Stoffabgrenzung Enterprise Applications

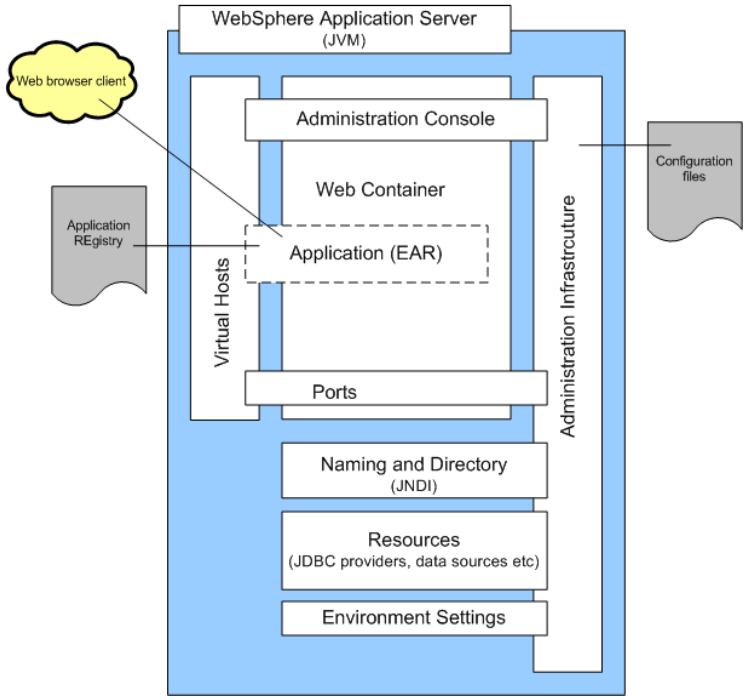
Inhalt

Fragen werden im Umfang aus den Vorlesungspräsentationen und Vorlesungsunterlagen auf ILIAS gestellt. Dabei ziehen Sie bitte ihre Vorlesungsnotizen zu Hilfe. Aus dem Tooling-Script werden auch Fragen gestellt. Die Fragen sind in ähnlichem Umfang wie die Repetitionsfragen jeweils zu Beginn der Vorlesungen. Nachfolgend ein Auszug von Topics aus dem Stoff die Gegenstand der Fragen sein können:

Abgrenzung

- Es werden Fragen gestellt die in den oben aufgeführten Vorlesungen diskutiert wurden.
- Es werden Fragen gestellt über Ihre in den Übungsstunden entwickelte Applikation. Z.B. „Skizzieren Sie die Architektur Ihrer Persistenzschicht“.
- Es werden keine Detailfragen bezüglich der Implementation ihrer Web Shop Applikation gestellt.
- Es werden Verständnis-Fragen gestellt um zu prüfen ob sie die grösseren Zusammenhänge begriffen haben (z.B. mit welcher Technik kann ich ein bestimmtes Problem lösen).

The Basic Architecture of WebSphere Application Server



Quelle: https://www.packtpub.com/sites/default/files/Article-Images/websphere-article1-image01.png

1 Tooling Script

1.1 Der Inhalt vom Tooling Script wird befragt. Sie müssen keinen Code schreiben.

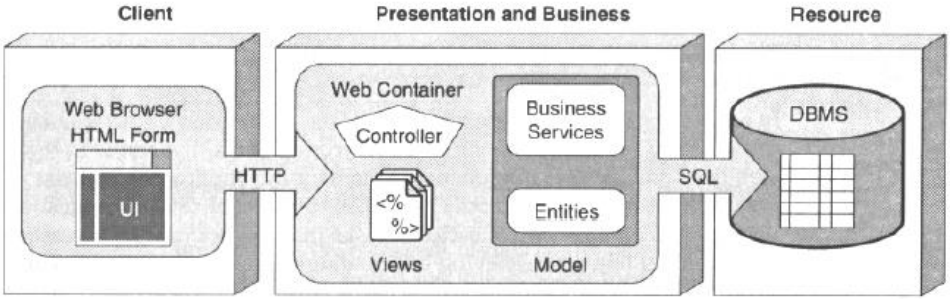
2 Servlets

2.1 Das Paper: “MVC Design Pattern for the multi framework distributed applications using XML, spring and struts framework” muss verstanden sein. Der Teil von “Struts” ist nicht Prüfungsrelevant.
http://www.enggjournals.com/ijcse/doc/IJCSE10-02-04-48.pdf

