



OLAT

Das
Open Source Learning Management System
(LMS)

OLAT

Vortrag von Ralf Rublack



OLAT

INHALT

1. Entwicklungsgeschichte
2. Technologie und Funktionsübersicht
3. Fazit im Bezug auf das Praktikum



OLAT

1. Entwicklungsgeschichte

- 1999
 - Beginn der Entwicklung in PHP
 - Universität Zürich
- 2002
 - PHP nicht performant und skalierbar genug
 - Entscheidung zum „rebuild“ in Java
- 2004
 - Release 3.0 komplett Java
 - Neue Javabasierte Architektur
 - Neues GUI, Ressourcen- und Gruppensystem
 - Produktiver Einsatz an der Universität Zürich
- Bis 2007
 - Erweiterung der Lernressourcen, Wiki, RSS, Jabber, SCORM, IMS
 - Letztes Release 5.1.2 vom 5.4.2007
 - Entwicklungsteam von zwölf Personen



OLAT

2. Technologie und Funktionsübersicht

2.1. Technologien

2.1.1. Servlet

2.2. Features

2.2.1. Konzept der Rollen und Gruppen

2.3. GUI

2.4. Velocity-Konzept

2.4.1. Beispiel

2.5. Extentions



OLAT

2.1. Technologien

- Basiert auf J2EE
- Java-Servlet-Technologie
- Hibernate-Persistenz-Framework
- Velocity-Konzept
- spring-Framework
- Eigenes GUI-Framework mit Ajax (Betastadium)



OLAT

2.1. Technologien

Servlet Container (Tomcat)

