Entwurfsbeschreibung-OLAT

von Vladislava Nadova

1. Allgemeines

OLAT basiert auf einem Rechtesystem mit mehreren Hierarchien. Der Benutzer hat jederzeit genau diejenigen Funktionen zur Verfügung, zu denen er eine dem gegebenen Kontext entsprechende Berechtigung hat. OLAT hat vier grundlegende Systemrollen, denen eine Anzahl von Rechten zugeordnet ist: Gäste, Benutzer, Autoren und Administratoren.
Das Rechte- und Rollensystem, orientiert sich stark am Policy-Konzept von Java. Es gibt Identities (OLAT-Benutzer), Securitygroups, die Identities zwecks Rechteverwaltung zusammenfassen, Rechte, Ressourcen und Policies.

2. Produktübersicht

<table>
<tr><td colspan="3">Präsentationsschicht</td></tr>
<tr><td>Persönlicher Bereich</td><td>Kollaborativer Bereich</td><td>Öffentlicher und administrativer Bereich</td></tr>
<tr><td colspan="3">System der Rechte und Verantwortlichkeiten</td></tr>
<tr><td colspan="3">Basisdienste – Ressourcenschicht</td></tr>
<tr><td colspan="3">Daten- und Kommunikationsschicht</td></tr>
</table>

Das Rechte-und Rollensystem baut auf der Basisdienste-Ressourcenschicht auf und nutzt die von ihr zur Verfügung gestellten Dienste. Es wird über Policies

implementiert, die eine Kombination von einer Gruppe, einem Recht und einer Ressource darstellen.
Ein Benutzer hat dann ein gewisses Recht (z.B. lesen) auf eine bestimmte Ressource (z.B. RepositoryEntry), wenn er in mindestens einer Gruppe ist, die von einer Policy referenziert wird. Diese Policy hat die Rechte „lesen“ für die Ressource, d.h. diese Policy hat einen Fremdschlüssel auf die Gruppe, einen Fremdschlüssel auf die Ressource und einen Text für das Recht. So wird die separate Zuordnung eines Rechtes zu jedem einzelnen Benutzer gespart und mehreren, die dieses Recht benutzen dürfen, zugewiesen.
In Securitygroups werden Identitys zusammengefasst, die ein bestimmtes Recht auf eine bestimmt Ressource haben.

3. Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem

OLAT setzt auf der Java 2 Enterprise Edition (J2EE) auf und verwendet eine Servlet-Architektur. Mit dem Model-View-Controller (MVC) Framework wird eine strikte Trennung zwischen Businesslogik, Ablauflogik, Darstellung und Datenhaltung erlaubt, die die schnelle und übersichtliche Entwicklung, die Wiederverwendbarkeit der Komponenten und eine leichte Erweiterbarkeit ermöglicht.

OLAT ist komponentenbasiert und kann mit Hilfe so genannter „Extension-Points erweitert werden, ohne dabei Änderungen am Hauptsystem durchzuführen. Dieses Erweiterungskonzept kann auch bei dem Rechte- und Rollen-Management eingesetzt werden, wobei es zu beachten ist, wie von den Erweiterungspunkten auf das Rechte- und Rollenmanagement zugegriffen wird.

3.1 Zugriff von Erweiterungspunkten auf das Rechte- und Rollenkonzept

Wir studieren dazu die Demoerweiterung im Paket ch.goodsolutions.demoextension, deren Quellen sich im Verzeichnis webapps/WEBINF/src befinden:

Die Klasse DemoExtension implementiert das Interface Extension. Dabei sucht die Methode getExtensionFor() unsere Extension, die in olat_extension.xml angegeben ist. Die Extensionpunkte *org.olat.home.HomeMainController*

und *org.olat.gui.control.generic.DTabs* werden benutzt, um den neuen DemoExtension-Tab zu erstellen. Dabei wird ein entsprechendes Controller-Objekt von der Klasse DemoController erzeugt. Die Methode *createController (UserRequest ureg, WindowControl wControl, java.lang.Object arg)* wird vom Interface *ActionExtension* in der Extension-klasse benutzt. Wenn eine DTabs erweitert werden soll, wird auch eine Methode *createController(UserRequest ureq, WindowControl wControl)* beim Erzeugen einer SiteInstance benutzt.

