

# OLAT CommunityDay 2009

Scalability



Universität Zürich



## OLAT Scalability

Patrick Brunner  
Christian Guretzki  
(Informatikdienste, MELS)

Contact:

[patrick.brunner@id.uzh.ch](mailto:patrick.brunner@id.uzh.ch)  
[christian.guretzki@id.uzh.ch](mailto:christian.guretzki@id.uzh.ch)

# OLAT CommunityDay 2009

Scalability



Universität Zürich



## *Ablauf Skalierbarkeit*

- 09:30 ~ 10:15 **Entstehung, Konzepte, Entwicklung**
- 10:20 ~ 11:05 **HowTo, Code Bsp, Konfiguration**
- 11:10 ~ 11:55 **Eclipse Cluster Setup, Debugging**
- 12:00 ~ 12:30 **Diskussion**

# OLAT Conference 2009

OLAT Skalierbarkeit



Universität Zürich



## OLAT Skalierbarkeit Entstehungsgeschichte

# OLAT Conference 2009

OLAT Skalierbarkeit



Universität Zürich



## Inhalt:

- *Entstehungsgeschichte*
- Problemstellung
- Ziele & Abgrenzung
- Ausblick



# OLAT Conference 2009

OLAT Skalierbarkeit - Entstehungsgeschichte

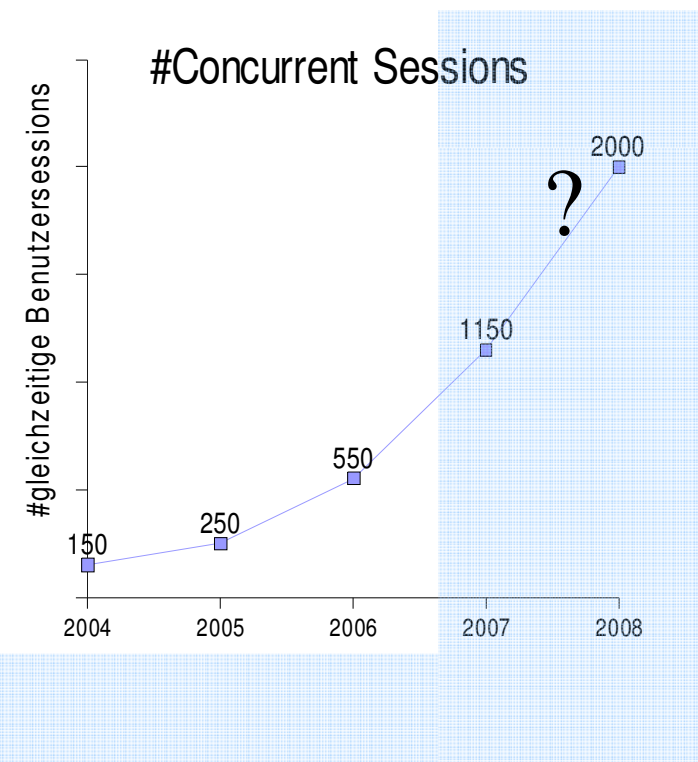


## AUSGANGSLAGE - Dez 2006

- OLAT attraktiv für E-Learning
  - mehr Benutzer
  - mehr (komplexe) Kurse
  - mehr Tests, e-Prüfungen

## Erwartung an Verfügbarkeit

- während ID Sys Updates
- während OLAT SW Update
- bei Hardware Ausfall



- Zukünftige Entwicklung?



# OLAT Conference 2009

OLAT Skalierbarkeit - Entstehungsgeschichte



*HEUTE - Mai 2009*

Benutzerzahlen

-> **Herbstsemester Peak**

Erwartung an Verfügbarkeit  
während ID Sys Updates

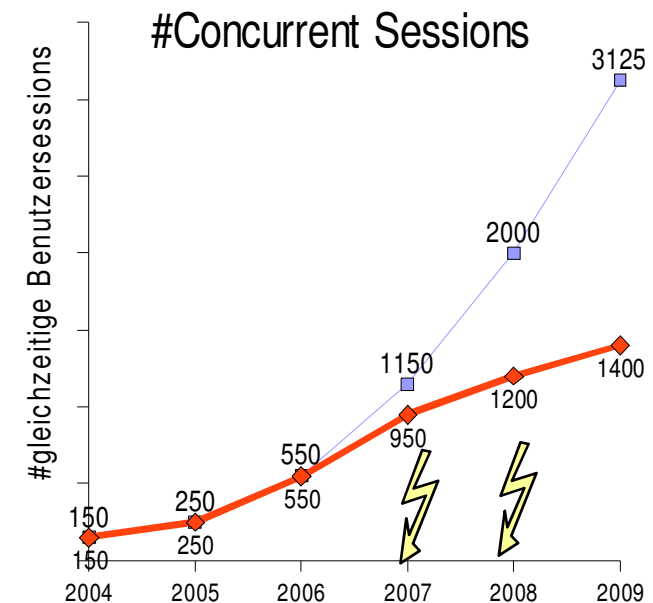
-> **möglich**

während OLAT SW Update

-> **nicht möglich (DB/FS)**

bei Hardware Ausfall

-> **Jein, SinglePointOfFailure!**





# OLAT Conference 2009

OLAT Skalierbarkeit - Entstehungsgeschichte



Zeitraum Dez 2006 – Oktober 2009:

- Skalierbarkeitsprojekte 1, 2 und 3
- Problem:
  - Concurrent Session Anzahl limitiert durch JVM RAM
  - JVM RAM limitiert wegen Garbage Collection Zeit
  - 1 Tomcat mit 2.5GB RAM rund **1000 aktive Sessions**
    - **zähflüssig, kritisch und out of memory gefährdet**
- Ziel:
  - Mehr Concurrent Sessions
  - n Tomcats 2.0 GB RAM x 500 Sessions > **1000**
    - **n = 2, schnell, unkritisch alle Funktionen zur Verfügung.**

