

# UNIRE – Développement

## Objectifs :

- Utiliser un environnement de développement efficace.
- Découvrir comment programmer un canal pour uPortal.
- Réaliser un exemple complet de canal en utilisant les outils développés par les développeurs ESUP.

## **PARTIE A : INSTALLATION**

### ***I. Installation du JDK :***

Installer le Java Development Kit dans le répertoire D:\JDK.

Le choix du répertoire d'installation est proposé au moment où il vous est demandé de sélectionner les différents composants à installer.

Créer une variable d'environnement utilisateur `JAVA_HOME` pointant vers le répertoire d'installation du JDK.

### ***II. Installation de la distribution esupdev :***

Décompresser l'archive `devel-nancy-esupdev.zip` à la racine du lecteur D. Un dossier `esupdev-2.3` est automatiquement créé.

### ***III. Installation et configuration de Eclipse :***

Décompresser l'archive `eclipse-SDK-3.0.1-win32.zip` à la racine du lecteur D. Un dossier `eclipse` est automatiquement créé.

Décompresser l'archive `LogWatcher v1.3.2.zip` dans le dossier :

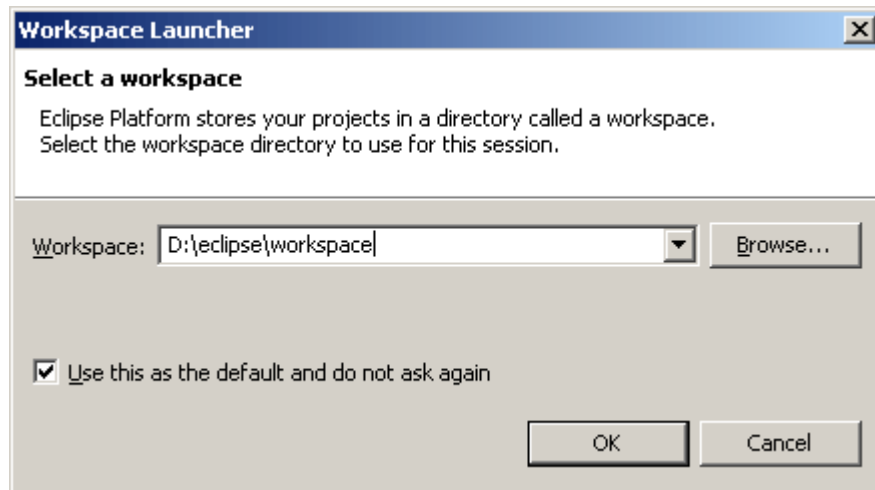
D:\eclipse\plugins.

Décompresser l'archive `XML Buddy v2.0.10.zip` dans le dossier :

D:\eclipse\plugins.

Eclipse est maintenant installé est prêt à fonctionner.

Lancer `eclipse.exe`, une fenêtre doit apparaître et vous demander les informations suivantes :



Conservez le répertoire par défaut, cochez la case en bas à gauche et validez. Eclipse s'ouvre et vous présente la page de bienvenue. Ouvrez le plan de travail en cliquant sur la flèche 'Go to the workbench' située en haut à droite.

Vous êtes désormais dans la perspective 'Ressource' qui est la perspective par défaut au démarrage de Eclipse.

Vous allez maintenant configurer quelques options, accessibles par le menu 'Window > Preferences'.

La première page concerne le plan de travail (Workbench). Configurez les options pour que les tâches s'exécutent toujours en arrière plan, que la compilation ne soit pas automatique et pour que les fichiers soient automatiquement sauvegardés au lancement de la compilation.

Dans la section 'Workbench > Editors > Text Editor', cochez l'option permettant d'afficher les numéros de ligne.

Dans la section 'Java > Editor', cochez l'option permettant d'afficher les numéros de ligne. Dans l'onglet 'Folding', cochez l'option permettant de réduire automatiquement les méthodes.

Dans la section 'Logwatcher', cochez l'option permettant de restaurer les watchers au démarrage d'Eclipse.

Dans la section 'XML Buddy > XML > Encoding' spécifiez un encodage par défaut en ISO-8859-1.

Validez l'ensemble de vos paramètres.

