Entwurfsbeschreibung - OLAT

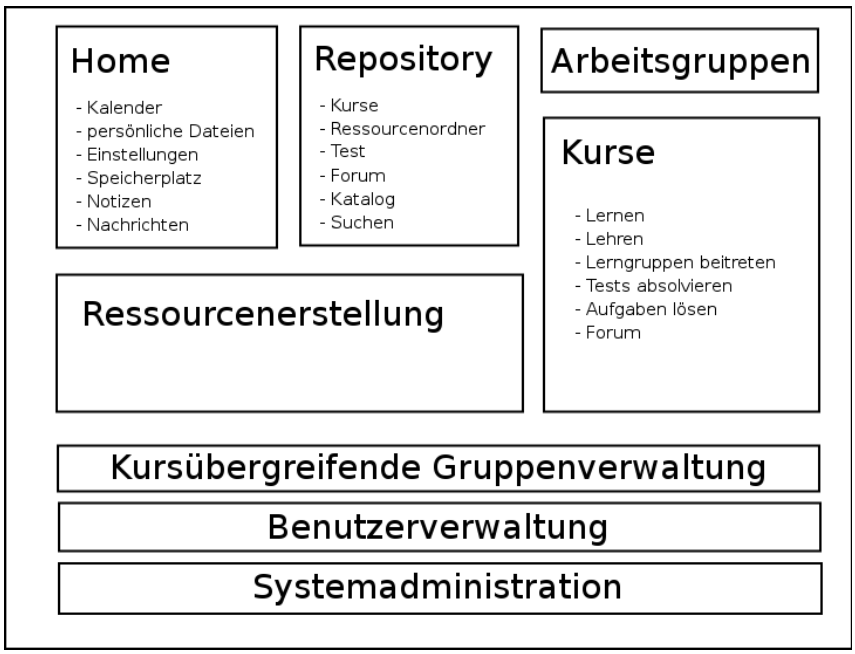
Inhaltsverzeichnis

1 Allgemeines

Das OLAT (Online Learning And Teaching) ist ein webbasiertes Learning Management System (LMS) mit einer großen Anzahl vielseitiger Funktionen, um einem Campusmanagement mit weitem Einsatz und einem breitem Anforderungsspektrum genügen zu können.
Es beinhaltet neben einem flexiblen Kurssystem auch kursunabhängige und kursübergreifende Funktionen. Hierzu zählen insbesondere auch eine allgemeine Verwaltung von Lernressourcen inklusive Bereitstellung von Editorenwerkzeugen für Tests, Fragebögen oder Kurse.
Das LMS OLAT ist eine Servlet-Anwendung und braucht somit einen Servlet-Container (z.B. Tomcat), in welches es eingebunden werden kann. Weiterhin benutzt es verschiedene "opensource frameworks" (z.B. velocity, hibernate) zum Erfüllen seiner Aufgaben und ist in der Programmiersprache Java geschrieben.
Zur persistenden Datenspeicherung kann eine Datenbank (MySQL oder PostgreSQL), ein Dateisystem oder eine Kombination aus Beiden genutzt werden.
Ein eigens entwickeltes Model-View-Controller (MVC) Framework erlaubt eine moderne, fehlerreduzierte und schnelle Entwicklung mit strikter Trennung zwischen Darstellungslogik, Ablauflogik, Businesslogik und der Datenhaltung. OLAT unterstützt die Erweiterung des Systems und stellt dazu einfache Mechanismen bereit.

2 Produktübersicht

Eine schematische Übersicht der wichtigsten Systemfunktionen:



Das OLAT unterscheidet verschiedene systeminterne Rollen: Gast, Benutzer, Autor und Administrator.
Je nach Rolle eines Nutzers, besitzt er verschiedene Rechte und kann auf verschiedene Funktionen und nur auf bestimmte, für die Rolle freigegebene, Ressourcen zugreifen.