Der createController()-Methode wird ein Objekt der Klasse UserResquest als Parameter übergeben. Durch dieses Objekt kann auf alle relevanten Nutzerdaten zugegriffen werden. Über die Methode getUserSession() in dieser Klasse wird ein neues UserSession-Objekt erzeugt, mit dem wir die Rollen durch *UserSession.getRoles()* auslesen können. Mittels *getRoles()* kann die Methode *createController()* entsprechend der Benutzeranfrage folgende *Controller*-Objekte zurückgeben:

- *BusinessGroupMainRunController*
- *BGMainController*
- *LaunchController*
- *GuestHomeMainController*
- *RepositoryMainController*
- *SystemAdminMainController*
- *UserAdminMainController*
- *BGContextManagementController*
- *HomeMainController*

So wird der Zugriff von einem Erweiterungspunkt auf Rechte- und Rollen-Management organisiert.

4. Grundsätzliche Strukturen- und Entwurfsprinzipien der einzelnen Pakete

4.1 org.olat.core.id

Das Paket org.olat.core.id wird bei der Verwaltung von Nutzern und Rollen eingesetzt. Wichtig dabei sind die Funktionen zur eindeutigen Identifikation von Benutzern, deren Rollen und entsprechenden Ressourcen.

4.1.1 Interfaces:

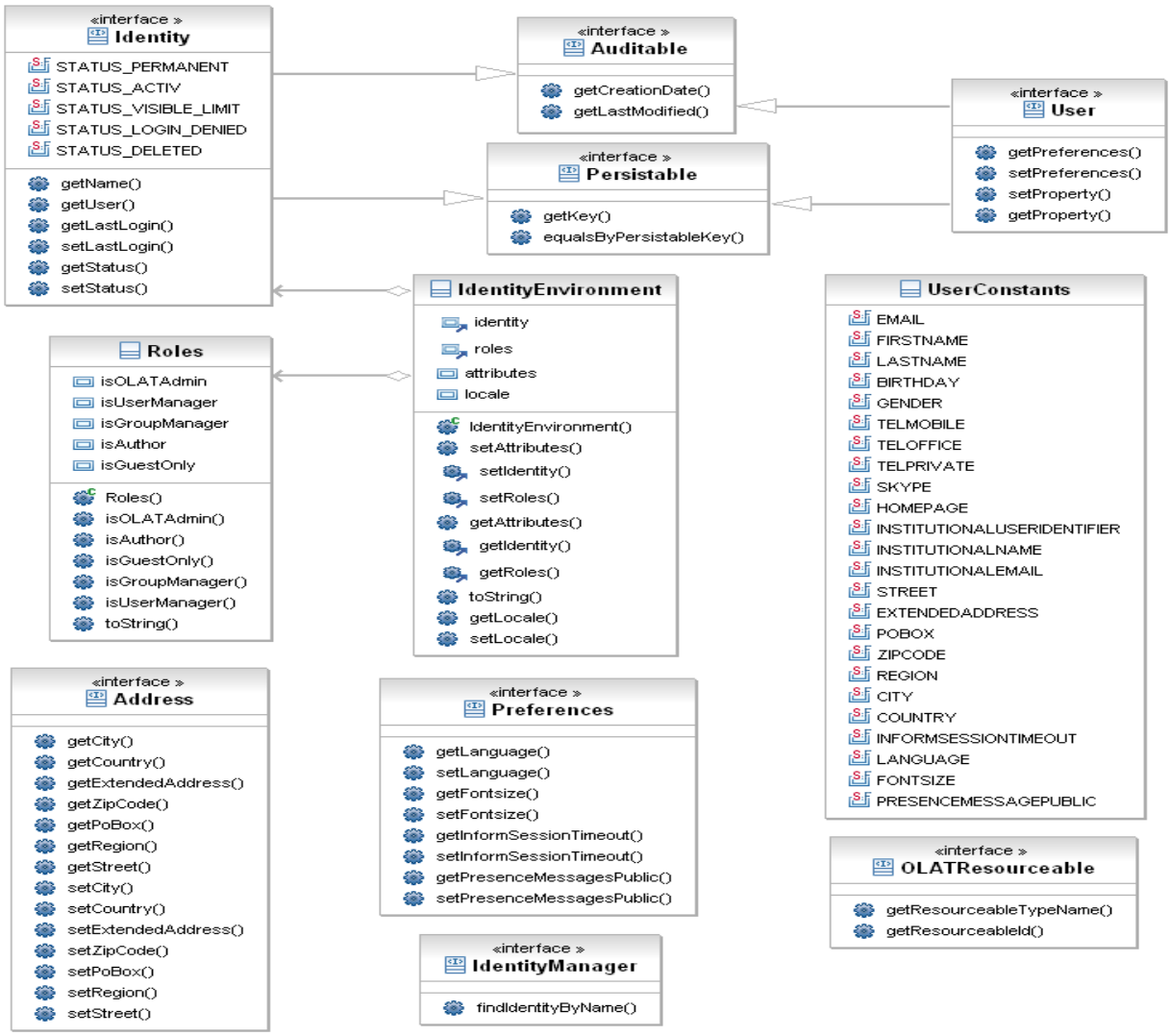
Address.java: Interface für eine internationale Adresse. Es ist ein Bestandteil von OLAT-User.
