**La génération d'une application JavaServer Faces 2.x CRUD à partir d'une base de donnees + Securite**

Cette application Web interagit avec une base de données en arrière-plan. Elle offre la possibilité de visualiser et modifier les données contenues dans la base de données - autrement appelée fonctionnalité CRUD (Create, Read, Update, Delete). Cette application repose sur les technologies suivantes.

• JavaServer Faces (JSF) 2.x pour les pages Web front-end, pout la manipulation de la validation et pour la gestion du cycle de requete-réponse.

• Java Persistence API (JPA) 2.0 à l'aide EclipseLink pour générer des classes entités à partir de la base de données, et de gérer les transactions. (EclipseLink est la mise en œuvre de référence pour l'APP, et est le fournisseur de persistance par défaut pour le serveur GlassFish.)

• Enterprise JavaBeans EJB (3.1), qui vous fournit des EJB stateless qui accèdent aux classes d'entités, et qui contiennent la logique métier de l'application.

1. **JPA :** 
   1. **Definition**

La Java Persistence API (JPA) est une spécification Java qui sert pour accéder, persister, et gérer les données entre objets Java / classes et une base de données relationnelle. JPA a été défini dans le cadre de la spécification EJB 3.0 comme un remplacement a la spécification des entités beans EJB 2 CMP. JPA est maintenant considérée comme etant l'approche standard de l'industrie pour le mappage Objet- Relationel (ORM) dans l'industrie Java.

JPA lui-même est juste une spécification, pas un produit; il ne peut pas effectuer la persistance ou toute autre chose lui-même. JPA est juste un ensemble d'interfaces, et nécessite une implementation. Il y’a des implémentations JPA open-source et commerciaux dependants du choix . JPA nécessite une base de données pour la persistance.

JPA permet la persistence aisee du POJO (Plain Old Java Objects) sans qu’il n’est nécessaire que les classes mettent en œuvre des interfaces ou des méthodes comme dans le cas de la spécification EJB 2 CMP. JPA permet les correspondances relationnel-objet d'un objet d’être définis par des annotations standards ou par XML définissant comment la classe Java est mappee à une table de base de données relationnelle. JPA définit également une API EntityManager a temps d'exécution pour le traitement des requêtes (Query) et des transactions sur les objets contre la base de données. JPA définit un langage de requete (Query) au niveau objet, JPQL, pour permettre l'interrogation (Querying) des objets de la base de données.

JPA est la dernière de plusieurs spécifications de persistance Java. Le premier était le service de persistance liaison Java OMG, qui n'a jamais été très réussie. Vient ensuite EJB 1.0 beans d'entité CMP, qui a très bien réussi à être adoptée par les grands fournisseurs Java EE (BEA, IBM), mais il y avait une réaction contre cette spécification par certains utilisateurs qui pensaient que les exigences de spécifications sur les Entity Beans étaient trop complexse et que les frais etaient eleveset la performance était pauvre. EJB 2.0 CMP a tenté réduire une partie de la complexité des Entity Beans en introduisant les interfaces locales, mais la majorité de la complexité est restée. EJB 2.0 manquait également la portabilité, en ce que les descripteurs de deploiement (deployment descriptors) définissant le mapping objet-relationnel ne sont pas précisées et étaient tous propriétaires.

Cette reaction a partiellemente aboutit à la création d'une autre spécification de persistance Java, (Java Data Objects) JDO. JDO a obtenu plusieurs vendeurs indépendants tels que Kodo JDO, et plusieurs implémentations open-source, mais n'a jamais eu beaucoup de succès avec les grands fournisseurs Java EE.

Malgré les deux concurrentes Java normes de persistance EJB CMP et JDO, la majorité des utilisateurs ont continué à préférer les solutions d'API propriétaires, principalement TopLink (qui avait été repandue depuis une certaine periode et possedait sa propre API POJO) et Hibernate (qui était relativement un nouveau produit open -source qui avait aussi sa propre API POJO et a été rapidement devenu le standard de l'industrie open-source). Le produit TopLink anciennement détenu par WebGain a également été acquis par Oracle, ce qui a augmente son influence sur la communauté Java EE.