Gast:
Ein Gast ist nicht registriert und ist in den Funktionen sehr eingeschränkt. Er kann weder an Tests teilnehmen, noch Forums- oder Wiki-Beiträge verfassen.

Benutzer:
Jeder registrierte Nutzer ist automatisch der Rolle Benutzer zugeordnet. Er kann sich in Gruppen einschreiben und an Tests teilnehmen. Je nach Gruppe stehen ihm dann weitere Funktionen oder Rechte zur Verfügung. Ein Benutzer kann eine Arbeitsgruppe anlegen und diese verwalten.

Autor:
Ein Autor hat zusätzlich zu den Benutzerrechten noch das Recht Lernressourcen zu erstellen und zu verwalten. Der Erstellter einer Ressource ist immer auch der Eigentümer dieser und hat volle administrative Rechte innerhalb der Ressource. So kann er auch anderen Benutzer zu Eigentümern machen, welche dann für diese Ressource die selben Rechte hat, wie der Ersteller.

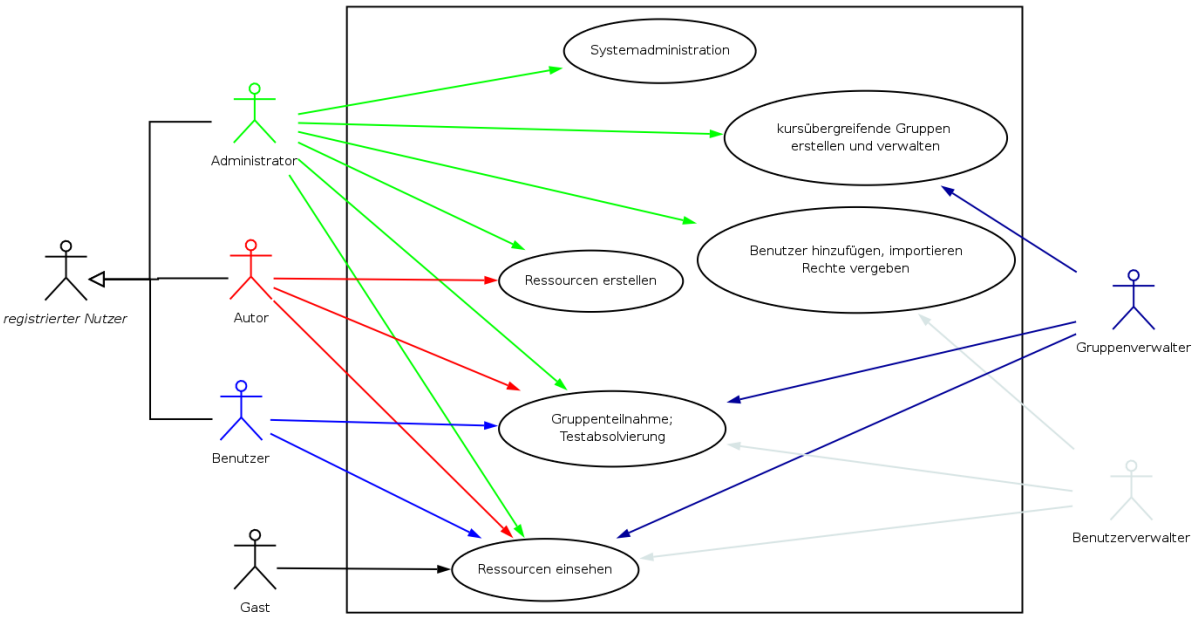
Ein Benutzer oder ein Autor kann zusätzlich noch in die Gruppe der Gruppenverwalter oder in die Gruppe der Benutzerverwalter aufgenommen werden. Daraus ergeben sich weitere Recht.

Gruppenverwalter:
Dieser kann kursübergreifende Lern- oder Rechtegruppemn erstellen und verwalten.