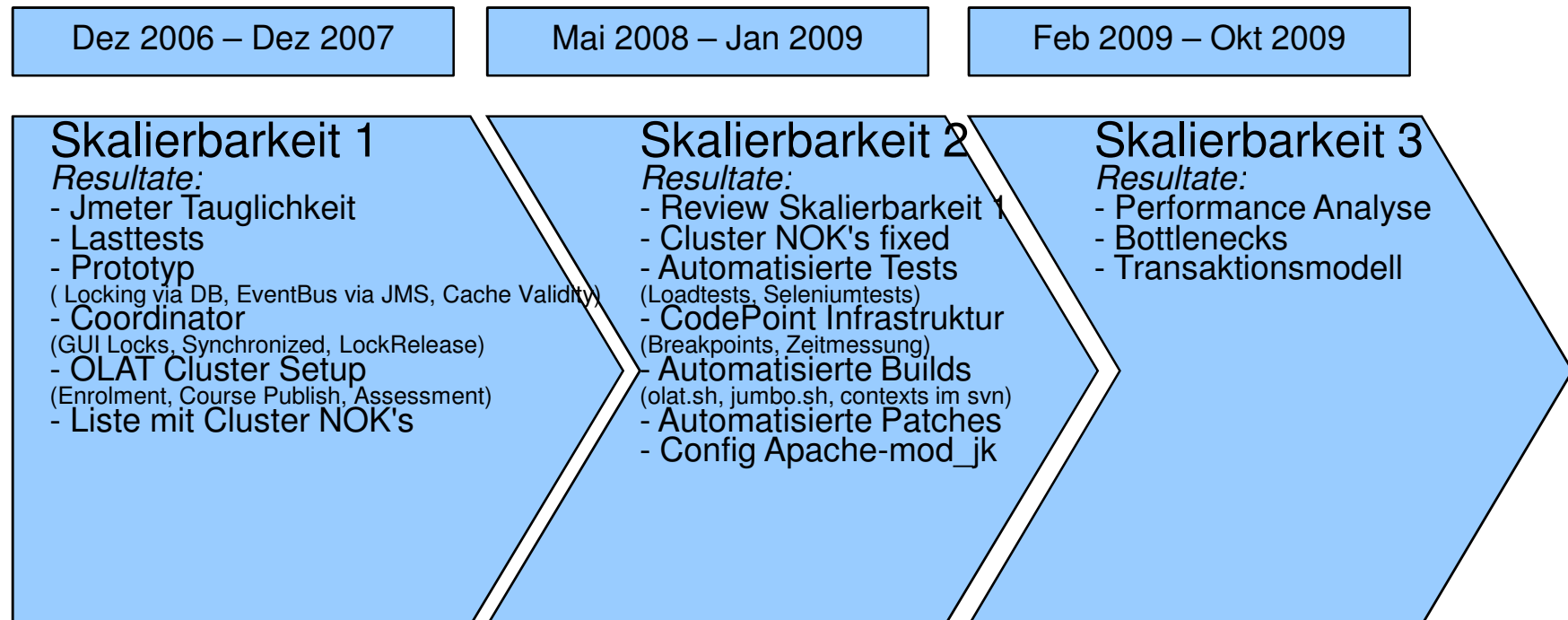


# OLAT Conference 2009

OLAT Skalierbarkeit - Entstehungsgeschichte



Dez 2006 – Oktober 2009:



# OLAT Conference 2009

OLAT Skalierbarkeit



Universität Zürich

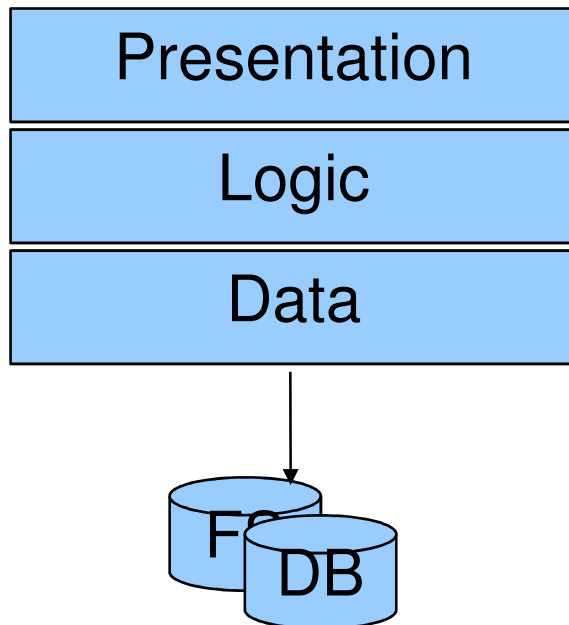


## Inhalt:

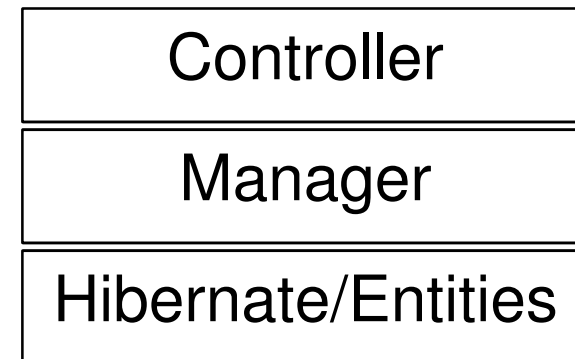
- Entstehungsgeschichte
- *Problemstellung*
- Ziele & Abgrenzung
- Ausblick



## 3-Schichten-Architektur



-> in OLAT



- + Optimistic Locking
- + MultiUserEventBus
- + GUI Lock Konzept

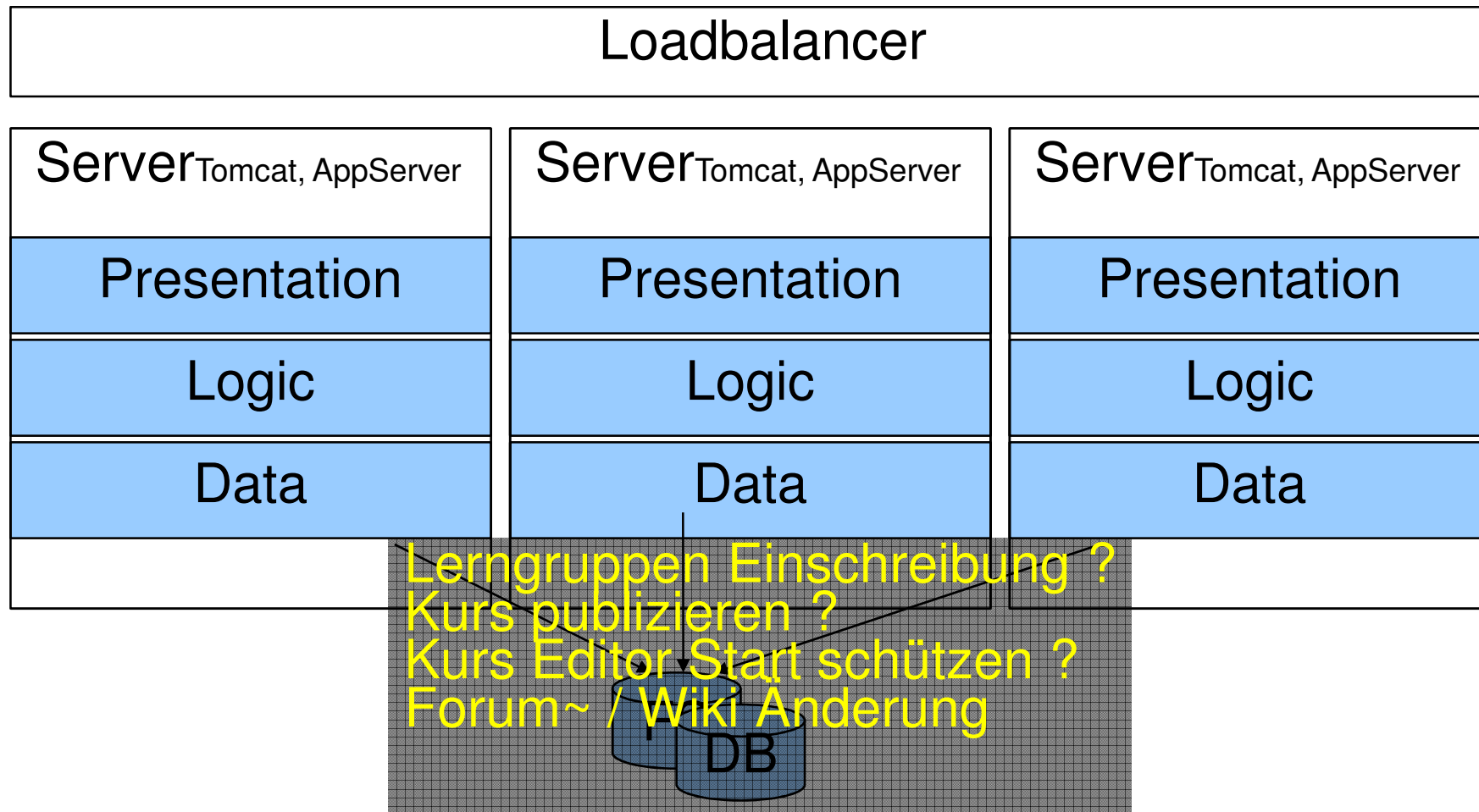


# OLAT Conference 2009

OLAT Skalierbarkeit - Problemstellung



(A) Cluster, alle Schichten kombiniert deployed:



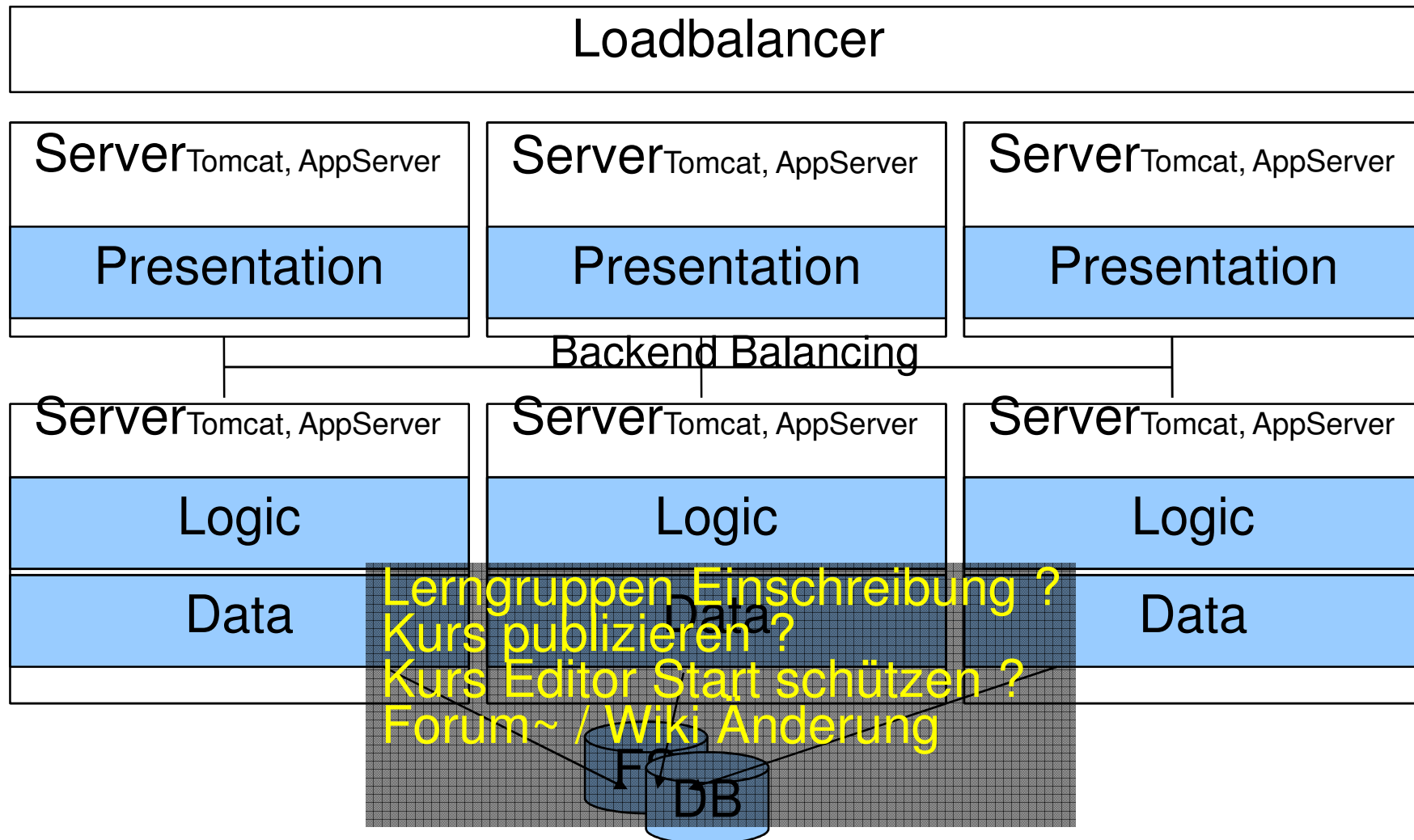


# OLAT Conference 2009

OLAT Skalierbarkeit - Problemstellung



(B) Cluster, Schichten getrennt deployed:





# OLAT Conference 2009

OLAT Skalierbarkeit - Problemstellung



## Problemstellung:

- Zugriff gemeinsamer Ressourcen über JVM Grenzen hinweg sicherstellen
  - Synchronized (in JVM) -> doInSync(..) (über mehrere JVMs)
  - z.B. Enrolment, muss Maximum prüfen.
- Klassenvariablen Referenzen (Caches) Änderungen über JVM Grenzen hinweg mitkriegen.
  - Referenz ändern (in JVM) -> CacheWrapper, reload
  - z.B. ICourse course = ... (Kurspublish)
- OLAT GUI Lock Mechanismus um schreibende Workflows zu schützen
  - Lock Objekte in Hashtable (in JVM)-> Lock Objekte in DB
  - z.B. Kurseditor starten -> nur ein Autor ist erlaubt.



## OLAT Conference 2009

OLAT Skalierbarkeit - Problemstellung



(B) Cluster, Schichten getrennt deployed:

- Nur Sinnvoll mit Umstieg auf EJB 3.0 + App.Server
  - Alles auf EJB umschreiben -> ganze Manager, Entities EJBifizieren
  - Build, Deploy, Config Prozess AppServer tauglich machen
- MultiUserEventBus? -> App.Server JMS
- (+) AppServer -> Transaktional, Ressourcen Pooling, Admin. Tool, Monitoring, Standard

