



Spring IDE

Bien que Spring mette à disposition d'intéressants mécanismes afin d'améliorer l'architecture des applications Java EE en se fondant sur l'injection de dépendances et la programmation orientée aspect tout en simplifiant les traitements de ces types d'applications.

Ces concepts permettent d'améliorer la modularisation des traitements et de décorréler les entités applicatives. Ainsi, certains aspects ne peuvent être testés qu'au démarrage de l'application au moment du chargement de la configuration par Spring. Cet aspect peut grandement nuire à la productivité des développements.

Spring IDE, un greffon de l'outil Eclipse, vise à adresser ces différentes problématiques afin de détecter les éventuelles erreurs en amont du lancement de l'application aussi bien au niveau de la configuration de l'injection de dépendances que de la programmation orientée aspect, ce greffon tirant parti de toutes les facilités de cet outil.

Mise en œuvre

Spring IDE correspond à un greffon Eclipse s'intégrant parfaitement avec les différentes caractéristiques de cet environnement de développement en offrant différents éditeurs, vues et entrées dans les menus ainsi que la complétion automatique afin de faciliter le développement d'applications Spring dans ce contexte.

Eclipse

Eclipse est un environnement de développement modulaire et fondé sur le concept des greffons. Initié à partir de la mise en Open Source par IBM d'un projet correspondant au successeur de

l'outil Visual Age, Eclipse est actuellement en un outil de développement éprouvé, robuste, très populaire et omniprésent dans le monde Java.

Cet outil n'adresse cependant pas que ce langage car des modules offrent la possibilité de réaliser notamment des développements en C, C++ ou PHP. Il sert également de socle à d'autres outils tels que les outils de modélisation UML et peut même être utilisé en tant que brique de base pour des développements d'interfaces graphiques riches.

Au niveau de son architecture, Eclipse se fonde sur des briques dédiées par l'intermédiaire de SWT et JFace, ce premier étant particulièrement performant. Les mécanismes de gestion des greffons sont pris en charge par l'intermédiaire d'Equinox, un framework OSGi enrichi avec la fonctionnalité des points d'extension.

Sur ces deux briques de base, reposent des modules relatifs aux composants dédiés au développement tels que l'espace de travail, les vues et les perspectives. Différents outils et assistants sont mis à disposition afin de créer des ressources dans différents contextes, de réaliser des recherches et des comparaisons. En parallèle de ces aspects, Eclipse offre en standard un intéressant mécanisme de gestion des greffons et de leurs mises à jour.

La figure A-1 récapitule les différentes briques constituant l'architecture interne d'Eclipse.

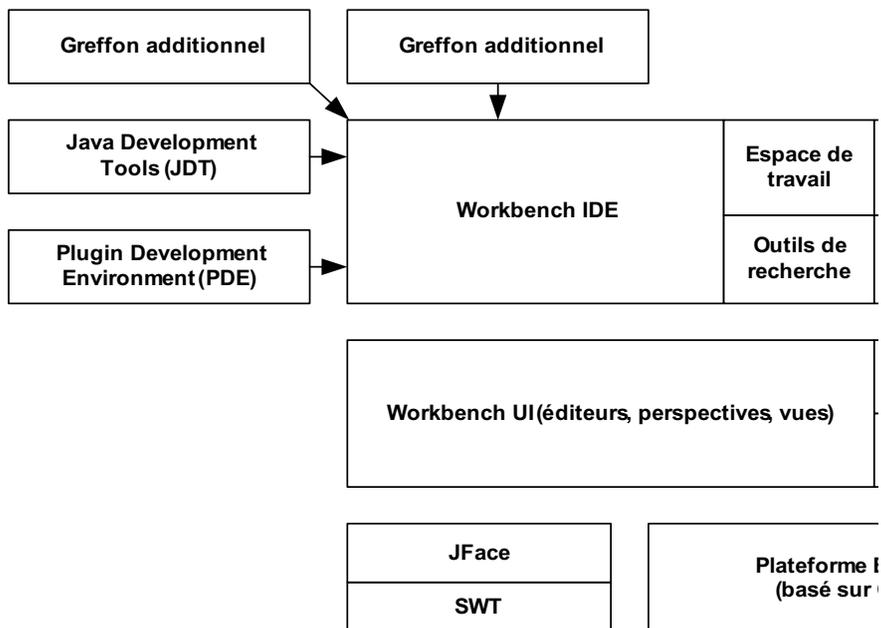


Figure A-1

Architecture de l'environnement de développement Eclipse

Revenons rapidement sur les éléments principaux éléments d'Eclipse afin de réaliser des développements. Les différentes ressources sont regroupées dans un espace de travail et structurées par l'intermédiaire du concept de projet.

Différentes représentations sont alors disponibles afin de parcourir et de modifier les ressources. Le concept de vue permet quant à lui d'afficher des informations, principalement de manière hiérarchique. L'édition d'une ressource textuelle ouvre alors un outil de visualisation dénommé éditeur et permettant d'afficher les données d'une ressource dans un contexte tout en offrant des facilités de modification. Des supports relatifs à la colorisation du contenu ainsi que la complétion sont également disponibles.

En complément des vues, la notion de perspective permet d'organiser ces vues pour un contexte donné. Au sein d'une perspective, il reste néanmoins possible d'offrir les vues souhaitées à tout moment.

Spring IDE se positionne en tant que greffon pour Eclipse afin d'offrir des facilités dans le développement d'applications Spring. Regardons maintenant comment mettre en œuvre et utiliser cet outil.

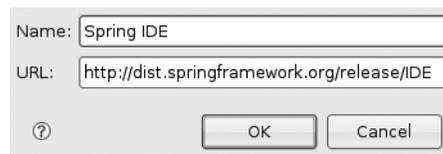
Installation

Afin d'installer Spring IDE dans Eclipse, il est recommandé d'utiliser la fonctionnalité intégrée de mise à jour de l'outil. Cette dernière est disponible par l'intermédiaire du menu Aide/Software Update/Find and Install. Dans l'écran correspondant, une liste de sites de mise à jour est proposée. Un site correspondant à Spring IDE 2.2 doit être alors ajouté par l'intermédiaire du bouton « New Remote Site ».

Pour une version d'Eclipse supérieure ou égale à 3.3, l'adresse <http://dist.springframework.org/release/IDE> doit être spécifiée, comme l'illustre la figure A-2.

Figure A-2

Configuration du site distant relatif à Spring IDE 2.2



The image shows a dialog box for configuring a remote site. It has two text input fields: 'Name' with the value 'Spring IDE' and 'URL' with the value 'http://dist.springframework.org/release/IDE'. Below the fields are three buttons: a help icon (question mark in a circle), an 'OK' button, and a 'Cancel' button.

Spring IDE offre une approche modulaire concernant ses différentes fonctionnalités. Il est ainsi possible de n'installer que les fonctionnalités souhaitées. Le tableau A-1 récapitule les différents modules disponibles.