La reaction EJB CMP était seulement une partie d'une réaction contre toute la Java EE qui a été considérée comme trop complexe en général, et a incité des produits tels que Spring container. Cela a conduit a la spécification EJB 3.0 qui avait un objectif principal de réduire la complexité, ce qui a dirigé le comité des spécification sur la voie de JPA . JPA a été conçue pour unifier le EJB 2 CMP, JDO, Hibernate, et TopLink, et il semble qu’elle a très bien succede achiever ce but.

Actuellement, la plupart des fournisseurs de persistance ont issué des implémentations de JPA qui confirment son adoption par l'industrie et les utilisateurs. Ceux-ci comprennent Hibernate (acquis par JBoss, acquis par Red Hat), TopLink (acquis par Oracle), et Kodo JDO (racheté par BEA, acquis par Oracle). Autres produits qui ont ajouté le support pour JPA incluent CocoBase (détenue par Thought Inc.) et JPOX.

**1.2 – Utilisation**

**1.2.1 - Generalites**

L'IDE (Integrated Development Environment) dispose de deux wizards (assistants) qui génèrent tout le code de l'application. Le premier est [Entity Classes from Database wizard](https://netbeans.org/kb/docs/web/jsf20-crud.html#generateEntity) (l'Assistant des classes entitéd’apres base de données) qui permet de générer des classes d'entités a partir de la base de données fournie. Après la creation des classes d'entités, on utilise le [JSF Pages from Entity Classes wizard](https://netbeans.org/kb/docs/web/jsf20-crud.html#jsfPagesEntityClasses)  (l'assistant des Pages JSF a partir des classes entité) pour créer JSF managed beans and EJBs pour les classes d'entités, ainsi que d'un ensemble de pages Facelets pour gérer les vues des données des classes d'entité.

|  |  |
| --- | --- |
| **Software or Resource** | **Version Required** |
| [NetBeans IDE](https://netbeans.org/downloads/index.html) | 7.2, 7.3, 7.4, 8.0, Java EE bundle |
| [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) | 7 or 8 |
| [GlassFish Server Open Source Edition](http://glassfish.dev.java.net/) | 3.x, 4.x |
| [mysql-consult.zip](https://netbeans.org/projects/samples/downloads/download/Samples%252FJavaEE%252Fmysql-consult.zip) (MySQL)  *or* [javadb-consult.zip](https://netbeans.org/projects/samples/downloads/download/Samples%252FJavaEE%252Fjavadb-consult.zip) (JavaDB) |  |

on crée un projet Web et on ajoute le JavaServer Faces framework au projet. Lorsque on crée le projet, on doit sélectionner JavaServer Faces dans le panneau de Frameworks du New Project wizard.

  l'IDE génère le projet d'application web et ouvre **index.xhtml** dans l'éditeur.

**1.2.2 - Génération des classes d'entité à partir de la base de données**

Après la connexion à une base de données dans l'IDE, vous pouvez utiliser le **Entity Classes from Database wizard** pour générer rapidement des classes d'entités basées sur les tables de la base de données. L'IDE peut générer des classes d'entités pour chaque table que vous choisissez.

New > Entity Classes from Database

OU bien

Other > Persistence > Entity Classes from Database

Sélectionnez "New Data Source" de la liste " Data Source " déroulante pour ouvrir la the “Create Data Source dialog”

Tapez le nom JNDI et sélectionnez “Database Connection”.

 Les tables de la base de données apparaissent dans la liste des tables Disponibles

Sélectionnez les tables nécessaires ou Cliquez sur le bouton “Add All” pour sélectionner toutes les tables contenues dans la base de données

Tapez un nom (**jpa.entities**) pour le nom du package.

Vérifiez que les checkboxes “générer des requêtes nommées” (generate named queries) et “ créer une unité de persistence” (create a persistence unit) sont sélectionnés

 l'IDE génère les classes d'entités dans le package **jpa.entities** du projet

l'IDE examine les relations entre les tables de base de données. Dans le nœud du paquet jpa.entities (jpa.entities package node), vous pouvez voir que l'IDE a généré une classe d'entité (.java) pour chaque table, sauf pour la (les) table(s) de jointure.

L'IDE génère également des classes supplémentaires pour les tables ayant des clés primaires composites. Les classes de clés primaires pour ces tables ont PK ajouté au nom.