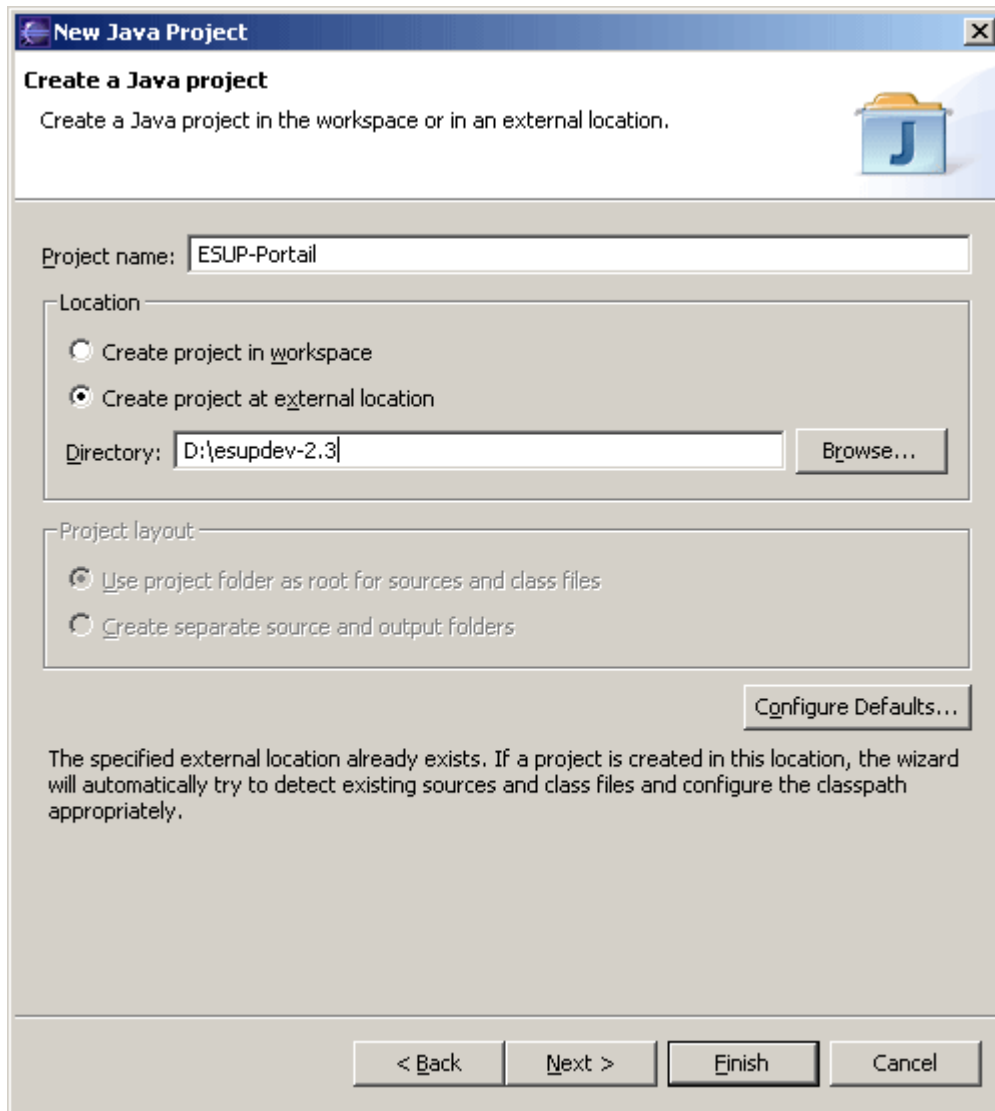
A l'aide du menu 'Window > Open Perspective', ouvrez la perspective Java. Supprimez les vues inutiles : Hierarchy, Javadoc, Declaration. Descendez la vues Package Explorer dans la section du bas.

A l'aide du menu 'Window > Show View' ajoutez les vues Ant et LogWatcher dans la section du bas.

## IV. Un premier projet : esupdev

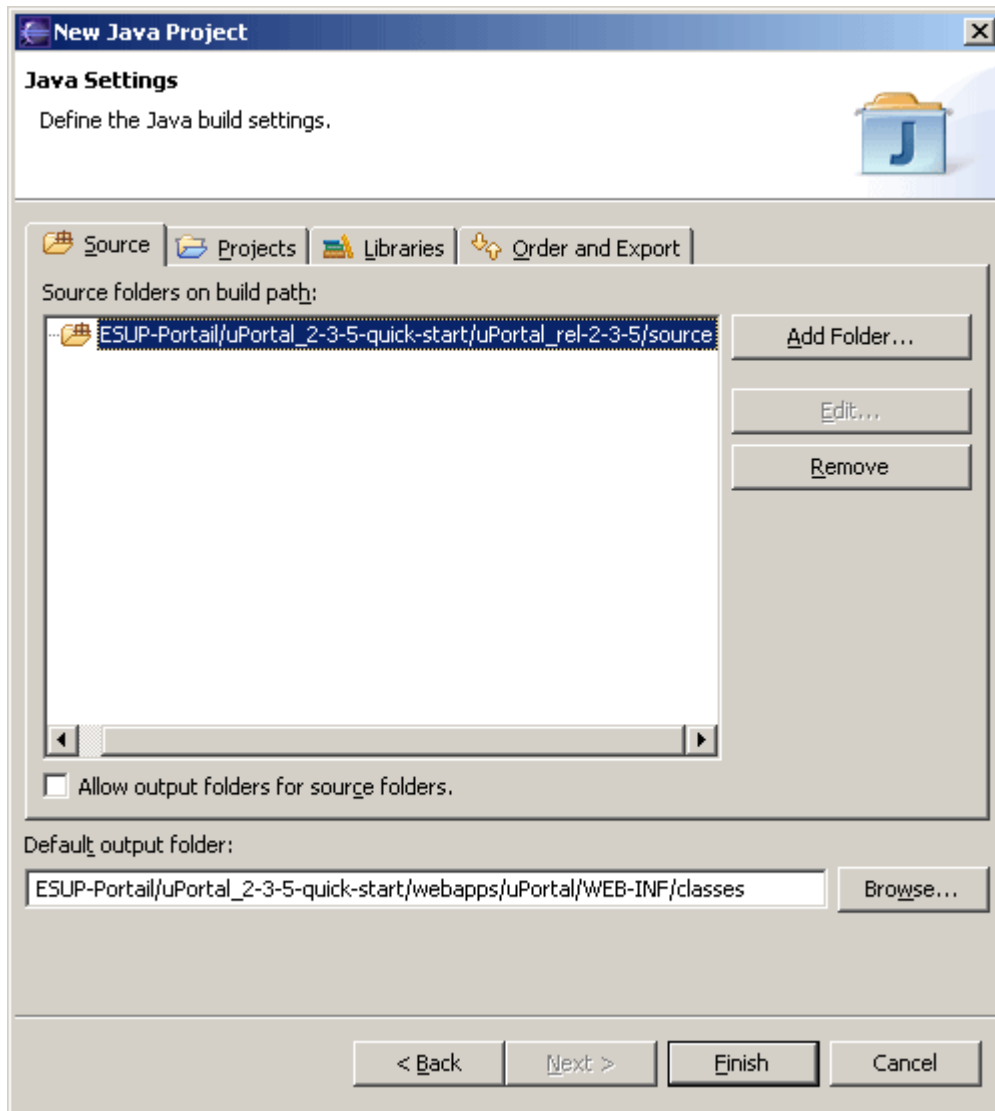
Nous allons maintenant créer directement dans Eclipse un projet Java correspondant au package esupdev.