Auditable.java: (prüffähig) wird durch das persistable Objekt implementiert. Es liefert einheitliche Methoden für Abruf von creation date und last modified Parameter.
Identity.java: Ist mit jedem OLAT-Nutzer verbunden. Es definiert eine Identifikation des Nutzers. Das Interface enthält eine Schnittstelle zum letzen Logindatum und zum Status (aktiv, deleted, permanent). Diese Schnittstelle wird von Persistable.java und Auditable.java abgeleitet.
IdentityManager.java: findet eine bestimmte Identifikation und damit einen Nutzer für den gegebenen Nutzernamen. Das Ergebnis der Suche ist eindeutig.
OLATResourceable.java: ist ein eindeutiger Identifikator (Bezeichner) von Objekten in OLAT. Der eindeutige Identifikator ist durch den Name des Typs und ID gebildet.
Persistable.java: Das Interface ermöglicht einen Vergleich mit dem anderen persistable Objekt (User, Identity) auf der Datenbankebene. Es stellt die Integrität der Daten sicher.
Preferences.java: ist ein Bestandteil von OLAT-User. Es definiert Präferenzen des Nutzers, z.B. Sprache.
User.java: Dieses Interface implementiert ein Userobjekt, das einen angemeldeten Nutzer repräsentiert und dessen Privatdaten wie Namen, E-Mail-Adresse usw. hält. Folgende Elementen sind enthalten: a) Profil: eine Liste von Nutzereigenschaften und b) Preferences: eine Liste von Nutzereinstellungen.