Benutzerverwalter:
Der Benutzerverwalter kann Benutzer anlegen oder importieren (über eine Tabelle mit Daten). Er kann weiterhin schon bestehenden Benutzern weitere Rechte geben.

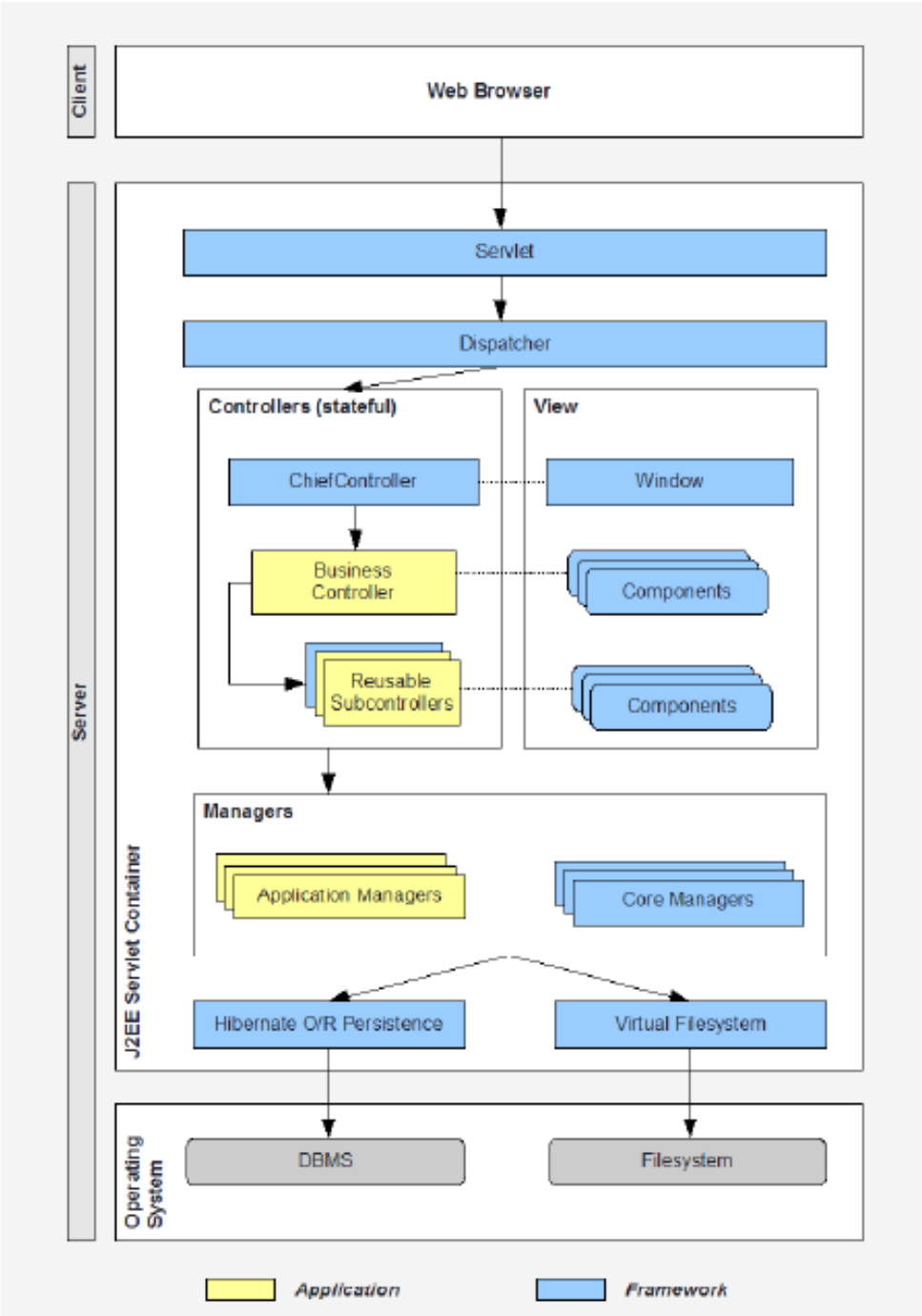
Administrator:
Der Administrator hat alle Rechte eines Benutzers, Autors, Gruppen- und Benutzerverwalters. Weiterhin hat er weitere technisch administrative Funktionen zur Verfügung, welche ihm die Verwaltung des Systems ermöglichen.

