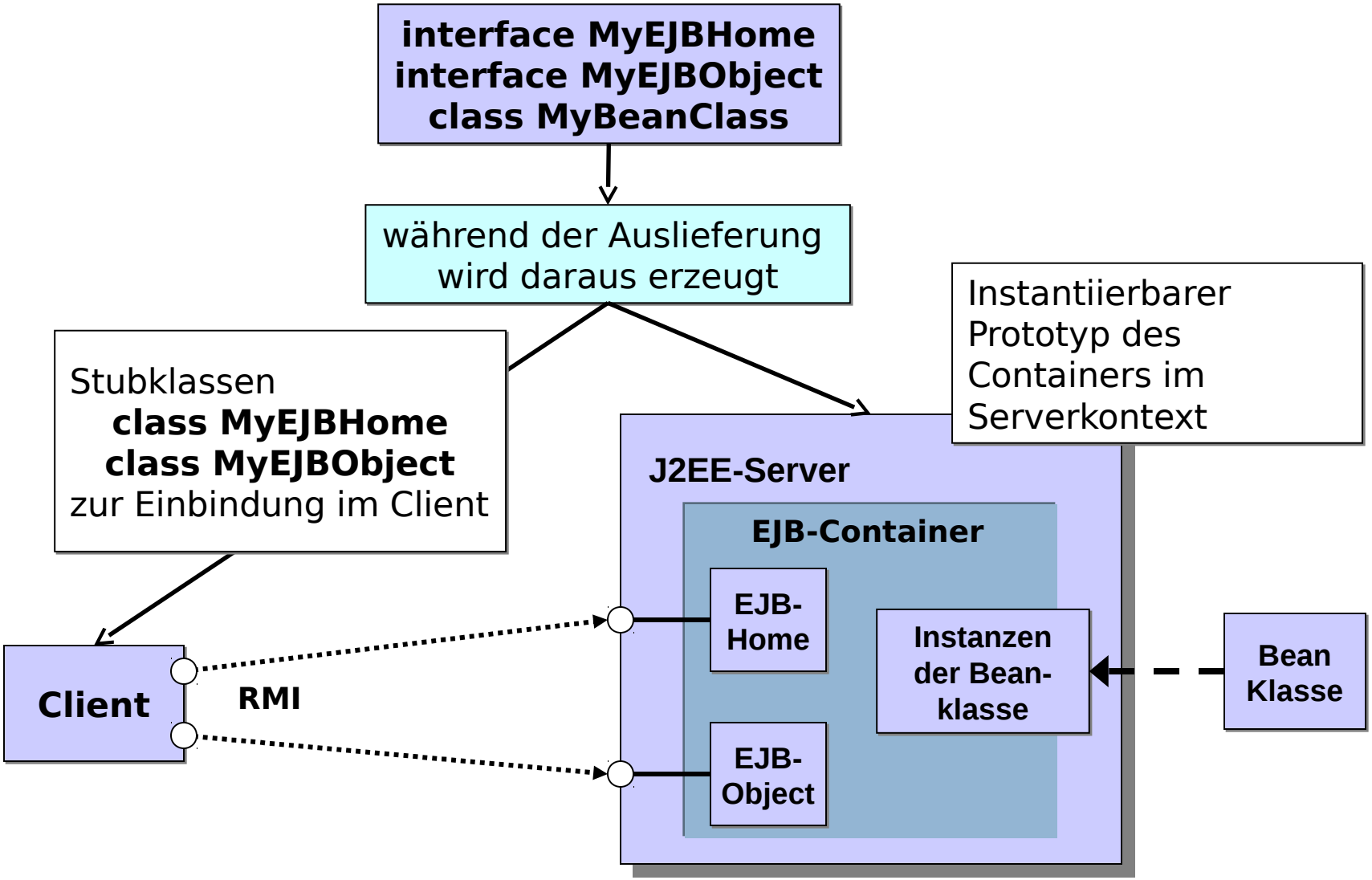
Vorlesung
Software aus Komponenten

3. Komponentenmodelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2015/16

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Anforderungen an den EJB-Container-Kontext

- Funktionalität der JVM wird benötigt, ist aber allein nicht ausreichend
- EJB-Standard spezifiziert Schnittstelle und Verhalten von Container und J2EE
- Container benötigt zur Verwaltung seiner Beans
  - Namensdienst (Java Name and Directory Interface)
  - Transaktionsmonitor (Java Transaction API)
  - Datenbankzugriff (Java Data Base Connectivity)
  - eMail (Java Mail API)
  - Standard Java API
- Greifen dabei gewöhnlich auf weitere Dienste im Rahmen des J2EE Applikationsservers zu

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bean-Schnittstelle

```
package SemOrg.Schnittstellen;
public interface Buchung extends javax.ejb.EJBObject {
    public void buchen(Kunde k, Seminartyp s)
        throws java.rmi.RemoteException;
}
```

