

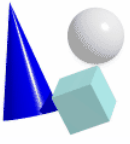
Doktorandenseminar
09.12.2004



Konzeption einer Persistenz-Architektur für verteilte, Java-basierte Anwendungen auf Basis des JDO-Standards

Doktorandenseminar am
Lehrstuhl für Wirtschaftsinformatik III
Prof. Dr. Martin Schader
Universität Mannheim

**Dipl.-Wirtsch.-Inf.
Matthias Merz**



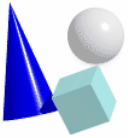
Agenda

Matthias
Merz

Doktorandenseminar
09.12.2004



- JDO 1.0.1
 - Überblick
 - Schwachstellen
- JDO 2.0
 - Neuerungen in JDO 2.0
 - Überschneidungen mit anderen Standards
 - Verschmelzung von JDO und EJB zu einem Persistenzstandard
- Problematiken bei der Verwendung von JDO 2.0
- Konzeption einer Persistenz-Architektur
 - Sicherheitsmodell
 - Ansatz zur Verteilung



Java Data Objects - Grundlagen

Matthias
Merz

Doktorandenseminar
09.12.2004

"The Java Data Objects (JDO) API is a standard interface-based Java model abstraction of persistence, developed as Java Specification Request 12 under the auspices of the Java Community Process. Application programmers use JDO to directly store their Java domain model instances into the persistent store (database)."

Quelle: Sun

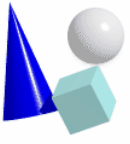
Aktueller Stand:

- Java Data Objects (JDO) Specification
- Seit Mai 2003: Version 1.0.1 (Maintenance Release)

In Planung:

- Java Data Objects 2.0 - An Extension to the JDO Specification
- Seit Juni 2004: Early Draft Review





Java Data Objects - Grundlagen

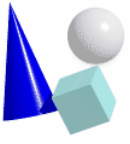
Matthias
Merz

Ziele von JDO:

- Transparente Persistenz
- Unabhängigkeit vom Datenspeicher
- Transaktionale Semantik
- Interoperabilität bezüglich
 - der JDO-Implementation
 - der verwendeten Plattform