Mit dem folgendem Use-Case-Diagramm soll eine Übersicht für die, teilweise hierarchisch angeordneten, Funktionen des OLAT gegeben werden. Dieses Diagramm ist bei weitem nicht vollständig, soll aber nur der Übersicht über die Funktionalitäten des Systems dienen und dabei die unterschiedlichen Rechte (und damit verbundenen Funktionen) verdeutlichen.



Die Rolle *registrierter Nutzer* ist abstrakt und soll den Unterschied zwischen einem registriertem Nutzer und einem Gast hervorheben.
Man beachte, dass Gruppenverwalter und Benutzerverwalter nur besondere Rechtegruppen sind, zu welchen ein Benutzer zugeordnet werden kann. Ein Gruppen- oder Benutzerverwalten ist somit auch immer ein registrierter Nutzer. Weiterhin können diese auch Autorenrechte besitzen. Beides ist, wegen der besseren Übersichtlichkeit, nicht im Diagramm berücksichtigt.

3 Grundsätzliche Strutkur- und Entwurfsprinzipien für das Gesamtsystem



3.1 Architekturübersicht

Das LMS OLAT hat eine klassische 3-Schichten-Architektur und besteht somit aus einer Clientschicht und, auf der Serverseite, aus einer Anwendungs- und einer Datenschicht.

3.1.1 Das User Tier

Der Zugriff auf OLAT erfolgt über einen WebBrowser auf der Clientseite. Dabei kann auf verschiedene Protokolle zurückgegriffen werden: HTTP, HTTPS (verschlüsselt), als Netzwerkverbindung über WEBDAV oder über RSS.

3.1.2 Das Business Tier

Hier wird die Prozess- und Anwendungslogik gesteuert. Die Benutzereingaben werden validiert und die Benutzersitzung wird gesteuert. Daten werden hier an die Datenschicht weitergegeben und die HTML-Antwortseite wird erstellt und an die Userschicht gesendet.
Um eine saubere Trennung dieser verschiedenen Aufgabe zu ermöglichen, werden innerhalb der Schicht Layers verwendet. Folgende Layers können dabei unterschieden werden:

- OLAT Servlet / Dispatcher
  Initialisierung des OLAT und Weiterleiten von Nutzeranfragen.
- Windows/ChiefController
  Bereitstellung des Inhaltes für Webbrowser und Navigation.
- Controllers
  Verantwortlich für einen speziellen Workflow.
- Managers
  Stellt Basisfunktionen bereit.
- VFS (Virtual File System) und Hibernate

3.1.3 Das Data Tier

Bei OLAT wird in der Anwendungsschicht die Datenschicht mittels Hibernate für die Datenbank(en), und mittels VFS (Virtual File System) für das Dateiensystem, abstrahiert. In der Datenschicht befinden sich die Datenbanken und das Dateiensystem.

3.2 Eingesetzte Frameworks und Technologien

3.2.1 Apache Velocity

Der Java-Template-Engine Velocity bietet die Grundlage der View-Komponente des Systems. Mittels der Velocity Template Language (VTL), können in eine eigentliche statische HTML-Seite dynamische Textersetzungen durchgeführt werden. Dies geschiet über ein Velocity-Servlet, welches die HTML-Ausgabe generiert.

3.2.2 Apache Commons

Apache Commons stellt eine ganze Reihe nützliche und wiederverwendbare Java-Bibliotheken und -komponenten zur Verfügung.