OLATServlet / Dispatcher

Windows / ChiefControllers

Controllers

Managers

VFS

Hibernate

FileSystem / Database

DispatcherAction
ConfigurationManager

UserSession

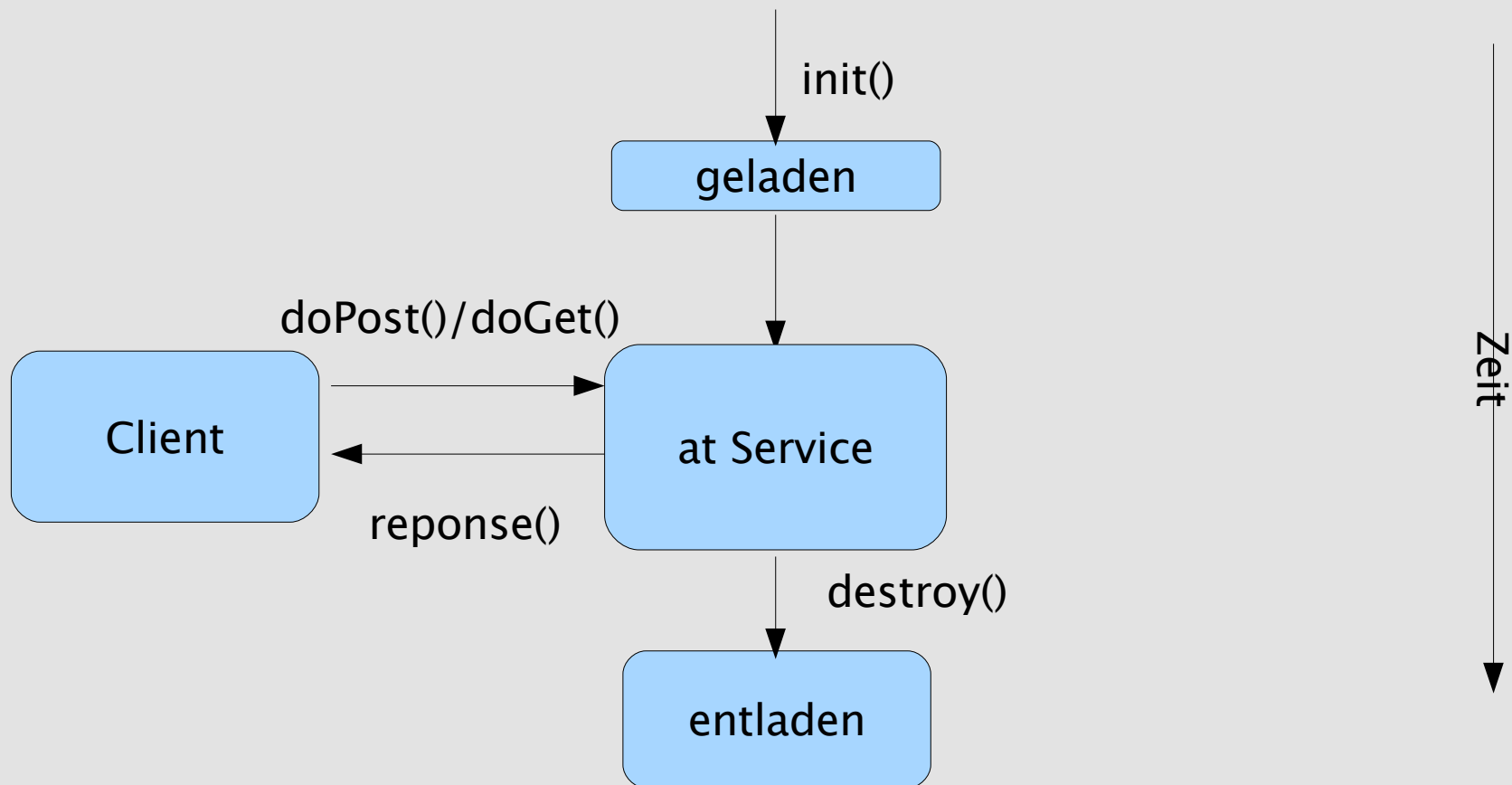
OLATResourceManager



OLAT

2.1.1. Servlets

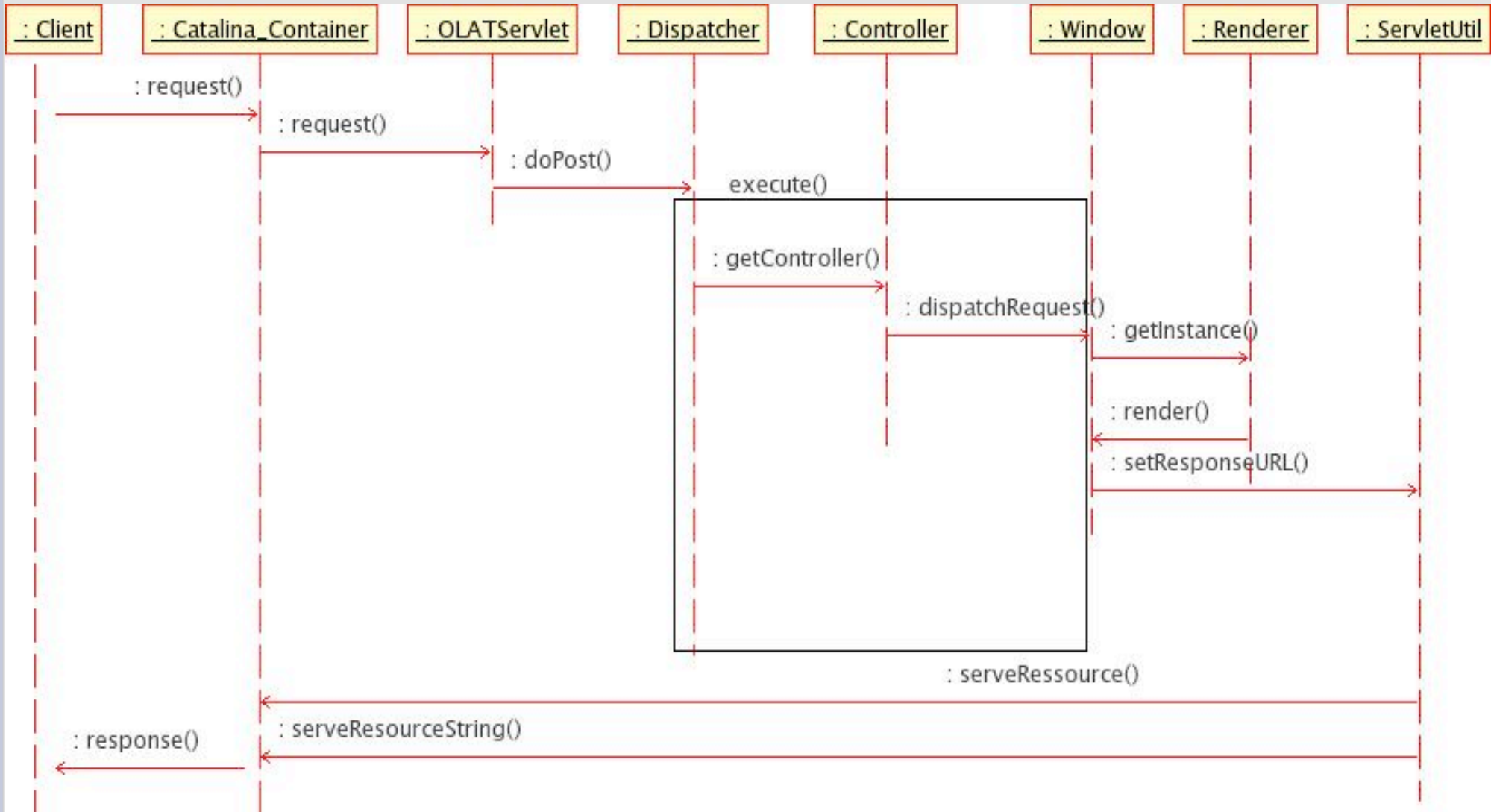
- Nutzung des Catalina-Containers von Tomcat mit Servlet-api 2.4