Dans le Package Explorer, à l'aide du bouton droit de la souris, créez un nouveau projet. Choisissez un projet Java et paramétrez comme suis :



Lors du clic sur 'Next', Eclipse va tenter de déterminer automatiquement les répertoires contenant des sources Java et ceux où seront placés les fichiers compilés.

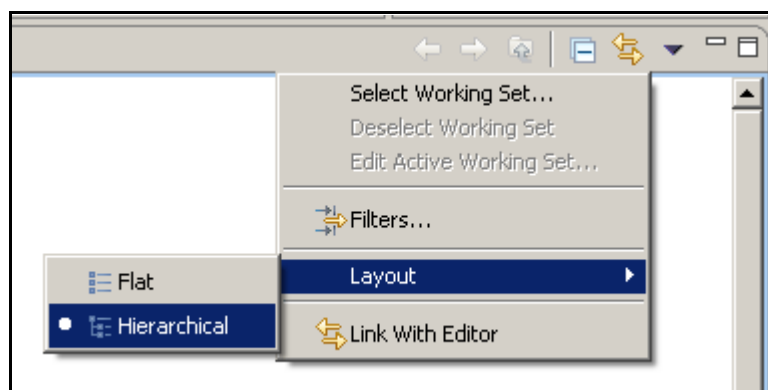
Supprimez tous les répertoires source inutiles de façon à obtenir le résultat suivant :



Dans l'onglet Libraries, supprimez tous les fichiers JAR (conservez la JRE System Library).

Après avoir terminé la création du projet, celui-ci apparaît dans le package explorer. On remarque un grand nombre de packages et l'affichage est assez flou.

Le passage en affichage arborescent se fait en réalisant la manipulation suivante :



## **V. Configuration et déploiement du package ESUP :**

Naviguez dans le projet et ouvrez le fichier `perso.properties` situé à la racine du package. Très peu de propriétés sont configurables étant donné que l'objectif de la séance n'est pas d'installer un portail. Indiquez dans le fichier les bonnes informations concernant le JDK et le répertoire où se trouve le package et remplacez les champs `XX` par le numéro du poste.

Allez ensuite dans la vue Ant et ajoutez le fichier `build.xml` situé à la racine du package. Lancez ensuite la tâche `esup.init`.

### Remarque :

*Comme toutes les bibliothèques ont été supprimées du projet, un nombre important d'erreurs de compilation vont apparaître dans la vue Problems, toutefois cela n'affectera pas le déroulement de la tâche Ant qui se construit son propre classpath.*

Lancez la tâche `esup.db.init`.  
Lancez la tâche `uportal.deploy`.

Maintenant que toutes les bibliothèques ont été copiées aux bons endroits et que le répertoire `webapps` a été créé, il est nécessaire de revenir à la configuration du projet pour finaliser le paramétrage.

Dans un premier temps il faut indiquer à Eclipse que des fichiers ont été modifiés en choisissant l'option rafraîchir du menu contextuel du projet (en cliquant du bouton droit sur la racine).

Accédez ensuite aux propriétés du projet de la même façon.

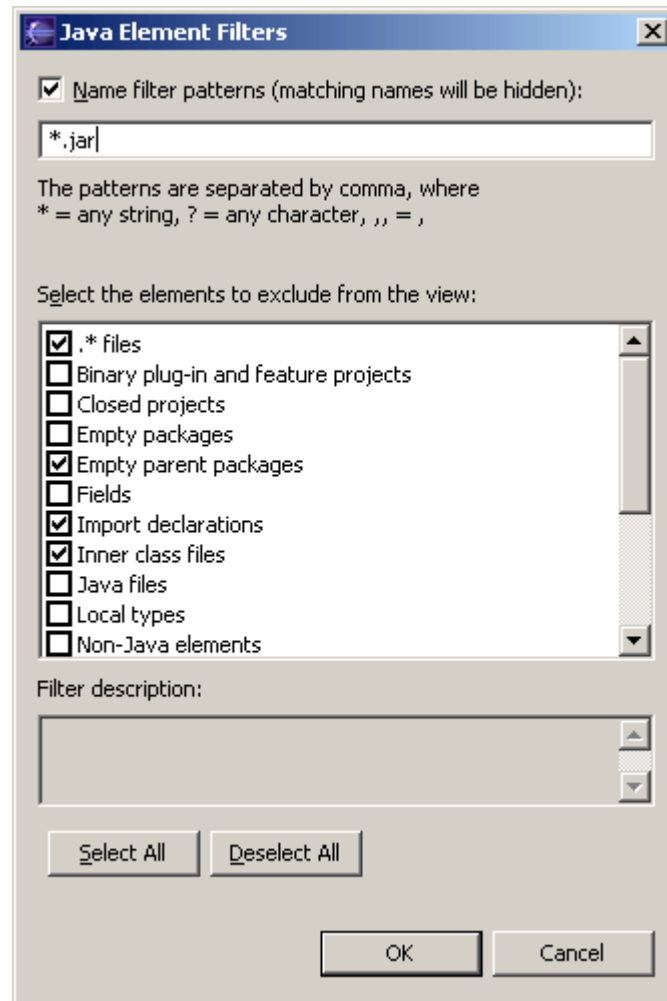
Une fois placés dans la section Java Build Path, on retrouve les options proposées à la création du projet.

Paramétrez le répertoire de destination des fichiers compilés dans :  
`uPortal_2-3-5-quick-start\uPortal_rel-2-3-5\build\WEB-INF\classes`.

Dans l'onglet Libraries, choisissez Add JARs et sélectionnez tous les JAR situés dans :  
`uPortal_2-3-5-quick-start\uPortal_rel-2-3-5\lib`.

Validez vos modifications et autorisez Eclipse à supprimer les fichiers compilés qu'il avait créés dans l'ancien répertoire.