=> **Entscheid dagegen, weil zu grosser Aufwand.**

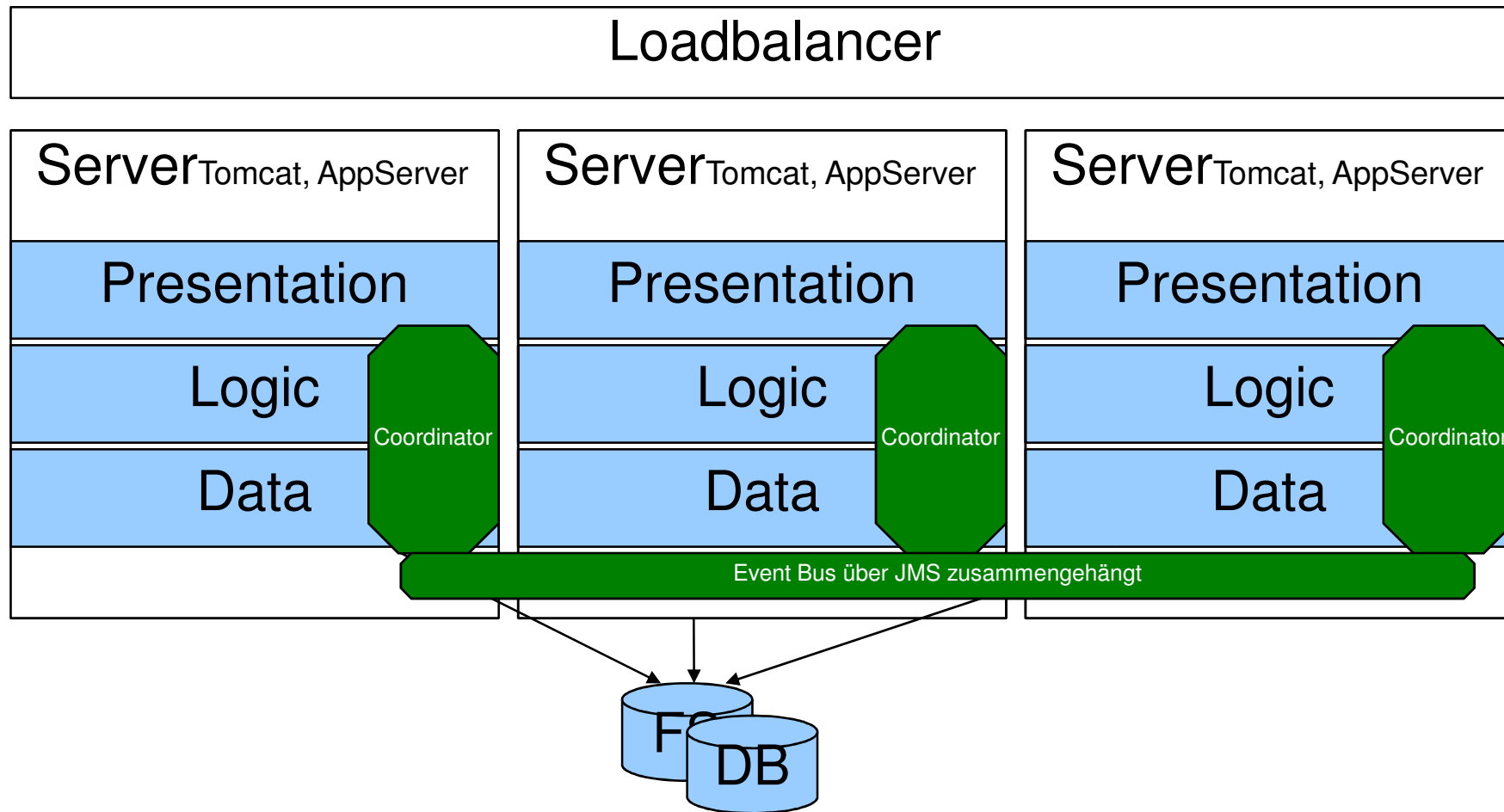


# OLAT Conference 2009

OLAT Skalierbarkeit - Problemstellung



(A) Cluster, alle Schichten kombiniert deployed:





## OLAT Conference 2009

OLAT Skalierbarkeit - Problemstellung



(A) Cluster, alle Schichten kombiniert deployed:

- Finde alle synchronized
  - => umschreiben auf doInSync
- Finde Klassenvariablen die komplexen BusinessObjekte Entitäten in Controllern cachen
  - => umschreiben auf immer wieder Reload über Manager

=> Überschaubare Menge von Änderungen

DONE

# OLAT Conference 2009

OLAT Skalierbarkeit



Universität Zürich



## Inhalt:

- Entstehungsgeschichte
- Problemstellung
- *Ziele & Abgrenzung*
- Ausblick

# OLAT Conference 2009

OLAT Skalierbarkeit – Ziele & Abgrenzung



Universität Zürich



- Ziel:
  - Mehr Sessions, „verteiltes shared Memory“
- Abgrenzung:
  - Nicht High Availability
  - Nicht Session Replication
  - Nicht Ausfallsicherheit bei Hardware Ausfall

# OLAT Conference 2009

OLAT Skalierbarkeit



Universität Zürich



## Inhalt:

- Entstehungsgeschichte
- Problemstellung
- Ziele & Abgrenzung
- *Ausblick*

# OLAT Conference 2009

OLAT Skalierbarkeit – Ausblick



Universität Zürich



- Resultate bisher
  - Loadtest-Infrastruktur
  - Selenium Funktionaltest-Infrastruktur
  - GoLive Frühlingssemester 2009 ohne Probleme
    - Peak Test kommt mit Herbstsemester 2009
  - Verteiltes OLAT auf 1 Servermaschine mit Apache mod\_jk als LoadBalancer, 3 Tomcat Instanzen, MySQL und Jabber, Filesystem via Fibrechannel angebunden.



- Ausblick / Skalierbarkeit 3
  - Performance Untersuchung, wo skaliert OLAT nicht.
    - DB? Filesystem? JMS? Workflows?
  - Transaktionsmodell
    - DoInSync macht Commit
      - commit–rollback-per-Click- Paradigma stimmt nicht!
    - Verteilte Transaktion
      - MultiEventBus -> via JMS asynchron -> Transaktion per VM
  - Managed Block
    - Code Utility / Core Infrastruktur
      - Commit-rollback issues entdecken
  - Herbstsemester 2009: Betrieb auf mehreren Rechnern

# OLAT CommunityDay 2009

## Scalability HowTo



Universität Zürich



### ***Skalierbarkeit HowTo Ziele :***

- Cluster Implementation verstehen
- Coordinator-Interface anwenden
- Cluster Problem-Stellen erkennen und mit Coordinator lösen können



### *Mit dem Coordinator werden 4 Hilfsmittel zum Clustering zur Verfügung gestellt :*

- Cluster-weit synchronisieren => **Syncer**
- Cluster-weiter Ereignisaustausch => **EventBus**
- Caching mit Cluster-weiter Invalidierung => **Cacher**
- Sperren von Workflows fuer andere Benutzer => **Locker**

Applikations-Code hat keinen cluster-spezifischen Teil. Alle Cluster Funktionen sind im Coordinator gekapselt. SingleVM oder Cluster Implementation des Coordinators wird via Spring definiert.