Doktorandenseminar
09.12.2004





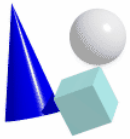
JDO-Klassentypen

Matthias
Merz

Doktorandenseminar
09.12.2004



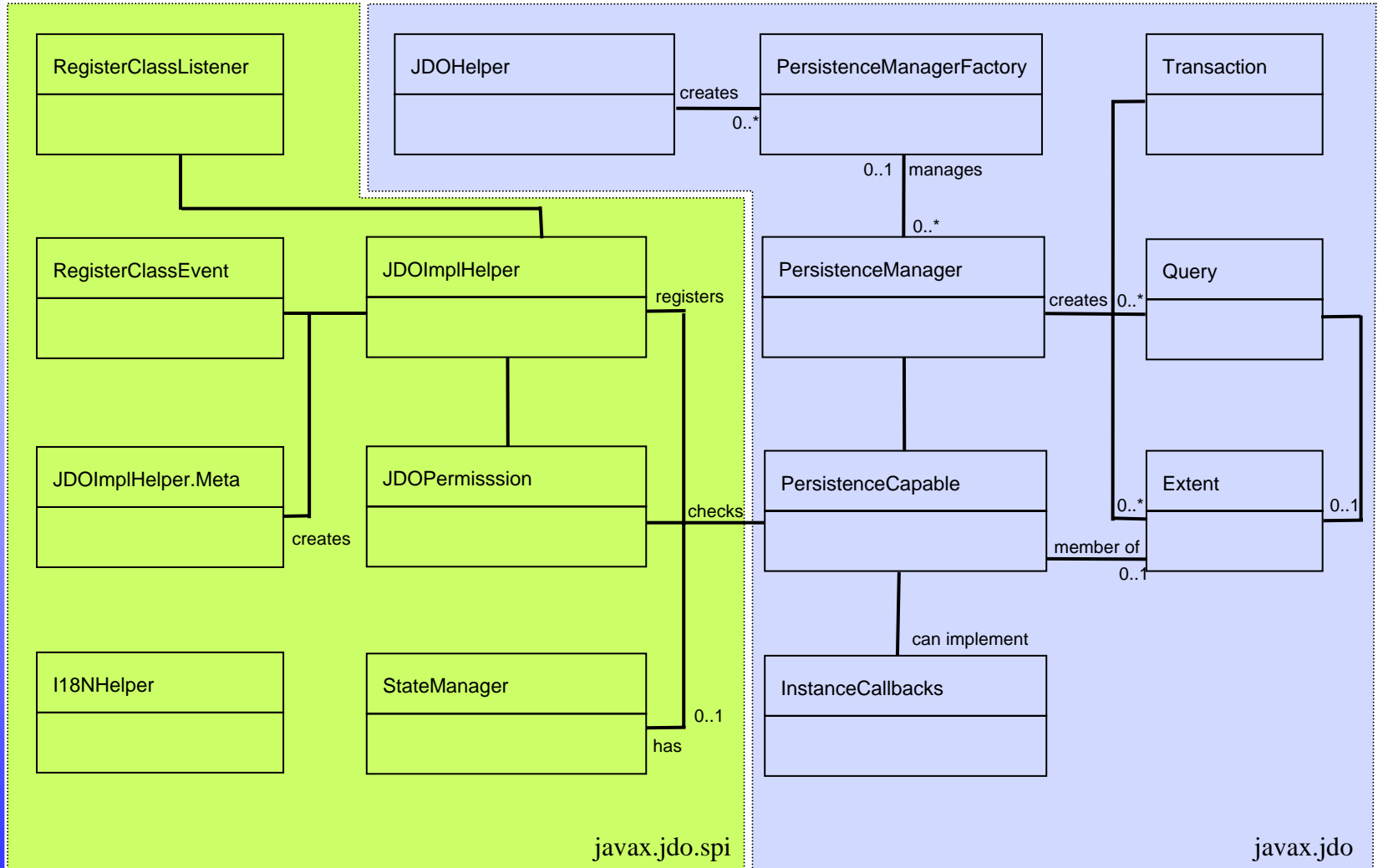
- *persistence-capable* (persistenzfähige Klassen)
 - Implementieren das `PersistenceCapable`-Interface
- *transient* (nicht persistenzfähig)
 - Klassen, die native Aufrufe verwenden
 - System-nahe Klassen in `java.lang`, `java.io`, `java.net` (z. B.: `Socket`, `Exception`, `Thread`)
- *persistence-aware*
 - Selbst nicht persistenzfähig
 - Greifen direkt auf `public` bzw. `protected` Attribute einer `PersistenceCapable` -Instanz zu



