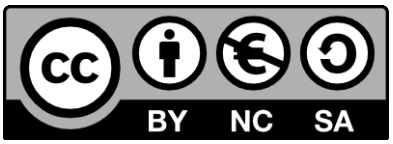
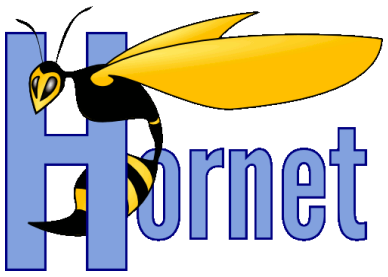


Création d'un projet Hornet



Débuter avec Hornet 3.6B

Version : 1.0 du 31/07/2014 - **Etat** : Validé

SUIVI DES MODIFICATIONS

Version	Auteur	Description	Vérification	Date
1.0	O. Rousseil	Renommage fichier + renumérotation Mise à jour pour Hornet 3.6B	S. Heurtematte	09/07/2014

DOCUMENTS DE REFERENCE

Titre
HORNET_GUI_Guide du développeur Hornet 3.6B_1.0A
HORNET_GUI_Guide de paramétrage_1.4
HORNET_CHA_CharteArchitecture_1.0
HORNET_GUI_Guide Deploiement du framework Hornet 3.6B_1.0A

SOMMAIRE

TABLEAUX

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

FIGURES

1 Introduction et pré requis

1.1 Objet

Ce document présente les étapes de création d'un projet Hornet 3.6B ainsi que l'intégration de nouveaux services au sein d'une application Hornet.

1.2 Pré requis

1.2.1 Connaissance de l'architecture Hornet

Consulter le document HORNET_CHA_Charte architecture.

1.2.2 Paramétrage du poste de développement

Appliquer le document HORNET_GUI_Guide de Paramétrage.

1.2.3 La disponibilité d'un serveur de framework hornetclient

Appliquer le document HORNET_GUI_Deploiement du framework Hornet.

2 Création d'un projet Hornet

2.1 Préparation

2.1.1 Récupération de hornettemplate

Récupérer le projet « **hornettemplate** » par un checkout sur le SVN de la forge Adullact.

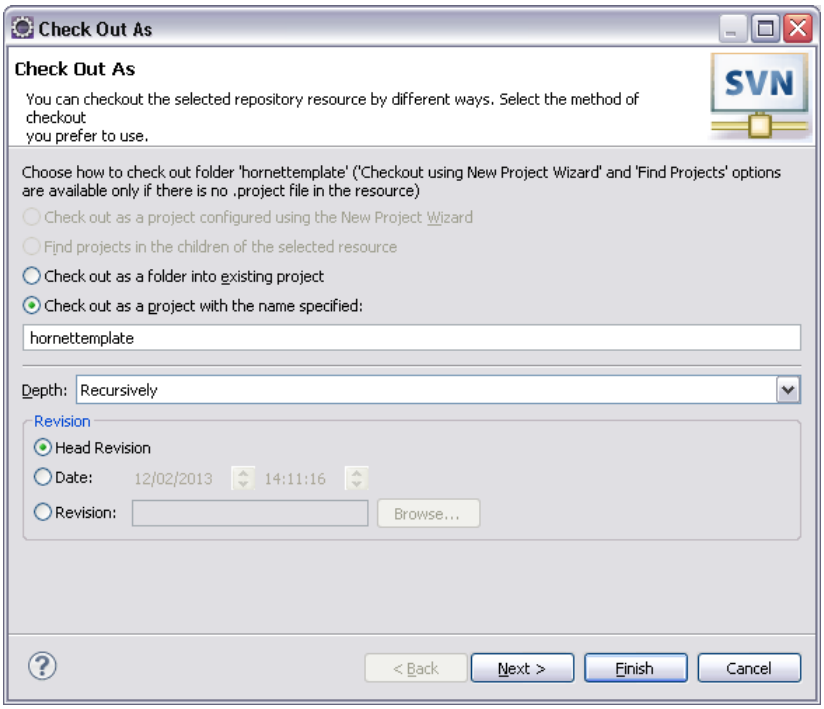


Figure 1 : Récupération du projet sous SVN

Le nom donné au projet sous Eclipse n'interfère pas dans le déroulement de la génération du projet. Le nom de projet spécifié doit être en minuscules et sans caractère espace.

> ➲ **Suite au bon déroulement du checkout, déconnecter le projet du SVN en supprimant les métadonnées SVN.**

2.1.2 Contenu initial de hornettemplate

Le projet se compose initialement des éléments suivants :

- Un répertoire « **dev** » : contient les éléments qui permettent la génération du template.
- Le répertoire est composé de deux sous-répertoires :
- « **dev/hornet** » : contient les sources du template permettant la génération du projet
- « **dev/style** » : contient l'ensemble des fichiers XSL qui seront utilisés pour générer des fichiers spécifiques
- Le fichier « **.project** » : fichier de configuration du projet sous Eclipse
- Le fichier « **buildTemplate.properties** » : contient les valeurs par défaut pour l'initialisation du template.
- Le fichier XML « **buildTemplate.xml** » : contient les tâches d'initialisation du template.
- Le fichier « **build.xml** » permet de publier sur le gestionnaire d'artefacts le template
- Le fichier « **build.properties** » : contient les propriétés pour le déploiement

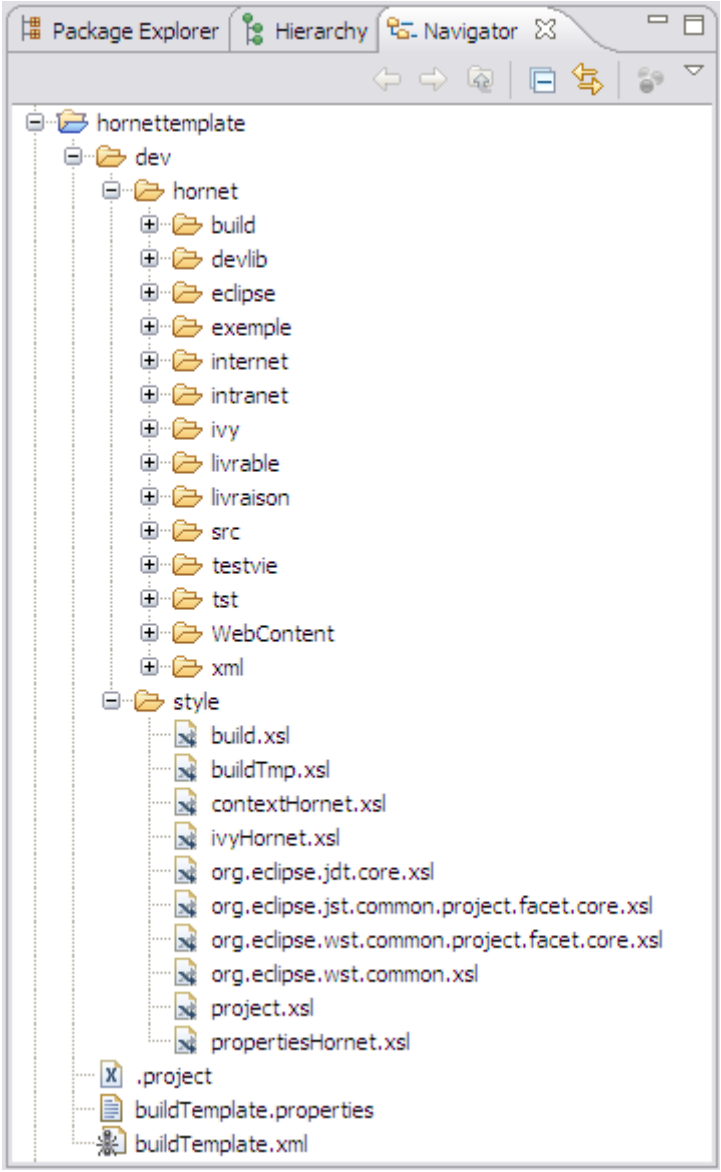


Figure 2 : Contenu initial du projet

2.2 Initialisation du projet

2.2.1 Nom de projet et nom de package

Dans la vue de navigation, éditer le contenu du fichier « **buildTemplate.properties** » afin de renseigner les variables de construction suivantes :

