I. Begriffe

- **OLAT(Online Learning and Training)**
  OLAT ist ein webbasiertes Lernmanagementsystem. Es unterstützt Studierende und Lehrende als Kommunikations- und Lernplattform. OLAT wurde an der Uni Zürich entwickelt und ist eine plattformunabhängige Open-Source-Software, die für viele Zwecke gebraucht und weiterentwickelt werden kann.

- **Clustering**
  bezeichnet den Verbund mehrerer Rechner zur Erhöhung der Rechenkapazitäten oder der Verfügbarkeit gegenüber einem einzelnen Rechner. Cluster werden von außen häufig als eine Maschine gesehen. Die einzelnen Rechner innerhalb eines Clusters bezeichnet man als Nodes.

- **Extension Point**
  OLAT verfügt mit seinem modularen Aufbau über mehrere Erweiterungspunkte, sogenannte Extension Points. So werden explizit Stellen definiert, an denen das System erweiterbar ist. Dies passiert häufig mittels einem XML-Schema (deklarativer Anteil der Erweiterung) und einem Java Interface, das die Erweiterung implementiert.

- **LMS (Learning Management System)**
  stellt Funktionalitäten zur Verfügung, mit denen sich Lehren und lernen managen lassen. (z. B. Verwaltung von Nutzern und Lerninhalten . Meistens ist ein LMS eine Webapplikation.

- **Milestone**
  (dt. Meilenstein) bezeichnet einen wichtigen Versionssprung bei der Entwicklung einer Computerapplikation, üblicherweise gekennzeichnet durch eine Erhöhung der Hauptversionsnummer. Mit einem Milestone gehen meist grundlegende Änderungen der Funktionalität einer Applikation einher.

- **Servlet**
  Als Servlets bezeichnet man Java-Klassen, deren Instanzen innerhalb eines Java-Webservers Anfragen von Clients entgegen nehmen und beantworten.
  Diese Antworten können sowohl statisch als auch dynamisch (z.B. eine Html-Seite) sein.

- **TOMCAT**
  Apache Tomcat stellt eine Umgebung zur Ausführung von Java-Code auf Webservern bereit. Es handelt sich um einen in Java geschriebenen Servlet-Container. OLAT ist vorrangig eine Servlet Applikation, die zum Betrieb einen Servlet Container braucht.

- **User (Benutzer)**
  In OLAT werden die folgenden Systemrollen unterschieden:
  Gäste: anonyme Benutzer mit eingeschränkten Rechten, können weder ihre Benutzeroberfläche anpassen noch Tests absolvieren noch Forumsbeiträge verfassen.
  Benutzer: können ihre Benutzeroberfläche selber gestalten, Arbeitsgruppen erstellen und einen Kurs als Teilnehmer starten.
  Autoren: Zusätzlich zu den Berechtigungen eines Benutzers können Autoren Lernressourcen herstellen, kopieren, archivieren, löschen etc.
  Gruppenverwalter: können zusätzlich zu den Rechten eines Benutzers Lern- und Rechtegruppen verwalten.

Benutzerverwalter: können neue Benutzer erstellen oder importieren und ihnen Rollen zuweisen.
Administratoren: administrative Tätigkeiten im gesamten OLATSystem., beinhaltet Rechte aller anderen Systemrollen

- **XML (Extensible Markup Language) / XML-Schema**
  XML ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Datensätze in Form von Textdaten. XML wird u. a. für den plattform- und implementationsunabhängigen Austausch von Datensätzen zwischen Computersystemen eingesetzt, insbesondere über das Internet.
  XML-Schema ist eine Empfehlung des W3C zum Definieren von Strukturen für XML-Dokumente.

- **URI (Universal/Uniform Ressource Identifier)**
  URIs sind Identifikatoren für Ressourcen. Ressourcen können hierbei jegliche Arten von Daten sein, am bekanntesten sind URL,s die zur Bezeichnung von Websites verwendet werden. Ihre einheitliche Syntax wurde vom W3C-Konsortium als Standard festgelegt. Eine URI gibt in verkürzter Form Aufschluss über die Zugriffsart (z.B. ein Netzwerkprotokoll) auf die Ressource, den zugehörigen Pfad und Bezeichner, sowie das Mutterobjekt.