1. **JSF :** 
   1. **Definition**

Signifie "JavaServer Faces." JSF est un cadre (framework) qui permet aux développeurs Web de créer des interfaces utilisateur (user interfaces) pour les applications JavaServer. Il est soutenu par des serveurs Web a Java Enterprise Edition (Java EE).

JSF simplifie la création des applications Web en fournissant un ensemble standard d'outils (ou un API) pour la construction des interfaces utilisateur. Par exemple, au lieu de coder une forme Web en HTML, un développeur peut appeler une simple function JSF qui génère la forme. Une autre fonction de JSF peut être utilisée pour traiter les données saisies par l'utilisateur. Ces fonctions sont traitées sur le serveur et les données qui en resultant sont diriges vers le navigateur du client.

JSF bénéficie les développeurs en fournissant des objets réutilisables qui peuvent être facilement insérés dans des pages Web. Toutefois, ces composants sont également bénéfiques pour les visiteurs du site car ils produisent des éléments d'interface standardizes (normalizes). Puisque le code Java est traite sur le serveur, l'aspect des objets Web générés est cohérent à travers plusieurs sites Web. En plus, les composants JSF sont testés sur de multiples plateformes, afin qu'ils fonctionnent bien sur tous les principaux navigateurs.

Alors que JSF est souvent utilisé pour créer des éléments de base pour les pages Web, il prend également des fonctions avancées en charge, telles que l'accès a la base de données, interaction Ajax, et les actions de page JavaScript. Ces capacités sont utiles pour la construction de sites web dynamiques qui génèrent des pages sur la volée.

* 1. **– Utilisation**

**2.2.1 - Generalites**

Maintenant que les classes d'entités sont créées, vous pouvez créer l'interface Web pour afficher et modifier les données. Vous pourrez utiliser le **JSF Pages from Entity Classes wizard** pour générer des pages JavaServer Faces. Le code généré par l'assistant est basé sur les annotations de persistance contenues dans les classes d'entité.

Pour chaque classe de l'entité l'assistant génère les fichiers suivants.

• un session stateless bean qui étend AbstractFacade.java

• un JSF session-scoped, managed bean

• un répertoire contenant quatre fichiers Facelets pour les capacités de CRUD (Create.xhtml, Edit.xhtml, List.xhtml et View.xhtml)

L'assistant génère également les fichiers suivants.

• la classe AbstractFacade.java qui contient la logique du métier pour la création, la recherche, la modification et la suppression des instances d’entite

• les classes entite utilisées par les JSF managed beans (JsfUtil, PaginationHelper)

• un bundle propriétés pour les messages localisés, et une entrée correspondante dans le fichier Faces configuration du projet (un fichier **faces-config.xml** est créé s’il n’existait déjà pas)

• fichiers auxiliaires web, y compris un stylesheet par défaut pour les composants rendus, et un fichier de modèle Facelets

**2.2.2 - Génération des pages JSF à partir des classes d'entité**

Pour générer les pages JSF:

1. Dans la fenêtre Projects, sur le noeud du projet choisissez New > JSF Pages from Entity pour ouvrir l'assistant. (Si cette option ne figure pas, choisissez Other. Puis, dans l'assistant de fichier, sélectionnez la catégorie JavaServer Faces, puis JSF Pages from Entity Classes)

La liste contient les classes Entité contenues dans le projet. La liste ne répertorie pas les classes intégrables-embeddable (\*PK.java).

2. Cliquez sur **Add All** pour déplacer toutes les classes vers la boîte **Selected Entity Classes**.

3. Tapez **jpa.session** dans **JPA Session Bean Package**.

4. Tapez **JSF** pour **JSF Classes Package**.

5. Tapez **'/ resources / Bundle'** dans le champ **Localization Bundle Name**. Cela va générer un package nommé “**resources”** dans lequel le fichier “**Bundle.properties”** va résider. (Si vous laissez ce champ vide, les propriétés bundle seront créés dans le package par défaut du projet- project's default package)

6. En général, vous pouvez accéder et faire des modifications sur tous les modèles (templates) tenus par l'IDE en utilisant le gestionnaire de modèle- **Template Manager** (Tools > Templates).

7. Cliquez sur “Terminer”. L'IDE génère les **stateless session beans** dans le package de **jpa.session**, et la **session-scoped** JSF, **managed beans** dans le package JSF. Chaque session bean gère les opérations pour la classe de l'entité correspondante, y compris la création, la modification et la destruction d'instances de la classe d'entité via la Java persistance API. Chaque JSF managed bean implémente l'interface **javax.faces.convert.Converter** et effectue le travail de conversion des instances de la classe de l'entité correspondant à des objets **String** et vice versa.