- **build.projectName** : Le nom du projet (par défaut : hornettemplate)
- **build.projectPackage** : Le nom du package racine (par défaut : net.adullact.hornet.hornettemplate

2.2.2 Configuration Ivy

2.2.2.1 Repository Ivy

Le repository compatible Hornet (disponible dans l'onglet Fichiers sur https://adullact.net/projects/hornet/) utilisé par Ivy doit être installé en local.

- Créer un répertoire local qui sert de référence dans la suite du document pour la configuration des propriétés Ivy. Le chemin vers ce répertoire est représenté par la variable <REPERTOIRE REPOSITORY>.
- Récupérer l'archive contenant le repository compatible Hornet 3.6B.

- Extraire l'archive et copier le contenu du répertoire « Repository_AAAAMMJJ » dans <REPERTOIRE REPOSITORY>.
  - L'arborescence obtenue sera la suivante :
    - <REPERTOIRE REPOSITORY>/repository-technique-local
    - <REPERTOIRE REPOSITORY>/repository-tiercesparties
    - <REPERTOIRE REPOSITORY>/repository-integration
    - <REPERTOIRE REPOSITORY>/repository-metier

2.2.2.2 Configuration Ivy

Les fichiers ivysettings.xml et ivysettings.properties doivent être stockés dans le profil utilisateur.
Ex : sous windows 7 : **C:\Users\NomUtilisateur\.ivy2**

Le fichier « **ivysettings.xml** » permet de configurer :

- Localisation des repositories
  - Fichier « **ivysettings.properties** »
- Accès aux repositories
- Cache
- Statuts

Le fichier ivysettings.properties contient la configuration des urls.

Note : il est fait référence au repository-technique-remote. Celui-ci représente les repos public type maven : http://repo1.maven.org/maven2
Il ne demande donc pas de configuration spécifique.

Exemple de ivysettings.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<ivy-settings>
  <properties file="ivysettings.properties" />

  <settings defaultResolver="main" />

  <resolvers>
    <chain name="main" returnFirst="true" >

            <filesystem name="repository-integration">
                  <ivy
pattern="${repository.integration.url}/${repository.integration.ivy.pattern}" />
                  <artifact
pattern="${repository.integration.url}/${repository.integration.artifact.pattern}" />
            </filesystem>

            <filesystem name="repository-technique-local">
                  <ivy
pattern="${repository.technique.local.url}/${repository.technique.local.ivy.pattern}" />
                  <artifact
pattern="${repository.technique.local.url}/${repository.technique.local.artifact.pattern}" />
            </filesystem>

            <ibiblio name="repository-technique-remote" m2compatible="true"/>

            <filesystem name="repository-technique-tiercesparties">
                  <ivy
pattern="${repository.technique.tiercesparties.url}/${repository.technique.tiercesparties.ivy.pat
tern}" />
                  <artifact
pattern="${repository.technique.tiercesparties.url}/${repository.technique.tiercesparties.artifac
t.pattern}" />
            </filesystem>
```

```
            <filesystem name="repository-metier">
                  <ivy pattern="${repository.metier.url}/${repository.metier.ivy.pattern}" />
                  <artifact
pattern="${repository.metier.url}/${repository.metier.artifact.pattern}" />
            </filesystem>

    </chain>
  </resolvers>

</ivy-settings>
```

Ivysettings.properties :

```
default.artifact.pattern=[organisation]/[module]/[revision]/[type]s/[module]-[revision](-
[classifier]).[ext]
default.ivy.pattern=[organisation]/[module]/[revision]/ivys/[module]-[revision]-ivy.xml

repository.url=${user.home}/.ivy2

# dépot en ligne pour les fichiers du framework acube et ses dépendances
repository.integration.url=${repository.url}/repository-integration
repository.integration.artifact.pattern=${default.artifact.pattern}
repository.integration.ivy.pattern=${default.ivy.pattern}

repository.metier.url=${repository.url}/repository-metier
repository.metier.artifact.pattern=${default.artifact.pattern}
repository.metier.ivy.pattern=${default.ivy.pattern}

repository.technique.local.url=${repository.url}/repository-technique-local
repository.technique.local.artifact.pattern=${default.artifact.pattern}
repository.technique.local.ivy.pattern=${default.ivy.pattern}

repository.technique.tiercesparties.url=${repository.url}/repository-tiercesparties
repository.technique.tiercesparties.artifact.pattern=${default.artifact.pattern}
repository.technique.tiercesparties.ivy.pattern=${default.ivy.pattern}
```

2.2.3 Tâche d'initialisation

Dans la vue Ant, lancer la tâche par défaut « **initHornet** » du fichier XML « **buildTemplate.xml** ».

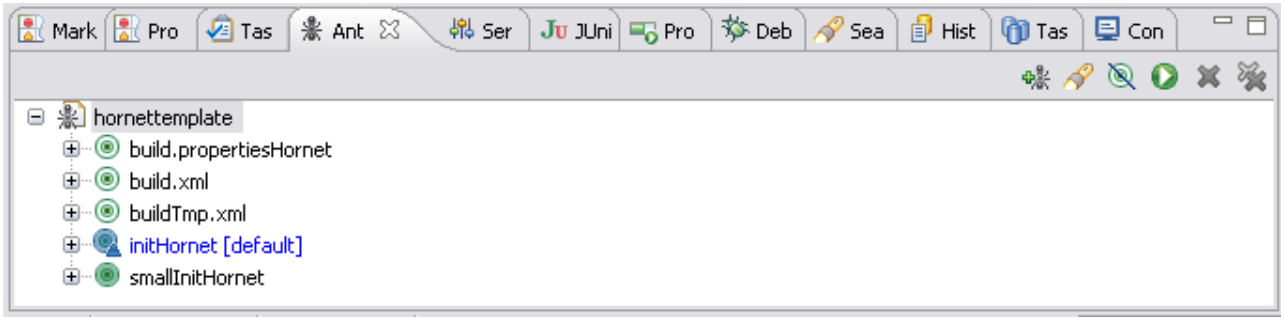


Figure 3 : Tâche « initHornet » du « buildTemplate.xml »

La console affiche le résultat suivant :

```
Buildfile: D:\devjava\workspace\hornettemplate\buildTemplate.xml
initHornet:
     [echo] initialisation du projet
    [mkdir] Created dir: D:\devjava\workspace\hornettemplate\lib\dev
     [copy] Copying 1 file to D:\devjava\workspace\hornettemplate\lib\dev
build.propertiesHornet:
```