- **SPARQL(SPARQL Protocol and** RDF **Query Language)**
  Eine Anfragesprache für RDF. Zu beachten ist hier, dass es sich bei RDF nicht um eine Datenbank, sondern um einen gerichteten Graphen handelt. Das W3C-Konsortium hat SPARQL zur empfohlenen Sprache für Anfragen an RDF erklärt.

II. Konzepte

- **OLAT Schichtenmodell**
  OLAT ist eine modular aufgebaute Web-Applikation. Im klassischen Sinne hat es eine 3-Tier-Architektur, geteilt in User Tier, Business Tier und Data Tier.
  Der User-Tier läuft auf dem Client im Webbrowser. Er enthält die sichtbare Applikation. Hier greift der User auf OLAT zu und ruft die verschiedenen Funktionen ab, die im GUI als Sites sichtbar sind.
  Der Business Tier enthält die allgemeine Ablaufsteuerung. Innerhalb des Tiers werden weitere Layer verwendet, um die Funktionen der Validierung, Businesslogik, Persistierung und HTML-Antwortaufbereitung voneinander zu trennen. Im Business Tier werden Benutzereingaben validiert, die Ablauflogik gesteuert, die HTML-Antwortseite wird aufbereitet und an den User Tier geschickt sowie die Daten an den Data Tier weitergegeben.
  Der Data Tier dient der Anbindung an das Dateisystem, die Datenbank oder andere Dienste wie Instant Messaging. Er enthält die Datenbank sowie das Filesystem. In der aktuellen OLAT Version wird mittels Hibernate der Zugriff auf die Datenbank, und mittels dem Virtual Filesystem VFS der Zugriff auf das Filesystem abstrahiert.
  Der User Tier läuft direkt auf dem Client. Business und Data Tier können auf einem oder mehreren Servern verteilt sein. Dies ermöglicht eine gute Skalierbarkeit.

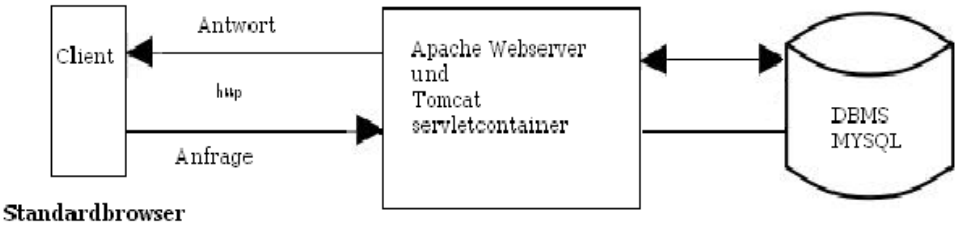
- **MVC**
  MVC ist ein Architekturmuster zur Strukturierung von Software-Entwicklung in die drei

Einheiten Datenmodell (Model), Präsentation (View) und Steuerung (Controller). Es ermöglicht einen flexiblen Programmentwurt und so eine gute Wiederverwendbarkeit der einzelnen Komponenten.
Das Model beinhaltet hierbei die internen Daten sowie Methoden zur Datenverarbeitung und Manipulation, unabhängig von Präsentation und Steuerung. Sie werden den anderen Einheiten mittels Schnittstellen zur Verfügung gestellt.
Der View dient zur Darstellung der Daten aus dem Modell. Für die Verarbeitung der vom Benutzer übergebenen Daten ist der View allerdings nicht zuständig.
Der Controller verwaltet die Darstellung im View und nimmt Benutzerinteraktionen entgegen. Sie werden ausgewertet und entsprechende ihrer Bedeutung dem Model zur Weiterverarbeitung übergeben. Dabei dient der Controller ebenfalls nicht der Datenmanipulation, sondern zur Steuerung, welche Daten im Model geändert oder berechnet werden müssen.
OLATs Applikationsschicht basiert auf einem selbstentwickelten MVC Framework, das weit über JSF oder Spring hinausgeht, dem Brasato-Framework. Model und View sind hier relativ eng, in einer Klasse org.olat.core.gui.components.Component verbunden. Diese dient als Schnittstelle zur Anmeldung von Observern und deren Benachrichtigung. Die Darstellung der Objekte selbst geschieht durch org.olat.core.gui.components.ComponentRenderer. Der Controller findet sich in org.olat.core.gui.control.Controller.