2.2 Tiers und Architekturen

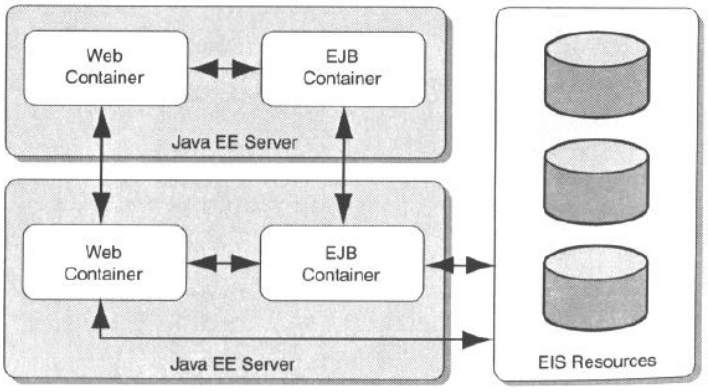
Web zentrische Architektur

- *Verwendet nur Web Container der EE Technologie*
- *Der Web Server beinhaltet alle benötigten Komponenten*



B2B Architektur

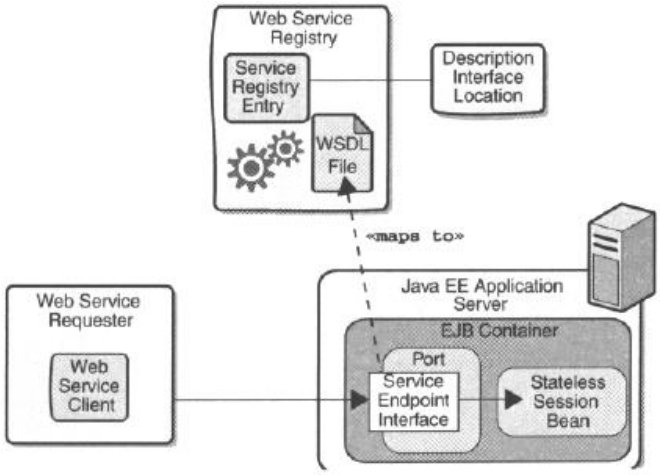
- *Erweiterung der Komponenten basierten Architektur*
- *Verwendet zwei EJB Server, jeder besteht aus Web- und EJB Container*



EIS: enterprise information systems

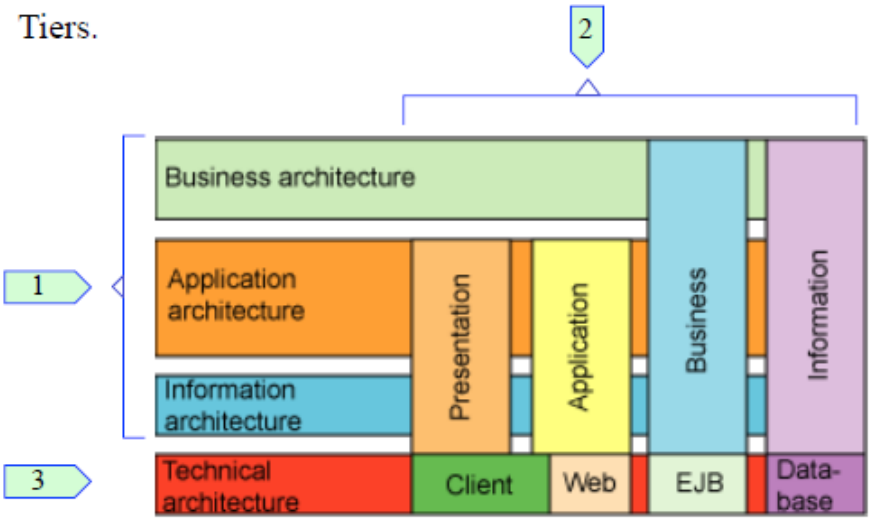
Web Service Architektur

- *Eine EE Business-Komponente veröffentlicht Funktionalität*
- *Stateless Session Bean dient als Zugriffspunkt für die Servicefunktion*