Tableau A-1 – Modules disponibles dans Spring IDE 2.2

Module	Groupe	Description
Spring IDE Core	Core	Correspond au cœur du greffon fournissant les différents outils relatifs afin de faciliter la mise en œuvre de Spring dans Eclipse.
Spring IDE AOP Extension	Extensions	Fournit un support pour l'espace de nommage aop de Spring ainsi que la configuration d'aspects AspectJ par annotation conjointement avec la configuration aop:aspectj-autoproxy.
Spring IDE OSGI Extension	Extensions	Fournit un support pour les développements utilisant Spring Dynamic Modules dans un environnement OSGi.
Spring IDE Security Extension	Extensions	Fournit un support pour Spring Security 2.0.

Tableau A-1 – Modules disponibles dans Spring IDE 2.2 (suite)

Module	Groupe	Description
Spring IDE Web Flow Extension	Extensions	Met à disposition tous les outils afin de faciliter le développement d'applications Spring Web Flow dans Eclipse.
Spring IDE Autowire Extension	Extensions (Incubation)	Fournit un outil afin de supporter les configurations automatiques (autowiring) dans Spring.
Spring IDE JavaConfig Extension	Extensions (Incubation)	Fournit un support de l'outil JavaConfig de Spring.
Spring IDE AJDT Integration	Integrations	Met à disposition les briques relatives à l'intégration de Spring IDE avec AJDT.
Spring IDE Mylyn Integration	Integrations	Met à disposition les briques relatives à l'intégration de Spring IDE avec Mylyn.

Il est à noter que les outils AJDT et Mylyn peuvent être sélectionnés à partir de la fenêtre de sélection des modules de Spring IDE dans le cas d'une utilisation des intégrations avec ces outils.

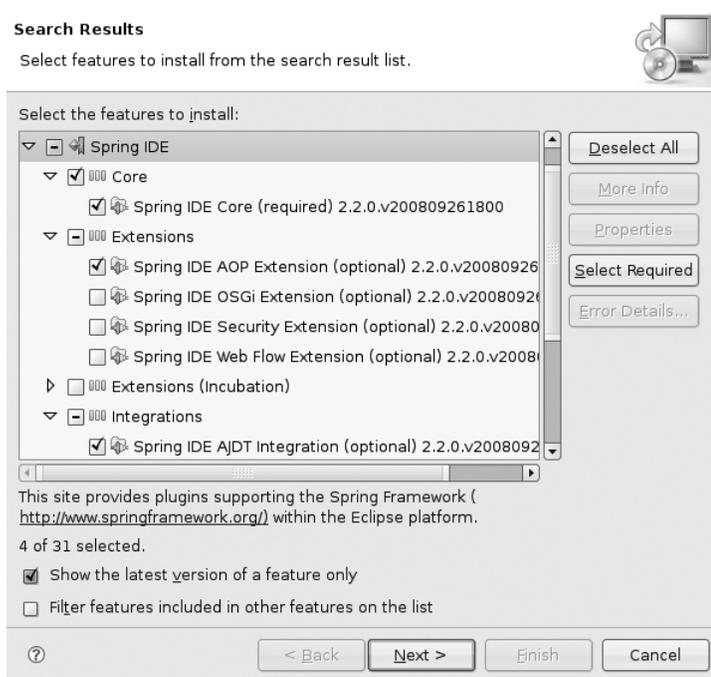
AspectJ Development Tools

L'AJDT correspond au griffon de l'outil Eclipse facilitant le développement d'applications utilisant AspectJ.

Puisque nous n'allons décrire que les fondations de Spring IDE, nous n'allons sélectionner que les modules « Spring IDE Core », « AOP Extensions » et « AJDT Integration ». Du fait du dernier module, l'outil AJDT doit également être sélectionné. La figure A-3 décrit l'écran de sélection correspondant.

Figure A-3

Sélection des modules de base de Spring IDE 2.2



Une fois l'acceptation des termes de la licence acceptés, Eclipse télécharge les différents fichiers correspondant puis propose leur installation. Il convient alors de sélectionner le bouton « Install all » pour les installer, Eclipse proposant un redémarrage afin que les modules soient pris en compte. Une fois le redémarrage réalisé, les différents modules sont présents la fonctionnalité « Product Configuration » ainsi que l'illustre la figure A-4.

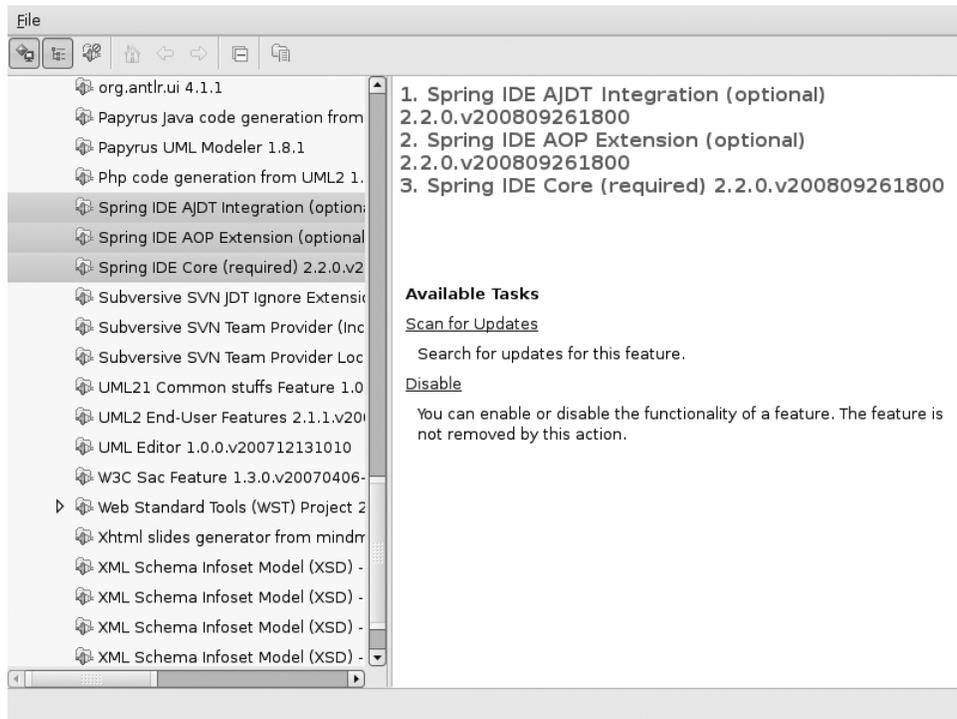


Figure A-4

Présence des modules installés de Spring IDE dans la configuration d'Eclipse

Configuration

La configuration générale de Spring IDE se situe au niveau des préférences de l'outil Eclipse, ces préférences étant accessibles depuis le menu « Menu/Préférences ». Deux entrées peuvent être utilisées à ce niveau :

- **Spring**, permettant de configurer les propriétés de Spring IDE notamment au niveau du paramétrage de la validation et des constructeurs d'Eclipse ;
- **Visualiser**, permettant de spécifier le fournisseur Spring IDE pour l'outil de visualisation de l'AJDT.