Les bibliothèques ajoutées apparaissent désormais dans le projet et nuisent à la qualité de l'affichage. Dans le menu d'options de la vue Package Explorer, créez un filtre et configurez le de la façon suivante :



Des erreurs apparaissent encore dans le projet, vu qu'il n'a pas été reconstruit. Commencez par éliminer les erreurs en sélectionnant la racine du projet et en choisissant l'option Clean du menu Project. Choisissez de nettoyer uniquement le projet courant et de ne pas le reconstruire automatiquement. Les erreurs disparaissent. Compilez maintenant votre projet :

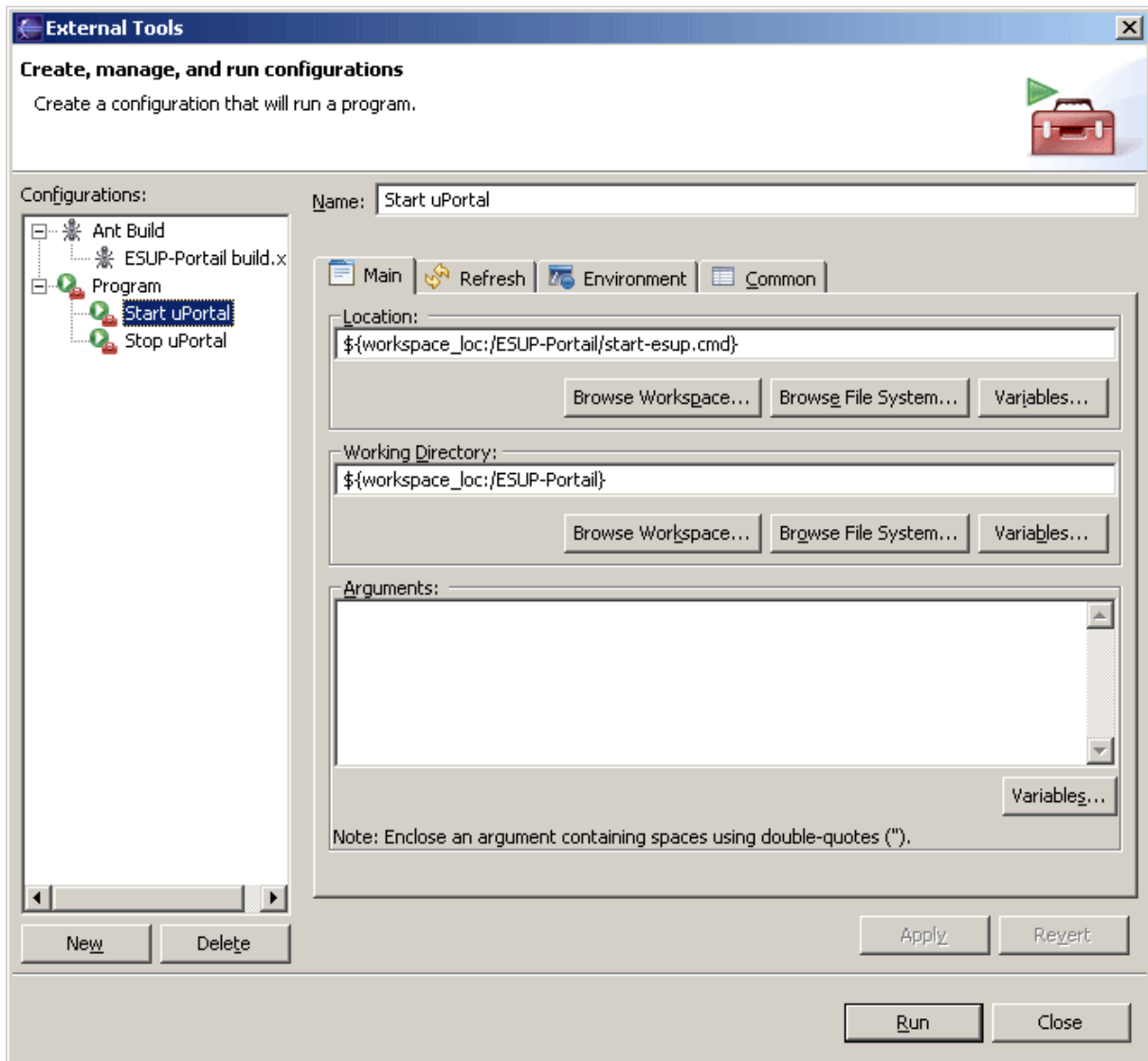
- Build Project dans le menu contextuel du projet
- Project > Build Project dans le menu du haut (une fois le projet sélectionné)

Si tout se passe correctement, le projet ne doit plus contenir aucune erreur (les icônes rouges ont disparus). Une icône jaune subsiste sur certains éléments pour indiquer des avertissements de compilation. La vue Problems résume l'ensemble des avertissements et des erreurs liés à la compilation d'un projet. De la même façon que pour la vue Package Explorer, il est possible de filtrer l'affichage afin de ne garder que les informations concernant des erreurs.

Il faut désormais tester le bon fonctionnement de notre portail. Pour cela, nous allons créer des raccourcis dans Eclipse pour lancer et arrêter directement le portail depuis l'environnement de développement.

Dans la barre d'outils, choisissez External Tools.

Configurez deux programmes externes de la façon suivante :



Sauvegardez vos modifications et quitter en cliquant sur Close (surtout pas Run pour le moment).

Retournez dans le sous-menu External Tools (en cliquant sur la petite flèche juste à côté). Choisissez 'Organize Favorites' et ajoutez vos deux programmes. Ils sont désormais disponibles dans le sous-menu External Tools. Lancez votre portail pour le tester et connectez vous à l'URL suivante afin de vérifier son fonctionnement :

<http://unire.multimedia.uhp-nancy.fr/portalXX>

Vous pouvez utiliser l'authentification CAS avec au choix les comptes ensXX ou etuXX.