4.1.2 Classes:

IdentityEnvironment.java: Verbindet eine Identifikation des Nutzers mit seiner Rolle im OLAT, Attributen und Einstellungen der Region.

Roles.java: Diese Klasse bestimmt, welche der im OLAT verfügbaren Rollen einem Nutzer zugewiesen ist (Admin, Usermanager, Groupmanager, Author, Guest). Hier werden die Rollenvariablen eingelegt und die Methoden isAuthor(), isAdmin(), isGroupManager, isUserManager() und isAnonymous definiert, die einen Booleanwert zurückgeben. Bei Auruf wird es quasi überprüft, ob der Nutzer diese entsprechenden Rollen hält.

UserConstants.java: Konstanten zum Richten von Nutzereigenschaften und Nutzereinstellungen .



4.2 org.olat.basesecurity

Dieses Paket beinhaltet Basisfunktionalität zur Umsetzung des Rechte- und Rollensystems von OLAT. Die beschriebenen in Punkt 1. Identities stellen eine Instanz eines OLAT-Benutzers dar.

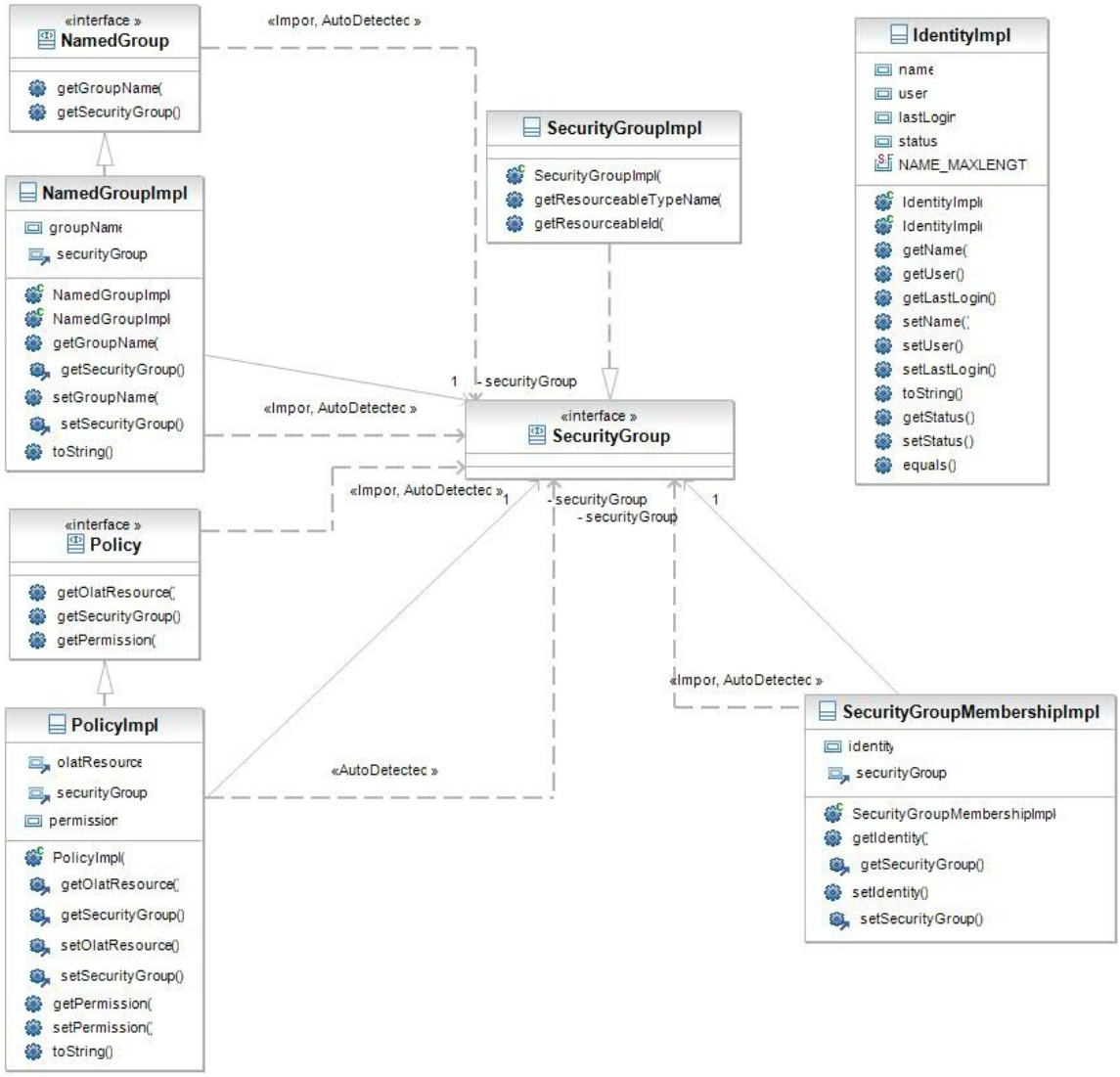
IdentityImpl.java:
Die Klasse implementiert org.olat.core.id.Identity.

IdentityManagerImpl.java:
Stellt eine Funktion zum Suchen von Identities bereit.

Securitygroups.java:
(SecurityGroupImpl) fassen Identities zu einer Gruppe zusammen. Somit sind etwa alle Admins in einer Securitygroup.

NamedGroupImpl.java:
Verbindet eine Securitygroup mit einem Namen (etwa „admins“ für die Admin-Gruppe).

SecurityGroupMembershipImpl.java:
Befindet sich eine Identity in einer Securitygroup, so ist dies in einer Instanz von SecurityGroupMembershipImpl gespeichert.



Constants.java:

Das gesamte OLAT-Rechte- und Rollensystem wird über die Policies implementiert (PolicyImpl). Ein Recht in einer Policy ist dabei nichts Weiteres als ein String mit einer vorgegebenen Bedeutung.
Constants stellt dabei derartige String-Konstanten, also Rechte und Gruppennamen, als auch resourceable-Typen (OLATResourceable) bereit.