Regardons maintenant le support offert par Spring IDE au niveau des projets Eclipse Java afin de faciliter la mise en œuvre de l'injection de dépendances et de la POA avec Spring.

Nature Spring

Une fois ces éléments configurés, le travail de paramètre se réalise au niveau des projets Java Eclipse dans l'espace de travail. À cet effet, Spring IDE définit la notion de nature Spring afin de marquer les projets utilisant Spring et contenant des fichiers de configuration XML relatifs.

Deux approches sont possibles afin de spécifier une nature Spring à un projet Java. La première consiste en la création d'un projet de type Spring par l'intermédiaire du menu ou du menu contextuel « New/Other » et de la sélection de l'élément « Spring Project » dans la rubrique « Spring ».

La nature Spring peut également être ajoutée manuellement par l'intermédiaire du menu contextuel accessible sur le nom d'un projet. Il suffit de sélectionner l'élément « Add Spring Project Nature » dans la rubrique « Spring Tools ».

Propriétés Spring des projets

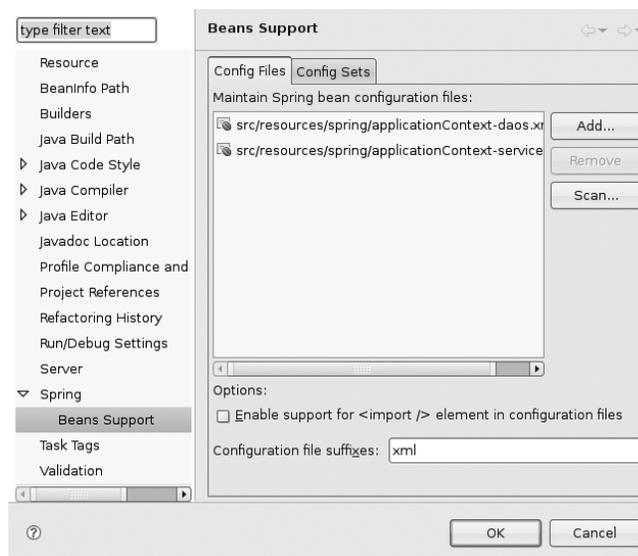
La présence de la nature Spring ajoute un élément `Spring` dans les propriétés du projet. Cet élément offre la possibilité de surcharge la configuration dans les préférences de l'espace de travail, de définir l'extension utilisée pour détecter les fichiers de configuration Spring et de configurer les ensembles de configuration.

Ce dernier aspect est particulièrement intéressant si la configuration d'un contexte applicatif se réalise à partir de plusieurs fichiers de configuration.

Par défaut, Spring IDE supporte l'extension `xml` pour les fichiers XML contenant une configuration Spring. Il convient néanmoins de sélectionner les fichiers utilisés pour une configuration de ce type dans le projet. La figure A-5 illustre la sélection des fichiers **applicationContext-daos.xml** et **applicationsContext-services.xml** localisés dans le répertoire **src/resources** pour le projet **tudu-core**.

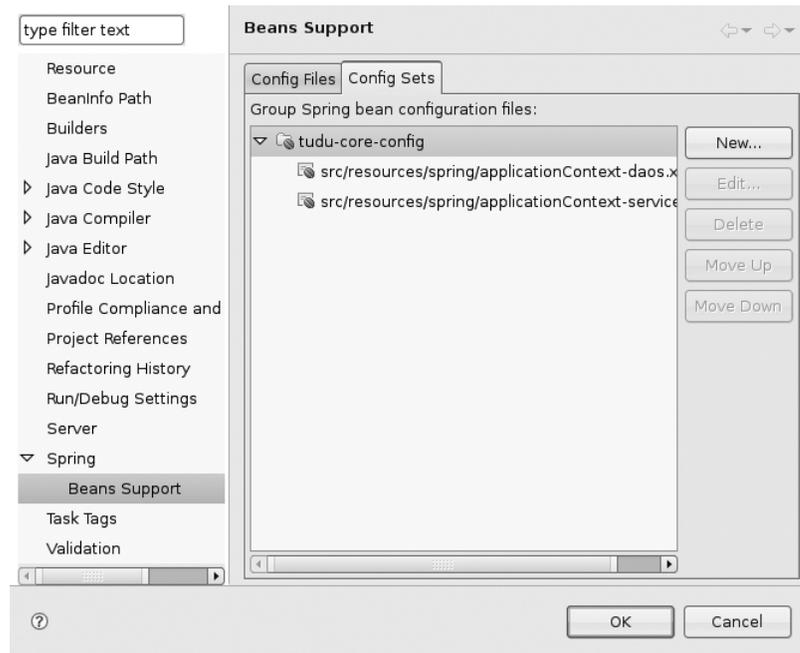
Figure A-5

Sélection des deux fichiers de configuration Spring pour le projet tudu-core



Comme ces deux fichiers sont utilisés pour configurer un même contexte applicatif Spring, il convient de les définir en tant qu'ensemble de fichiers de configuration. Ainsi, Spring IDE saura résoudre les liens des beans d'un fichier vers un autre. La figure A-6 décrit la manière de configurer cet aspect dans les propriétés du projet.

Figure A-6
Définition d'un ensemble de fichiers de configuration Spring pour le projet tudu-core



Maintenant que le projet est correctement configuré au niveau par rapport à Spring IDE, regardons les apports de cet outil dans le développement et le débogage d'applications Spring dans Eclipse.

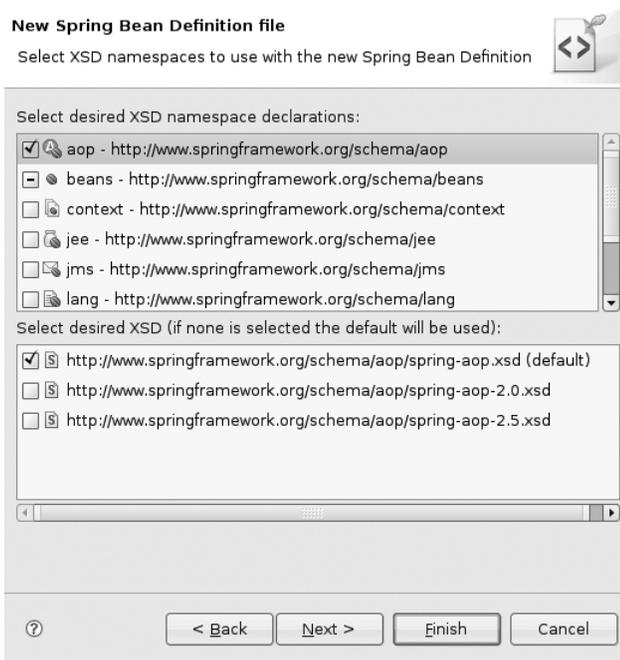
Support de l'injection de dépendance

Une des premières caractéristiques de Spring IDE consiste dans le support de l'espace de nommage de configuration par défaut de Spring. Il offre notamment ainsi la complétion au niveau aussi des balises bean et property ainsi de leurs attributs.