```
     [echo] generation du fichier build.properties Hornet
     [xslt] Processing D:\devjava\workspace\hornettemplate\.project to
D:\devjava\workspace\hornettemplate\build.properties
     [xslt] Loading stylesheet D:\devjava\workspace\hornettemplate\dev\style\propertiesHornet.xsl
build.xml:
     [echo] generation du fichier build.xml
     [xslt] Processing D:\devjava\workspace\hornettemplate\dev\hornet\build\build.xml to
D:\devjava\workspace\hornettemplate\build.xml
     [xslt] Loading stylesheet D:\devjava\workspace\hornettemplate\dev\style\build.xsl
buildTmp.xml:
     [echo] generation du fichier buildTmp.xml
     [xslt] Processing D:\devjava\workspace\hornettemplate\dev\hornet\build\buildTmp.xml to
D:\devjava\workspace\hornettemplate\buildTmp.xml
     [xslt] Loading stylesheet D:\devjava\workspace\hornettemplate\dev\style\buildTmp.xsl
   [delete] Deleting: D:\devjava\workspace\hornettemplate\buildTemplate.xml
   [delete] Deleting: D:\devjava\workspace\hornettemplate\buildTemplate.properties
BUILD SUCCESSFUL
Total time: 500 milliseconds
```

*Figure 4 : Résultat de la console après la tâche « **initHornet** »*

Un rafraichissement du projet donne la vue suivante :

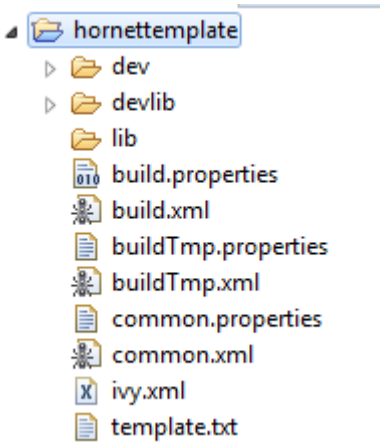


Figure 5 : Vue du projet après initialisation du template

Les changements sont :

- Les fichiers « **buildTemplate.xml** » et « **buildTemplate.properties** » ont été supprimés.
- Les répertoires suivants ont été créés :
  - **lib**
  - **devlib**
- Les fichiers suivants ont été créés :
  - **build.properties**
  - **build.xml**
  - **buildTmp.properties**
  - **buildTmp.xml**
  - **common.properties**
  - **common.xml**
  - **ivy.xml**
  - **template.txt**

2.3 Choix du type de projet

2.3.1 Types de projet disponibles

Il existe 4 types de projet avec le thème par défaut :

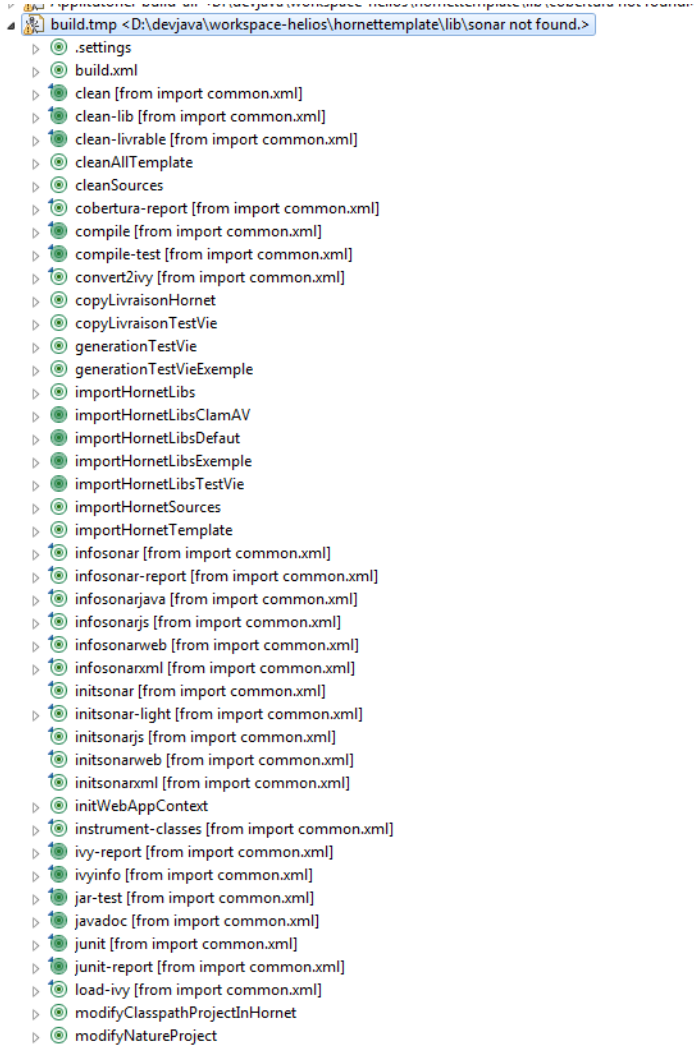
- **HornetDefaut :** projet basique et vide.
- **HornetWithClamAV :** projet avec la configuration pour ClamAV.
- **HornetWithExample :** projet avec des pages d'exemples.

- **HornetWithTestVie** : projet avec la configuration pour les Tests de Vie.

2.3.2 Taches de génération de projet typé

2.3.2.1 Génération du projet

Le fichier « **buildTmp.xml** » contient les tâches de génération du projet.



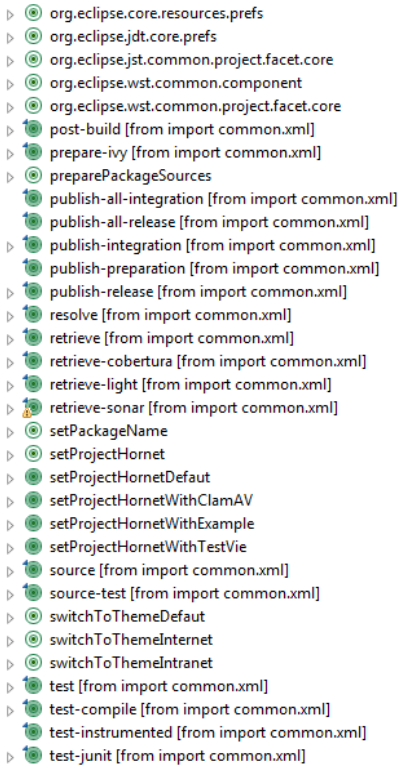


Figure 6 : Tâches du fichier « buildTmp.xml »

> ➲ **Il faut ensuite impérativement lancer une des tâches suivantes, selon le type de projet à créer :**

- « **setProjectHornetDefaut** » : permet la génération d'un projet avec le thème CSS par défaut.
- « **setProjectHornetWithClamAV** » : permet la génération d'un projet avec le thème CSS par défaut et la configuration pour ClamAV.
- « **setProjectHornetWithExample** » : permet la génération d'un projet avec le thème CSS par défaut et des pages d'exemples.
- « **setProjectHornetWithTestVie** » : permet la génération d'un projet avec le thème CSS par défaut et la configuration pour les Tests de Vie.

2.4 Etat par type de projet

2.4.1 Pour tout type de projet

Un rafraichissement du projet donne la vue suivante :

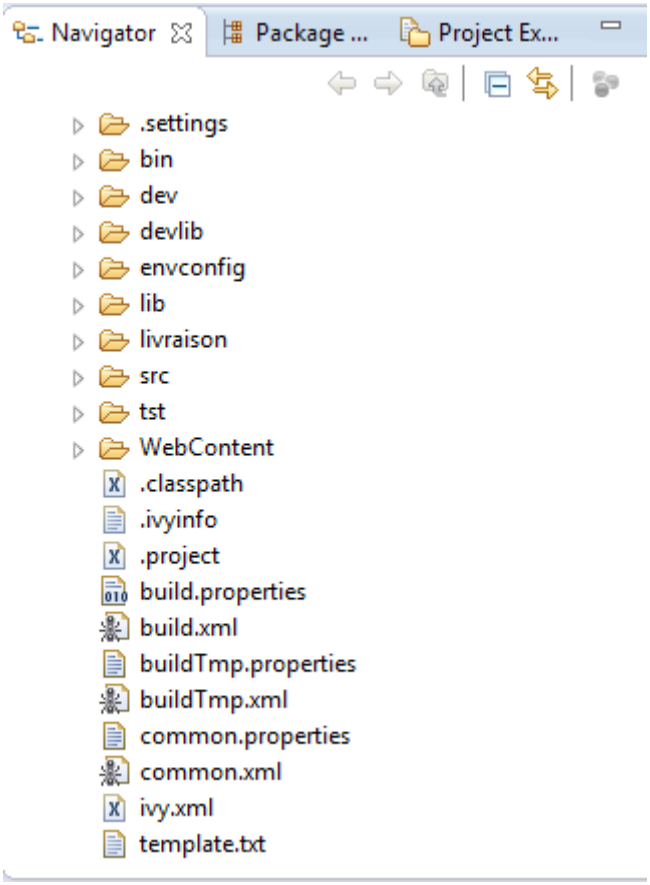


*Figure 7 : Vue du projet après exécution de la tâche « **setProjectHornetDefaut** »*

Les changements sont :

- Création du répertoire « **.settings** » qui contient des fichiers de configurations propre à Eclipse. Cela permet de déclarer le projet comme un projet Web sous Eclipse qui utilise Apache Tomcat et qui a pour répertoire source « **src/config** », « **src/java** », et « **tst/java** ».
- Création du répertoire « **bin** », utilisé par Eclipse et qui contient les sources java compilées du projet
- Ajout de librairies dans le répertoire « **lib** » : le projet est déclaré sous Eclipse comme utilisant les librairies présentes dans « **lib/test** ». Ce sont les librairies utilisés pour les tests, la librairie contenue dans « **devlib** » étant utilisé pour la récupération des librairies Hornet par Ivy et le répertoire « **WebContent\WEB-INF\lib** ».
- Création des répertoires « **livrable** » et « **livraison** », utilisés lors de la création des livrables du projet par les tâches du fichier « **build.xml** ».
- Création du répertoire « **envconfig** » contenant les fichiers properties, ex : « **log4j.properties** », « **mail.properties** » ainsi qu'un modèle de fichier « **context.xml** » pour le développement.
- Création du répertoire « **src/config** » contenant les fichiers de configuration Struts, Spring, Spring Security, Tiles et MyBatis et les fichiers « **package_fr.properties** » pour la gestion de l'internationalisation et de l'externalisation des libellés.
- Création du répertoire « **src/java** » contenant les sources java du projet
- Création du répertoire « **tst** » contenant les fichiers de test du projet.
- Création du répertoire Web « **WebContent** » par défaut, contenant les ressources web de l'application : les libraires Hornet, les pages JSP, les ressources statiques…
- Création du fichier « **.classpath** », fichier de configuration d'Eclipse.
- Création du fichier « **ivy.xml** » qui contient la configuration Ivy des librairies du projet.
- Création des fichiers globaux liés à la construction du projet en environnement d'intégration continue : « **build.xml** », « **common.xml** », « **common.properties** »

2.4.2 Vérifications pour tout type de projet

Il faut vérifier la présence des librairies nécessaires dans le répertoire « WebContent/WEB-INF/lib ». Si ce répertoire est vide :

- Vérifier la configuration Ivy

Il faut aussi vérifier que l'URL du serveur de framework est bien configurée dans le fichier « **envconfig/hornet.properties** » du projet.

2.4.3 Configuration du serveur de framework pour tout type de projet

La configuration initiale du serveur de framework hornetclient peut être effectuée dans le fichier « **hornet.properties** » sous « **envconfig/** » en modifiant les paramètres « **fwkRoot** » et « **yui3Root** » (<URL DU SERVEUR DE FRAMEWORK> étant l'URL sur laquelle est déployé le serveur de framework hornetclient).