- **Open Source**
  Software, deren Quellcode frei verfügbar ist. Derartige Software kann von anderen Menschen leichter verbessert oder angepasst werden, wobei Klassifizierung als Open Source allein dies nicht zwangsläufig auch erlaubt. Dafür ist die Lizenz, unter der die Software veröffentlicht wurde, ausschlaggebend. Open source Software wird jedoch meist unter sogenannten freien Lizenzen, wie beispielsweise der general public license(GPL) veröffentlicht.

- **Hibernate**
  Das Hibernate-Framework ermöglicht es, Objekte in einer relationalen Datenbank zu speichern und umgekehrt aus Datensätzen einer relationalen Datenbank Objekte zu erzeugen. Im Gegensatz zu einer reinen Datenbankanfragesprache, die man in den Quellcode einbettet, bietet es aber diverse Vorteile.
  Zum einen ist es unabhängig von der verwendeten Datenbank. Solange ein JDBC-Treiber für die Datenbank existiert, kann Hibernate darauf angewendet werden. Auch der Verbindungsaufbau zur Datenbank ist in eine vorgefertigte Klasse gekapselt. Desweiteren orientiert sich Hibernate stark an den Konzepten der objektorientierten Programmierung. Datenbankoperationen können mittels HQL, der Anfragesprache von Hibernate, gestellt werden oder aber mit Hilfe der criteria-API, die sich syntaktisch nahtlos in Javaquellcode einfügt. Werden Datensätze gefunden, kann Hibernate automatisch Objekte generieren, auf die die Datensätze abgebildet werden. Umgekehrt kann Hibernate auch Objekte aus Java in die Datenbank übertragen. Beziehungen zwischen den Objekten werden dann als Relationen abgespeichert.
  In unserem Projekt werden wir aber nicht vollständig auf Hibernate zurückgreifen können, so wie dies beim OLAT der Fall ist, da die von uns anzufragende Datenquelle keine Datenbank, sondern ein gerichteter Graph (siehe RDF) ist. Allerdings existiert spezifisch für RDF ein Framework, welches sich in puncto Objektorientiertheit stark am Konzept von Hibernate orientiert: JRDF.

- **Client-Server-Prinzip**
  Das Client/Server-Konzept ist eine Programmarchitektur für verteilte Systeme in einem Netzwerk. Auch OLAT nutzt diese Architektur. Hierbei wird ein Programm in zwei Teile gespalten, den Client, der meist lediglich die Benutzeroberfläche beinhaltet und den Server, der die Geschäftslogik enthält und die Daten speichert (im Falle von OLAT als Apache Server mit Tomcat-Servlet-Container). OLAT nutzt Servlets für die Erstellung dynamischer Webseiten. Die Schnittstelle zwischen dem Client, einem Standard-Webbrowser, und dem Server ist das Übertragungsprotokoll HTTP. Der Server liefert Daten in HTML, Javascript und CSS. Zur persistenten Speicherung der Daten nutzt OLAT MySQL.



  Auf diese Art und Weise kann der Client auf mehreren Rechnern laufen und jeweils eine Verbindung zum Server herstellen, ohne dass Daten oder Programmteile mehrfach vorhanden sein müssen. Die Nutzer des Programms können über die Clients auf Daten zugreifen oder diese manipulieren ohne dass ein Abgleich zwischen mehreren Datensätzen stattfinden muss.