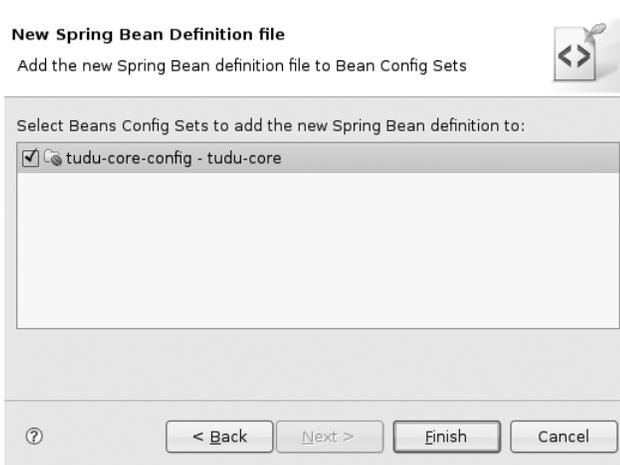
Il est à noter qu'il est également possible de créer un fichier de configuration Spring par l'intermédiaire du menu « New – Other » avec l'élément Spring Bean « Definition » de la rubrique « Spring ». Cet élément permet de créer un squelette minimal pour un fichier de configuration Spring, à savoir la balise beans avec les différents espaces de nommage. Le wizard de création permet de sélectionner les espaces de nommage à inclure et si l'utilisateur souhaite ajouter le fichier à un ensemble de configuration, comme l'illustrent les figures A-7 et A-8.

Figure A-7

Sélection des espaces de nommage à ajouter dans la balise beans lors d'une création

**Figure A-8**

Rattachement de la configuration à un ensemble de configuration lors d'une création



Les principales facilités offertes dans l'édition d'un fichier XML de configuration de Spring sont les suivantes :

- Support relatif à la recherche des classes pour l'attribut `class` de la balise `bean` avec la complétion des éditeurs Eclipse ;
- Support relatif à la détection des propriétés utilisables pour un bean dans l'attribut `name` de la balise `property` avec la complétion ;

- Support relatif à la recherche des identifiants de beans pour un référencement de beans dans l'attribut ref de la balise property.

Elles offrent un important gain de productivité dans le développement d'applications Spring puisqu'elles permettent de voir les erreurs dans la configuration XML avant de lancer l'application correspondante.

Les figures A-9 à A-11 illustrent respectivement ces trois aspects différents mettant en œuvre la complétion des éditeurs d'Eclipse.

```

<!-- DAO -->
<bean id="todoListDAO" class="TodoListD">
  <property name="sessionFactory" ref="
</bean>
<bean id="todoDAO" class="tudu.domain.d
  <property name="sessionFactory" ref
</bean>
<bean id="userDAO" class="tudu.domain.d
  <property name="sessionFactory" ref
</bean>
<!-- Hibernate -->
<bean id="sessionFactory" class="org.sp
  <property name="mappingResources">
    <list>
      <value>/tudu/domain/model/User.hbm.xml</value>

```

Figure A-9

Complétion du nom des classes pour l'attribut class de la balise bean

```

<!-- DAO -->
<bean id="todoListDAO" class="tudu.domain.dao.hibernate.TODOListDAOImpl">
  <property name="session
</bean>
<bean id="todoDAO" class="t
  <property name="session
</bean>
<bean id="userDAO" class="t
  <property name="session
</bean>
<!-- Hibernate -->
<bean id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
  <property name="mappingResources">
    <list>
      <value>/tudu/do
      <value>/tudu/domain/model/TODO.hbm.xml</value>

```

Figure A-10

Complétion du nom des propriétés pour l'attribut name de la balise property

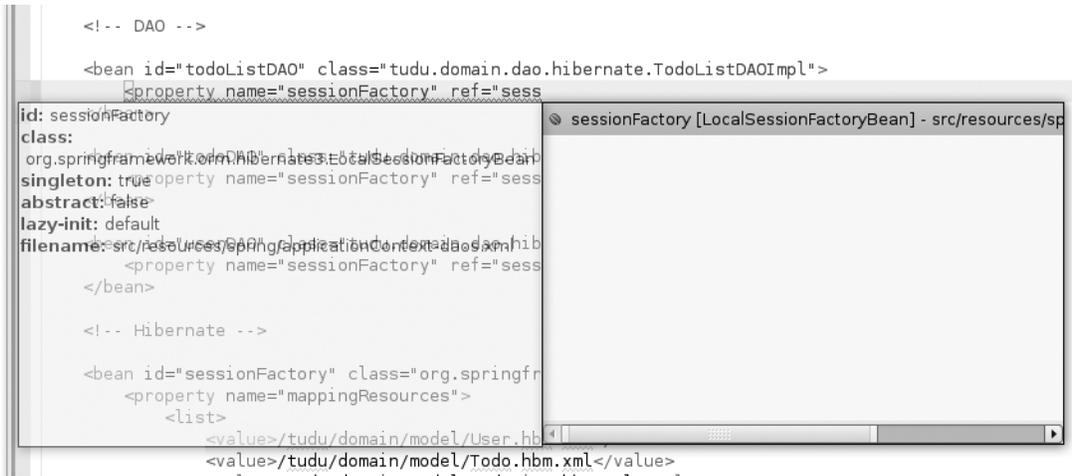


Figure A-11

Complétion des identifiants de beans pour l'attribut ref de la balise property

En parallèle de ces aspects, Spring IDE met à disposition les erreurs et les avertissements de configuration au niveau de la vue Problems d'Eclipse au même niveau que les erreurs Java. Ces erreurs peuvent correspondre notamment aussi bien à des erreurs de syntaxe XML, de classes et de propriétés non trouvées que des références non résolues vers des beans. La figure A-12 illustre l'affichage d'une erreur de configuration dans la vue Problems.

