en savoir plus, cf. « HORNET_GUI_Guide du développeur Hornet Sprint 3 », section 3.3.6.5 « Onglet ».

4 Mise à jour fonctionnelle

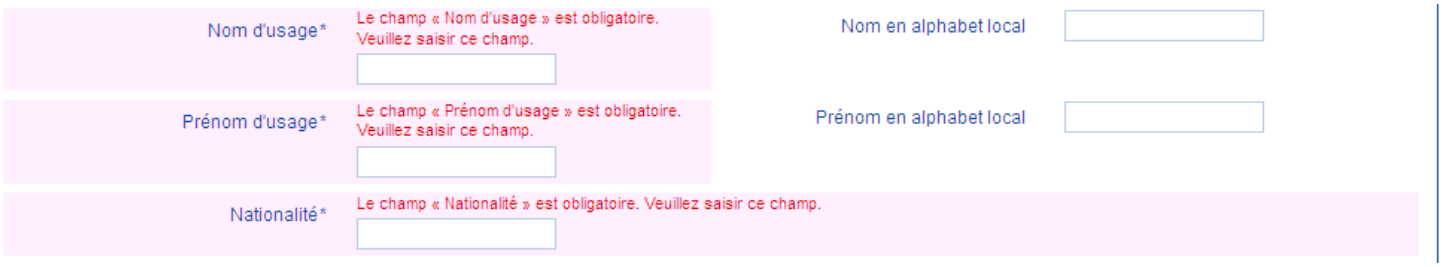
4.1 Composant Formulaire

4.1.1 Amélioration de la restitution des erreurs

Type : Evolution.

Amélioration de la restitution des erreurs lors de la saisie de formulaires :

- Dans la zone de notification, un clic sur un lien de message d'erreur met le focus dans le champ erroné correspondant.
- Après validation du formulaire, pour les champs erronés :
  - une mise en valeur via un fond rouge pâle
  - le message d'erreur est affiché au-dessus du champ (ou groupe de champs).



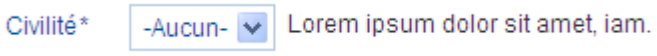
Pour en savoir plus, cf. « HORNET_GUI_Guide du développeur Hornet Sprint 3 », section 3.3.6.2.5 « Validation de formulaire ».

4.1.2 Amélioration des styles de formulaire

Type : Evolutions

1) De la même façon que pour les formulaires de recherche, il existe désormais un style à appliquer sur des éléments de texte dans un formulaire classique.

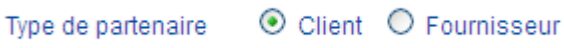
Exemple :




```
<form class="form"…>
…
  <p class="texte">Lorem ipsum dolor sit amet, iam.</p>
```

2) Par défaut, les boutons radio sont alignés verticalement. Il existe désormais un style pour qu'ils puissent être alignés horizontalement.

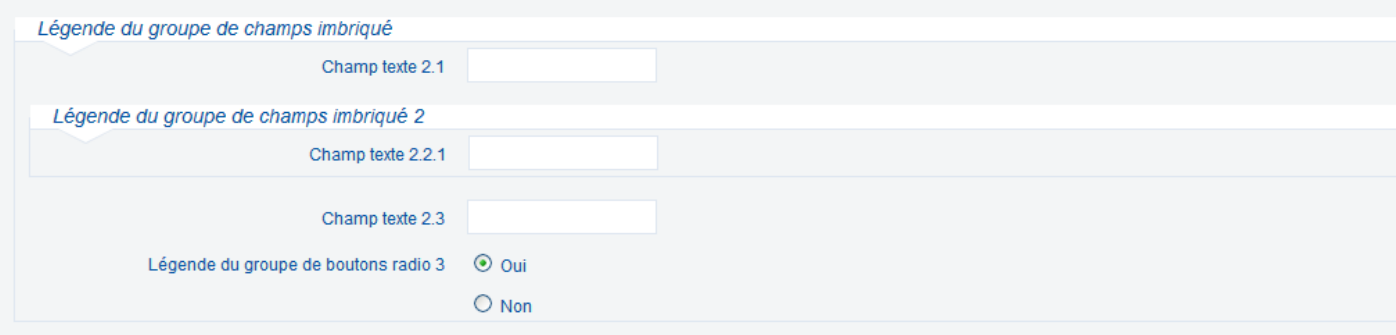
Exemple :




```
<div class="yui3-u-1-2 liste-radio-inline">
  <div class="formmgr-message-text"></div>
  <a tabindex="-1" id="partenaireEnCours.isClient_anchor" href="#partenaireEnCours.isClient_anchor"></a>
  <div>
    <input type="radio" value="true" checked="checked"
id="formEditionPartenaire_partenaireEnCours_isClienttrue" name="partenaireEnCours.isClient"><label
for="formEditionPartenaire_partenaireEnCours_isClienttrue">Client</label>
  </div>
```

```
   <div>
     <input type="radio" value="false" id="formEditionPartenaire_partenaireEnCours_isClientfalse"
name="partenaireEnCours.isClient"><label
for="formEditionPartenaire partenaireEnCours isClientfalse">Fournisseur</label>
   </div>
</div>
```

3) L'affichage d'un groupe de champs dans un autre groupe de champs a fait l'objet d'un travail d'ergonomie.



4.1.3 Composant Autocomplete

Type : Nouvelle fonctionnalité.

Le composant "Autocomplete" permet d'afficher un champ d'aide à la saisie couplé à une liste déroulante.

Pour en savoir plus, cf. « HORNET_GUI_Guide du développeur Hornet Sprint 3 », section 3.3.6.8 « Arborescence ».

4.2 Composant Onglet

4.2.1 Titrage des onglets

Ajout de titres (balises `hn`) pour chaque zone de contenu.

Pour en savoir plus, cf. « HORNET_GUI_Guide du développeur Hornet Sprint 3 », section 3.3.6.5 « Onglet ».

4.3 Masquage du markup HTML le temps du chargement des composant JavaScript

Les composants sont désormais masqués le temps du chargement JavaScript.

Utilisation des classes `yui3-[composant]-loading, hornet-[composant]-loading`.

Pour en savoir plus, cf. « HORNET_GUI_Guide du développeur Hornet Sprint 3 », chapitres dédiés à chaque composant.