OLAT

2.1.1. Servlets





OLAT

2.2. Features

- Mehrsprachig (neue Sprachen hinzufügar mit OLAT translation tool)
- Komplet in UTF-8
- Shibboleth- oder User/Password Authentifizierung
- Dateifreigabe über http oder WebDAV
- Volltextsuche in vielen Dokumentformaten
- integrierter iCal basierender Kalender für Privat-/Gruppen-/Kursnutzung
- personalisierte RSS-Feeds
- Instant Messenger Jabber
- E-Mail Benachrichtigung
- Erweitertes Gruppen-Management
- Eigenorganisation des Lernens



OLAT

2.2. Features

Kurssystem

- Kurserzeugung mit OLAT Kurseditor
- Integration von Inhalten per IFrame
- Kursimport und -export
- Kursarchivierungssystem
- Kursvorschau und -simulation
- Änderungshistorie von Dokumenten

Lernressourcen

- IMS Content Packaging
- IMS QTI Test und Fragebogen
- SCORM Inhalte (SCORM 1.2)
- Lokale Inhalte per Upload oder WebDAV
- Externe Inhalte (Bilderdatenbank, Webseiten)
- Forum, Chat, Wiki, Dateifreigaben, Benachrichtigungen



OLAT

2.2.1 Konzept der Rollen und Gruppen

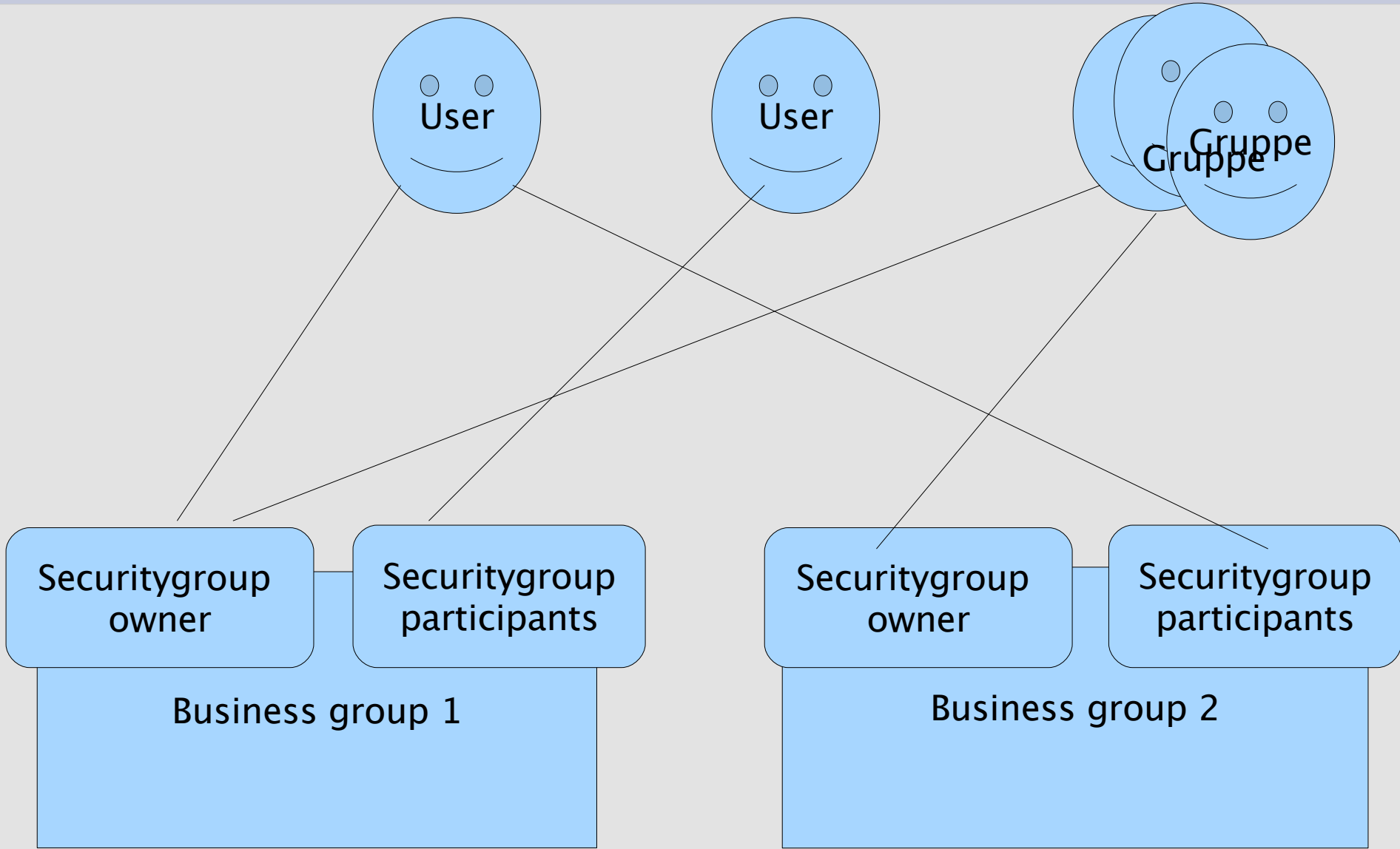
Gruppen

- System roles
 - guest
 - user
 - author
 - administrator
- Resource owners
 - course owner
 - catalog entry owner
 - groupmanagement owners
- Business groups
 - learning groups
 - × areas
 - right groups
 - × rights
 - Buddy groups
 - × IM roster sync
- SecurityGroup: owner, participants