## PARTIE B : QUELQUES CANAUX

### *I. Premier canal CHelloWorld :*

Décompressez l'archive CHelloWorld.zip où vous le souhaitez sur la partition D. Créez un nouveau projet Java dans Eclipse pointant vers ce dossier. Dans un premier temps ne vous souciez pas des répertoires sources et compilation. Une fois le projet créé, éditez et adaptez le fichier build.properties. Ajoutez ensuite à la vue Ant le fichier build.xml du projet.

Lancez la tâche prepare, rafraîchissez l'affichage du canal et modifiez les propriétés du projet de façon à bien spécifier le répertoire source et à utiliser le répertoire build/WEB-INF/classes pour les fichiers compilés. Dans l'onglet Projects, créez une dépendance avec le projet ESUP. Enfin dans l'onglet Libraries ajoutez des JARs que vous irez prendre dans les librairies du projet ESUP :

- commons-logging.jar
- log4j-1.2.8.jar

Recompilez le projet pour vous assurer qu'il n'y a pas d'erreur. Déployer le à l'aide de la tâche Ant all. Dans la vue LogWatcher créez un watcher sur le fichier portal.log. Redémarrez votre portail et connectez vous en tant qu'administrateur pour publier votre canal HelloWorld. Mettez le canal à la disposition de tout le monde, dans la rubrique divers et n'oubliez pas de préciser que le canal possède un mode 'A propos de'. Reconnectez-vous en tant que ensXX et allouez vous le canal dans un nouvel onglet. Utilisez-le et examinez dans le LogWatcher les différentes lignes qui apparaissent dans le fichier de log du portail.

### *II. Passage du canal en mode debug*

Dans Eclipse copiez le fichier :

```
uPortal-2-3-5-quick-start/webapps/uPortal/WEB-INF/classes/properties/Logguer.properties
```

dans :

```
Perso/uPortal/properties
```

Editez la version du répertoire Perso et ajoutez à la fin les lignes suivantes :

```
# Fichier de log canaux ESUP
#
log4j.category.org.esupportail.portal.channels = DEBUG, esup
log4j.additivity.org.esupportail.portal.channels = false

log4j.appender.esup = org.apache.log4j.RollingFileAppender
log4j.appender.esup.File = D:/esupdev-2.3/esup.log
log4j.appender.esup.Encoding=ISO-8859-1
log4j.appender.esup.MaxFileSize=3000KB
log4j.appender.esup.MaxBackupIndex=10
```



```
log4j.appender.esup.layout=org.apache.log4j.PatternLayout
log4j.appender.esup.layout.ConversionPattern=%5p [%t]
%c{2}.[%x] %d{MMM/dd HH:mm:ss} - %m%n
```

Ces nouvelles instructions vont permettre de loguer séparément toutes les classes Java situées dans le package `org.esupportail.portal.channels` (donc tous les canaux ESUP) avec un niveau de log différent du niveau du portail.

Arrêtez votre portail et exécutez les tâches `esup.init` et `uportal.deploy` afin de prendre en compte vos modifications. Redémarrez votre portail, utilisez le canal `HelloWorld` et créez ensuite un nouveau watcher qui surveillera le fichier `esup.log`.

### **III. Utilisation du cache**

Décompressez l'archive `CCacheable.zip` et créez un projet sous Eclipse pointant sur ce répertoire. Une fois le projet créé, éditez et adaptez le fichier `build.properties`. Ajoutez ensuite à la vue Ant le fichier `build.xml` du projet.

Déployez le canal et publiez le pour tout le monde, dans la rubrique 'divers'.

Enfin, souscrivez le canal et vérifiez qu'il affiche bien le `TimeStamp` et que celui-ci est bien rechargé à expiration du cache (au bout de 60 secondes). Dans le code `CGetLogin.java`, la méthode `setKey()` permet de générer la clé. La clé de cache est générée à partir d'une chaîne de caractères contenant l'ID de la personne connectée.

La méthode `setKeyValidity(new Long(System.currentTimeMillis()))` permet de positionner la date de début. C'est à partir de cette date que l'on vérifiera si le cache est toujours valide ou non.

Modifiez le code pour qu'il affiche, dans les logs dans un premier temps puis dans le canal, un compteur sur le nombre de fois où le cache est rechargé (C'est-à-dire, incrémentation de 1 lorsqu'il n'est plus valide). Le cache aura une durée de validité de 30 secondes.

Une correction est disponible dans `CorrectionCCacheable.zip`.

### **IV. Récupération et utilisation de l'objet IPerson**

Décompressez l'archive `CGetLogin.zip` et créez un projet sous Eclipse pointant sur ce répertoire. Une fois le projet créé, éditez et adaptez le fichier `build.properties`. Ajoutez ensuite à la vue Ant le fichier `build.xml` du projet.

Déployez le canal et publiez le pour tout le monde, dans la rubrique 'divers'.

Enfin, souscrivez le canal et vérifiez qu'il affiche bien votre `Login`. Dans le code `GetLogin.java` c'est `staticData.getPerson()` qui permet de récupérer l'objet `IPerson` instancié à la connexion de l'utilisateur et stocké dans les `StaticData`.

Modifiez le code pour qu'il affiche, dans les logs dans un premier temps puis dans le canal, la liste des attributs - valeurs d'une personne. Pour cela vous utiliserez les méthodes `getAttributes()` et `getAttributeNames()` de `IPerson` décrites dans l'API : [http://www.unicon.net/api/uPortal\\_ref2-1-3/](http://www.unicon.net/api/uPortal_ref2-1-3/).

Une correction est disponible dans `CorrectionCGetLogin.zip`.

## ***V. Le Digester***

Décompressez l'archive `esup-canal-digester.zip` et créez un projet sous Eclipse pointant sur ce répertoire. Ajoutez les bibliothèques `commons-beanutils.jar`, `commons-digester.jar`, `commons-logging.jar` situées dans le répertoire `lib` de `uPortal`. Adaptez le fichier `build.properties` puis ajoutez à la vue Ant le fichier `build.xml`. Compilez et déployez puis publiez le canal afin de le tester.