```
#Éléments de configuration du framework
fwkRoot=http://<URL DU SERVEUR DE FRAMEWORK>/hornetclient/<VERSION HORNET>/fwk
yui3Root=http://<URL DU SERVEUR DE FRAMEWORK>/yui/yui/<VERSION YUI>
themeName=defaut
themeVersion=
```

2.4.4 Spécificités projet avec ClamAV

L'ajout de la configuration ClamAV dans le projet ajoute la librairie « **hornetserver-clamav-X.X.X.jar** » dans le répertoire « **WEB-INF/lib** » du projet.

2.4.5 Spécificités projet avec exemples

L'ajout des pages d'exemples dans le projet impacte les fichiers suivants :

- Les fichiers « **struts.xml** » et « **tiles.xml** » pour ajouter les chemins relatifs aux deux nouvelles pages d'exemple.
- Le fichier « **package_fr.properties** » contenant les libellés des deux nouvelles pages d'exemple.
- Le fichier « **menu.xml** » qui contient la définition des pages de l'application et qui embarque donc les deux nouvelles pages d'exemple. Ce fichier XML est utilisé pour la génération du menu, du plan du site et du fil d'Ariane.
- Les pages JSP « **page1.jsp** » et « **page2.jsp** ».

Les pages contiennent le texte latin « **Lorem ipsum** » (cf. Wikipédia).

A noter que les deux pages d'exemple sont affichées dans le menu mais seule la page 1 apparait dans le Plan du site (provient de la configuration dans le fichier « menu.xml).

2.4.6 Spécificités projet avec test de vie

L'ajout des Tests de Vie dans le projet impacte les fichiers suivants :

- Le fichier de configuration Struts « **struts.xml** » pour inclure le fichier de configuration « **struts-hornet-testvie.xml** » propres aux Test de vie.
- Le fichier de configuration Tiles « **tiles.xml** » pour ajouter les nouvelles pages de Tests de Vie.
- Le fichier de configuration de l'application web « **web.xml** » pour faire référence aux fichiers de configurations Spring pour les Tests de Vie.
- Les fichiers de configurations Spring (« **spring-appContext-testvie-encode.xml** », « **spring-appContext-testvie.xml** ») propres aux Tests de vie et les fichiers de propriétés associés (« **testvie-service.properties** », « **testvie.properties** »).
- Le fichier « **package_fr.properties** » contenant les libellés des nouvelles pages de Tests de Vie.
- Le fichier « **menu.xml** » qui contient la définition des pages de l'application et qui embarque donc les nouvelles pages de Tests de Vie. Ce fichier XML est utilisé pour la génération du menu, du plan du site et du fil d'Ariane.
- Les pages JSP « **detail.jsp** », « **infoserveur.jsp** », « **system.jsp** » et « **testvie.jsp** » ainsi que les ressources statiques (« **tableau.css** » et « **tableau.js** » pour les Tests de Vie).

- Les fichiers « **ivy-template.xml** »et « **ivy.xml** » pour inclure les librairies nécessaires aux Tests de Vie.
- Les fichiers de construction du projet et du livrable applicatif (« **build.xml** » et les fichiers de propriétés associés) qui contiennent des taches Ant supplémentaires utiles aux Tests de Vie