Si vous ouvrez le **Web Pages node**, vous pouvez voir que l'IDE a généré un dossier pour chacune des classes d'entité. Chaque dossier contient les fichiers **Create.xhtml**, **Edit.xhtml**, **List.xhtml** et **View.xhtml**. L'IDE a également modifié le fichier **index.xhtml** en insérant des liens vers chacune des pages **List.xhtml**.

Chaque JSF managed bean est spécifique aux quatre fichiers Facelets correspondants et inclut le code qui invoque des méthodes dans le session bean approprié.

Ouvrez le nœud du dossier de ressources pour localiser la feuille de style (**stylesheet**) par défaut **jsfcrud.css** qui a été généré par l'assistan- wizard. Si vous ouvrez la page de bienvenue de l'application (**index.xhtml**) ou le fichier des modeles des Facelets- **Facelets template file** (**template.xhtml**) dans l'éditeur, vous verrez qu'il contient une référence à la feuille de style.

<h: outputStylesheet name = "css / jsfcrud.css" />

Le fichier modèle Facelets (**Facelets template file**) est utilisé par chacun des quatre fichiers Facelets pour chaque classe d'entité.

Si vous ouvrez le noeud “Source Packages” vous pouvez voir les **session beans**, **JSF managed beans**, les **utility classes**, et le **properties bundle** que l'assistant a généré.

L'assistant a également généré un fichier **Faces Configuration** (**faces-config.xml**) afin d'enregistrer le location du **properties bundle**. Si vous ouvrez le noeud **Configuration Files** et **openfaces-config.xml** dans l'éditeur XML, vous pouvez voir que l'entrée suivante est incluse.

<application>

    <resource-bundle>

        <base-name> / resources / Bundle </ base-name>

        <var> bundle </ var>

    </ resource-bundle>

</ application>

Aussi, si vous décompressez le package de **new resources** , vous trouverez le fichier **Bundle.properties** qui contient des messages pour la langue par défaut du client. Les messages ont été déduite des propriétés de la classe de l'entité.

1. **Securite :** 
   1. **Definition**

Ce tutoriel se concentre sur la sécurité des applications Web. Lors de la sécurisation des applications web, il ya deux principales préoccupations qui doivent être abordées:

1. Prévenir les utilisateurs non autorisés d'accéder à un contenu protégé.

2. Prévenir le contenu protégé d'être révlé alors qu'il est en cours de transmission.

La première préoccupation, le contrôle d'accès, est généralement un processus en deux étapes qui implique (1) déterminer si un utilisateur est bien celui qu'il prétend être (c.-à-authentification), puis (2) soit accorder ou refuser l'accès d’un utilisateur a la ressource demandée (par exemple, autorisation). Une façon simple et commune pour mettre en œuvre le contrôle d'accès pour les applications web est avec une “Login Form” qui permet au serveur de comparer les informations d'identification avec une liste pré-existante d'utilisateurs authentifiés.

La deuxième préoccupation, la protection des données en transit, consiste typiquement en utilisant Transport Layer Security (TLS), ou son prédécesseur, Secure Sockets Layer (SSL), afin de crypter toutes les données communiquées entre le client et le serveur.

nous aurons besoin des moyens suivants pour garantir l'application:

• Mettre en place une forme de connexion de la console d'administration qui permet aux membres du personnel l'accès aux services de la console, et empêche les utilisateurs non autorisés.

• Configurer le transport sécurisé des données à la fois pour les transactions, et pour toutes les données transmises vers et à partir de la console d'administration.

* 1. **Utilisation**

Comment configurer l'authentification de sécurité en utilisant une fenêtre de connexion de base (basic login window) et en utilisant aussi une forme de connexion dans une page Web (login form). Après la création des utilisateurs sur le serveur GlassFish, vous créez ensuite les rôles de sécurité en définissant les propriétés de sécurité dans le descripteur de deployment (deployment descriptor). Ce document montre également comment vous pouvez utiliser l'authentification JDBC pour sécuriser votre application lors du déploiement sur le serveur GlassFish.