- **RDF**
  Das Resource Description Framework bezeichnet mehrere Standards des World Wide Web Consortiums (W3C) zur formalen Beschreibung von Informationen über Objekte. Ähnlich einem relationalen Datenbanksystem beschreiben jene Standards eine Syntax, mit welcher Informationen gespeichert werden können. Es wurde entwickelt, um die Konzepte des WWW (Verknüpfung, Offenheit, Heterogenität) auf sämtliche Daten zu übertragen.
  Das Modell von RDF ist das eines gerichteten Graphen. Jede Information wird in einem Tripel aus Subjekt (der betrachtete Gegenstand), Prädikat (Eigenschaft) und Objekt (Eigenschaftswert) gespeichert. Zur Identifizierung der einzelnen Teile des Tripels werden URIs (Uniform Ressource Identifier) genutzt. Das Objekt kann aber auch eine nicht nach URI-Standard formulierte Zeichenkette sein. Die Darstellung von RDF ist prinzipiell nicht festgelegt, in unserem Fall werden RDF-Graphen aber in XML-Dateien abgespeichert.

III. Beschreibung der Applikation OLAT

i) Allgemeine Beschreibung

OLAT ist ein unter der Apache2.0-Lizenz stehendes Learning Management System (LMS) und dient als serverseitige Web-Applikation der Organisation von Lehrveranstaltungen. Zu diesem Aufgabengebiet zählen u.a. die Bereitstellung einer Infrastruktur zur Verteilung von Lernressourcen und zur Kontrolle des individuellen Lernfortschrittes, womit der Lernbetrieb nebst Verwaltung vereinfacht werden kann.

OLAT (**O**nline **L**earning **a**nd **T**raining) bildet hierfür viele Elemente des modernen Internets auf den Lehrbetrieb der Universität ab. Hierzu gehören z.B.

- die Einbindung externer Quellen,
- Diskussionsforen,
- Wikis,
- Dateidiskussionen,
- Hausaufgabenverteilung und -abgabe,
- Tests und Selbsttests,
- Fragebögen,
- Chat (via XMPP).

Die Software ist ein Projekt der Universität Zürich und wurde dort zunächst als studentisches Projekt auf PHP-Basis entwickelt, ab 2001 jedoch federführend von der IT-Abteilung der Universität. 2004 erfolgte ein Wechsel auf eine komplett neue Java-Codebase und das System wurde vollständig OpenSource. Durch diesen Wechsel wurde die Grundlage für eine verbesserte Skalierbarkeit gelegt, die mit einem Milestone schließlich durch die Möglichkeit der Clusterung vollständig hergestellt wurde. Seitdem wird die Software, vorwiegend im europäischen Raum, vermehrt von Hochschulen zur Organisation ihrer Lehrveranstaltungen eingesetzt, da sie nun auch für den gesamt- oder interuniversitären Bereich geeignet ist und problemlos mehrere zehntausend Benutzer ermöglicht.

OLAT unterstützt außerdem Formatstandards wie IMS Content Packaging, IMS QTI und SCORM. Damit setzt das System auf die gebräuchlichsten offenen Formate zum Austausch von Lerninhalten und gewährleistet damit eine Interoperabilität mit anderen LMS und Content Management Systemen (CMS). Auch insgesamt setzt OLAT auf die verbreitetsten offenen Standards, um damit ein nicht-proprietäres System zum E-Learning zu etablieren.