Tiers / Tier to Layer Mapping

1. *Conceptual Layer: (Business-, Application-, Information- and Technical Architecture.*
2. *Logical Tiers: Client-, Web-, EJB- and Information Tier.*
3. *Physical Tiers: mapping of Technical Tiers and Locical Layers to Physical Tiers.*



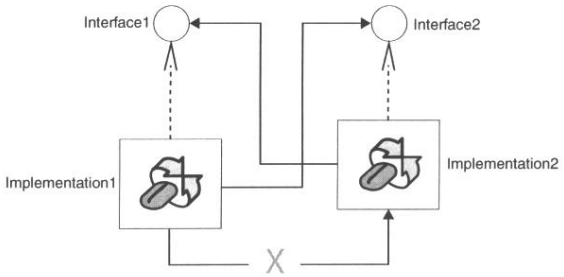
2.3 Komponenten basiertes entwickeln

Allgemein

Komponenten…

- *haben Status und Eigenschaften*
- *werden vom Container gekapselt (verwaltet)*
- *unterstützen lokale und verteilte Komponenten Interaktionen*
- *erzeugen Standort Transparenz*
- *können referenziert werden unter Verwendung von Namensservices*

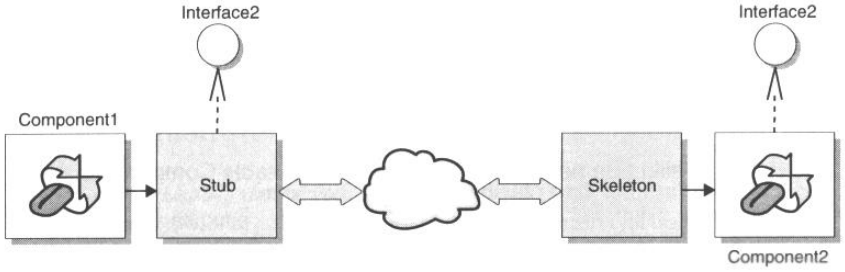
Interaktion von EE Komponenten



- *Die meisten Komponenten sind definiert und kontrolliert durch Interfaces*
- *Die eine Komponente kommuniziert durch das Interface der anderen*
- *Auch wenn Beide in der gleichen Java Virtual Machine laufen!*
- *Die Komponenten Infrastruktur stellt dafür auch Proxies zur Verfügung*

RMI Infrastruktur für Verteilte Komponenten

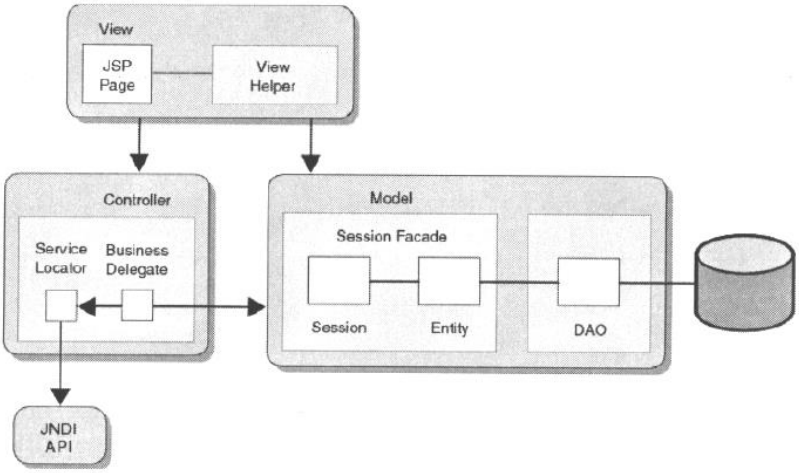
- *Marshalling und Unmarshalling von Argumenten und Rückgabewerten*
- *Übergeben von verteilten Exceptions*
- *Übergeben von Security- und Transaktions- Kontext zwischen Aufrufer und Ziel*



2.4 Das MVC Pattern und die Service Locator, View Helper, Session Facade und DA-Patterns.

Java EE Patterns

- ***Service Locator***
  Business Tier Pattern, abstrahiert omponenten vom Lookup und Connect Mechanismus zu entfernten Objekten.
- ***View Helper***
  Präsentations- Tier Pattern, für Komponenten die Hilfsfunktionen anbieten (z.B. Datenobjekte erzeugen) die von View Komponenten benötigt werden.
- ***Session Facade***
  Business Tier Pattern, das business Funktionen veroeffentlicht, die als grobkörnige Dienste in den Businesslogik Komponenten des EJB Tiers implementiert sind.
- ***Data Access Object (DAO)***
  Integrations Pattern, erzeugt Komponenten die Daten kapselt. Diese werden zur Interaktion mit der Datenbank verwendet.