Dans cet exemple Digester est utilisé pour lire un fichier XML contenant un arbre récursif et le transformer en un graphe d'objets respectant la même hiérarchie. Digester peut également appliquer d'autres règles que la création dynamique d'objets comme l'appel dynamique de méthodes ou le positionnement de variables.

## ***VI. Le ticket CAS***

Décompressez l'archive `esup-canal-ticket.zip` et créez un projet sous Eclipse pointant sur ce répertoire. Une fois le projet créé, éditez et adaptez le fichier `build.properties`. Ajoutez ensuite à la vue Ant le fichier `build.xml` du projet.

Déployez le canal et publiez le pour tout le monde, dans la rubrique 'divers'.

Enfin, souscrivez le canal et vérifiez qu'il récupère bien le `proxyTicket`. Le principe de ce canal est de montrer comment récupérer un PT. Une fois ce PT obtenu, il est possible de l'utiliser pour accéder à un service authentifié par CAS sans que l'utilisateur s'en rende compte.

## **PARTIE C : INTEGRATION D'UNE APPLICATION EXISTANTE**

Nous allons intégrer une application PHP existante dans le portail grâce à un Canal de type WebProxy.

### ***I. Installation d'EasyPHP et de l'application***

Lancer easyphp1-7\_setup.exe et installer EasyPHP sur D:\.

Décompresser le fichier Appli\_Existante\_PHP.zip dans D:\EasyPHP1-7\www.

Lancer EasyPHP puis consulter le Web local (click droit sur l'icône en bas à droite). Vous pouvez alors directement utiliser l'application PHP que nous allons intégrer à Esup.

### ***II. Le Canal Web Proxy***

Sur le portail cliquer sur  puis sur « Publish a new channel »

Choisir un canal de type WebProxy et saisir les noms, titres etc. (Attention le Channel Timeout ne doit pas être vide ou à 0).

Saisir l'URL : `http://localhost:80/Appli_Existante_PHP/` (Laisser le reste des paramètres par défaut).

Mettre le canal à la disposition de tout le monde, dans la rubrique divers.

Souscrire le canal et tester l'intégration.

### ***III. Personnalisation de l'intégration***

Modifier les options de publications afin que toutes les pages s'ouvrent dans le portail.

Vérifier que cela fonctionne bien.

Afficher dans la page index.php le login(uid), le nom(cn) et l'e-mail(mail) de la personne connectée.

Vérifier que cela fonctionne bien.

#### **Remarque :**

*Par défaut, à l'installation nous avons positionné :*

*org.jasig.portal.channels.webproxy.CWebProxy.person\_allow=\* vous n'avez donc pas besoin de le faire.*

Enfin, vérifier dans la page bonjour.php que l'uid saisi est bien le même que l'uid récupéré du portail. Dans le cas contraire l'utilisateur est un menteur.

#### **Remarque :**

*Pour les personnes qui ne connaissent pas php, une correction est disponible dans CorrectionAppli\_ExistantePHP.zip*

## **PARTIE D : DEVELOPPEMENT COMPLET**

### ***I. Installation du MAG :***

Décompressez l'archive `esup-utils-mag.zip` et créez un projet Eclipse.

### ***II. Objectifs :***

Réaliser une application d'un degré de complexité supérieur à celles vues précédemment à l'aide des outils MAG :

- Etape N°1 :
  - Reproduire le canal HelloWorld en affichant Hello suivi du nom de la personne connectée au portail.
- Etape N°2 :
  - Créer une action affichant un formulaire de saisie des informations.
  - Créer une action de traitement du formulaire qui ajoute la personne saisie à la base de données.
- Etape N°3 :
  - Créer une action de visualisation de l'ensemble des personnes stockées dans la base.
- Etape N°4 :
  - Créer une action permettant d'envoyer à l'application un fichier XML contenant une liste de personnes.
  - Créer une action permettant de traiter le fichier XML afin d'ajouter à la base les personnes décrites à l'intérieur.
  - Afficher un compte rendu du traitement du fichier XML à l'aide du plugin Message.
- Etape N°5 :
  - Créer une action permettant de télécharger la liste des personnes enregistrées au format CSV.

### ***III. Réalisation :***

#### **Etape N°1 :**

Commencez par créer un nouveau projet Java dans Eclipse que vous appellerez `esup-canal-agenda`. Dans un premier temps, ne vous souciez pas des répertoires sources et compilation. Récupérez les fichiers `build.properties` et `build.xml` d'un autre canal et copiez les à la racine du nouveau projet. Adaptez le `build.properties` et modifiez le nom du canal dans le `build.xml`. Ajoutez-le à la vue ANT et exécutez la tâche `prepare`. Rafraîchissez l'affichage du projet et ouvrez ses propriétés. Définissez les bons répertoires pour les sources Java et pour les fichiers compilés. Créez une dépendance avec les projets ESUP et MAG. Enfin ajoutez les librairies suivantes : `common-beanutils`, `common-digester`, `common-logging`, `cos`, `activation`, `log4j`.

Copiez la DTD du MAG dans le répertoire properties et créez un fichier de configuration CAgenda.xml au même endroit s'appuyant sur cette DTD. Dans un premier temps, créez une action 'default' pointant sur la classe 'org.esupportail.portal.channels.CAgenda.actions.Default' et utilisant une feuille de style XSL 'Default.xml' (que l'on créera plus tard).

Nous allons maintenant créer le fichier principal du canal, la MainChannel. Dans le package des sources Java, 'org.esupportail.portal.channels.CAgenda', créez une classe CAgenda.java héritant de org.esupportail.portal.utils.channels.MainChannel. Pour le moment, cette classe ne fait rien du tout...