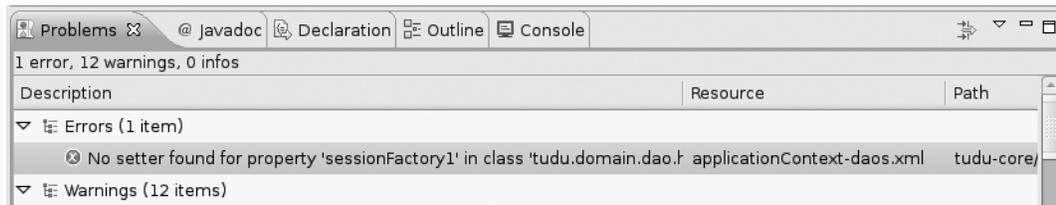


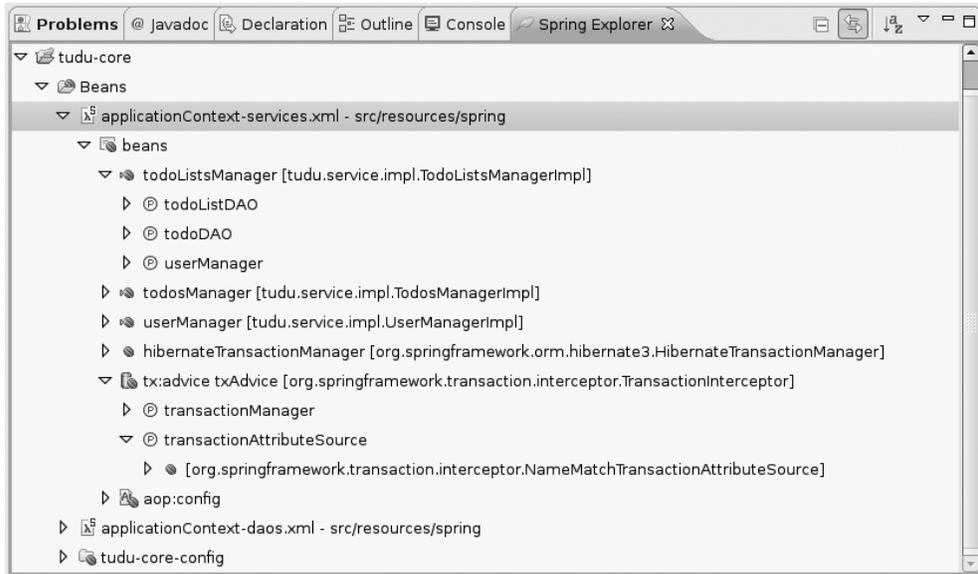
Figure A-12

Affichage des erreurs de configuration Spring dans la vue Problems

Spring IDE offre également un support dans des vues existantes d'Eclipse et la mise à disposition de vues spécifiques afin d'afficher d'une manière différente les informations présentes dans les fichiers de configuration de Spring. Les différentes vues supportées sont les suivantes :

- Vue Outline, permettant d'affichage une représentation hiérarchique directement calqué sur la structure du fichier XML correspondant ;
- Vue Spring Explorer, offrant également une représentation hiérarchique mais plus élaborée que la précédente. Elle offre en effet un support des différentes espaces de nommage ainsi que les relations entre les beans.

La figure A-13 illustre la représentation du contenu de la configuration Spring contenu dans le fichier **applicationContext-services.xml**.

**Figure A-13**

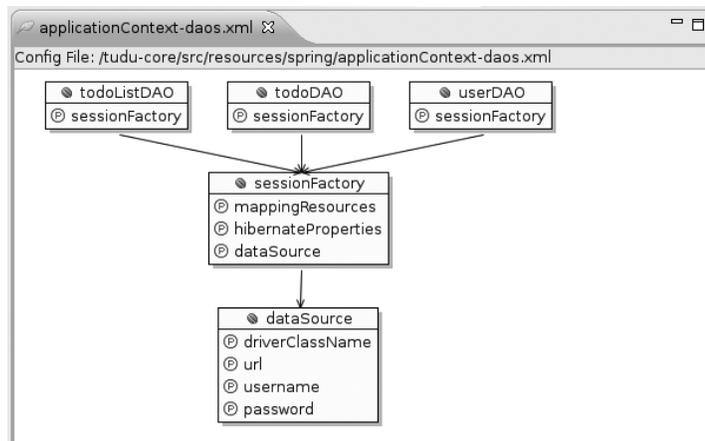
Affichage de la structure hiérarchique d'une configuration avec la vue Spring Explorer

En complément de la vue Spring Explorer, Spring IDE met à disposition une représentation graphique des beans contenus dans un fichier de configuration Spring, cette représentation s'affichant par l'intermédiaire d'un éditeur Eclipse dont les éléments sont en affichage seul. Cet éditeur est accessible depuis la vue Spring Explorer en se fondant sur le menu contextuel sur le nœud d'un fichier de configuration ou d'un élément de configuration. Dans ce dernier cas, une représentation restreinte s'affiche.

La figure A-14 illustre la représentation graphique des éléments contenus dans le fichier **applicationContext-daos.xml**.

Figure A-14

Éditeur graphique de Spring IDE



Support de la POA

Le support de la programmation orientée aspect par Spring IDE correspond à la caractéristique la plus appréciable car elle permet d'avoir accès en temps réel dans la phase de développement aux impacts d'un tissage d'un aspect ainsi qu'aux éventuelles erreurs dans les coupes AspectJ.

Impacts des tissages

À l'instar de l'espace de nommage de Spring, Spring IDE supporte l'espace de nommage aop relatif à la configuration de la POA au niveau de la syntaxe XML. Il permet par complétion d'avoir accès aux balises ou attributs possibles dans un contexte donné.

L'apport le plus intéressant à ce niveau consiste en l'affichage de l'impact d'un aspect à différents niveaux en utilisant une approche similaire à celle de l'outil AJDT :

- Affichage de l'impact des aspects dans la marge au niveau des définitions des beans dans la configuration Spring. Le concept d'ensemble de configuration est supporté à ce niveau ;
- Affichage de l'impact des aspects dans la marge au niveau des méthodes impactées dans le code des classes correspondantes.

Il est à noter que, dans les deux cas ci-dessus, le type de code advice de l'aspect est signalé avec différentes flèches dans la marge des éditeurs. De plus, dans le cas d'une information relative à un tissage sur un bean, les flèches dans la marge sont tournées vers le bean, l'inverse se produisant pour la configuration d'un aspect.

Les figures A-15 et A-16 illustrent respectivement l'impact du tissage au niveau de la configuration de Spring et du code d'une méthode d'une classe impactée.



```

<bean id="todosManager" class="tudu.service.impl.TodosManagerImpl">
  <property name="todoDAO" ref="todoDAO"/>
</bean>
<bean id="userManager" class="tudu.service.impl.UserManagerImpl">
  <property name="userDAO" ref="userDAO"/>
</bean>
<!-- Transactions -->
<bean id="hibernateTransactionManager"
  class="org.springframework.orm.hibernate3.HibernateTransactionManager">
  <property name="sessionFactory" ref="sessionFactory"/>
</bean>
<tx:advice id="txAdvice"
  transaction-manager="hibernateTransactionManager">
  <tx:attributes>
    <tx:method name="create*" />
    <tx:method name="update*" />
    <tx:method name="delete*" />
    <tx:method name="*" read-only="true" />
  </tx:attributes>
</tx:advice>
<aop:config>
  <aop:advisor pointcut="execution(* *.*/ManagerImpl.*(..)"
    advice-ref="txAdvice" />
</aop:config>

```

The image shows a code editor window displaying XML configuration for Spring. A tooltip is visible over the line containing the method name pattern in the tx:attributes section, displaying the text "Element : method" and "Press 'F2' for focus." The tooltip is a small rectangular box with a light background and a dark border.