OLAT

2.2.1 Konzept der Rollen und Gruppen

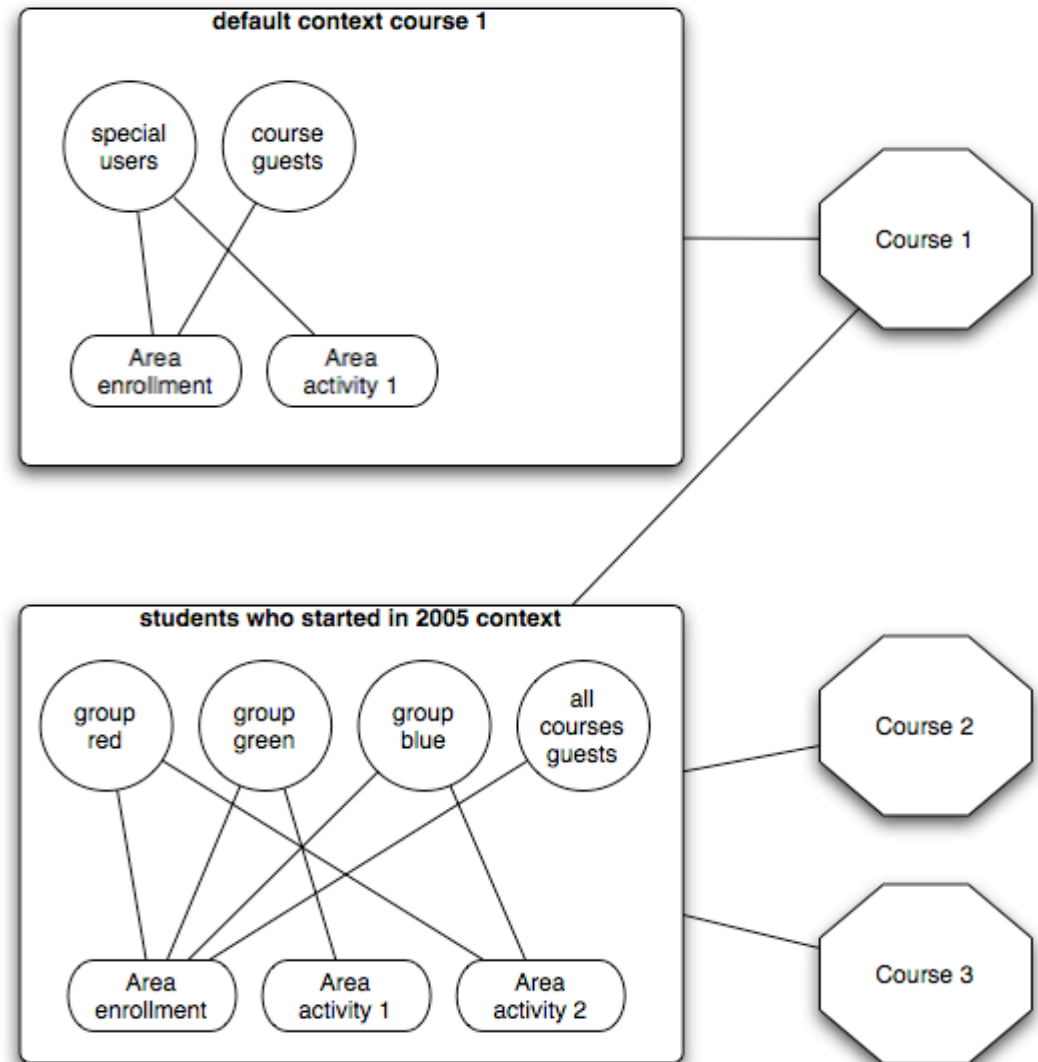




OLAT

2.2.1 Konzept der Rollen und Gruppen

- Areas
Bündelung von Ressourcen
unterschiedliche Workflows
- Rights
Zugriffsbeschränkungen
auf Ressourcen





OLAT

2.3. GUI

- Konzeptionierung nach Swing/SWT
- Ausgliederung des GUI als brasato framework
- für Entwicklung Debugmodus
- Ausgabe Html oder angefragtes Dokument
- Struktur durch Velocity
- Layoutverfeinerung mit CSS



OLAT

2.3. GUI

Verfügbare GUI-Komponenten
im Package `org.olat.core.gui.components`


• Form	form	passive
• Image	image	
• HTML Site	htmlsite	
• Panel	panel	
• HTML Header	htmlheader	
• Progressbar	progressbar	
• Wiki to HTML	wikiToHtml	
• Table	table	
• Delegating	delegating	
• Tabbed Pane	tabbedpane	
• Link	link	
• Choice	choice	
• Tree	tree	interactive





OLAT


2.3. GUI


- Abstrakte Klasse Component


 **Component**


 HTMLRendererSingleton


 VALIDATE_EVENT: Event


 name: String


 fireValidateEvent(): boolean


 listeners: Controller


 translator: Translator


 latestDispatchedController: Controller


 latestFiredEvent: Event


 parent: Container


 Component(name: String)


 Component(name: String, translator: Translator)