**3.2.1 - La Creation de l’application Web**

  Vous créez d'abord le projet d'application Web et la structure de répertoire. Vous créez ensuite des fichiers html simples dans chacun des répertoires sécurisés. L'application Web utilise le “basic login authentication” pour accéder aux répertoires sécurisés. Si vous souhaitez utiliser un formulaire de connexion pour l'authentification, vous pouvez ajouter une page JSP avec la forme.

**3.2.1.1 - Création des répertoires sécurisés**

1. Choisissez File > New Project, sélectionnez Application Web à partir de la catégorie Java Web

2. Sélectionnez le serveur sur lequel vous souhaitez déployer votre application (Glassfish).

3. Vous ne devez ajouter un cadre(framework)

4. Pour générer un descripteur spécifique au serveur. Cliquez-droit sur le noeud du projet et sélectionnez New > Other > GlassFish > GlassFish Descriptor.

5. dans le server-Specific Descriptor. Acceptez toutes les valeurs par défaut et cliquez sur Terminer. Le descripteur spécifique au serveur, nommé GlassFish-web.xml apparaît dans le projet dans le dossier des fichiers de configuration.

6. Dans la fenêtre des projets de l'IDE, cliquez-droit sur “Web Pages” et choisissez New > Other. sélectionnez “Other” dans la catégorie et “Folder” comme type de fichier. Nommez-le SecureAdmin et cliquez sur Terminer. Le dossier SecureAdmin apparaît dans la fenêtre des projets dans le dossier Web Pages. Répétez l'opération pour créer un autre dossier nommé secureUser.

7. Créez un nouveau fichier HTML dans le dossier secureUser (pageU)

8. creez un dossier SecureAdmin et puis créez un nouveau fichier HTML nommé PageA.

**3.2.1.2 - Création de la page Indice de JSP (JSP index page)**

Vous créez maintenant la page d'index JSP (index.jsp) contenant des liens vers les zones sécurisées. Lorsque l'utilisateur clique sur le lien, ils sont invités à entrer le nom d'utilisateur et mot de passe. Si vous utilisez “basic login”, l’authentification va av oir lieu par la fenêtre de connexion par défaut du navigateur. Si vous utilisez “login form page”, l'utilisateur entre le nom d'utilisateur et mot de passe dans une forme.

<p>Request a secure Admin page <a href="secureAdmin/pageA.html">here!</a></p>

<p>Request a secure User page <a href="secureUser/pageU.html" >here!</a></p>

**3.2.2 - Création d'utilisateurs sur le serveur cible**

Pour pouvoir utiliser l'authentification utilisateur / mot de passe (login base ou connexion basée sur un formulaire) de sécurité dans les applications web, les utilisateurs et leurs rôles appropriés doivent être définis pour le serveur cible. Pour vous connecter à un serveur, le compte d'utilisateur doit exister sur ce serveur.

la definition des utilisateurs et des rôles varie selon le serveur cible que vous avez spécifié. Dans ce tutoriel, l'utilisateur admin et utilisateur sont utilisés pour tester la configuration de la sécurité. Vous devez confirmer que ces utilisateurs existent sur les serveurs respectifs, et que les rôles appropriés sont affectés aux utilisateurs.

**3.2.2.1 - Définition des utilisateurs sur le serveur GlassFish**

Pour ce scénario, vous devez utiliser la console d'administration du serveur GlassFish pour créer deux nouveaux utilisateurs nommés utilisateur et admin. L'utilisateur d'utilisateur nommé aura un accès limité à l'application, tout en administration aura des privilèges d'administration.

1. Ouvrez la Admin Console en allant à la fenêtre IDE des services, et clickez Servers > GlassFish server > View Domain Admin Console. La page login pour le serveur GlassFish ouvre dans la fenêtre de votre navigateur. Vous devez vous connecter en utilisant le nom d'utilisateur admin et mot de passe pour accéder à la console d'administration.

2. Dans “Admin Console”, accédez à Configurations > server-config > Security > Realms > File. Le ”Edit Realm panel” ouvre.

**3.2.3 - Configuration de la méthode de connexion**

Lors de la configuration de la méthode de connexion de votre application, vous pouvez utiliser la fenêtre de connexion fournie par votre navigateur pour l'authentification “basic login”. Sinon, vous pouvez créer une page web avec un formulaire de connexion. Les deux types de configuration de connexion sont basés sur l'authentification utilisateur / mot de passe.