JDO Klassendiagramm

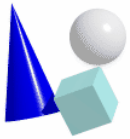
Matthias
Merz

Doktorandenseminar
09.12.2004



javax.jdo.spi

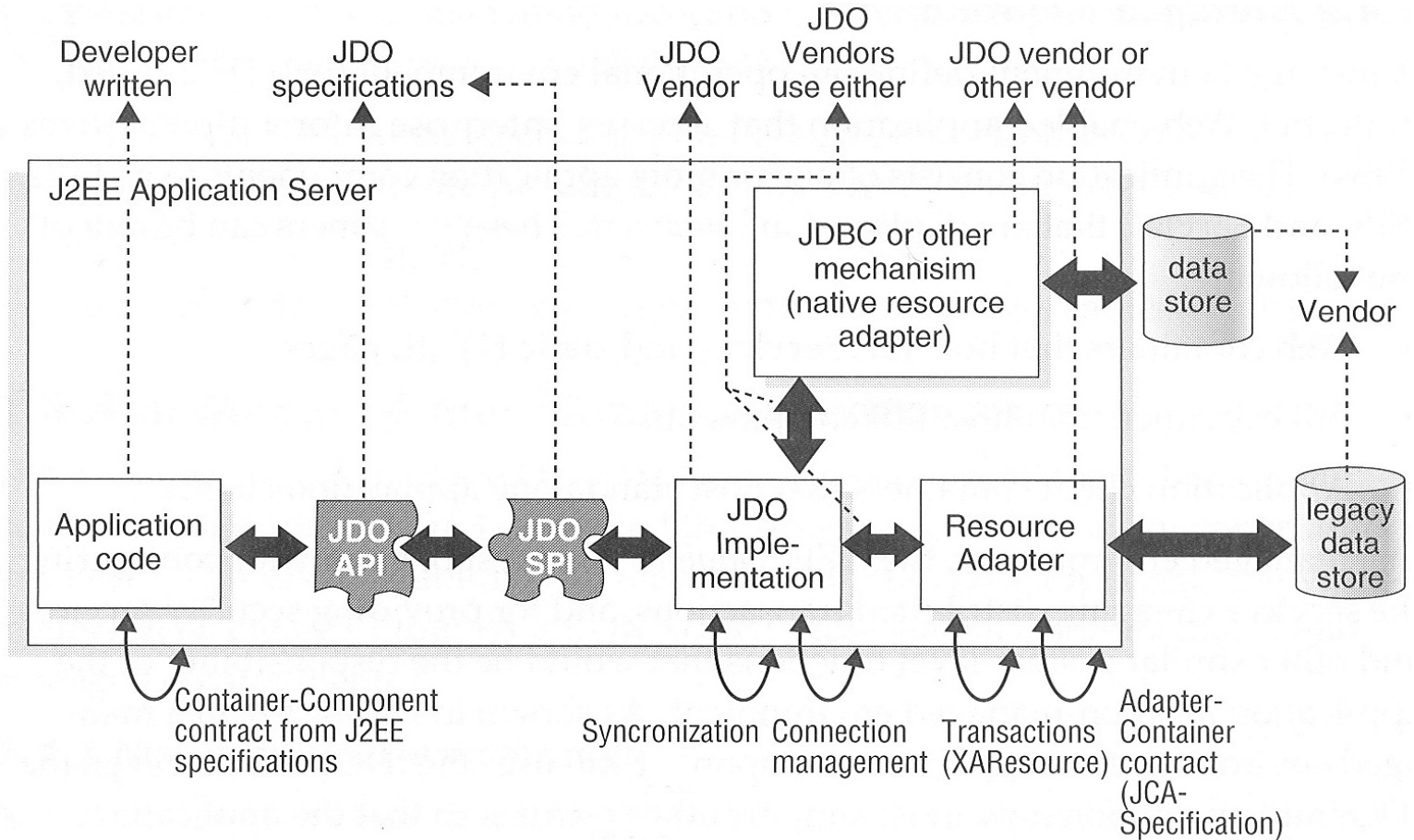
javax.jdo



Managed Environments

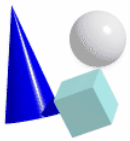
Matthias Merz

Doktorandenseminar
09.12.2004



Quelle: Core Java Data Objects

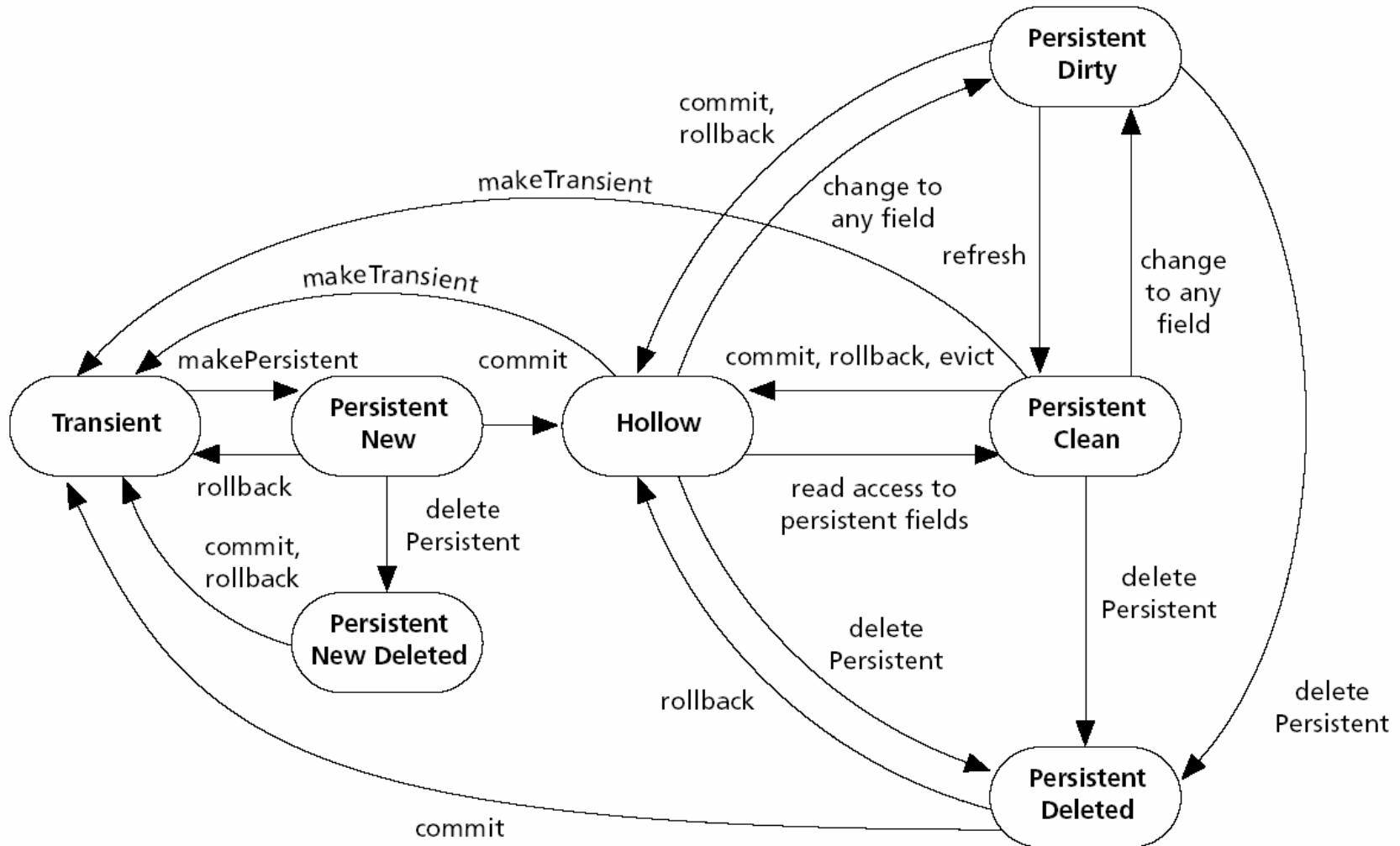




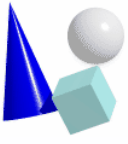
JDO-Objektlebenszyklus

Matthias
Merz

Doktorandenseminar
09.12.2004



Quelle: Roos 2002, S. 44



Kritische Würdigung von JDO 1.0.1

Matthias
Merz

Doktorandenseminar
09.12.2004

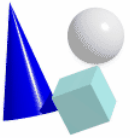


Positiv:

- + Unabhängigkeit vom Datenspeicher
- + Transparente Persistenz
- + Vererbungskonzept wird voll unterstützt
- + Integrationsmöglichkeit in J2EE
- + Lazy Data Loading, Default Fetch Group
- + Unabhängigkeit von der konkreten JDO-Implementierung (?)