 GetComponentName(): String


 dispatchRequest(req: UserRequest)

 doDispatchRequest(req: UserRequest)

 fireEvent(req: UserRequest, event: Event)

 addListener(controller: Controller)

 getAndClearLatestFiredEvent(): Event

 loadReplayableEvent(baseTypeMap: Map <K,V>): ReplayableEvent

- Jede GUI Komponente besteht mindestens aus

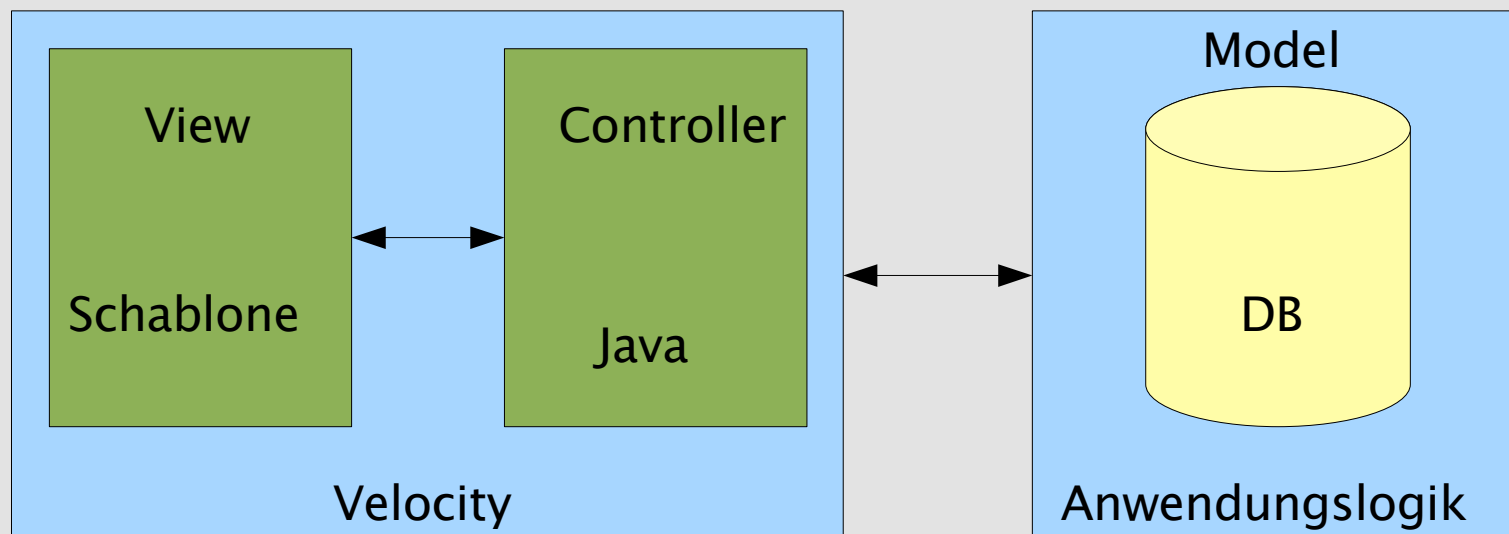
- Komponentenkategorie wird von Component erweitert
- Renderer der ComponentRenderer implementiert
- Eventklasse



OLAT

2.4. Velocity

- Open source Projekt
- Apache Software Foundation
- Erstellung dynamischer Webseiten
- Unterstützt MVC





OLAT

2.4. Velocity

Vorgehensweise bei Velocity

1. Initialisierung der Velocity-Engine
2. Anlegen eines Kontextobjektes
3. Füllen des Kontextobjektes mit Objekten
4. Auswahl eines Templates
5. Zusammenführen von Template und Kontextobjekt („mergen“)



OLAT

2.4. Velocity

Velocity Template Language (VTL)

Templates in Html, XML u.a geschrieben

Referenz auf Kontextobjekte mit \$-Zeichen

Direktiven (z.B. Kontrollstrukturen) mit #-Zeichen

Interpretation bei merging



OLAT

2.4.1 Velocity-Beispiel

Template

HelloName.vm

```
<html>
  #if ( $name )
    <b>Hallo $name!</b>
  #else
    <b>Hallo Fremder</b>
  #end

</html>
```