OLAT läuft in einer Java-VM auf den gebräuchlichen Betriebssystemen, die Voraussetzungen sind vor allem:

- Java SDK
- Tomcat Servlet Engine
- MySQL oder PostgreSQL Database

Die Architektur ist grundsätzlich als Java Servlet aufgebaut. Ein eigens entwickeltes Model-View-Controller (MVC) Framework erlaubt hierbei eine strikte Trennung zwischen Datenhaltung, -darstellung und -verarbeitung (vgl. OLAT-Schichtenmodell). Bei der Entwicklung wird grundsätzlich auf eine hohe Modularisierung und Entkopplung der einzelnen Komponenten geachtet, um eine hohe Wiederverwendbarkeit zu gewährleisten und Fehlerquellen möglichst schnell auffindbar zu machen.
Durch diesen Aufbau und durch die Einführung von speziellen Extension Points lässt sich OLAT sehr leicht mittels sog. Extensions um weitere Funktionen erweitern, ohne dabei Änderungen am Grundsystem vornehmen zu müssen. Die Erweiterungen docken an bestehenden Schnittstellen an und fügen sich nahtlos in das LMS ein.
Die Plattform setzt auf die im Web üblichen Benutzerrollen Administrator, verschiedene Autorenarten und normale User. Auch ein Gastzugang ist prinzipiell möglich, kann aber dank umfassender Konfigurationsmöglichkeiten auch deaktiviert werden. Zudem werden mit OLAT bereits einige Analyse-Tools mitgeliefert, mit denen z.B. die Auslastung des Servers oder die Benutzeraktivität anschaulich gemacht werden können.

ii) Verzeichnisstruktur

Verzeichnis	Funktion
bin	Hilfsscripte (Shellscripte für Linux und MacOS)
cluster	Scripte zum Clustering (ab Version 6.1) von Olat, bessere Skalierbarkeit
conf	Konfigurationsdateien für Tomcat und Instant-Messaging
database	SQL-Datenbankvorlagen für Setup und zum Upgraden älterer Olat-Versionen
doc	Dokumentation
htdocs	Fehlercode-Seiten
interfaces	Dateien für Flash-Integration in OLAT QTI
monitoring	Vorlagen für die Traffic-Überwachung
olatdata	OLATdaten
+bcroot	Laufzeitdaten
++course	Kursdaten
++cts	Gruppenordner und Gruppenwerkzeuge
++ homepages	Webseiten der Nutzer
++ homes	persönliche Ordner der Nutzer
++ repository	angelegte Lernressourcen
++ tmp	Temporäre Dateien
+ calendars	Kalenderdaten
+ customizing	Anpassungen
+ logs	Logdaten
+ monitoring	Überwachungsdaten
+ system	installierte Upgrades
+ tmp	Temporäre Dateien
temp	Temporäre Dateien
webapp	OLAT-Webapplikation
+ examples	Beispieldateien (Demokurse)
+ help	OLAT-Hilfe
+ static	Layoutdateien (movie, themes, images)
+ WEB-INF	OLAT - Konfigurationsdateien
++ classes	OLAT-Java-Klassen
++ lib	Java-Bibliotheken, die OLAT verwendet
++ patches src	Quelldateien für Patches
++ src	OLAT-Quelldateien

Die **Verzeichnisse für Entwicklungsarbeiten** sind monitoring und webapp. Im Unterordner WEB-INF befindet sich die Datei olat_extensions.xml, welche ebenfalls für Entwicklungsarbeiten erforderlich ist.

Die **für den Betrieb einer Instanz erforderlichen Verzeichnisse** sind die Verzeichnisse in olatdata und der webapp-Ordner.

iii) Formen der modularen Erweiterung

Erweiterungen werden im OLAT über die sogenannten *Extension-Points* realisiert, welche den Plugins im Eclipse recht ähnlich sind. Dieses Verfahren bietet den Vorteil, dass der originale Quellcode und somit das Grundsystem unverändert bleiben.

Eine neue Erweiterung muss in der Datei *olat_extensions.xml* eingetragen werden, in der sie mittels der ID und des Klassen- bzw Packagenamens identifiziert werden kann. Ist die Erweiterung als .jar Paket in den Ordner WEB-INF/lib eingefügt, kann sie vom OLAT-Extension Manager geladen, und somit ins OLAT integriert werden.

OLAT verfügt über bereits fertig definierte Extensions, deren Funktionalitäten jetzt aufgeführt werden:

- Extension interface: org.olat.extensions.globalmapper
- Extension point: org.olat.dispatcher.DispatcherAction
- Klassen: org.olat.dispatcher.DispatcherAction
- Beschreibung: Die Extension kann einen *Mapper* erhalten und kennt den Pfad, mit dem er assoziiert wurde.