Figure A-15

Impact du tissage d'un aspect au niveau de la configuration de Spring

```
/**
 * Find a Todo by ID.
 *
 * @see tudu.service.TodosManager#findTodo(java.lang.String)
 */
public Todo findTodo(final String todoId) {
    System.out.println("--> TodosManagerImpl.findTodo");
    if (log.isDebugEnabled()) {
        log.debug("Finding Todo with ID " + todoId);
    }
    Todo todo = todoDAO.getTodo(todoId);
    TodoList todoList = todo.getTodoList();
    User user = userManager.getCurrentUser();
    if (!user.getTodoLists().contains(todoList)) {
        if (log.isInfoEnabled()) {
            log.info("Permission denied when finding Todo ID '" + todoId
                + "' for User '" + user.getLogin() + "'");
        }

        throw new PermissionDeniedException(
            "Permission denied to access this Todo.");
    }
    return todo;
}
```

Figure A-16

Impact du tissage d'un aspect au niveau du code des classes

Cette fonctionnalité de Spring IDE offre la possibilité de valider au niveau du développement la validité d'une expression d'une coupe. Aucune exécution de l'application correspondante n'est nécessaire.

Vue Bean Cross References

Afin d'avoir accès à des informations plus précises sur les impacts du tissage d'un aspect, Spring IDE met à disposition la vue Bean Cross References. Cette dernière permet de voir simplement les différents aspects tissés sur un bean et sur quels beans est tissé un aspect.

En cochant l'icône avec les deux flèches dans la barre d'outils de la vue, le contenu de cette dernière se synchronise automatiquement avec l'élément sélectionné dans le fichier de configuration de Spring.

Cette vue possède une structure hiérarchique afin d'afficher tout d'abord les différents aspects ainsi que les codes advice contenus. Ensuite, pour chacun d'entre eux, sont rattachées en dessous les informations relatives aux beans tissés avec les méthodes impactées.

Il est à noter que Spring AOP considère un intercepteur comme un aspect bien que cela ne soit pas tout à fait exactement rigoureusement.

La figure A-17 décrit l'affichage des beans impactés par le tissage d'un aspect relatif aux transactions dans le fichier **applicationContext-services.xml**.

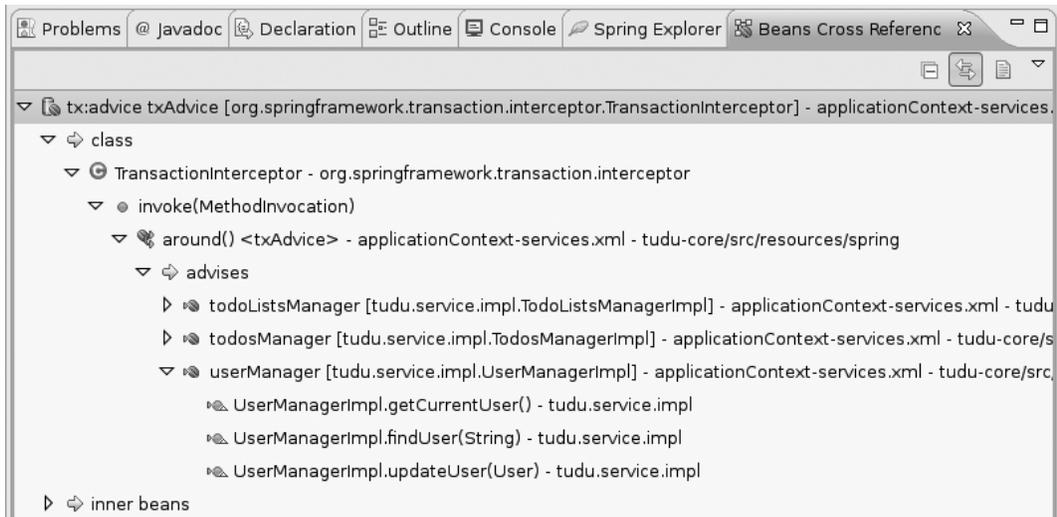


Figure A-17

Impacts détaillés du tissage d'un aspect par l'intermédiaire de la vue Beans Cross Reference

D'un autre côté, cette même structure hiérarchique offre la possibilité d'afficher pour un bean sélectionné les listes des aspects tissés sur ses méthodes avec une indication sur le type de code advice correspondant.

La figure A-18 décrit l'affichage des aspects tissés sur le bean `userManager` configuré dans le fichier `applicationContext-services.xml`.

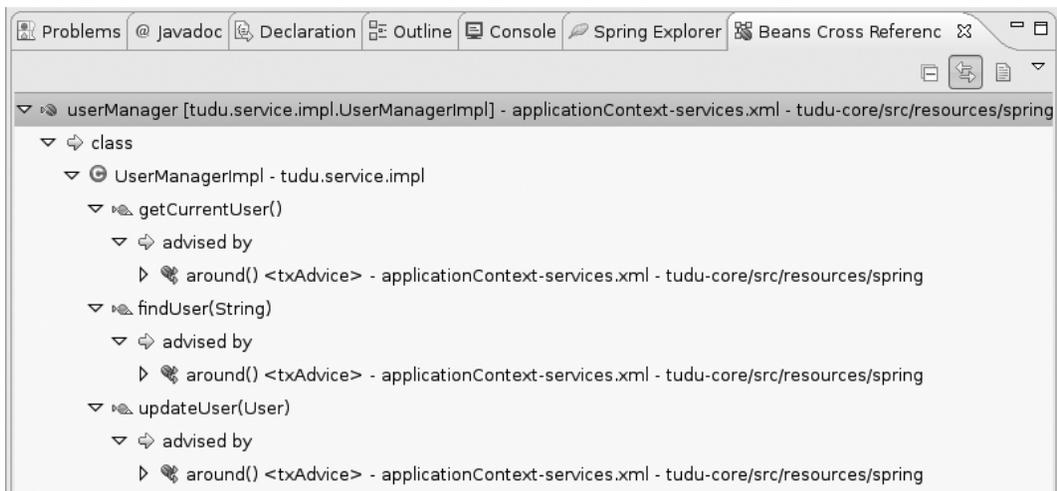


Figure A-18

Aspects tissés sur un bean par l'intermédiaire de la vue Beans Cross Reference

Cette vue est particulièrement intéressante afin d'avoir une vision globale par bean des aspects tissés facilitant ainsi la correction de problèmes au niveau des coupes durant la phase de développement et sans avoir à lancer l'application correspondante.

Outil Visualiser

Au niveau de la programmation orientée aspect, Spring IDE offre la possibilité de s'intégrer avec l'outil Visualiser de l'AJDT. Cet outil permet d'afficher l'impact des coupes sur les classes d'une manière globale sous forme de traits.

Afin de pouvoir utiliser cette fonctionnalité, il suffit de spécifier Spring IDE en tant que fournisseur pour l'outil Visualiser dans les préférences d'Eclipse, comme nous l'avons décrit dans la section « Configuration ».

Les vues correspondantes sont accessibles dans la rubrique Visualiser de la liste des vues disponibles :

- Visualiser Menu, décrivant les légendes associées aux couleurs des traits sur les classes et les packages ;
- Visualiser, correspondant à la vue affichant à proprement parlé l'impact des aspects sur les classes et les packages.