# OLAT CommunityDay 2009

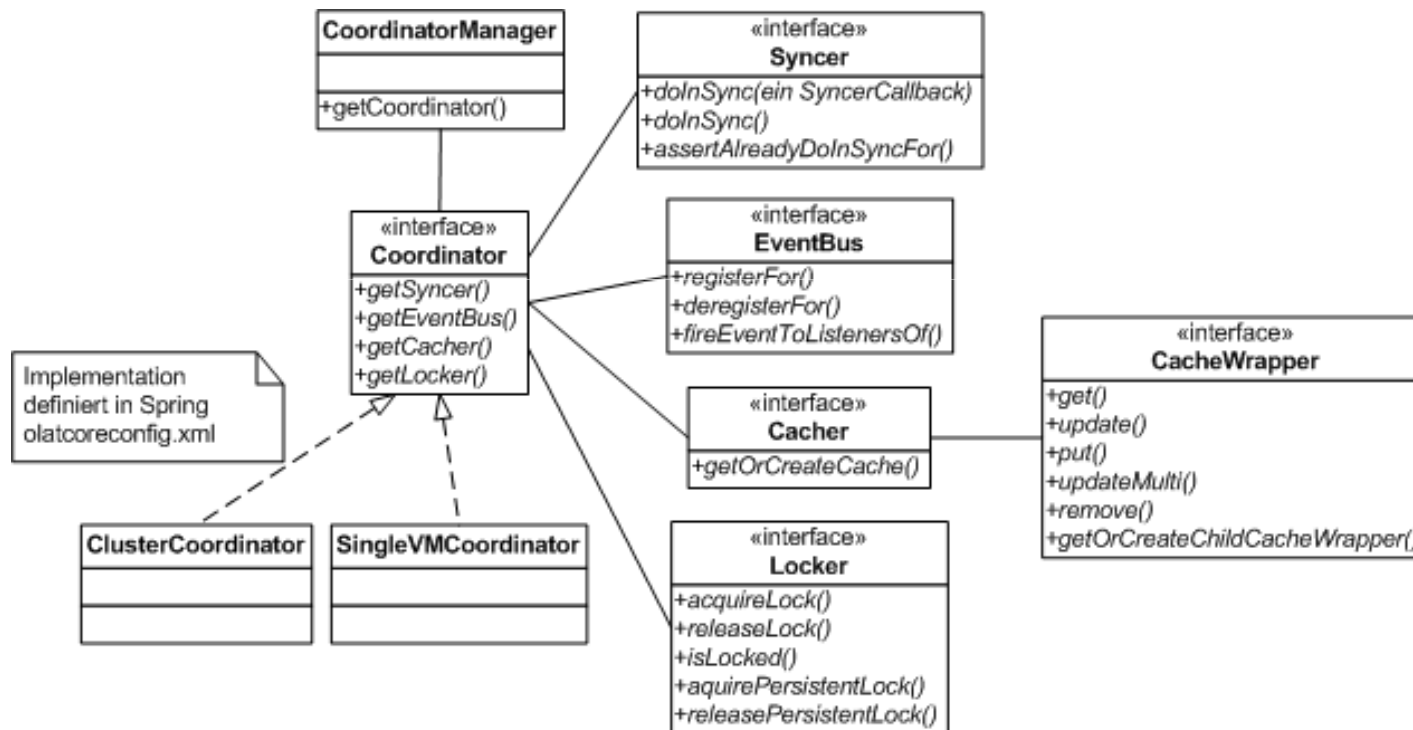
Scalability HowTo Coordinator



Universität Zürich



## Coordinator Class Diagram



# OLAT CommunityDay 2009

Scalability HowTo Coordinator



Universität Zürich



## ***Coordinator API :***

- Referenz auf Coordinator

```
CoordinatorManager.getCoordinator()
```

- **Zugriff auf Subcomponenten**

```
CoordinatorManager.getCoordinator().getSyncer()
```

```
CoordinatorManager.getCoordinator().getEventBus()
```

```
CoordinatorManager.getCoordinator().getCacher()
```

```
CoordinatorManager.getCoordinator().getLocker()
```



## ***Syncer : Cluster-weite Synchronisation von Code-Stellen, ersetzt java synchronized :***

- Mit DB und **SelectForUpdate** umgesetzt  
=> siehe PessimisticLockManager
- **OLATResourceable** (definiert durch Namen und Key) reserviert eine Zeile in der plock-Tabelle für einen update. Die Datenbank stellt damit sicher, dass diese Zeile bis zum commit oder rollback reserviert ist.
- Wichtig : OLATResourceable definiert den Bereich, welcher synchronisiert wird, d.h. Es muss so genau wie möglich definiert sein  
Bsp eine Gruppe und nicht alle Gruppen

Anwendungs-Beispiele : Einschreibung, Lesen oder Erzeugung von Business-Objekten

# OLAT CommunityDay 2009

Scalability HowTo Syncer



Universität Zürich



## ***Syncer API :***

- Sync-Block ohne return Wert

```
public void doInSync(OLATResourceable ores,  
                    SyncerExecutor action);
```

- Sync-Block mit return Wert

```
public <T> T doInSync(OLATResourceable ores,  
                    SyncerCallback<T> action)
```

## **Wichtig :**

1. möglichst kurzer doInSync-Block
2. OLATResourceable möglichst fein granular wählen
3. Mit OresHelper können beliebige OLATResourceable Sync-Objekte erzeugt werden.

# OLAT CommunityDay 2009

Scalability HowTo Syncer



Universität Zürich



## *doInSync Beispiel ohne return-Wert mit einer BusinessGroup als OLATResourceable :*

```
BusinessGroup enrolledGroup;
CoordinatorManager.getCoordinator()
    .getSyncer()
    .doInSync(enrolledGroup,
        new SyncerExecutor() {
            public void execute() {
                // do something synchronized ....
            }
        });
```

=> Beispiel siehe EndrollmentManager

# OLAT CommunityDay 2009

Scalability HowTo Syncer



Universität Zürich



## ***doInSync Beispiel mit Boolean return-Wert und einer BusinessGroup als OLATResourceable :***

```
OLATResourceable calOres =  
    OresHelper.createOLATResourceableType(  
        getKeyFor(cal.getType(), cal.getCalendarID()));  
  
Boolean persistSuccessful =  
    CoordinatorManager.getCoordinator().getSyncer()  
        .doInSync(calOres, new SyncerCallback<Boolean>() {  
        public Boolean execute() {  
            // do something...  
            return new Boolean(value);  
        }  
    });
```

=> Beispiel siehe ICalFileCalendarManager

# OLAT CommunityDay 2009

Scalability HowTo Syncer



Universität Zürich



## *Wann wird der Syncer verwendet ?*

Sobald man bei der Entwicklung ein `synchronized` verwendet, muss man sich überlegen ob die Synchronization nur VM-weit oder Cluster-weit erfolgen muss.

Falls nur VM-weit synchronisiert wird, **muss** dies mit einem Kommentar dokumentiert werden :

```
synchronized (upcomingWork) { //o_clusterOK by:ld
    // synchronized OK - only one cluster node
    // must update the EfficiencyStatements
    // (the course is locked for editing)
    // (same as e.g. file indexer)
```

# OLAT CommunityDay 2009

Scalability HowTo EventBus



Universität Zürich



## ***EventBus : Cluster-weiter Austausch von Ereignissen :***

- Der Austausch erfolgt asynchron via JMS (ActiveMQ)
- Adressierung erfolgt via OLATResourceable z.B. Course
- Nur eine JMS Message wird pro Cluster-Node verschickt und dann OLAT-intern an die entsprechenden Listener verteilt
- Anwendungs-Beispiele : Publish, BookmarkManager verlangt ein Reload der Daten in PortletController
- Events müssen von der Klasse MultiUserEvent abgeleitet sein und serialisierbar sein d.h. möglichst wenig Daten enthalten.