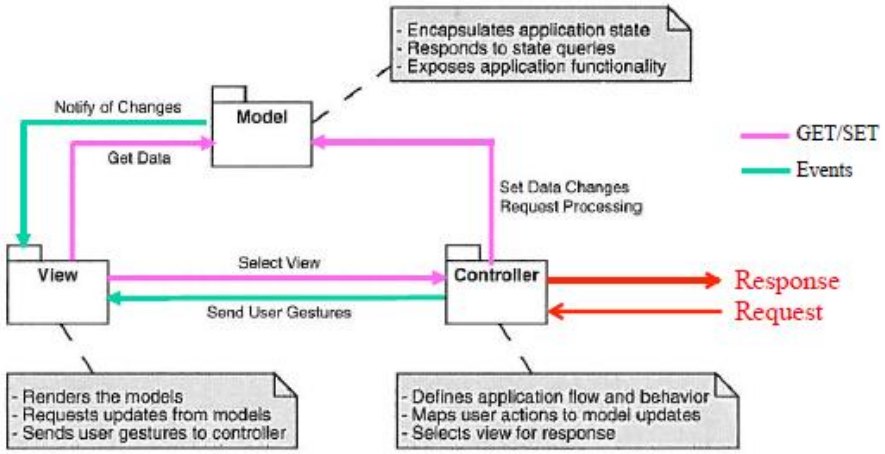


MVC Limitation:

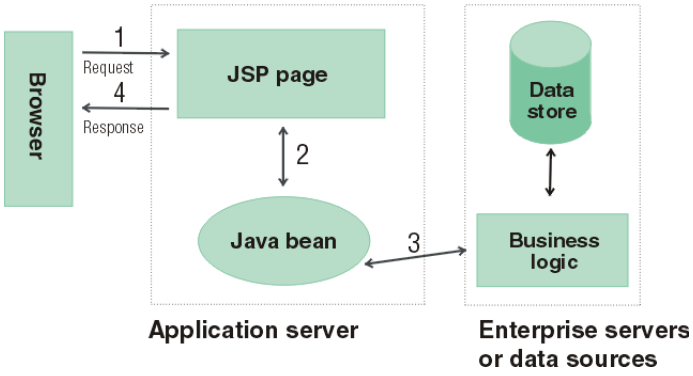
Das MVC Pattern ist für Web Applikationen geeignet. Weil das Web aber „stateless" ist, die Clients verteilt sind und das Web User Interface Limitationen aufweist, braucht es einige Anpassungen. Um genau zu sein, im traditionellen MVC Pattern beobachtet die View das Model um bei Änderungen automatisch das Data-Display anzupassen wenn die Modeldaten ändern. Web Applikationen zeigen normalerweise geänderte Daten nur wenn der Benutzer eine neue Anfrage startet (Get/Post).