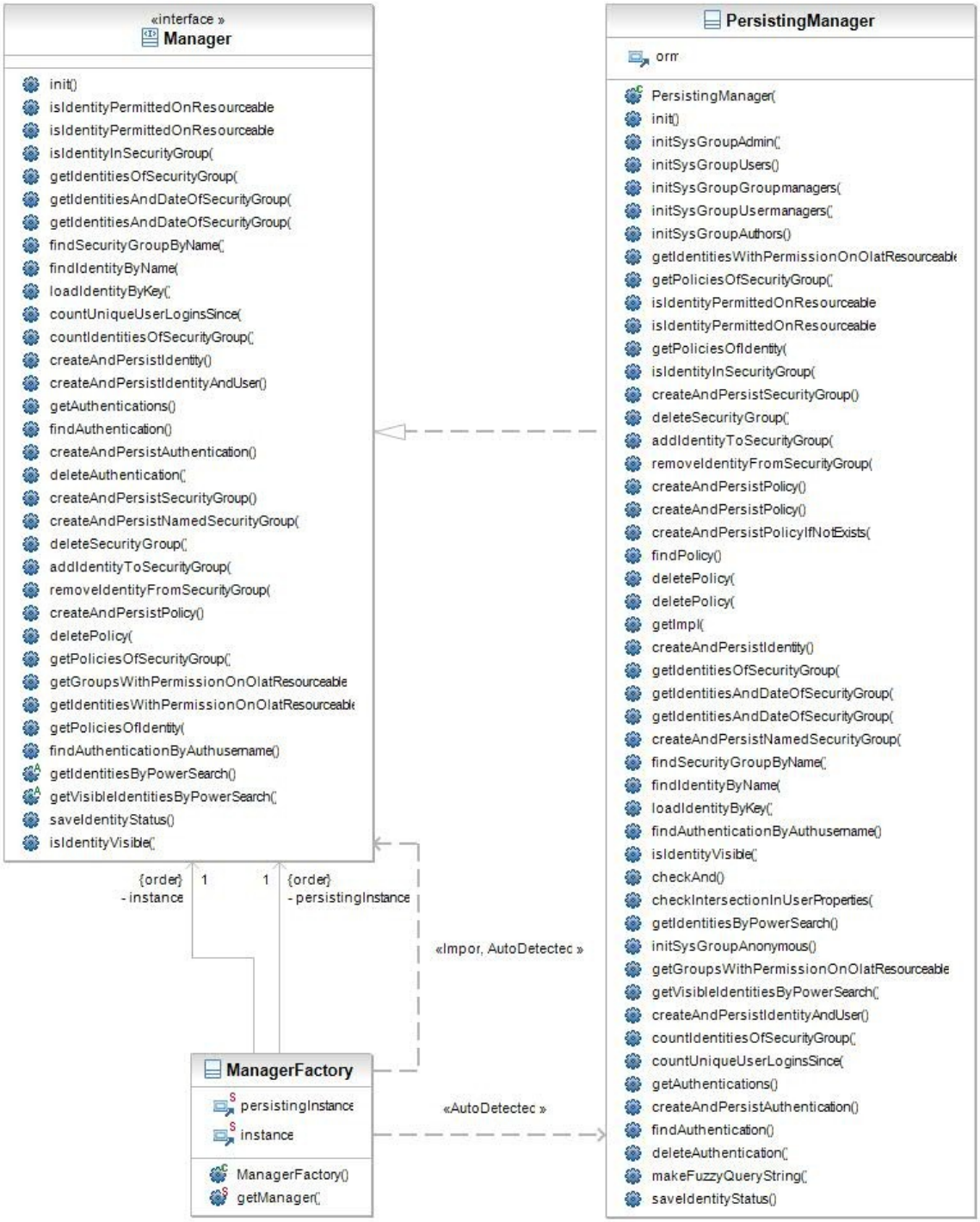
PermissionOnResourceable.java:

Ist ein Container, der eine Ressource und ein Recht beinhaltet. Er kann bei der Suche nach einem Benutzer oder einer Policy genutzt werden.

PersistingManager.java:

Stellt Funktionen zum Verwalten von Benutzern, Securitygroups und

Policies bereit, insbesondere das Finden, Erzeugen und Löschen solcher Objekte in der Datenbank; Singleton, auf das über ManagerFactory zugegriffen werden kann.



BaseSecurityModule:

Wird beim Systemstart vom ConfigurationManager geladen und initialisiert, mit Hilfe des PersistingManager, das Rechtesystem

4.3 org.olat.admin.user

SystemRolesAndRightsController.java:

Hier werden die Systemrollen und Rechte der Benutzer manipuliert. Mit Hilfe dieser Klasse können die Systemrollen und Rechte bearbeitet werden und eine neue SystemRolesAndRightsForm für die entsprechende Identity initialisiert werden.

UserCreateController.java:

Ein Controller, der die verschiedenen Formen für Anlegen neuer Nutzer auf der OLAT-Plattform.

UserAdminController.java:

Die Klasse überprüft, ob dem User erlaubt ist, eine andere Identity zu modifizieren. Nur Veränderungen bei Nutzern mit hierarchisch weniger Rechten ist erlaubt. Der Administrator nur darf Gruppenverwalter und andere Admins verändern.