Pour configurer la connexion, vous créez des contraintes de sécurité et d'assigner des rôles à ces contraintes de sécurité. Contraintes de sécurité définissent un ensemble de fichiers. Lorsque vous attribuez un rôle à une contrainte, les utilisateurs ayant ce rôle auront accès à l'ensemble des fichiers définis par la contrainte. Par exemple, dans ce tutoriel, vous affectez le AdminRole à l'AdminConstraint et UserRole et AdminRole à l'UserConstraint. Cela signifie que les utilisateurs avec le AdminRole avoir accès aux deux fichiers Admin et les fichiers de l'utilisateur, mais les utilisateurs avec le UserRole avoir accès uniquement aux fichiers de l'utilisateur.

Vous configurez la méthode de connexion pour l'application en configurant web.xml. Le fichier web.xml peut être trouvé dans le répertoire des fichiers de configuration de la fenêtre des projets.

**3.2.3.1 - Basic Login**

Lorsque vous utilisez la configuration de “basic login”, la fenêtre de connexion est fournie par le navigateur. Un nom valide d'utilisateur et un mot de passe sont nécessaires pour accéder au contenu sécurisé.

Les étapes suivantes montrent comment configurer une connexion de base pour le GlassFish.

Pour configurer **basic login**:

1. Dans la fenêtre Projects, ouvrez fichiers le nœud Configuration Files et double-cliquez sur le fichier **web.xml** du projet. Le fichier web.xml ouvre dans l'éditeur

2. Cliquez sur Security dans la barre d'outils pour ouvrir le fichier en mode de sécurité.

3. Ouvrez le nœud “Login Configuration” et choisir “Basic” pour “Login Configuration”.

4. Entrez un nom pour “realm”, en se basant sur la configuration sur serveur

GlassFish: Entrez “file” comme Nom de “realm”. Ceci est le nom de domaine par défaut où vous avez créé les utilisateurs sur le serveur GlassFish.

5. Ouvrez le nœud “Security Roles” et cliquez sur “Add” pour ajouter un nom de rôle.

Ajouter les “Security Roles” suivants:

6. AdminRole : Les utilisateurs ajoutés à ce rôle auront accès au répertoire **SecureAdmin** du serveur.

**UserRole** : Les utilisateurs ajoutés à ce rôle auront accès au répertoire **secureUser** du serveur.

7. Créer et configurer une contrainte de sécurité nommé **AdminConstraint** en procédant comme suit: Cliquez sur Ajouter contrainte de sécurité. Une section d'une nouvelle contrainte de sécurité apparaît. Entrez **AdminConstraint** pour le Nom d'affichage de la nouvelle contrainte de sécurité. Cliquez sur “**Add**”. La boîte de dialogue “**Add Web Resource**” s’ouvre.

Dans la boîte de dialogue “Add Web Resource”, définissez le nom de la resource (Resource Name) à “**Admin”** et la URL Pattern a **/ SecureAdmin / \*** et cliquez sur OK. Le dialogue se ferme. Sélectionnez “Enable Authentication Constraint” et cliquez sur “Edit”. La boîte de dialogue “Edit Role Names” s’ouvre. Dans la boîte de dialogue “Edit Role Names”, sélectionnez **AdminRole** dans le volet gauche, cliquez sur “Add”, puis cliquez sur OK.

**3.2.3.2 - Form Login**

L'utilisation d'une forme pour le Login vous permet de personnaliser le contenu des pages de Login et Error. Les étapes de configuration de l'authentification en utilisant une forme sont les mêmes que pour la configuration de Basic Login, sauf que vous devez spécifier les pages de connexion et d'erreur que vous avez créés.

Les étapes suivantes montrent comment configurer une Forme Login :

1. Dans la fenêtre des projets, double-cliquez sur le fichier **web.xml** situé dans le répertoire “Web Pages/WEB-INF ” pour ouvrir le fichier dans l'éditeur

2. Cliquez sur “Security” dans la barre d'outils pour ouvrir le fichier en “Security view” et ouvrez le nœud “Login Configuration”

3. Définissez la “Login Configuration” à “Form”.

4. définir le “Form Login Page” en cliquant sur “Browse” et localiser **login.jsp**

5. Définissez la “Form Error Page” en cliquant sur Browse et localisez **loginError.html**

6. Saisissez un nom de “Realm”, en fonction de votre serveur.

GlassFish: Entrez “file” comme Nom de “Realm”. Ceci est le nom de domaine par défaut où vous avez créé les utilisateurs sur le serveur GlassFish.