3.2.3 Hibernate

Das Hibernate-Framework stellt Funktionen bereit, welche das persitente Speichern von Objektern in einer Datenbank ermögichen. Der Programmierer muss dabei keine SQL-Abfragen erstellen, sondern arbeitet mit den Objekten von Hibernate. Dies wird auch als Object-Relational-Mapping bezeichnet.

3.2.4 Spring

Unter Verwendung eines leichtgewichtigem Containers wird ein konsistenter Mechanismus zur Verfügung gestellt um Applikationen in eine J2EE-Umgebung zu integrieren. Mit leichtem Container wird dabei die Fähigkeit eines Containers gemeint, Applikationscode in verschiedenen Umgebungen zu verwalten, ohne das dafür spezielle Abhängigkeiten im Code bzgl. der Container-API erforderlich sind.

3.2.5 Xalan, Xerces, JDom

Bibliotheken für den Umgang mit XML-Dateien.

3.2.6 Lucene

Mit Hilfe dieser Bibliothek lassen sich Volltextsuchen für beliebige Inhalte implementieren.

3.3 Entwurfsprinzipien

3.3.1 Komponenten basiert

Eine Webseite wird aus Komponenten zusammengesetzt und durch Traversierung des Komponentenbaumes anschliessend als HTML generiert und als HTML-Seite an den Webbrowser geschickt.
Die Komponenten sind einem Business-Controller zugeordnet, welcher die Ablauflogik eines momentanen Bildschirmausschnittes steuert.

3.3.2 Event-basiert

Wenn ein Nutzer auf eine Stelle im Webbrowser klickt, so wird diesem Klick immer genau eine Komponente zugeordnet, welche dieses Ereigniss verarbeitet und ein Event an dem Business-Controller, der der Komponente zugeordnet ist, weiterleitet.

3.3.3 wiederverwendbare GUI-Workflows/Buisiness-Controllers

Das OLAT-GUI Framework ermöglicht eine einfache Wiederverwendung von Buisiness-Controllers, welche die Workflows representieren. Der Ablauf eines Workflows sieht aus Sicht der GUI immer gleich aus.

3.3.4 Trennung zwischen Code, Übersetzung und HTML-Templates

Die meisten Packete enthalten neben den Code-Dateien die zwei Unterordner '_content' und '_i18n' und eventuell '_static' für statische Dateien, wie Bilder oder CSS-Dateien.
In '_content' befinden sich die HTML-Templates und in '_i18n' die Übersetzungsdateien für die verschiedenen Sprachen. Dadurch können neue Sprachen leicht eingebunden werden und die Templates leicht, durch einen Designer, verändert werden ohne sich in den Quelltext reinlesen zu müssen.

4 Grundsätzliche Strutkur- und Entwurfsprinzipien der einzelnen Pakete

4.1 org.olat.core.id.*

Das Packet org.olat.core.id spielt eine wichtige Rolle bei der Verwaltung vom Nutzer- und Rollenmanagment, denn es stellt wesentliche Funktionen für die eindeutige Identifikation von Benutzern, deren Rollen und Resourcen. Die wichtigsten Klassen und Schnittstellen in diesem Paket sind:

4.1.1 Auditable.java

Ist eine Schnittstelle, um für persistente Objekte ein einheitliches Format für die Darstellung von Erstellungs- und zuletzt editierter Zeit zu gewährleisten.

4.1.2 Persistable.java

Stellt die Integrität der Daten sicher.

4.1.3 User.java

Ist ein Interface. Ein Userobjekt, welches dieses Interface implementiert, repräsentiert einen bekannten und angemeldeten Benutzer und hält dessen Daten wie Name, Email etc. Jeder Session eines Benutzers wird ein solches Objekt zugeordnet, um jederzeit seine Daten zur Verfügung zu haben.

4.1.4 Roles.java

Ist eine Klasse die festhält, welchen der fünf in OLAT verfügbaren Rollen ein Benutzer angehört. Ein und derselbe Benutzer kann mehreren Rollen zugeordnet werden.