Negativ:

- JDOQL
- Enhancement-Vorgang
- Keine Verwaltung bidirektionale Objekt-Beziehungen
- Fehlendes Sicherheitskonzept
- Keine Unterstützung für verteilte Anwendungen



Java Data Objects 2.0

Matthias
Merz

JDO 2.0

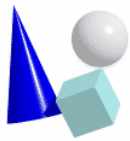
- JSR 243: JDO 2.0 - An Extension to the JDO Specification
- Early Draft Review seit Juni zum download bereit
- Final Release angekündigt für Mitte 2005

Erweiterungsvorhaben:

- Vereinfachung und Erweiterung der JDOQL
- Standardisiertes Object/Relational (O/R)-Mapping
- Managed Relationships
- Detach/Attach Methoden
- Standardisierung von JDO Vendor-Exceptions

Doktorandenseminar
09.12.2004





Java Data Objects 2.0

Matthias Merz

JSR #243















Java™ Data Objects 2.0 - An Extension to the JDO specification

JSR Review Ballot

These are the final results of the JSR Review Ballot for JSR #243. The Executive Committee for SE/EE has approved this ballot.

Votes

SE/EE

Apache Software Foundation 	Apple Computer, Inc. 	BEA Systems 	Borland Software Corporation <input type="checkbox"/>
Caldera Systems 	Fujitsu Limited 	Hewlett-Packard 	IBM 
IONA Technologies PLC 	Lea, Doug 	Macromedia, Inc. 	Nokia Networks 
Oracle 	SAP AG 	Sun Microsystems, Inc. 	

Icon

Legend

Yes 

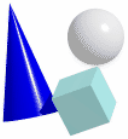
No 

Abstain

Not voted

Doktorandenseminar
09.12.2004





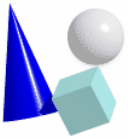
Überschneidungen mit anderen Standards

Matthias
Merz

- Service Data Objects (SDO)
 - Standard von IBM und BEA zur Übertragung eines Objektgraphen zwischen Client und Server
 - Eingereicht als JSR-235
 - Basiert auf dem DTO-Pattern
- Enterprise JavaBeans
 - Eigenes Persistenzkonzept für J2EE-Umgebungen
 - CMP/BMP basiert auf relationalen Datenbanken
 - Keine Nutzung in J2SE möglich
 - Aktuell: JSR-220: Enterprise JavaBeans 3.0

Doktorandenseminar
09.12.2004





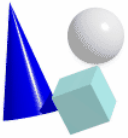
A Letter to the Java Technology Community

Matthias
Merz

- September 2004: Offener Brief der Specification Leads Linda DeMichiel und Craig Russell
- Ziel: Zusammenführen beider Persistenz-APIs:
 - Erweiterung der JSR-220 Expert Group um Mitglieder der JSR-243-Gruppe
 - Entwicklung eines einzigen Persistenzmodells
 - Nutzbar für J2SE und J2EE
 - Persistenzmodell als separate Spezifikation innerhalb des JSR-220
 - JDOQL wird neben EJBQL für das neue Persistenz-Framework nutzbar sein

Doktorandenseminar
09.12.2004





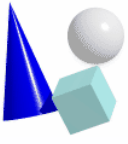
Offen in JDO 2.0

Matthias
Merz

Doktorandenseminar
09.12.2004



- Verteilung:
 - In J2EE: Ansätze derzeit mit Session Bean Facade und Übertragung primitiver Typen (DTO-Pattern)
 - Außerhalb von J2EE: Verteilung bleibt vollständig dem Anwendungsentwickler überlassen - lediglich in JDO 2.0 ist die Abkoppelung von **PersistenceCapable** Instanzen vom **PersistenceManager** vorgesehen
- Sicherheit:
 - Keine Authentifizierung auf Ebene der Persistenzschicht
 - Keine Definition von Benutzerrollen
 - Keine Prüfung von Rechten zur Persistenzierung bzw. Löschen von Objekten