OLAT

2.4.1 Velocity-Beispiel

Javaklasse

HelloName

```
import java.io.StringWriter;
import org.apache.velocity.app.VelocityEngine;
import org.apache.velocity.Template;
import org.apache.velocity.VelocityContext;
public class HelloName
{
    public static void main( String args[] ) throws Exception {
        // (1) Engine initialisieren
        VelocityEngine ve = new VelocityEngine();
        ve.init();
        // (2) Kontextobjekt anlegen
        VelocityContext context = new VelocityContext();
        // (3) Objekte über Schlüssel hinzufügen
        context.put("name", args[0]);
        // (4) Template auswählen
        Template t = ve.getTemplate( "HelloName.vm" );
        // (5) Template und Kontextobjekt zusammenführen
        StringWriter writer = new StringWriter();
        t.merge( context, writer);
        // Im Stringwriter steht der fertige String
        System.out.println( writer.toString() );
    }
}
```



OLAT

2.4.1 Velocity-Beispiel

Aufruf nach Kompilieren

```
/> java HelloName Frank
```

erzeugt die Ausgabe

```
Hallo Frank!
```



OLAT

2.5. extension concept

Softwareerweiterung

1. Erweiterung des Programms
2. Komponentenbasierter Ansatz

Erweiterungen werden in `olat_extensions.xml` definiert

Erweiterung als jar in `WEB-INF/lib`

Beispiel einer Erweiterung im Package
`ch.goodsolutions.demoextension`

verschiedene Erweiterungspunkte definiert



OLAT

2.5. extension concept

Erweiterungspunkte

Interface:	<code>org.olat.extensions.globalmapper</code>
Erweiterungspunkt:	<code>org.olat.dispatcher.DispatcherAction</code>
Verwendene Klasse:	<code>org.olat.dispatcher.DispatcherAction</code>
Beschreibung:	Erweiterung bekommt durch den Mapper den Path
Interface:	<code>org.olat.extensions.action.ActionExtension</code>
Erweiterungspunkt:	<code>org.olat.home.HomeMainController</code>
Beschreibung:	Erweiterung enthält Link, Beschreibung und Aktionsdefinition
Interface:	<code>org.olat.extensions.css.CSSIncluder</code>
Erweiterungspunkt:	<code>org.olat.gui.components.Window</code>
Verwendene Klasse:	<code>org.olat.gui.css.CSSGenerator</code>
Beschreibung:	neue CSS-Stylsheets



OLAT

2.5. extension concept

Interface: `org.olat.extensions.hibernate.HibernateConfigurator`
Erweiterungspunkt: `org.olat.persistence.DB`
Verwendene Klasse: `org.olat.persistence.DB`
Beschreibung: Erweiterung um neu Hibernatemappings

Interface: `org.olat.extensions.sitescreator.SitesCreator`
Erweiterungspunkt: `org.olat.gui.control.generic.dtabs.DTabs`
Verwendene Klasse: `org.olat.FullChiefController`
Beschreibung: Neue Seiten, muss SitesCreator enthalten der Liste von SiteDefinition Objekten enthält



OLAT

3. Fazit

- OLAT ist eine modular aufgebaute Webapplikation
- Einarbeitung in Konzepte nötig
- dadurch sollte Verständnis der Architektur entstehen
- Entwicklung durch gut definierte Schnittstellen einfach
- Debug auch auf GUI-Ebene
- modulare Erweiterbarkeit abhängig von Verfügbarkeit der Erweiterungspunkte



OLAT

Literatur

OLAT

- http://www.olat.org/downloads/presentations/OLAT_Projekthandbuch_mitAnhang.pdf
- <http://www.olat.org/docu/dev/index.html>
- <http://www.olat.org/api/index.html?org/olat/package-summary.html>

Servlet und Tomcat

- <http://tomcat.apache.org/>
- <http://java.sun.com/products/servlet/>

Velocity

- <http://velocity.apache.org/>
- <http://velocity.apache.org/engine/index.html>

Spring

- <http://www.springframework.org/>



OLAT

Ende

Ich bedanke mich für Ihre Aufmerksamkeit

Für Fragen stehe ich Ihnen ab jetzt zur Verfügung