# OLAT CommunityDay 2009

Scalability HowTo EventBus



Universität Zürich



## ***EventBus API :***

- Event Listener registrieren, identity kann null sein

```
public void registerFor(GenericEventListener gel,  
                        Identity identity,  
                        OLATResourceable ores);
```

- Event Listener de-registrieren

```
public void deregisterFor(GenericEventListener gel,  
                           OLATResourceable ores);
```

- Fire Event, event muss ein MultiUserEvent sein

```
public void fireEventToListenersOf(  
                        MultiUserEvent event,  
                        OLATResourceable ores);
```

# OLAT CommunityDay 2009

Scalability HowTo EventBus



Universität Zürich



## ***EventBus Beispiel 'BookmarkEvent' Listener (Teil1):***

1. **Klasse muss Interface `GenericEventListener` implementieren**

2. **EventListener im Constructor registrieren (in diesem Beispiel für eine bestimmte Identity)**

```
// identity is no OLATResourceable
// => create Helper Object eventBusThisIdentityOres =
OresHelper.createOLATResourceableInstance(
    Identity.class, identity.getKey());

CoordinatorManager.getCoordinator().getEventBus()
    .registerFor(this, ureq.getIdentity(), eventBusThisIdentityOres);
```



## ***EventBus Beispiel 'BookmarkEvent' Listener (Teil2):***

### **3. event Methode implementieren**

```
public void event(Event event) {
    if(event instanceof BookmarkEvent) {
        if(((BookmarkEvent)event).getUsername().equals(identity.getName())
            || ((BookmarkEvent)event).isAllUsersEvent()) {
            reloadModel(sortingCriteria);
        }
    }
}
```

### **4. de-registrieren der Listener in doDispose()**

```
protected void doDispose() {
    CoordinatorManager.getCoordinator().getEventBus()
        .deregisterFor(this, eventBusThisIdentityOres);
    super.doDispose();
    // dispose Controller
}
```

=> Beispiel siehe `BookmarksPortletRunController`

# OLAT CommunityDay 2009

Scalability HowTo EventBus



Universität Zürich



## ***EventBus Beispiel fire 'BookmarkEvent' :***

```
BookmarkEvent bookmarkEvent = new BookmarkEvent(identity.getName());  
eventBusOres = OresHelper.createOLATResourceableInstance(  
    Identity.class, identity.getKey());  
CoordinatorManager.getCoordinator().getEventBus()  
    .fireEventToListenersOf(bookmarkEvent, eventBusOres);
```

=> Beispiel siehe `BookmarkManagerImpl`

# OLAT CommunityDay 2009

Scalability HowTo EventBus



Universität Zürich



## ***EventBus Beispiel 'BookmarkEvent' :***

```
public class BookmarkEvent extends MultiUserEvent {  
    private static final String ALL_USERS = "all";  
    private final String username;  
  
    public BookmarkEvent(String username) {  
        super("bookmark_event");  
        this.username = username;  
    }  
  
    public BookmarkEvent() {  
        this(ALL_USERS);  
    }  
  
    public String getUsername() {  
        return username;  
    }  
  
    public boolean isAllUsersEvent() {  
        return ALL_USERS.equals(getUsername());  
    }  
}
```

=> Beispiel siehe *BookmarkEvent*

# OLAT CommunityDay 2009

Scalability HowTo Cacher



Universität Zürich



## ***Cacher : Caching von Business-Objekten :***

- Mit cluster-weiter Invalidierung bei 'update' (keine Replizierung!)
- Cache-Parameter können in Spring Konfiguration definiert werden (timeToLive, timeToldle, maxElementsInMemory)
- Cache kann hierarchisch aufgebaut werden (Bsp *NewCachePersistingAssessmentManager*)





### ***Cacher Configuration :***

- Die Caches werden im *olatcore\_cluster\_config.xml* definiert
- Beispiel des Wiki Cache Eintrages:

```
<entry key="org.olat.modules.wiki.WikiManager_wiki">  
  <bean class="org.olat.core.util.cache.n.CacheConfig">  
    <property name="timeToLive" value="14400" />  
    <property name="timeToIdle" value="1800" />  
    <property name="maxElementsInMemory" value="50" />  
  </bean>  
</entry>
```

- Key ergibt sich aus Klassen-Namen '*org.olat.modules.wiki.WikiManager*' und dem Cache-Namen '*wiki*' mit einem '\_' als Delimiter
- Die Cache-property Werte müssen ev. Auf den verschiedenen Nodes entsprechend angepasst werden z.B. Course-Cache
- Cache Konfiguration im Admin Tab im Betrieb überprüfen und allenfalls Konfiguration anpassen

# OLAT CommunityDay 2009

Scalability HowTo Cacher



Universität Zürich



## ***Cacher HowTo :***

**1. Ersetzen Map,List mit CacheWrapper**

**2. Erzeugen eines Caches**

```
calendarCache = CoordinatorManager.getCoordinator().getCacher()  
                    .getOrCreateCache(this.getClass(), "calendar");
```

**3. Methode put für initial 'put-in-cache'**, muss in cluster-synchronized block sein (zum Verhindern von lost-invalidate)

**4. Methode get : für read-only cache Objekte**

**5. update, updateMultiple** : update Calls müssen in cluster-synchronized Block sein (zum Verhindern von lost-invalidate)

**6. Add implements Serializable zu den cached Objekten** für cache-Interface. Cached Objekte werden via key invalidatiert. Cached Objekte werden nicht repliziert!

**7. Cached-Objekte nicht lokal speichern** (keine Attribute)

**8. Cacher nur in Manager und nicht in Controller verwenden.**

=> Beispiel ICalFileCalendarManager

# OLAT CommunityDay 2009

Scalability HowTo Cacher



Universität Zürich



## ***Cacher Beispiel : Element aus Cache lesen oder falls nicht vorhanden im Cache speichern :***

```
Kalender cal = CoordinatorManager.getCoordinator().getSyncer()
    .doInSync( calOres, new SyncerCallback<Kalender>() {
    public Kalender execute() {
        String key = getKeyFor(callType, callCalendarID);
        Kalender cal = (Kalender)calendarCache.get(key);
        if (cal == null) {
            cal = loadOrCreateCalendar(callType, callCalendarID);
            calendarCache.put(key, cal);
        }
        return cal;
    }
});
```





### ***Locker : Sperren von Workflows über mehrer Requests :***

- Unterscheidung von persistent Locks und nicht persistent Locks :
  - **nicht persistente** Locks werden beim **Beenden** der Session wieder **freigegeben**
  - **persistente** Locks **bleiben** auch nach Beenden der Session **gespeichert** und müssen via Workflow wieder freigegeben werden.
- Persistent Locks sind als Property umgesetzt (siehe DBPersistentLockManager)
- Nicht persistente Locks sind mit ClusterLockManager und oc\_lock Tabelle umgesetzt.
- Anwendungsbeispiele Nicht persistente Locks : Lösch Workflow nur für einen Administrator, Course-Editor nur 1x Offen pro Kurs
- Anwendungsbeispiele persistente Locks : QTIEditor

# OLAT CommunityDay 2009

Scalability HowTo Cacher



Universität Zürich



## Locker API :

```
// all locks will be relased by signoff
public LockResult acquireLock(OLATResourceable ores,
                             Identity identity,
                             String locksubkey);

public void releaseLock(LockResult le);
public boolean isLocked(OLATResourceable ores, String locksubkey);

// A persistent lock will be stored over
// user-sessiontimeout, logout and
// server-restart. It will be released only by
// calling releasePersistentLock
public LockResult aquirePersistentLock(OLATResourceable ores,
                                       Identity ident,
                                       String locksubkey);

public void releasePersistentLock(LockResult le);

// LockResult API
// =====
public boolean isSuccess();
public Identity getOwner();
public long getLockAquiredTime();
```