Créez un package 'config' dans le package CAgenda dans lequel vous allez créer une classe Config.java héritant de la classe ConfigChannel du MAG. Implémentez un mécanisme de singleton dans cette classe et implémentez la méthode getConfigFile(). Celle-ci doit indiquer au MAG où est le fichier de configuration soit :

```
/properties/channels/org_esup/CAgenda/CAgenda.xml
```

Créez un package 'actions' dans le package CAgenda dans lequel vous allez créer une classe Default.java héritant de la classe SubChannel du MAG. Laissez Eclipse s'occuper du constructeur. Surchargez les méthodes init() et setXML(). La méthode init n'effectue aucun traitement particulier et se contente de stocker les runtimeData dans une variable de classe et de renvoyer la bonne valeur booléenne pour que le MAG poursuive son cycle de vie. La méthode setXML positionne le XML généré par l'action avec le nom de la personne connectée et retourne également une valeur booléenne pour indiquer son succès.

Faites en sorte que le nom de l'utilisateur soit indiqué de cette façon dans le XML :

```
"<user name="nomdepersonne" />"
```

Enfin, faites le fichier XSL correspondant (qui s'appellera Default.xml). Il devra afficher le nom de la personne accessible dans le XML grâce au chemin XPATH : ./user/@name. Vous pouvez vous du fichier XSL du canal HelloWorld. La feuille de style doit se situer dans le répertoire webpages\stylesheets puis dans une arborescence correspondante à la hiérarchie des packages. Ici :

```
webpages\stylesheets\org\esupportail\portal\channels\CAgenda\actions.
```

Il ne vous reste plus qu'à vérifier l'action par défaut dans le fichier de configuration CAgenda.xml. Elle présente la «page d'accueil» du canal c'est pourquoi cette action doit forcément s'appeler « default ».

Déployez le canal et publiez le pour tout le monde, dans la rubrique 'divers'. Souscrivez le et vérifiez que tout fonctionne. En cas de problème consultez le fichier portal.log il présente le XML généré pas la classe et interprété par la XSL.

## Etape N°2

Nous allons maintenant modifier le canal afin qu'il permette de saisir et enregistrer les contacts d'un agenda.

Pour cela nous allons modifier l'action par défaut afin qu'elle affiche un formulaire de saisie d'un nouveau contact et créer une nouvelle action qui traitera le formulaire et enregistrera le nouveau contact dans la base.

Modifiez le fichier Default.xsl afin qu'il affiche un formulaire de saisie des coordonnées : Le nom, le prénom, le téléphone et l'email sous la forme `<input type="text" name="nom" />`. Dans son fonctionnement MAG utilise le paramètre `baseActionURL` pour rediriger les requêtes. Ce paramètre doit donc être déclaré en début de fichier : `<xsl:param name="baseActionURL" />`.

Ensuite le formulaire doit pointer vers cette URL :

```
<form method="post" action="{ $baseActionURL }">
```

et contenir un champs caché qui indiquera le nom de l'action appelée lors du "submit" du formulaire :

```
<input type="hidden" name="action" value="addperson" />. (ici nous appelleront l'action qui traitera le formulaire et enregistrera le contact "addperson").
```

Nous allons maintenant créer l'action `addPerson`.

Elle va d'abord utiliser un objet métier "Person" qui correspondra à un contact (avec son nom, prénom, téléphone et e-mail). Pour cela créez un package `org.esupportail.portal.channels.CAgenda.beans`, copiez le fichier `Person.java` dedans et observez le rôle de cette classe.

Elle va d'abord utiliser un objet "BDDConnection" qui se chargera d'enregistrer un contact dans la base. Elle utilisera un objet `Person` pour récupérer les diverses coordonnées Pour cela créez un package

`org.esupportail.portal.channels.CAgenda.data` copiez le fichier `BDDConnection.java` dedans et observez les méthodes disponibles pour cette classe. On constate la ligne de code suivante :

```
Query query = Config.getInstance().getConnexionDefault();
```

Elle indique que les paramètres de connexions à la base de données sont indiqués dans le fichier de configuration du canal.

Ajoutez donc la connexion vers la base (utilisez la DTD pour le formalisme).

On peut choisir le pool Tomcat `PortalDb` (si les tables sont dans la même base que celles du portail) ou créer un autre pool. On l'indiquera comme cela dans le fichier de configuration :

```
<pool name="default"
      type="JNDI"
      url="PortalDb" />
```

On peut choisir une connexion JDBC traditionnelle vers une base de donnée. On l'indiquera comme cela dans le fichier de configuration :

```
<pool name="default"
      type="JDBC"
      url="http://unire.multimedia.uhp-nancy.fr/portalXX"
      driverClassName="com.mysql.jdbc.Driver"
      username="portaldev"
      password="portalN2" />
```

Il nous faut maintenant créer la classe action qui traite le formulaire et enregistre le contact dans la base. De la même façon que pour l'action `Default`, vous allez créer une classe `AddPerson.java` héritant de la classe `SubChannel` du MAG. Laissez Eclipse s'occuper du constructeur. Surchargez la méthode `init()`.

Récupérez les valeurs saisie dans les champs du formulaire par :

`runtimeData.getParameter("nom");` et vérifiez quelles ne sont pas nulles ou vide. Dans le cas contraire afficher un message d'erreur qui redirigera l'utilisateur vers l'action par défaut. Pour cela :

```
Message.message(mainChannel, runtimeData, new MessageBean(
  "Tous les renseignements sont obligatoires" ), "default");
```

Si tout est bon créez un objet `Person` et enregistrez ses coordonnées grâce à `BDDConnection.newPerson`.

Si l'enregistrement c'est bien déroulé la méthode `init` affichera un message redirigeant vers l'action par défaut et retournera `Boolean.FALSE`. Ainsi l'action n'a pas d'affichage propre. Elle ne surchargera pas la méthode `setXML` et ne nécessite pas de XSL correspondant.

Créez l'action `addperson` dans le fichier `CAgenda.xml` qui pointerà vers la classe `AddPerson.java`. Testez le canal.

### Etape N°3

Nous allons créer une nouvelle action "`viewpersons`" qui va afficher la liste des contacts de l'utilisateur.