Une fois la vue Visualiser ouverte, il faut par contre sélectionner les entités à partir de la vue Package afin qu'elles soient prises en compte. La vue offre la possibilité de visualiser les impacts aussi bien au niveau des packages que des classes individuellement en utilisant les icônes de sa barre d'outils.

Les figures A-19 et A-20 illustrent respectivement les vues Visualiser Menu et Visualiser, vues décrivant les impacts des aspects sur les classes contenues dans le package `tudu.service.impl`.

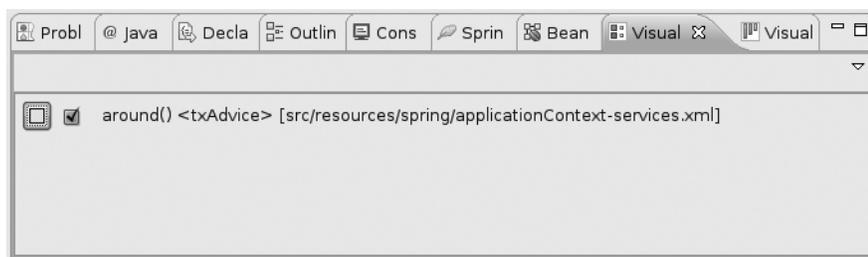


Figure A-19

Configuration des couleurs utilisées dans la vue Visualiser

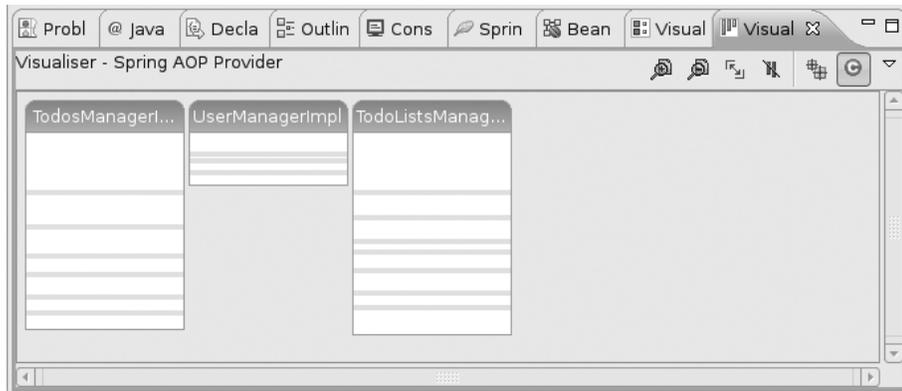


Figure A-20

Impacts des aspects sur les classes contenues dans le package tudu.service.impl

Refactoring

Une intéressante fonctionnalité de Spring IDE consiste en la mise à disposition de fonctionnalités relatives au refactoring. Cet aspect offre notamment la possibilité de garder une synchronisation entre les entités et leurs configurations relatives dans Spring à la suite d'un renommage ou un déplacement.

Spring IDE s'intègre parfaitement dans les mécanismes de refactoring mis à disposition pour l'environnement de développement Eclipse et accessible par l'intermédiaire du menu « Refactoring ». Il permet ainsi aussi bien de visualiser les impacts sur les fichiers de configuration de Spring que de les réaliser.

Entités Java

Spring IDE s'insère de manière transparente dans le refactoring d'éléments Java afin de permettre la répercussion des impacts au niveau des fichiers de configuration Spring. Le tableau A-2 récapitule les différents cas supportés par l'outil.

Tableau A-2 – Différentes possibilités de refactoring sur les entités Java

Refactoring	Impacts sur la configuration XML
Modification du nom d'une classe	Mise à jour de la valeur de l'attribut <code>class</code> dans la définition des beans relatifs à cette classe.
Déplacement d'une classe	Mise à jour de la valeur de l'attribut <code>class</code> dans la définition des beans relatifs à cette classe.
Modification du nom d'un attribut d'une classe	Mise à jour de la valeur de l'attribut <code>name</code> de la balise <code>property</code> dans la définition des beans relatifs à cette classe.

Les figures A-21 et A-22 illustrent la visualisation des impacts suite aux modifications respectives d'un nom de classe et d'un nom de propriété toutes deux présentes dans un fichier de configuration Spring.