2.5 Contenu Web par type de projet

Ce chapitre décrit le contenu des projets initiaux après déploiement sur Tomcat.

2.5.1 Authentification

Les projets fournissent systématiquement un mécanisme d'authentification. Les logins et mots de passes pré-configurés se trouvent dans le fichier **user.properties** (user/usermdp par exemple).

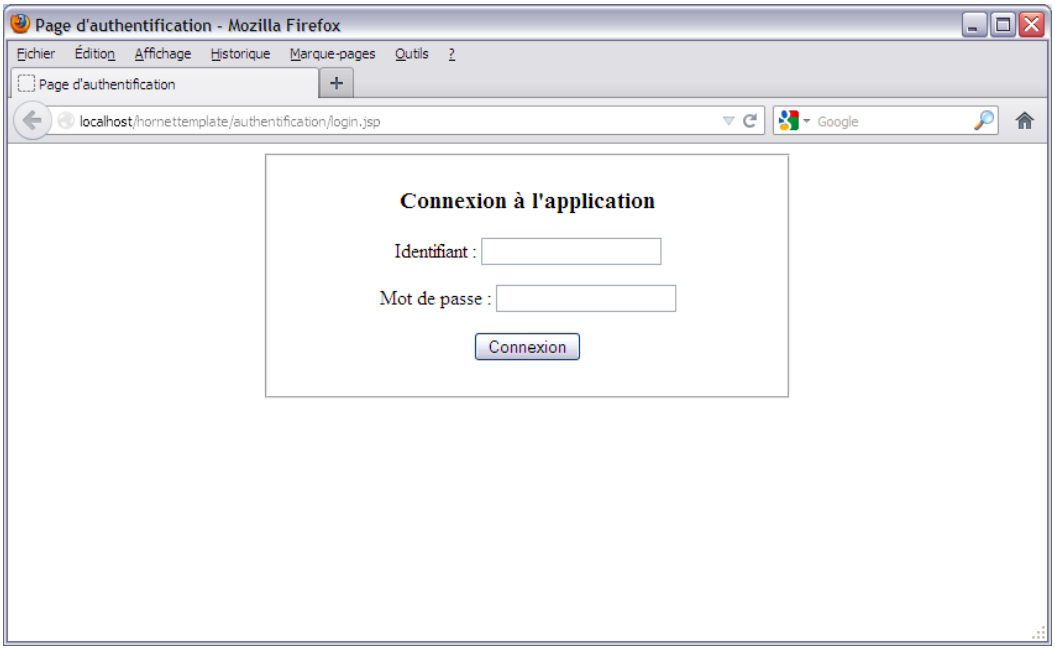


Figure 8 : Page d'authentification

La gestion de configuration de sécurité de l'application est effectuée dans le fichier « **spring-security.xml** ».

La définition des utilisateurs habilités à se connecter à l'application est effectuée dans le fichier « **users.properties** », où des comptes (actif/inactif) d'utilisateurs ou d'administrateurs peuvent être configurés.

Se reporter au « Guide du développeur Hornet 3.6B » pour les détails de mise en œuvre et de configuration de la sécurité.

2.5.2 Projet basique ou avec Clamav

Le projet contient 3 pages par défaut :

- La page « Accueil » : page d'accueil de l'application

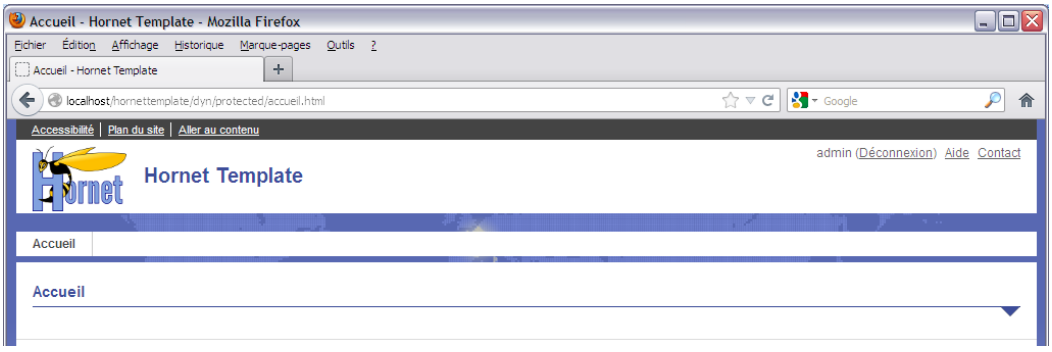


*Figure 9 : Projet basique / Page « **Accueil** »*

- La page « Plan du Site » : page montrant le plan du site de l'application

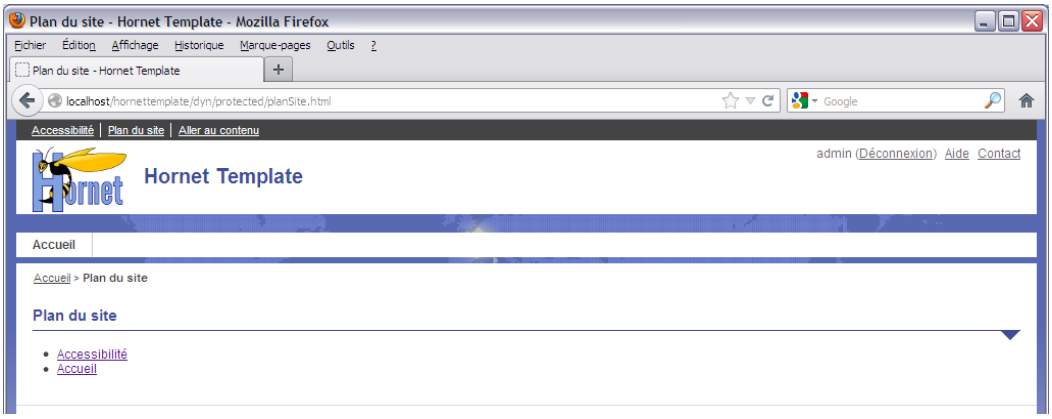


*Figure 10 : Projet basique / Page « **Plan du Site** »*

- La page « Accessibilité » : page présentation un exemple de page sur l'accessibilité à implémenter pour l'application

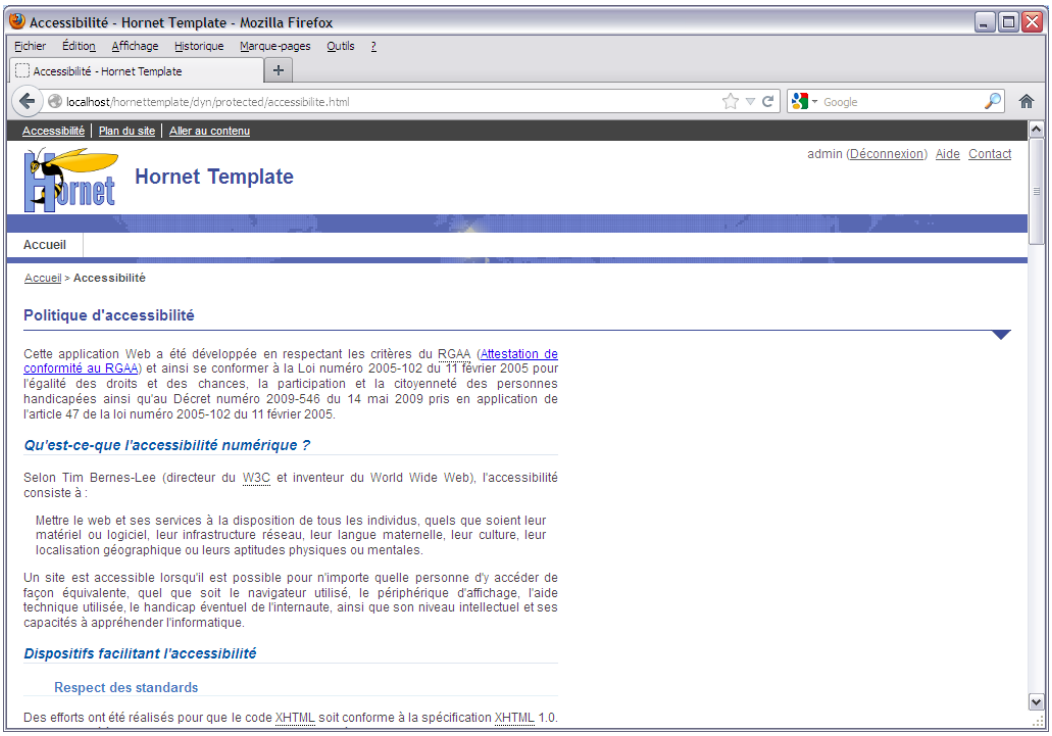


*Figure 11 : Projet basique / Page « **Accessibilité** »*

2.5.3 Projet avec exemples

Le projet contient 2 pages supplémentaires:

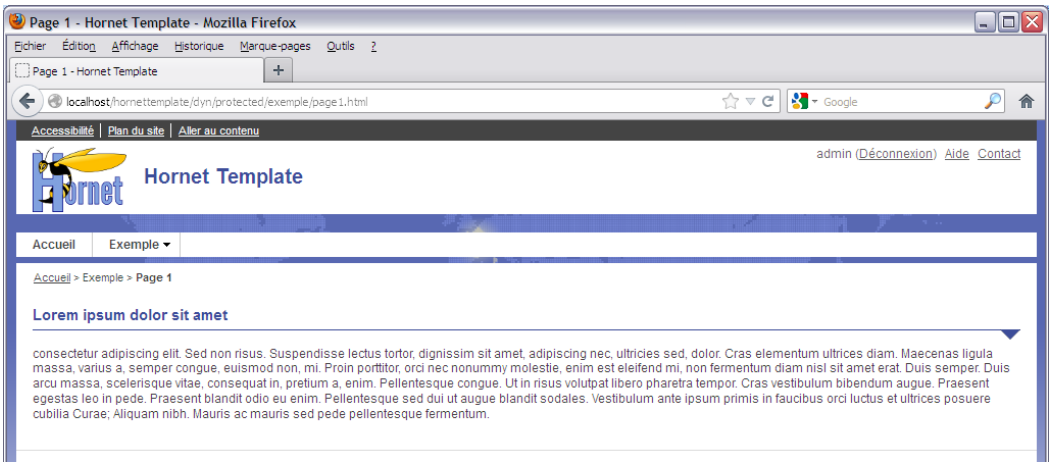


*Figure 12 : Projet avec des pages d'exemples / Page « **Exemple 1** »*

*Figure 13 : Projet avec des pages d'exemples / Page « **Exemple 2** »*

Ceci est répercuté aussi bien au niveau du menu que du plan du site :

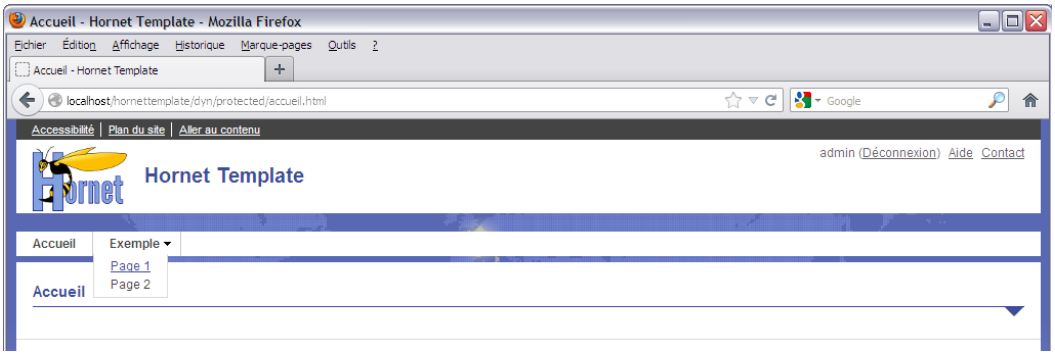


Figure 14 : Projet avec des pages d'exemples / Détail du Menu

2.5.4 Projet avec test de vie

Le déploiement du projet sous Tomcat donne le résultat suivant :

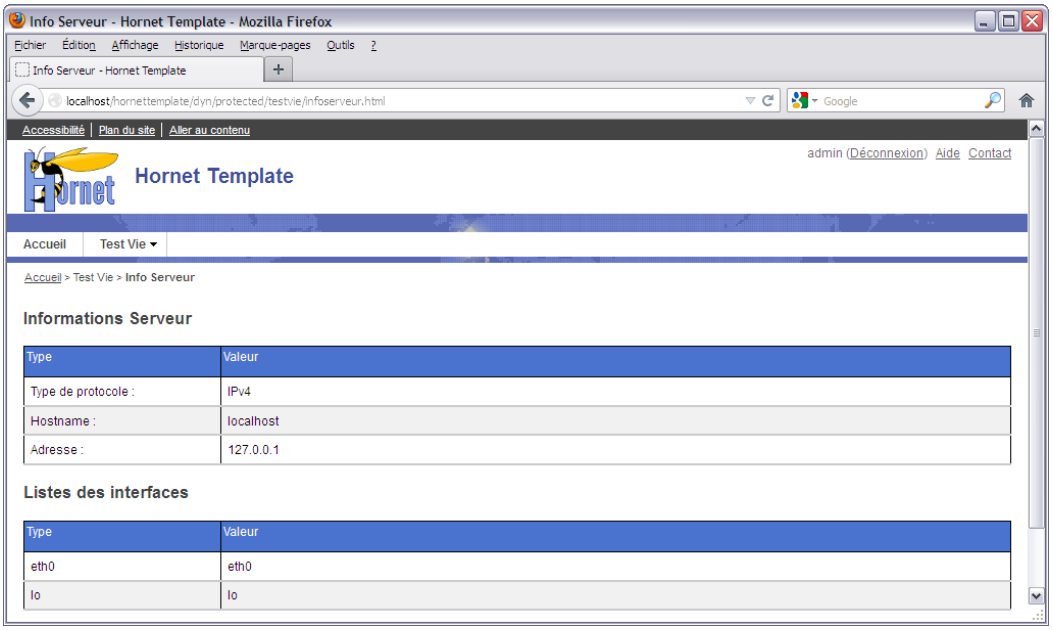


*Figure 15 : Projet avec la configuration Test de Vie / Page « **Info Serveur**» (Test de Vie)*

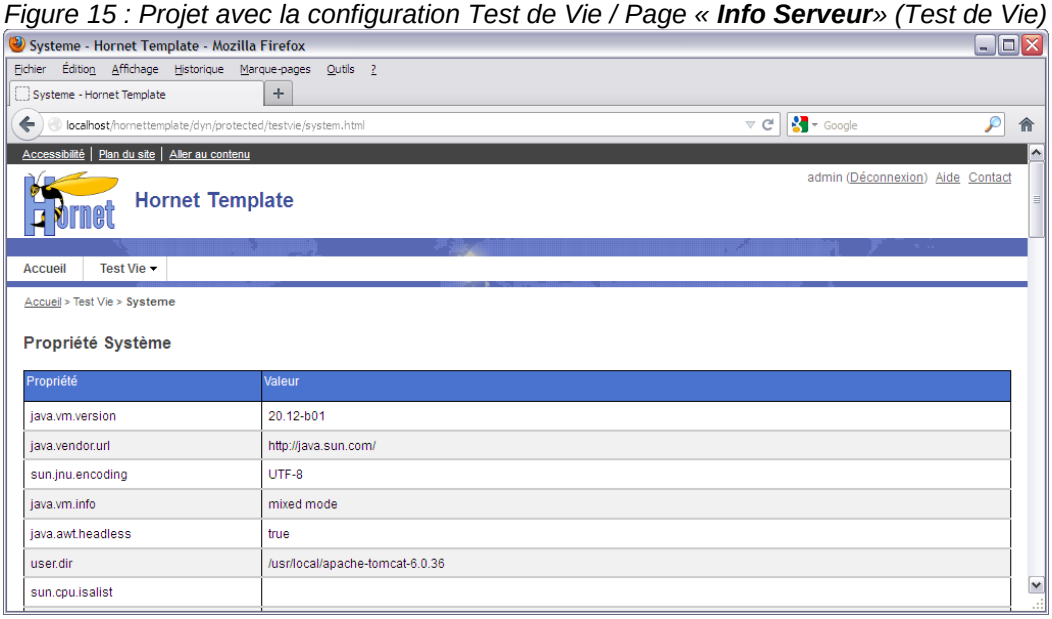


*Figure 16 : Projet avec la configuration Test de Vie / Page « **Système**» (Test de Vie)*

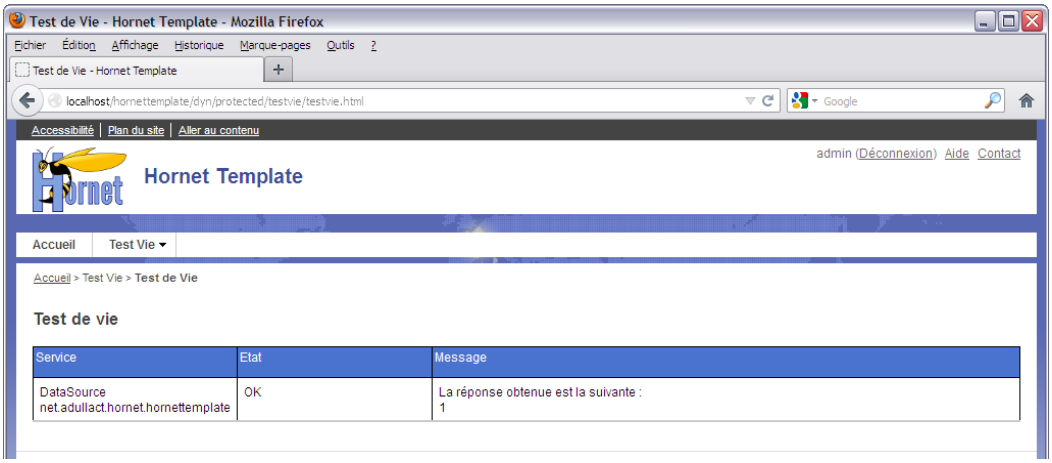


*Figure 17 : Projet avec la configuration Test de Vie / Page « **Test de Vie**» (Test de Vie)*

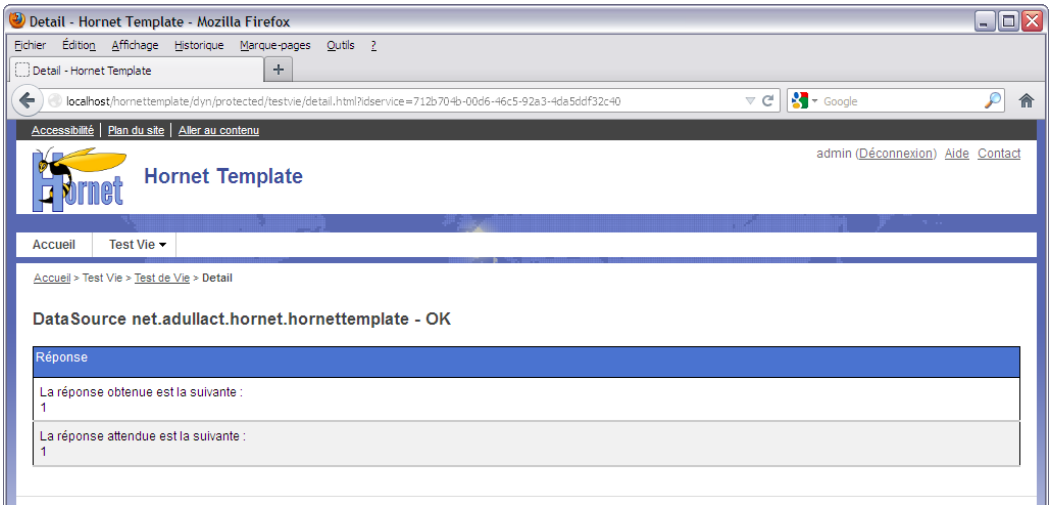


*Figure 18 : Projet avec la configuration Test de Vie / Page « **Détail**» (Test de Vie)*

2.6 Fin d'utilisation du template

Lorsqu'il n'y a plus besoin d'utiliser le template, il est nécessaire de supprimer tout ce qui est lié au template, avec la tâche « **cleanAllTemplate** » du buildTmp.xml.

La console affiche le résultat suivant :