Container-Schnittstelle

```
package SemOrg.Schnittstellen;
public interface BuchungHome extends javax.ejb.EJBHome {
    public Buchung create(int owner_id)
        throws java.rmi.RemoteException, javax.ejb.CreateException;
}
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bean-Klasse

```
package SemOrg.Server;
public class BuchungBean implements javax.ejb.SessionBean {

    public BuchungBean() {}

    // Geschäftsmethoden, die vom Client aufgerufen werden.
    public void buchen(Kunde k, Seminartyp s)
        throws java.rmi.RemoteException {
        // Code zur Ausführung der Buchung
    }

    // EJB-eigene Methoden, die vom Container aufgerufen werden,
    // niemals aber vom Client.
    private javax.ejb.SessionContext ctx;
    public void setSessionContext(javax.ejb.SessionContext sc) {
        this.ctx=sc; // Session-Context für Callback-Methoden
    }
    public void ejbCreate() {/* ... */ }
    public void ejbRemove() {}  ...
}
```

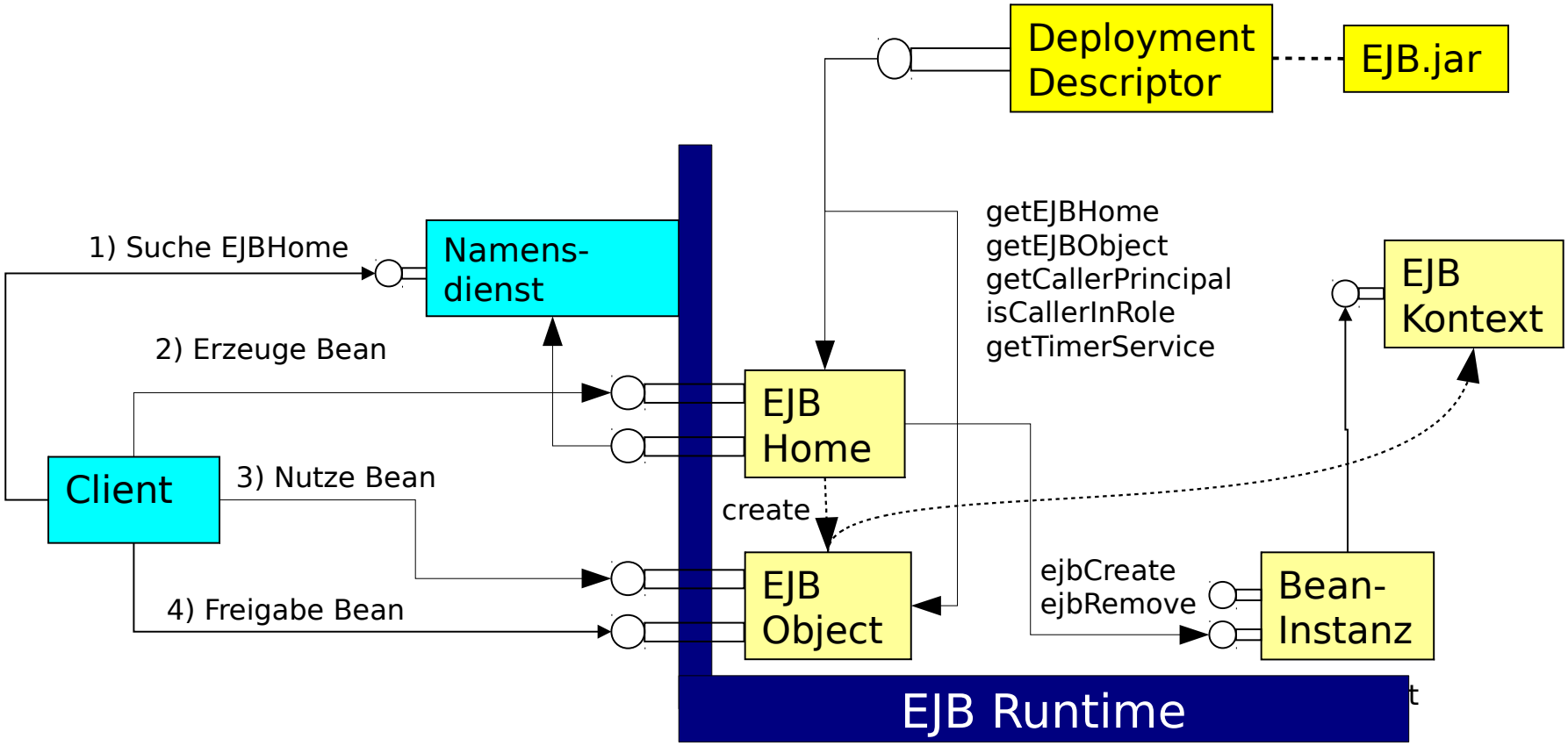
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Deployment-Deskriptor

```
<!-- Deployment descriptor -->
<enterprise-beans>
  <session>
    <ejb-name>SemOrg/Buchung</ejb-name>
    <home>SemOrg.Schnittstellen.BuchungHome</home>
    <remote>SemOrg.Schnittstellen.Buchung</remote>
    <ejb-class>SemOrg.Server.BuchungBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Container</transaction-type>
  </session>
</enterprise-beans>
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Architektur von EJB 2



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Container

- Die EJB 2.x Architektur geht davon aus, dass der EJB Server Hersteller auch der Containerhersteller ist.
- Der Container wird im Rahmen des Deployments der Komponente automatisch vom Server bereitgestellt und die Komponente im Namensdienst registriert.