MVC nach GoF

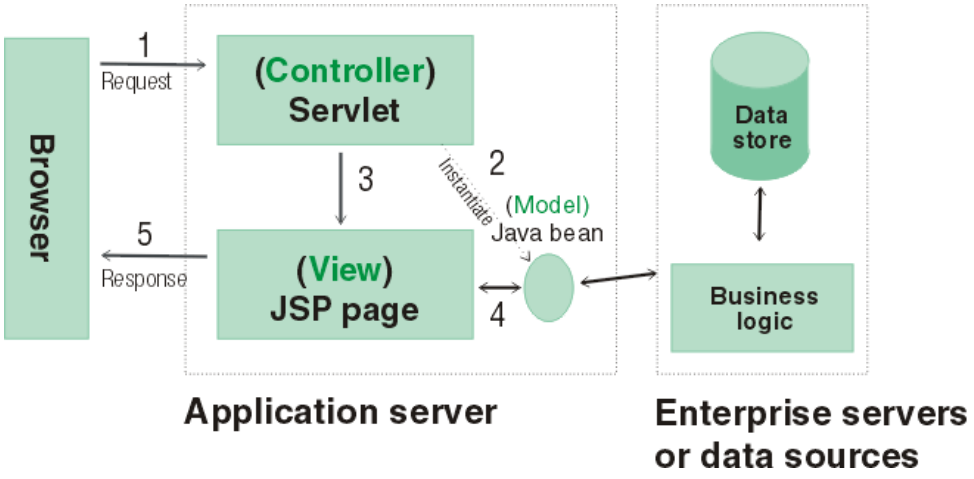
- *Model: Kapselt den Applikationsstatus, Gibt Antworten zu Statusanfragen, legt die Applikationsfunktionalität frei*
- *View: Rendering, Erzwingt Updates vom Model, gibt Useraktionen zum Controller, Erlaubt dem Controller die View zu selektieren*
- *Controller: legt Applikationsverhalten fest, verbindet Useraktionen mit Model-Updates, Selektiert eine View für die Antwort, Ein Controller für jede Funktionalität*