Beispiel: Ausschnitt einer Session Bean

Matthias
Merz

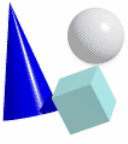
```
public Collection getAddresses(){
    PersistenceManager pm = null;
    List results = null;

    pm = jdoPMF.getPersistenceManager();
    Query q = pm.newQuery(Address.class);
    Collection jdoResult
        = (Collection) q.execute();
    results = new ArrayList(jdoResult);

    pm.close();
    return results;
}
```

Doktorandenseminar
09.12.2004





Konzeption

Matthias
Merz

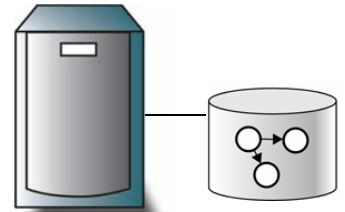
Verteilung

- Aufteilung des **PersistenceManagers** in eine *local* und *remote* Komponente



```
pml.newQuery(Address.class);
```

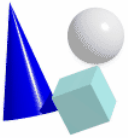
```
pmr.newQuery(Address.class);
```



- Kommunikation zwischen beiden Teilen noch offen.
Zu Prüfen ist der Einsatz von:
 - Service Data Objects (SDO)
 - Remote Message Invocation (RMI)
 - etc.

Doktorandenseminar
09.12.2004





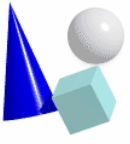
Fehlendes Sicherheitsmodell

Matthias
Merz

- Sicherheit nur bezüglich des Auslesens der Metadaten von **PersistenceCapable** Klassen gewährleistet
- Nach Anmeldung an einer Ressource kann über den **PersistenceManager** mit
 - **getObjectById(...)** auf beliebige Objekte zugegriffen bzw. mit
 - **deletePersistence(...)** beliebige Objekte gelöscht werden
- Zudem besteht prinzipiell die Möglichkeit
 - Laufende Transaktionen zu deaktivieren
 - Persistente Objekte transient zu machen