4.1.5 Identify.java

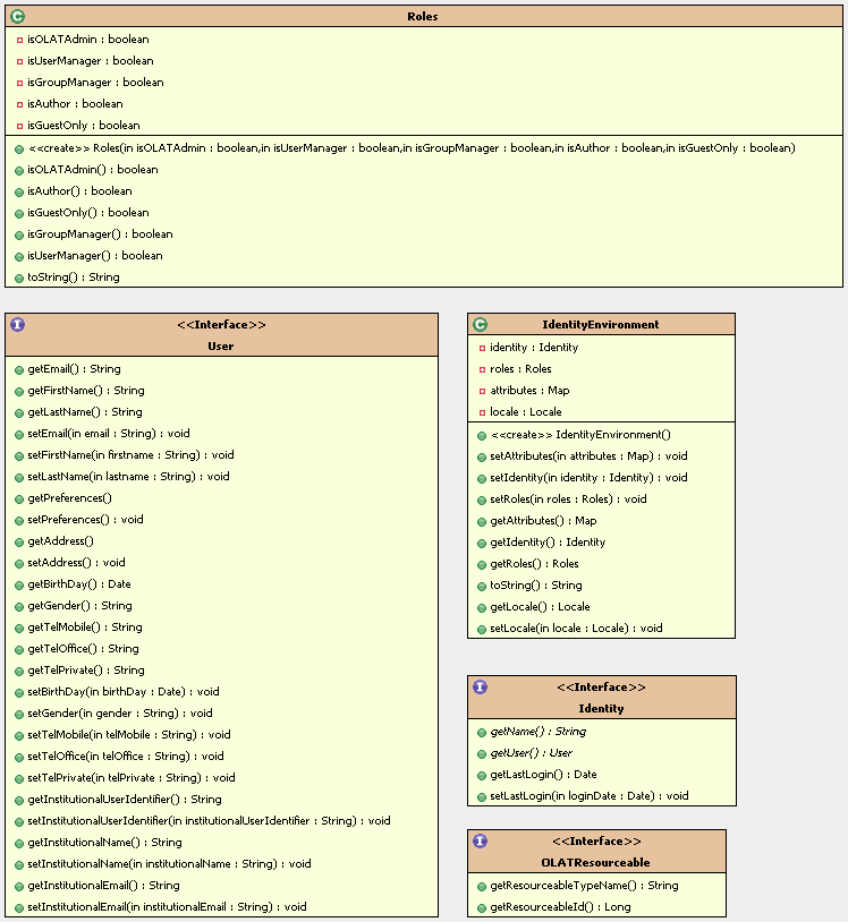
Ist eine von Persistable.java und auditable.java abgeleitete Schnittstelle. Gibt das mit ihr assozierte Userobjekt, nur dessen Namen und Informationen über letzte Änderungen, also unkritische Informationen aus.

4.1.6 IdentifyEnvironment.java

Ist ein Container für ein Identify-Objekt, eine Rolle, und weitere Attribute.

4.1.7 OLATResourceable.java

Dieses Interface ist ein eindeutiger Bezeichner für Objekte in OLAT, welcher den Typ der Klasse zurück gibt.



4.2 org.olat.group.*

BusinessGroups sind die Oberklasse aller Gruppen, verwendet wird ein BusinessGroupManager um alle Aktionen zu verwalten und Events auszulösen etc.

4.2.1 BusinessGroup

Interface, welches Oberklasse für alle BusinessGroups ist, wie BuddyGroups oder LearningGroups oder RightGroups. Definiert die grundsätzlichen Getter und Setter für alle relevanten Daten solcher Gruppen.

4.2.2 BusinessGroupAddResponse

Leitet 'fired Events' vom BusinessGroupManager weiter (also vor allem von addXXX-AndFireEvent).