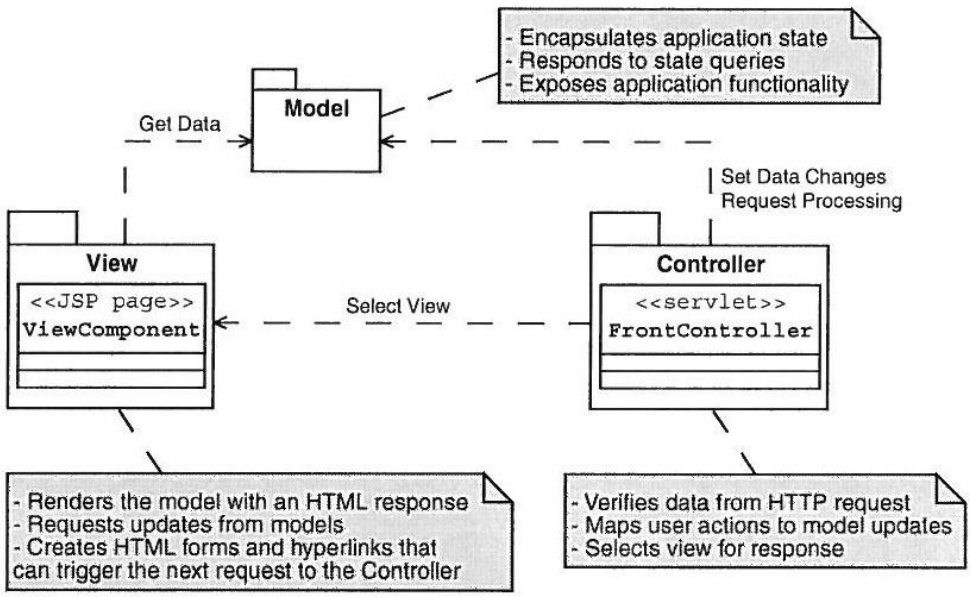


MVC Model 1



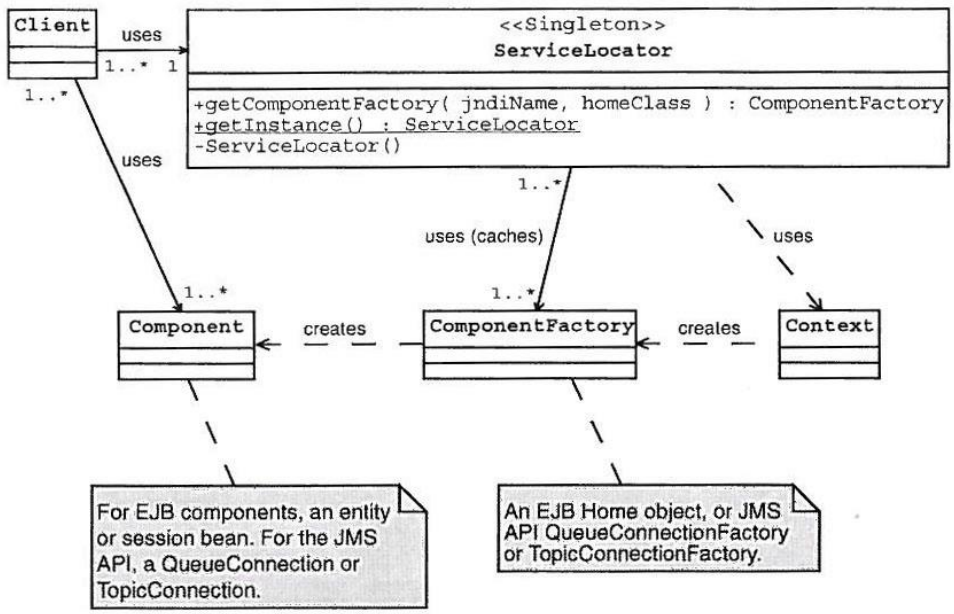
MVC Model 2





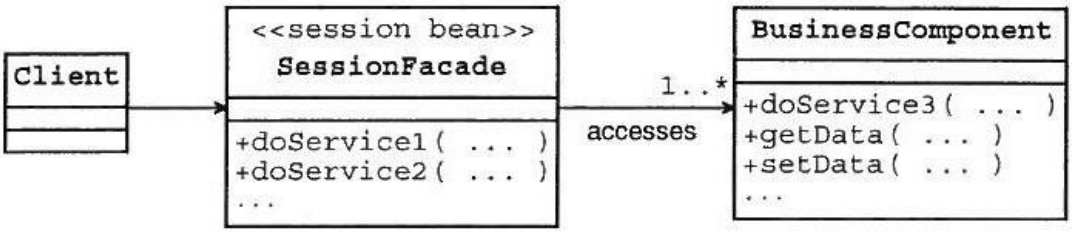
- *Benutzer-Anfragen gehen zum „FrontController“ Servlet*
- *allgemeine Aufgaben wie Authentifizierung und Validierung. Dieses Servlet benachrichtigt Business Prozesse*
- *sendet Prozess Aufrufe zu den Model-Komponenten*
- *Model-Komponenten können JavaBean Komponenten, Enterprise JavaBean Komponenten oder PoJOs sein*
- *FrontControlle Servlet entscheidet (aus Anfrage sowie Resultaten vom Business Prozess welche View der Benutzer erhält*
- *FrontController Servlet oder ein Helper Objekt versendet die Anfrage zur passenden View / JSP Page*

Service Locator Pattern

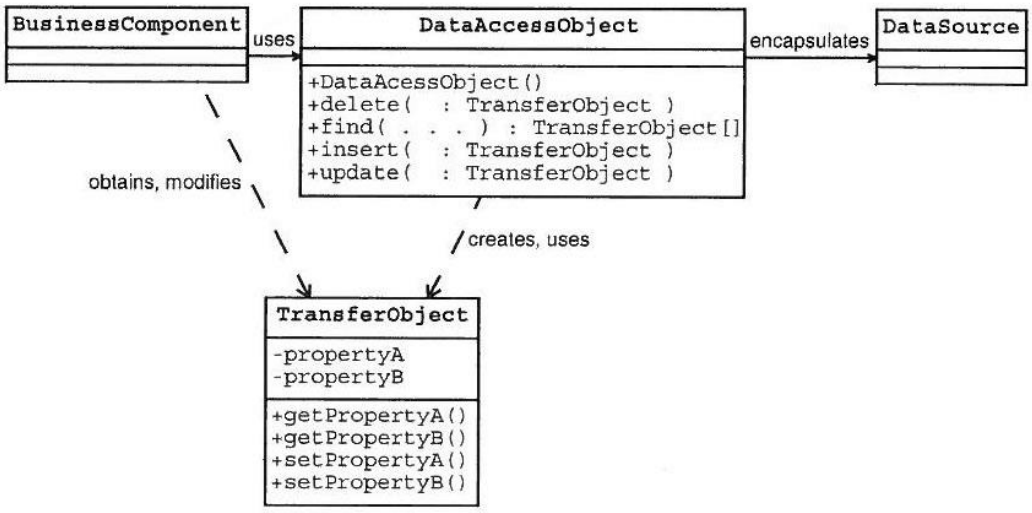


- *ServiceLocator Klasse ist normalerweise Singleton.*
- *ServiceLocator fuehrt lookup Prozess aus*
- *Client verlangt nach Service der ServiceLocator Klasse.*
- *ServiceLocator Objekt erzeugt JNDI Kontext Objekt*
- *ServiceLocator Objekt macht lookup auf ComponentFactory.*
- *ComponentFactory Objekt*
  - *EJB home*
  - *JMS API Connection Factory*
  - *JDBC Data Source sein*
- *ServiceLocator Objekt gibt ComponentFactory Objekt zurück.*
- *Client erzeugt damit Component Instanz.*
- *ServiceLocator Objekt kann Component Instanzen erzeugen.*
- *Component Instanzen:*
  - *Beans*
  - *JMS API Connections*
  - *JDBC Connections sein*