Doktorandenseminar
09.12.2004





Konzeption

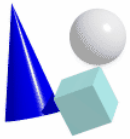
Matthias
Merz

Doktorandenseminar
09.12.2004



Sicherheit

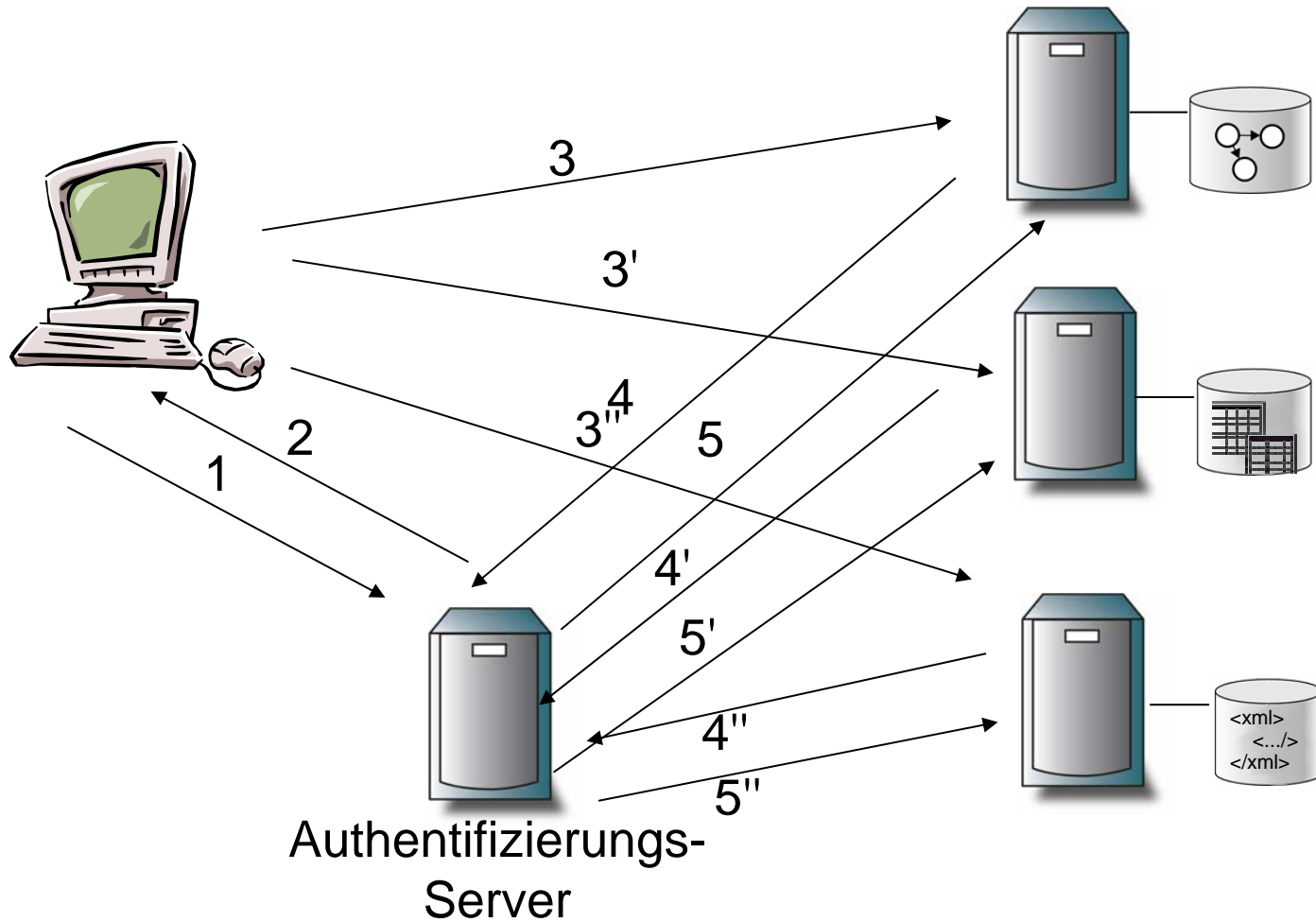
- Definition von Benutzer-Rollen
- Explizite Rechte-Verwaltung unterteilt nach
 - Funktionalen Aspekten: Löschen, Erzeugen, Update, etc.
 - Fein-granular auf Ebene von Objekten, Klassen, Subklassen und Paketen
- Verwaltung der dazu notwendigen Informationen in einem LDAP-Server (Zugriff über eigene JDO-Implementation)
- Single-Sign-On Mechanismus

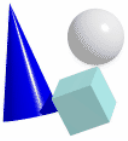


Beispiel: Heterogene DB-Landschaft

Matthias
Merz

Doktorandenseminar
09.12.2004





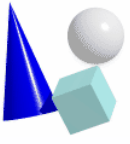
Fazit:

Matthias
Merz

Doktorandenseminar
09.12.2004

- Mit dem vorgestellten Ansatz können zwei wesentliche Probleme im JDO-Umfeld behoben werden
- Bisher scheinen die angesprochenen Probleme nicht durch die Verschmelzung von JDO und EJB zu einem neuen Persistenz-Standard berührt zu werden
- Herausforderungen liegen im Detail
 - Weiterentwicklung des neuen Persistenz-Standards; noch viele Überraschungen möglich!
 - Komplexität des JDO-Lebenszyklus





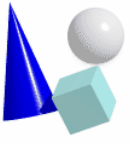
Weiteres Vorgehen:

Matthias
Merz

- Analyse des neuen Persistenz-Standards
- Konzept des Sicherheitsmodells ausarbeiten und verschiedene Realisierungsmöglichkeiten prüfen
- Unterschiedliche Szenarien zur Aufteilung des **PersistenceManagers** in eine *locale* und *remote* Komponente bewerten und verschiedene Ansätze zur Implementierung gegenüberstellen

Doktorandenseminar
09.12.2004





Danke für die Aufmerksamkeit!

Matthias
Merz

- Fragen?

Doktorandenseminar
09.12.2004