La classe `ViewPersons.java` devra appeler `BDDConnection.getAllPersons()`; afin de récupérer une "collection" de personne. Puis la méthode `setXML` devra parcourir cette collection et créer le xml correspondant.

Créez la feuille de style qui interprètera le XML et affichera les contacts sous forme d'un tableau.

Pensez à créer l'action correspondante dans le fichier `CAgenda.xml` et le lien dans l'action `Default` qui appellera cette action :

```
<a href="{ $baseActionURL }?action=viewpersons">Liste des
personnes</a>.
```

Testez le canal.

### Etape N°4

Nous allons maintenant proposer à l'utilisateur de charger un fichier XML contenant l'ensemble de ses contacts sous la forme de l'exemple `mes_contacts.xml`.

En fait on aura deux actions qui utiliseront la même classe (mais deux méthodes différentes de celle-ci). Elles seront déclarées de la manière suivante dans le fichier de configuration :

```
<action name="uploadform"
  classname="org.esupportail.portal.channels.CAgenda.actions.Upload"
  xslfile="Upload.xsl" />
```

```
<action name="parsefile"
  classname="org.esupportail.portal.channels.CAgenda.actions.Upload"
  init="parseinit" />
```

La première action proposera un formulaire permettant d'uploader le fichier de contacts. La seconde action récupèrera et parsèra le fichier.

Créez la première action qui affiche le formulaire de la façon suivante :

```
<form enctype="multipart/form-data" method="post"
action="{ $baseActionURL }">
  <input type="hidden" name="action" value="parsefile" />
  <table width="100%" border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td nowrap="true">Fichier :</td>
      <td nowrap="true"><input type="file" name="fichier"
/></td>
      <td width="100%">&#160;</td>
    </tr>
  </table>
  <br/>
  <input class="uportal-button" type="submit" value="Envoyer" />
</form>
```

Faites un lien permettant d'appeler cette action et testez le canal.

Pour réaliser la seconde action vous aurez besoin de :

- Créer un package `org.esupportail.portal.channels.CAgenda.utils` dans lequel vous copierez le fichier `UploadWorker.java`. Cette classe s'occupe du chargement du fichier.
- Copier le fichier `ParseXML.java` dans le package `org.esupportail.portal.channels.CAgenda.config`. Cette classe se charge d'interpréter le XML grâce au `digester` et de retourner une "Collection" d'objet `Person`. Attention, vous devez compléter cette classe afin d'ajouter les règles de parsing.

Créez maintenant la méthode `parseinit()` qui va :

- Récupérer le contenu du fichier chargé
- Parser ce contenu
- Récupérer la "Collection" d'objet `Person`
- Parcourir la liste et enregistrer chaque contact

Testez le canal.

## Etape N°5

Nous allons enfin permettre à l'utilisateur de télécharger ses contacts sous forme d'un fichier Excel CSV.

Nous allons donc créer une action "download" qui sera liée à la classe existante : `ViewPerson.java`. Ainsi on aura :

```
<action name="download"
  classname="org.esupportail.portal.channels.CAgenda.actions.ViewPersons"
  init="downloadinit" />
```

Elle devra implémenter `IMimeResponse` et donc les méthodes :

```
public String getContentType() {
    return "text/plain";
}
```



```
public InputStream getInputStream() throws IOException {
    return null;
}

public void downloadData(OutputStream out) throws IOException
{
    PrintWriter writer = new PrintWriter(out);
    Iterator i = persons.iterator();
    while(i.hasNext()) {
        Person tmp = (Person)i.next();
        writer.println(tmp.getName() + ";" +
tmp.getFirstname() + ";" + tmp.getPhone() + ";" +
tmp.getMail());
    }
    writer.close();
}

public String getName() {
    return null;
}

public Map getHeaders() {
    HashMap map = new HashMap();
    map.put("Content-disposition", "attachment;
filename=agenda.csv");
    return map;
}

public void reportDownloadError(Exception e) {
}
```

C'est la méthode `downloadData` qui écrit le contenu du fichier CSV.

Créez ensuite la méthode `downloadinit()` qui ne fera rien et retournera `Boolean.FALSE` (pas d'affichage correspondant).

Créez ensuite dans la XSL le lien permettant le download. Attention, MAG utilise une autre URL pour les redirections de download : `<xsl:param name="baseDownloadURL" />`

Testez le canal.

## Annexe A

### *Création d'un projet Eclipse à partir d'une archive ZIP*

- Décompresser l'archive.
- Créer un nouveau projet Java pointant sur ce répertoire sans se soucier des répertoires sources, des bibliothèques ou des projets liés.
- Ouvrir et adapter le fichier build.properties.
- Ajouter le fichier build.xml à la vue ANT.
- Exécuter la tâche prepare.
- Rafraîchir l'affichage.
- Ouvrir les propriétés du projet.
- Définir le ou les répertoires sources.
- Définir le répertoire de compilation.
- Choisir le ou les projets liés.
- Ajouter les bibliothèques nécessaires pour que le projet compile sans erreur.

### *Création d'un projet Eclipse vide*

- Créer un nouveau projet Java pointant sur le répertoire voulu ou laisser Eclipse choisir l'emplacement.
- Récupérer les fichiers build.properties et build.xml d'un autre projet.
- Adapter les deux fichiers de build.
- Ajouter le fichier build.xml à la vue ANT.
- Exécuter la tâche prepare.
- ...

## Annexe B

### *Sites utiles*

- <http://www.esup-portail.org>
- <http://www.uportal.org>

### *Documentations*

### *Contacts*

- <http://listes.uhp-nancy.fr>
- La liste dédiée au développement UNIRE est [unire.developpement@uhp-nancy.fr](mailto:unire.developpement@uhp-nancy.fr)
- <http://listes.esup-portail.org>
- [celine.bissler@uhp-nancy.fr](mailto:celine.bissler@uhp-nancy.fr)
- [mathieu.larchet@univ-nancy2.fr](mailto:mathieu.larchet@univ-nancy2.fr)