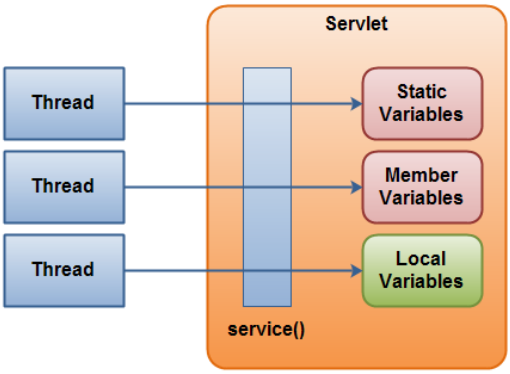
Session Facade Pattern Structure



DAO Pattern Structure



2.5 Thread safety

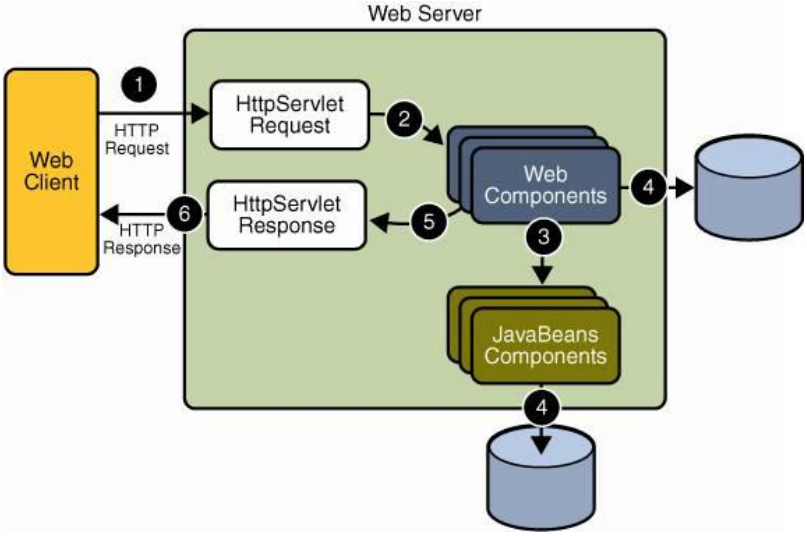


mehrere Threads gleichzeitig → alle greifen auf die gleichen Variablen zu → Concurrency beachten!

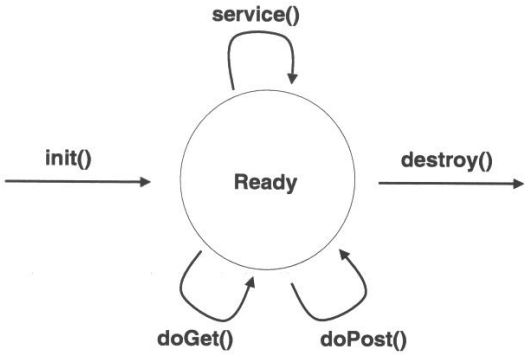
Jegliche Anzahl Threads können in der Service Methode gleichzeitig ausgeführt werden.