# OLAT CommunityDay 2009

Scalability HowTo Cacher



Universität Zürich



## ***Locker Beispiel :***

```
// aquire lock in EditorMainController constructor
ICourse course = CourseFactory.loadCourse(ores);
lockEntry = CoordinatorManager.getCoordinator().getLocker()
    .acquireLock(course, ureq.getIdentity(), CourseFactory.COURSE_EDITOR_LOCK);

// check if we have the lock
if (lockEntry.isSuccess()) {
    // ok we can edit course
} else {
    // Sorry course is already locked by :
    Identity lockedBy = lockEntry.getOwner();
}

// call doReleaseEditLock when closing editor
// and in doDispose() Release Lock
private void doReleaseEditLock() {
    if (lockEntry.isSuccess()) {
        CoordinatorManager.getCoordinator().getLocker().releaseLock(lockEntry);
    }
}
```

# OLAT CommunityDay 2009

## Scalability Configuration



Universität Zürich



### ***Skalierbarkeit Konfiguration Ziele :***

- Cluster Setup
- Cluster konfigurieren

# OLAT CommunityDay 2009

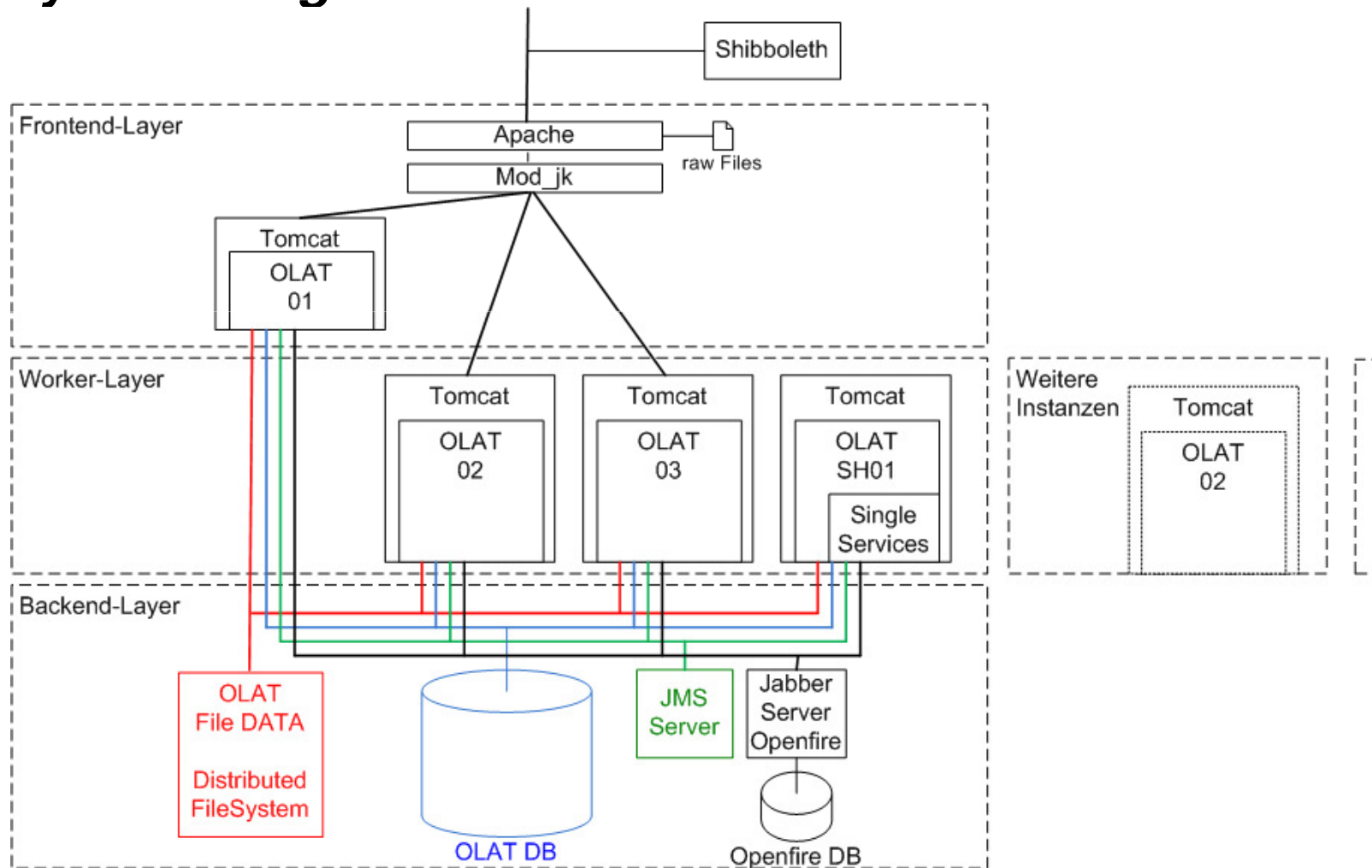
## Scalability Configuration



Universität Zürich



### System Diagramm Cluster



# OLAT CommunityDay 2009

Scalability Configuration



Universität Zürich



## *Single Services :*

- **RemoteAuditManagerService**  
Asynchroner Kurs-Logger, entstanden da Kurs-Logger  
zuerst synchronisiert war und sich dies als Performance  
Engpass zeigte.  
Implementiert via JMS
- **NotificationManager**
- **UpgradeManager**
-

# OLAT CommunityDay 2009

## Scalability Configuration



Universität Zürich



### ***Apache Konfiguration :***

- Loadbalancer : Apache übernimmt das Tomcat Loadbalancing.  
Konfiguriert im **worker.properties**

#### Bsp worker.properties

```
worker.list=tomcatolat01,tomcatolat02,loadbalancerworker,status  
worker.status.type=status  
  
worker.loadbalancerworker.type=lb  
worker.loadbalancerworker.balance_workers=tomcatolat02,tomcatolat01  
worker.loadbalancerworker.sticky_session=true  
worker.loadbalancerworker.sticky_session_force=false  
worker.loadbalancerworker.method=s
```

- Static Files : alle olat/raw Files (js, css, images, extensions, themes, yaml) sollten aus Performance Gründen direkt via Apache ausgeliefert werden.  
Dies wird im **mod\_jk.conf** File konfiguriert; Command JkUnMount .

#### Bsp für JS Files

```
JkUnMount /olat/raw/610/js/* loadbalancerworker
```

# OLAT CommunityDay 2009

## Scalability Configuration



Universität Zürich



### **OLAT Konfiguration (Teil1):**

- **build.properties für jeden Node (grün Node spezifisch Werte) anpassen**

```
server.modjk.enabled=true
server.modjk.jvmRoute=tomcatolat01
# optional for direct tomcat access
server.internal.http.port=8081
```

```
cluster.mode=Cluster
# Singleton Services auf einem Node enabled
cluster.singleton.services=disabled
# Port für jeden Node anpassen
cluster.catalinaport=8015
cluster.ajpport=8019
# Unique Node.Id vergeben
node.id=1
# JMS Broker Url analog Active setzen
jms.broker.url=failover:(tcp://localhost:61616?wireFormat.maxInactivityDuration=30000)
search.broker.url=failover:(tcp://localhost:61616?wireFormat.maxInactivityDuration=30000)
codepoint.jms.broker.url=failover:(tcp://localhost:61616?wireFormat.maxInactivityDuration=30000)
treecache.xml.file=treecache.xml
# Auf Netzwerk freier multicast Port definieren
multicast.port=45586
```