Componenten-Container-Modell

Um Komponenten von den Spezifika der unterliegenden Infrastruktur abzuschirmen, werden Container als standardisierte Laufzeitumgebungen eingeführt.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Ein einfacher Client

```
public class Client {

   public static void main(String[] args) {
      Client einClient=new Client();
      einClient.run();
   }

  public void run() {
      Kunde einKunde=getKunde();
      Seminar einSeminar=getSeminar();

      try {
      // Namenskontext des Servers finden
      javax.naming.Context ctx = new javax.naming.InitialContext();
      Object temp = ctx.lookup(“java:comp/env/Buchung”);
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

```
   // EJB-Container-Objekt vom Typ BuchungHome
    // auf dem Server finden und instanziieren
    BuchungHome home= (BuchungHome) PortableRemoteObject.narrow(
                  temp, BuchungHome.class);

    // Bean-Instanz anfordern und Buchung auslösen
    Buchung bean = home.create();
    bean.buchen(einKunde, einSeminartyp);
  }
  catch(Exception e) { }
}
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Deployment-Kontrakt

Komponente wird in einem speziellen Paketformat ausgeliefert, das spezifische Metainformationen im XML-Format enthält

- Dienst-Schnittstelle, Factory-Schnittstelle, Bean-Implementierung, Ressourcen-Zuordnung, Laufzeitbedingungen

Lebenszyklus-Kontrakt

Komponente implementiert spezielle Lebenszyklus-Events, die vom Container „automagisch“ aufgerufen werden können

- OnActivate() {…}
- OnPassivate() { … }

Container-Service-Kontrakt

Der Container stellt der Komponente ein Kontext-Objekt zur Verfügung, über welches die Komponente transparent Dienste aus dem Kontext in Anspruch nehmen kann.

- WhoIsCaller() { … }
- AccessDataBase() { … }
- LocateComponent() { … }

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Umgebungs-Kontrakt

Der Container sichert eine funktionierende Umgebung für die Komponente entsprechend der Deployment-Information.

Erweiterungs-Kontrakt

Der Container kann selbst erweiterbar sein und Verhaltensänderungen ausgerollter Komponenten unterstützen

- nutzergetriebene Unterbrechungen
- Unterstützung der Laufzeitkonfiguration von Einheiten
- Einbindung weiterer Dienste zur Laufzeit
- Versionsmanagement ausgerollter Komponenten
- Aspektorientiertes Verhalten

Client-Container-Kontrakt

Client nutzt Komponentendienste über den Container.

Framework bietet Dienste zum Auffinden der Komponente.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Probleme des Standards

- Viele Schnittstellen zu implementieren (EJBHome, EJBObject, callback zum Kontext)
- Nur eine Geschäftsmethoden-Schnittstelle pro EJB Bean
- Weitere Konventionen sind zu beachten (RMI, spezielle Basis-Schnittstellen)
- Zusätzliche und andere Konventionen für EJB-Klassen im Vergleich zu normalen (plain old) Java-Klassen
- Konfiguration von Beans und Applikationen für Beans erfolgt durch riesige XML-basierte Deployment-Deskriptoren
- EJB-Laufzeit zu rudimentär spezifiziert
- Komplexität der Interaktion mit der Persistenzschicht
- Keine Schnittstellen für Logging, Tracing und andere Basisdienste, um Komponenten zu testen und in der Laufzeit zu verfolgen.
- Beans müssen vom Client manuell gesucht und angesprochen werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Ziele von EJB 3

- Genauere Spezifikation des Bean-Lebenszyklus durch mehr Erweiterungspunkte im Kontext
- Dependency Injection: Einzelne Attributwerte, Methoden- oder Klassendefinitionen werden zur Laufzeit aus dem Kontext importiert
- Verwendung von Meta-Daten
- Interzeptoren und aspektorientierte Ansätze für Infrastrukturdienste
- Vereinfachung der Persistenzbehandlung
- Reduktion der Zahl der Artefakte, die ein Bean-Entwickler bereitstellen muss
- Vereinfachung der EJB Typen durch Reduktion der Zahl der Schnittstellen
- Verbesserung der Testmöglichkeiten außerhalb des Containers