```
Buildfile: C:\dev\worspace\hornetTEST\hornettemplateTEST2\buildTmp.xml
cleanAllTemplate:
     [echo] suppression des elements du template
   [delete] Deleting directory C:\dev\worspace\hornetTEST\hornettemplateTEST2\dev
   [delete] Deleting: C:\dev\worspace\hornetTEST\hornettemplateTEST2\buildTmp.xml
   [delete] Deleting: C:\dev\worspace\hornetTEST\hornettemplateTEST2\buildTmp.properties
BUILD SUCCESSFUL
Total time: 378 milliseconds
```

*Figure 19 : Résultat de la console après la tâche « **cleanAllTemplate** »*

Un rafraichissement du projet donne la vue suivante :

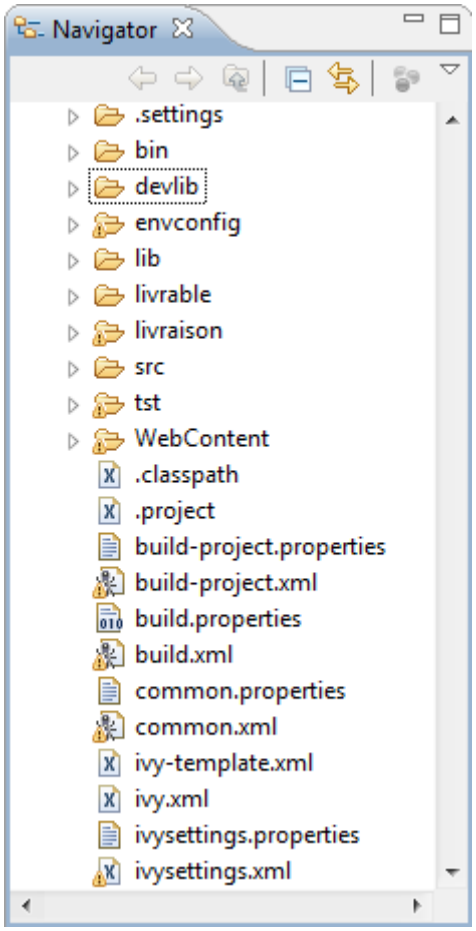


*Figure 20 : Vue du projet après exécution de la tâche « **cleanAllTemplate** »*

Les changements sont :

- Suppression du répertoire « **dev** ».
- Suppression du fichier « **buildTmp.xml** »

3 Ajout de nouveau service

3.1 Intégration des tests de vie

L'intégration des « **tests de vie** » à une application existante passe par plusieurs ajouts et configurations obligatoires :

- Récupération et intégration des artefacts :
  - Les pages JSP ;
  - Les ressources statiques ;
  - Les librairies pour les tests de vie ;
  - Les fichiers de configuration ;
  - Les fichiers de construction de l'application et du livrable applicatif.
- Configuration de l'application pour les « **tests de vie** » :
  - web.xml : déclaration des properties Spring;
  - Properties : propriétés utilisées par l'application ;
  - Struts : navigation et gestion des rôles dans les pages web ;
  - Spring : déclaration des services à tester et des rôles ;
  - Tiles : déclaration des pages web ;
  - menu.xml : arborescence des pages.

3.1.1 Ajout des artefacts

3.1.1.1 Récupération des éléments

Initialiser un nouveau projet Hornet avec la configuration « **Test de vie** ».

3.1.1.2 Copie dans l'application existante

Les éléments suivants sont nécessaires pour pouvoir utiliser les tests de vie dans une application.

- Les pages JSP et ressources statiques :

Recopier le contenu des dossiers suivants en gardant la même arborescence:

  - **WebContent/WEB-INF/static/testvie**
  - **WebContent/WEB-INF/tiles-jsp/testvie**

- Les librairies :

Recopier les dépendances nécessaires pour les tests de vie dans le fichier ivy.xml