# OLAT CommunityDay 2009

## Scalability Configuration



Universität Zürich



### ***OLAT Spring Konfiguration (Teil2):***

- **olatcore\_cluster\_config.xml**  
@ Serviceconfig/org/olat/core/ \_spring/  
=> Anpassung der Cache-Werte  
im config-all ant-target wird dieses File in  
olatcore\_config.xml umkopiert
- **olatdefaultconfig.xml**  
@ Serviceconfig/org/olat/search/service/ \_spring/  
=> Comment-in searchClient und searchServer Bean

# OLAT CommunityDay 2009

## Scalability Configuration



Universität Zürich



### ***Backend Konfiguration :***

- Datenbank : MySQL  
keine spezielle Cluster-Konfiguration  
Nicht vergessen die Permission für die Cluster-Nodes  
(Hosts) zu setzen!
- ActiveMQ (JMS)  
keine spezielle Cluster-Konfiguration



### ***Vorgehen Cluster Konfiguration:***

1. Anzahl Nodes und Verteilung festlegen
2. Definieren wo SingleServices und Search laufen sollen
3. Apache worker.properties mit den entsprechenden Tomcat konfigurieren
4. JkUnMount für static File konfigurieren
5. Build.properties der einzelnen Nodes entsprechend dem Setup konfigurieren (Single-Services, Tomcats, Ports)
6. Cache-Properties im olatcore\_cluster\_config.xml anpassen (ev. Später nochmals anpassen)
7. Search Proxy und Search-Server konfigurieren
8. Ant config-all
9. Start aller Backend Services und Apache
10. Zuerst Tomcat mit SingleService starten
11. Dann restlichen Tomcat starten

# OLAT CommunityDay 2009

Scalability Cluster Configuration in Eclipse



Universität Zürich



## ***Cluster Konfiguration in Eclipse:***

4 Projekte im Eclipse anlegen :

1. olatcore als Code-Basis
2. olat3 als Code-Basis und SingleVM Setup
3. olat3\_clusternode\_1 (olat3 Projekt ausgecheckt)  
Als Java Source Code wird olat3 Projekt konfiguriert.  
Konfiguration mit serviceconfig wird aber von diesem  
Projekt verwendet (Java Build Path : src\_olat3 Excluded  
serviceconfig).
4. olat3\_clusternode\_2 analog zu Node1

# OLAT CommunityDay 2009

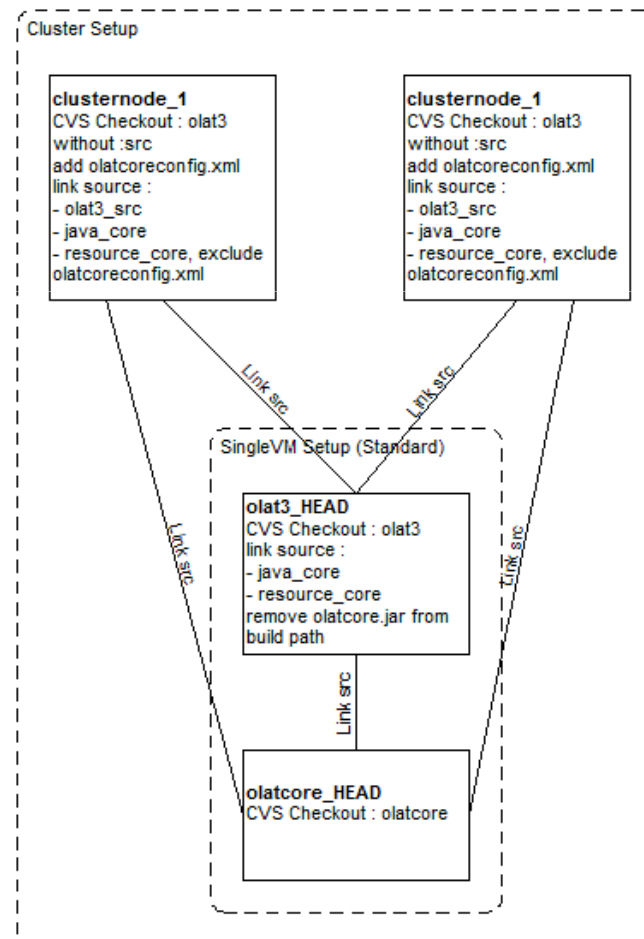
## Scalability Cluster Configuration in Eclipse



Universität Zürich



### Cluster Projekt Setup in Eclipse:



# OLAT CommunityDay 2009

## Scalability Cluster Configuration in Eclipse



Universität Zürich



### *Clusternode1 Konfiguration in Eclipse:*

Properties for olat3\_clusternode\_1

type filter text

- Resource
- BeanInfo Path
- Builders
- Checkstyle
- CVS
- FindBugs
- Java Build Path**
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Metrics
- Project References
- Refactoring History
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation

JRE\_LIB is deprecated: Use the JRE System Library instead.

Source Projects Libraries Order and Export

Source folders on build path:

- olat3\_clusternode\_1/java\_core - C:\home\cg\workspace\olatcore\_HEAD\src\main\java
- olat3\_clusternode\_1/patchesSrc\_olat3 - C:\home\cg\workspace\olat3\_HEAD\webapp\WEB-INF\patchesSrc
- olat3\_clusternode\_1/resources\_core - C:\home\cg\workspace\olatcore\_HEAD\src\main\resources
- olat3\_clusternode\_1/src\_olat3 - C:\home\cg\workspace\olat3\_HEAD\webapp\WEB-INF\src
  - Included: (All)
  - Excluded: serviceconfig/
  - Native library location: (None)
- olat3\_clusternode\_1/webapp/WEB-INF/src
  - Included: serviceconfig/
  - Excluded: META-INF;/ch;/com;/de;/org;/velocity/
  - Native library location: (None)

Default output folder:

olat3\_clusternode\_1/webapp/WEB-INF/classes

OK Cancel

# OLAT CommunityDay 2009

Scalability Cluster Configuration in Eclipse



Universität Zürich



## ***Cluster Konfiguration in Eclipse (Teil 1):***

Setup olat3 with core-project (SingleVM)

Checkout olat3 as clusternode\_1

Checkout olat3 as clusternode\_2

Remove all src in build path

Link source :

- java\_core

- resource\_core

- src\_olat3, exclude serviceconfig (because we must have one for each cluster-node)

- patches src

remove olatcore.jar and reference in classpath for it

Update build.properties for both clutsernodes

olat\_config.xml.in : enable ClusterModule

ant config-all

# OLAT CommunityDay 2009

Scalability Cluster Configuration in Eclipse



Universität Zürich



## ***Cluster Konfiguration in Eclipse (Teil 2):***

create olatcoreconfig.xml in new src dir with same path in each clusternode

Setup node-ID in olatcoreconfig.xml und brokerURL weiter unten

Create Server Tomcat 'clusternode\_1' with clusternode\_1/webapp as docBase

Create Server Tomcat 'clusternode\_2' with clusternode\_2/webapp as docBase

Tomcat 'clusternode\_1' : HTTP : 8080, RMI : 3000, Shutdown 8005

Tomcat 'clusternode\_1' : HTTP : 8082, RMI : 3001, Shutdown 8007

ActiveMQ starten (download from:<ftp://mirror.switch.ch/mirror/apache/dist/activemq/apache-activemq/5.1.0/apache-activemq-5.1.0-bin.tar.gz>)

MySQL Starten