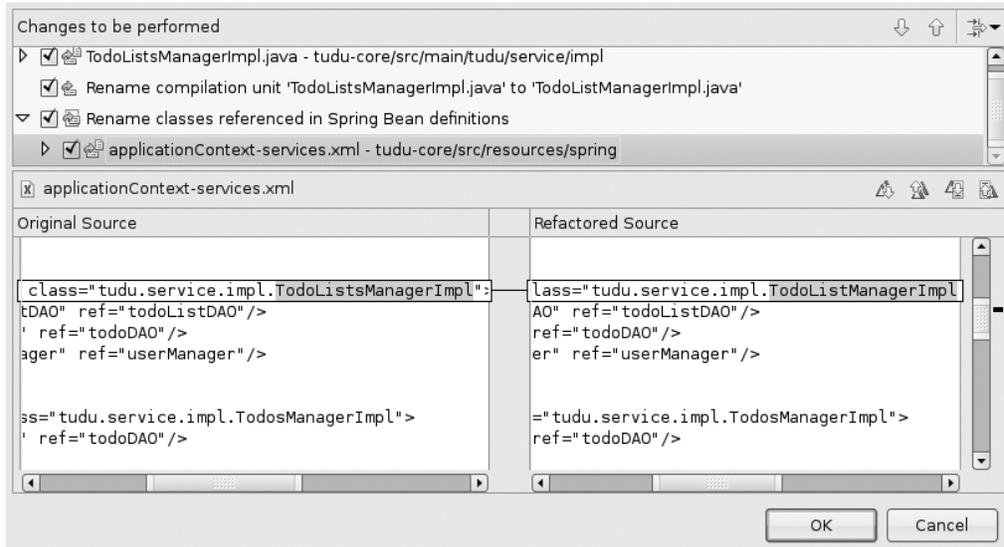


Figure A-21

Impacts de la modification d'un nom de classe

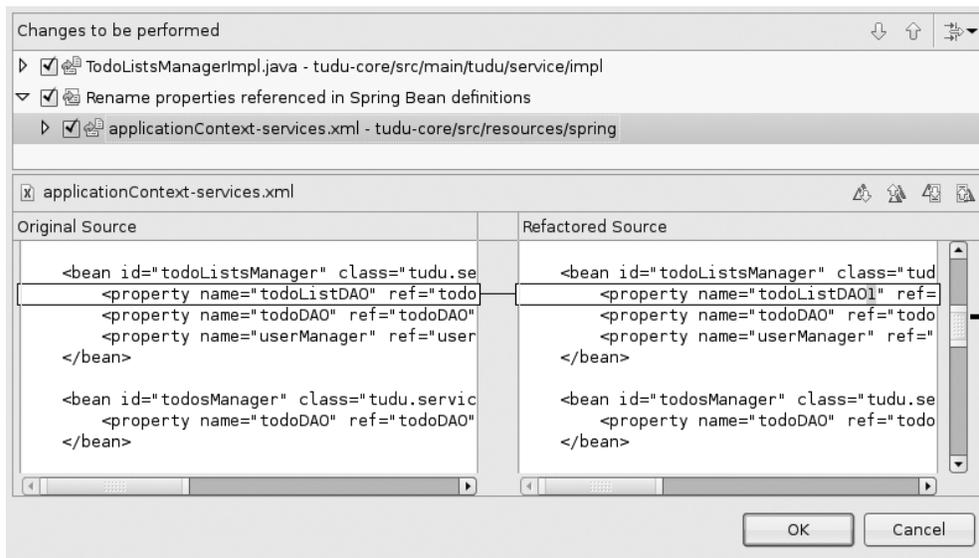


Figure A-22

Impacts de la modification d'un nom de propriétés d'une classe

Au niveau de la configuration XML de Spring

Au sein même d'un fichier de configuration XML de Spring, la modification d'éléments d'un bean peut avoir un impact aussi bien sur les autres beans que sur les entités Java correspondantes. La fonctionnalité de refactoring permet encore à ce niveau de garder une cohérence dans la configuration ainsi qu'une synchronisation avec les entités Java.

Le tableau A-3 récapitule les différentes opérations de refactoring supportées à ce niveau.

Tableau A-3 – Différentes possibilités de refactoring sur la configuration XML

Refactoring	Impacts sur la configuration XML
Modification de l'identifiant d'un bean	Mise à jour de la valeur de l'attribut <code>id</code> dans la définition du bean ainsi que de tous les liens vers cet identifiant.
Modification du nom d'une classe	Mise à jour de la valeur de l'attribut <code>class</code> dans la définition de tous les beans relatifs à cette classe et modification du nom dans la classe.
Déplacement d'une classe	Mise à jour de la valeur de l'attribut <code>class</code> dans la définition de tous les beans relatifs à cette classe et modification du nom dans la classe.
Modification du nom d'un attribut d'une classe	Mise à jour de la valeur de l'attribut <code>name</code> de la balise <code>property</code> dans la définition de tous les beans relatifs à cette classe et modification correspondante dans la classe si l'option relative à la mise à jour du mutateur a été sélectionnée.

Les figures A-23 et A-24 illustrent la visualisation des impacts suite aux modifications respectives d'un identifiant de beans et d'un nom de propriété toutes deux présentes dans un fichier de configuration Spring.

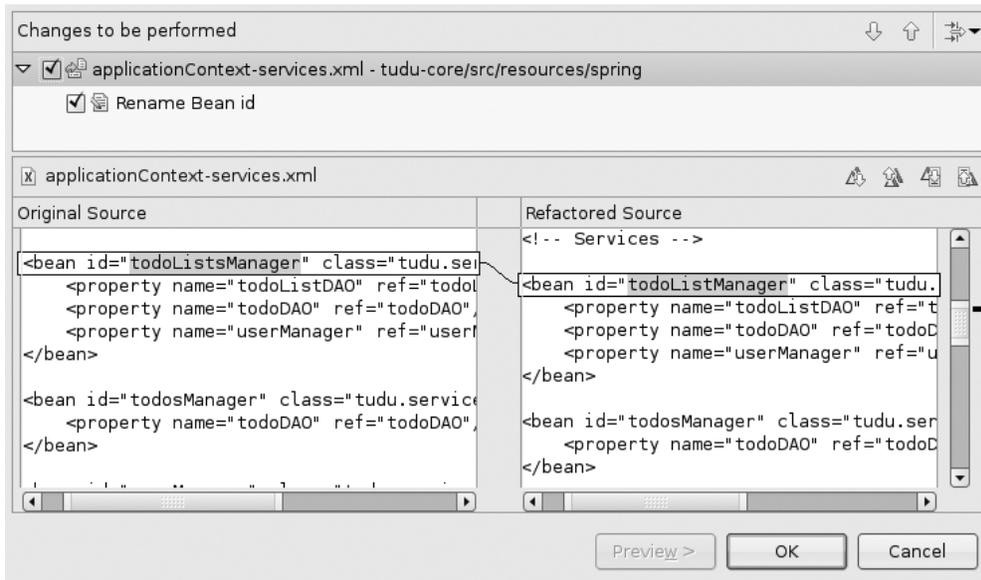
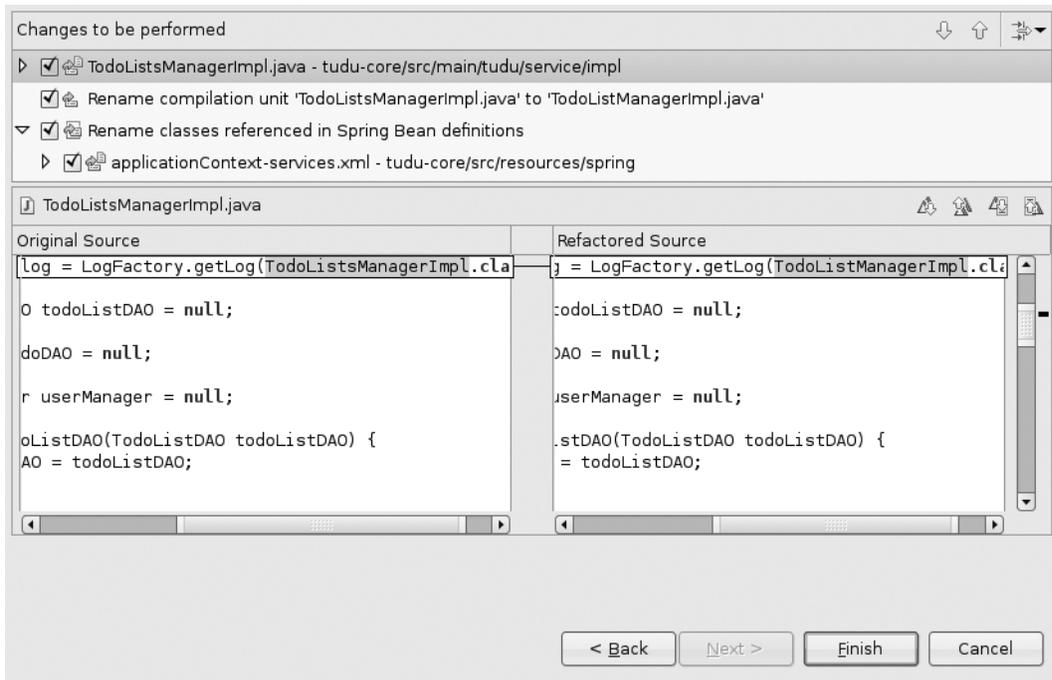


Figure A-23

Impacts sur la modification d'un identifiant de bean

**Figure A-24**

Impacts sur la modification de l'attribut class d'un bean