```
<dependency org="fr.gouv.diplomatie.testvie" name="testvie"
        rev="1.0" conf="compile-&gt;default" transitive="true">
        <artifact name="testvie" type="jar" ext="jar" />
</dependency>
<dependency org="fr.gouv.diplomatie.testvie" name="testvie-encode"
        rev="1.1" conf="compile-&gt;default" transitive="true">
        <artifact name="testvie-encode" type="jar" ext="jar" />
</dependency>
```

- Les fichiers de configuration Spring par défaut :

Recopier les fichiers suivant dans « src/config » :

  - **spring-appContext-testvie.xml**
  - **spring-appContext-testvie-encode.xml**
  - **testvie.properties** (préfixé éventuellement par un nom de package)
  - **testvie-service.properties**

- Le fichier de configuration Struts spécifique aux tests de vie :

Recopier les fichiers suivant dans « src/config » :

  - **struts-hornet-testvie.xml**

- Les fichiers de construction contenant des taches supplémentaires pour les tests de vie :

Recopier les fichiers suivant :

- **build.xml**

3.1.2 Configuration de l'application

3.1.2.1 Configuration de la webapp

Ajouter au contexte de l'application les fichiers de configuration Spring à utiliser pour les tests de vie.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<web-app id="WebApp_ID" version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app 2 5.xsd">

    <!-- Indique au ContextLoaderListener où sont les fichiers de configuration Spring -->
  <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
               classpath*:spring-security.xml
               classpath*:spring-appContext.xml
               classpath*:spring-appContext-dao.xml
               classpath*:spring-appContext-testvie.xml
        </param-value>
  </context-param>
```