- Extension interface: org.olat.extensions.action.ActionExtension
- Extension point: org.olat.home.HomeMainController
- Beschreibung: Die Extension bietet einen Link mit Text und Beschreibung an und definiert dessen ActionEvents.

- Extension interface: org.olat.extensions.css.CSSIncluder
- Extension point: org.olat.gui.components.Window
- Klassen: org.olat.gui.css.CSSGenerator
- Beschreibung: Die Extension bietet den Pfad zum CSS-Stylesheet an, welches jedes OLAT Fenster nutzt.

- Extension interface: org.olat.extensions.hibernate.HibernateConfigurato
- Extension point: org.olat.persistence.DB
- Klassen: org.olat.persistence.DB
- Beschreibung: Die Extension kann zusätzliche hibernate mappings hinzufügen, wobei angemerkt sein, dass das Erstellen von Tabellen schwierig ist und zusätzliche SQL Skripte genutzt werden müssen.

- Extension interface: org.olat.extensions.sitescreator.SitesCreator
- Extension point: org.olat.gui.control.generic.dtabs.DTabs
- Klassen: org.olat.FullChiefController
- Beschreibung: Wenn die Extension eine neue Seite zu OLAT hinzufügen soll (einen neuen Tab neben “Home”, Groups” etc.) muss sie von SitesCreator erben und eine Liste von SiteDefinition Objekten zurückgeben, von denen jedes verantwortlich für die Erstellung einer SiteInstance ist.

- Extension interface: org.olat.login.SupportsAfterLoginInterceptor
- Extension point: oorg.olat.login.AfterLoginInterceptorController

- Klassen: org.olat.user.UserModule
- Beschreibung: Module, die unmittelbar nach dem Einloggen in Erscheinung treten sollen, können diesen Controller hinzufügen. Die Konfiguration wird von jedem Modul selbst verwaltet.

In der *olat_extensions.xml* finden wir verschiedene auskommentierte Demo-Extensions, die durch Editierung leicht aktiviert werden können und dann im OLAT Fenster erscheinen. Sie geben Beispiele zur Erstellung eines statischen Tabs ("Demonstration Site") oder neuer Einträge im Navigationsmenü des Home Tabs.

Es folgt eine Beschreibung der Menüpunkte des Tabs "gui_demo", die über das Extension interface *org.olat.extensions.action.ActionExtension* realisiert werden, damit sie auch im Navigationsbereich auf der linken Seite des OLAT Fensters erscheinen. Dieser Tab dient vor allem Demonstrationszwecken, aber auch zum Einlesen in den Quellcode, da dieser jeweils angezeigt werden kann. Im Einzelnen sind dies die folgenden Unterpunkte:

- GUI Demo
- Poll Demo
- Forms
- Flexi Forms
- Steps
- Panes
- Links & Buttons
- GUI SOA
- Window Control
- Tables
- FlexiTables
- Glossar and marker
- Tooltips based on Extjs library
- Floating panel based on Extjs library
- Errors
- Clipboard
- Dynamic CSS & JS
- Dialogs
- Ajax Tree
- File Chooser
- Bread crumb path
- Autocompletion

Nach diesen Definitionen folgt ein auskommentierter Block, der eine Vorlage (engl. Template) enthält um eine eigene Extension einzufügen, die ebenfalls über das Extension interface *org.olat.extensions.action.ActionExtension* realisiert wird, also auch einen Menüeintrag, z.B. in der GUI Demo, erzeugt.