4.2.3 BusinessGroupFactory

Dient der Erstellung neuer Instanzen von BuddyGroups, LearningGroups und Right-Groups. Dabei wird getestet, ob die Gruppe vielleicht schon exisitert. Weiterhin wird die neue Gruppe gleich persistent gemacht.

4.2.4 BusinessGroupImpl

Implementierung des BusinessGoup-Interfaces, eine solche Gruppe bekommt unter anderem die Attribute: Beschreibung, Name, Typ, minimale und maximale Teilnehmerzahl, Zeit des letzten Zugriffs, sowie 3 SecurityGroup-Objekte für Besitzer, Teilnehmer und Warteliste.

4.2.5 BusinessGroupManager

Interface, welche folgende Arbeitsabläufe beschreibt: Erstellen einer neuen Gruppen-Instanz, welche persistent ist und als Besitzer die <Identity> enthält, die die Gruppe erstellt hat. Weiterhin kann man Gruppen finden die einer bestimmten <Identity> zugeordnet sind. Geplant ist weiterhin, eine Gruppe zu aktualisieren und zu löschen.

4.2.6 BusinessGroupManagerImpl

Persistente Implementierung des BusinessGroupManager-Interfaces. Sichert die Daten in einer Datenbank. Ist eine Singleton-Klasse. Implementiert alle Funktionen des BusinessGroup-Interfaces.

4.2.7 BusinessGroupManagerImplTest

Implementiert org.olat.core.gui.control.WindowControl und erweitert *org.olat.core.test.OlatTestCase*. Initialisiert sich ein Test-Setup, also spezielle Anwendungsfälle und loggt dabei die Ausgaben/Fehler/etc bei der Benutzung der BusinessGroupManager-Implementierung.

4.2.8 BusinessGroupTest

Erweitert ebenfalls OlatTestCase um mit einem Test die Funktionalität der BusinessGroup-Implementierung zu testen und alle relevanten Informationen zu protokollieren.

4.2.9 GroupfoldersWebDAVProvider

Implementiert das Interface WebDAVProvider und bietet Methoden um Gruppeninformationen zu transferieren (von anderen Medien, HDDs etc.).

4.3 org.olat.group.right.*

Verarbeitet die Rechtegruppen (RightGroup).

4.3.1 BGRightManager

Ist ein Interface. Gibt die Methodenstümpfe für Abfragen vor, Finden, Setzen und Löschen von Rechten für Rechtegruppen (spezielle BusinessGroup).

4.3.2 BGRightManagerImpl

Implementiert BGRightManager als Singleton-Klasse, überschreibt dabei alle Methoden des Interfaces mit sinnvollem Inhalt.

4.3.3 BGRights

Ist ein Interface. Enthält einzig die Methoden .getRights():List zur Ausgabe der gespeichterten Rechte des Objektes, sowie transateRight(String):String, das zur Übersetzung (Translation) genutzt werden kann. Der BGRightManager nutzt derzeit allerdings dieses Interface für die Rechte nicht, sondern übergibt einfach Strings.

4.4 org.olat.resource.*

Dient der Verwaltung von OLAT-Ressourcen, also Klassen, die org.olat.core.id.OLATResourceable implementieren.

4.4.1 OLATResource

Leeres Interface zum Markieren von Ressourcen, implementiert OLATResourceable, Persistable und Auditable.

4.4.2 OLATResourceImpl

Erweitert PersistentObject und implementiert OLATResource. Wird als Singleton instanziert. Belegt dabei alle für Hibernate relevanten Methoden, sowie toString() und grundlegende Ressourcenmanagement-Methoden.

4.4.3 OLATResourceManager

Wird als Singleton instanziert. Dient zum verwalten von OLATResource-Objekten, diese können erstellt, modifiziert, gefunden und gelöscht werden.

4.4.4 OLATResourceManagerTest

Erweitert OlatTestCase und implementiert OLATResourceable um den Resourcen-Manager in einem Testfall zu überprüfen und die Ergebnisse zu loggen.