3.1.2.2 Inclusion des pages web

L'application doit inclure dans ses fichiers de configuration un certain nombre d'éléments afin qu'elle puisse accéder aux pages de « **test de vie** ».

- Modifier le fichier de configuration Struts :

Inclure le fichier de configuration Struts spécifique aux tests de vie.

```
<struts>

    <include file="struts-hornet-testvie.xml" />
…
</struts>
```

- Modifier le fichier de configuration Tiles :

Ajouter la définition de page suivante héritant du modèle de base.

```
    <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
    <!-- Test de vie -->
    <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
    <definition name="WILDCARD:testvie.*.tile" extends="baseLayout">
        <put-attribute name="filArianeKey" value="menu.testvie.{1}" />
        <put-attribute name="content" value="/WEB-INF/tiles-jsp/testvie/{1}.jsp" />
        <put-list-attribute name="appCssItems">
            <add-attribute value="/static/css/global.css" />
            <add-attribute value="/static/testvie/css/tableau.css" />
        </put-list-attribute>
    </definition>
```

- Modifier le fichier contenant les libellés des pages de l'application:

Ajouter les libellés des pages dans le fichier package_fr.properties.

```
# Test Vie
menu.testvie.libelle=Test Vie
menu.testvie.libelleLong=Test Vie
menu.testvie.infoserveur.libelle=Info Serveur
menu.testvie.infoserveur.libelleLong=Test Vie - Info Serveur
menu.testvie.system.libelle=Systeme
menu.testvie.system.libelleLong=Test de Vie - Systeme
menu.testvie.testvie.libelle=Test de Vie
menu.testvie.testvie.libelleLong=Test de Vie - Test de Vie
menu.testvie.detail.libelle=Detail
menu.testvie.detail.libelleLong=Test de Vie - Detail
```

- Modifier le fichier de définition du menu :

Ajouter dans le fichier menu.xml l'arborescence pour les pages de test de vie.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<pagelet>
```

```
    <root id="root" href="/dyn/protected/accueil.html"></root>
    <menu>
        <menu-item
            id="menu.testvie"
            rolesAutorises="ROLE_AppliHornet_USER">
            <menu-item
               id="menu.testvie.infoserveur"
               rolesAutorises="ROLE_AppliHornet_USER"
               href="/dyn/protected/testvie/infoserveur.html">
            </menu-item>
            <menu-item
               id="menu.testvie.system"
               rolesAutorises="ROLE_AppliHornet_USER"
               href="/dyn/protected/testvie/system.html">
            </menu-item>
            <menu-item
               id="menu.testvie.testvie"
               rolesAutorises="ROLE_AppliHornet_USER"
               href="/dyn/protected/testvie/testvie.html">
               <menu-item
                 id="menu.testvie.detail"
                    rolesAutorises="ROLE_AppliHornet_USER"
                 href="/dyn/protected/testvie/detail.html"
                 visibleDansMenu="false" visibleDansPlan="false">
              </menu-item>
            </menu-item>
        </menu-item>
    </menu>
</pagelet>
```

Remarque : les attributs ***visibleDansMenu*** et ***visibleDansPlan*** sont optionnels et leur valeur par défaut est « **true** ». Pour ne pas afficher les pages dans le menu (ou le plan du site), valoriser ces attributs à « **false** ». L'attribut ***rolesAutorises*** liste les rôles de sécurité nécessaires à l'affichage du menu (et plan du site).

4 Problèmes connus

4.1 Les modifications des pages ne sont pas prises en compte

Si après exécution de tâches les modifications des pages ne sont pas prises en compte, cela peut venir d'un problème de rafraîchissement des fichiers sous Eclipse.

Relancer le serveur Tomcat après avoir rafraîchi le projet (touche « F5 » sur le projet ou action « **Refresh** ») et effectué un « **Clean** » puis « **Publish** » sur le serveur.

Vider le cache navigateur et supprimer les cookies de session pour forcer le téléchargement du thème courant.