Alles nächster Block folgen die Definitionen der statischen, also immer vorhandenen Tabs, die immer zu sehen und zu nutzen sind:

- Home

- Groups
- Learning resources
- Group administration
- User managment
- Administration
- gui_demo

iv) Erstellung und Einbindung einer Erweiterung als Tab

Das Grundprinzip des Einbindens einer Extension ist bereits im obigen Punkt erläutert, soll hier aber nochmals vertieft werden. Die .jar Datei, die den gesamten Sourcecode der Erweiterung beinhaltet, muss folgende Verzeichnisstruktur aufweisen:

```
your.New.Extension
|- controller
|- _content
|- _i18n
|- _static
|-  yourNewExtension.java
```

Im Detail:

- controller: Enthält den Controller, der wiederum alle visuellen Komponenten (Links, Buttons, Velocity Container etc.) enthält und das Event Management steuert.
- _content: Enthält die dynamischen Inhalte, die zur Laufzeit dargestellt werden, d.h Velocity Templates mit referenzierbaren Platzhaltervariablen und HTML Code.
- _i18n: Enthält Einstellungen für den PackageTranslator der den Inhalt in die Sprache des eingeloggten Users übersetzt, falls dessen Sprache unterstützt wird.
- _static: Enthält statische Inhalte wie CSS Stylesheets oder Bilder die für die Extension genutzt werden.
- yourNewExtension.java: Diese Datei ist verantwortlich für die Instanziierung des jeweiligen Controllers und setzt den sichtbaren Link im OLAT-Layout. Sie muss das Interface *Extension* implementieren.

Für einen neuen statischen Tab und dessen Inhalt geht man nun prinzipiell auf folgende Art und Weise vor:
Man erstellt die obligatorische Ordnerstruktur und beginnt mit der Implementierung der Extensionklasse. Diese muss wie bereits erwähnt das Interface *Extension* implementieren und den Extension Point *org.olat.extensions.sitescreator.SitesCreator* nutzen, da ja zunächst ein neuer Tab erstellt werden soll. Diese Datei wird auch den spezifischen Controller instanziieren.
Der zu erstellende Controller ist für die Visualisierung der statischen Komponenten verantwortlich. Dies geschieht analog zur Nutzung von SWING oder SWT unter anderem über die Erstellung von Panels, Buttons und Containerelementen oder vorgebildeten Objekten wie Tabellen und Formularen. Für diese Elemente werden auch ActionListener erstellt, die Eingaben durch den Benutzer registrieren und definiert bearbeiten.

Außerdem muss ein Velocity Container für die veränderbaren Inhalte der Seite erstellt werden,

der die entsprechende HTML Datei im *_content* Ordner darstellt. Diese HTML Datei enthält Variablen, die im Controller erstellt wurden und je nach Logik verschieden "befüllt" werden.
Die Konfigurationsdateien für den PackageTranslator werden auch durch den Controller erstellt. Dies geschieht in Form einer *LocalStrings_xy.properties* Datei, wobei für xy die spezifische Länderkenung (z.b. "de" für deutsch) manuell angepasst werden muss.
Anschließend müssen noch eventuell benutzte Bilder und Stylesheets in den *_static* Ordner eingefügt werden und die Extension ist funktionsfähig, nachdem sie als .jar Datei gepackt und in das Verzeichnis *WEB-INF/lib* kopiert wurde.
Um der Extension weitere Inhalte hinzuzufügen, wie zum Beispiel Unterpunkte im Navigationsmenü, müssen wir dafür weitere Extensions schaffen, die dann aber das Extension Interface *org.olat.extensions.action.ActionExtension* nutzen müssen.

Abschließend muss die Extension noch in der *olat_extention.xml* eingetragen werden, damit sie der OLAT-Extension Manager laden kann. Wichtig ist dabei eine eindeutige ID und dass der Eintrag *class* auf die Datei zeigt, die das Interface *Extension* implementiert:

```
<!-- generic OLAT extensions -->
<bean id="extmanager" class="org.olat.extensions.ExtManager" singleton="true">
 <property name="extensions">
  <list>
   <!-- Klasse die das Interface Extension implementiert -->
   <bean id="eindeutigeID" class="hier.gehts.zur.neuen.Extension" singleton="true" />
  </list>
 </property>
</bean>
```