7. Développez le nœud “Security Roles” et cliquez sur “Add” pour ajouter un nom de rôle.

8. Ajouter les rôles de sécurité suivants:

|  |  |
| --- | --- |
| **Server role** | **Description** |
| AdminRole | Users added to this role have access to the secureAdmin directory of the server. |
| UserRole | Users added to this role have access to the secureUser directory of the server. |

9. Créez et configurez une contrainte de sécurité nommé **AdminConstraint** en procédant comme suit:

Cliquez sur Ajouter contrainte de sécurité pour créer une nouvelle contrainte de sécurité. Entrez **AdminConstraint** pour le Nom d'affichage (Display Name) de la nouvelle contrainte de sécurité. Cliquez sur Ajouter pour ajouter une collection de ressources Web. Dans la boîte de dialogue “Add Web Resource”, définissez le nom de la ressource à **Admin**  et l’URL Pattern” a **/ SecureAdmin / \*** et cliquez sur OK. Sélectionnez “Enable Authentication Constraint” et cliquez sur “Edit”. La boîte de dialogue “Edit Role Names” ouvre. Dans la boîte de dialogue “Edit Role Names”, sélectionnez **AdminRole** dans le volet de gauche, cliquez sur Ajouter, puis cliquez sur OK.

**3.2.4 - Configuration de descripteurs de déploiement du serveur**

Si vous déployez votre application sur un serveur GlassFish, vous devez configurer le descripteur de déploiement du serveur pour mapper les rôles de sécurité définis dans web.xml. Le descripteur de déploiement du serveur est répertorié sous le nœud des fichiers de configuration de votre projet dans la fenêtre Projects.

**3.2.4.1 - Configuration du descripteur de déploiement serveur GlassFish**

Le GlassFish déploiement de serveur descripteur est nommé GlassFish-web.xml. Le descripteur de déploiement du serveur est dans le dossier des fichiers de configuration. Si elle n'y est pas, créez-la en cliquant droit sur le noeud du projet et d'aller au Nouveau> Autre> GlassFish> GlassFish descripteur de déploiement. Acceptez toutes les valeurs par défaut.

Notez que les valeurs que vous avez inscrits dans le fichier web.xml sont affichés dans GlassFish-web.xml. L'IDE tire ces valeurs de web.xml pour vous.

Pour configurer le descripteur de déploiement GlassFish:

1. Dans la fenêtre des projets, développez le nœud des fichiers de configuration du projet et double-cliquez sur GlassFish-web.xml. Le descripteur de déploiement GlassFish-web.xml ouvre dans un éditeur à onglets spécial pour GlassFish descripteurs de déploiement.

2. Sélectionnez l'onglet Sécurité pour révéler les rôles de sécurité.

3. Sélectionnez le nœud du rôle de sécurité AdminRole pour ouvrir le volet d'assignation de rôles de sécurité.

4. Cliquez sur Ajouter principal et entrez admin pour le nom principal. Cliquez sur OK.

5. Sélectionnez le nœud du rôle de sécurité UserRole pour ouvrir le volet d'assignation de rôles de sécurité.

6. Cliquez sur Ajouter et entrez principal utilisateur pour le nom principal. Cliquez sur OK

7. Enregistrez vos modifications GlassFish web.xml.

Vous pouvez également afficher et modifier GlassFish-web.xml dans l'éditeur XML en cliquant sur l'onglet XML. Si vous ouvrez GlassFish-web.xml dans l'éditeur XML, vous pouvez voir que GlassFish-web.xml dispose de l'information de correspondance de rôles de sécurité suivant:

<security-role-mapping>

<role-name>AdminRole</role-name>

<principal-name>admin</principal-name>

</security-role-mapping>

<security-role-mapping>

<role-name>UserRole</role-name>

<principal-name>user</principal-name>

</security-role-mapping>

Par défaut, le projet a été créé avec la fonction Compile sur Enregistrer permis, de sorte que vous ne devez compiler votre code en premier afin d'exécuter l'application dans l'IDE.