- ***Instanzvariablen mit Vorsicht verwenden***
  Sollten local - oder auch final – scope haben
- ***Klassenvariablen mit Vorsicht verwenden***
  synchronized() verhindert nicht, dass Klassenvavriablen nicht von anderen Threads gleichzeitig verwendet werden.
- ***Den Zugriff auf externe Ressourcen vorsichtig abwickeln***
  DB ist selten das Problem – eher Zugriff auf Filesystem
- ***Benutzen sie Synchronisationsprimitiven für kritische Abschnitte***
  synchronized(this){
  //Hier ist nur ein Thread gleichzeitig
  }

2.6 Servlet Programmier Modell

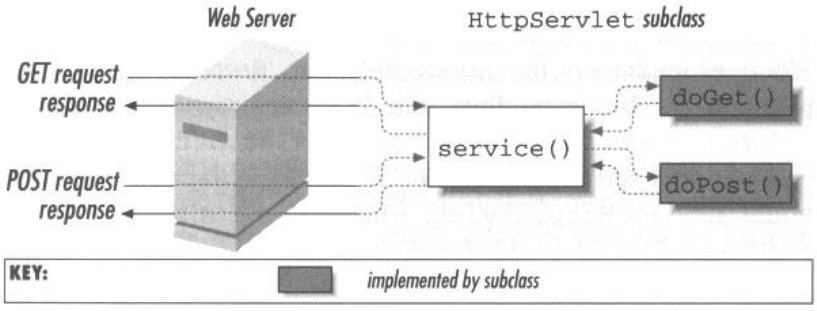


The Servlet Lifecycle

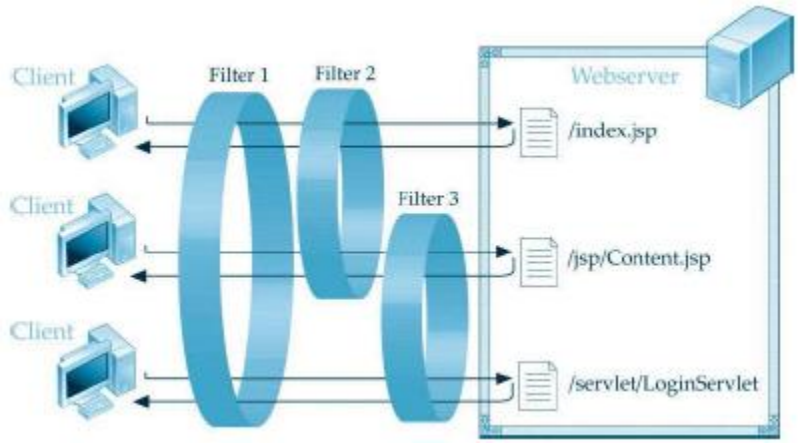


Was ist die Aufgabe eines Servlets?

- *Der Server identifiziert das Servlet gemäss der URL (web.xml)*
- *Der Server delegiert die Anfrage zum Servlet*
- *Das Servlet macht seine Arbeit (Aufruf EJB etc.)*
- *Das Servlet produziert eine Antwort (formatiert html Seite)*
- *Der Server sendet die Antwort zum Client zurück*



2.7 Server Domains mappings und Filter



Es können mehrere Filter hintereinander gekoppelt (kaskadieren) werden.
Die Filter werden im web.xml definiert und zugewiesen.

3 High Availability (JGroups)

3.1 Nicht Prüfungsrelevant

☺

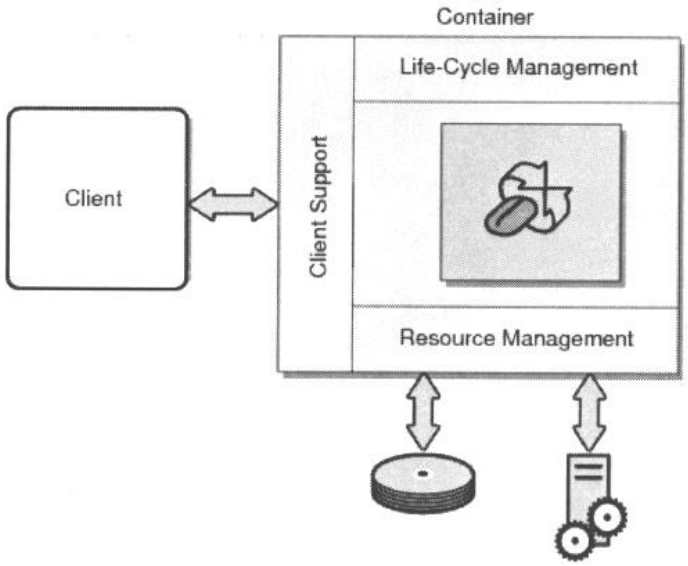
4 Enterprise Java Beans

4.1 Die Rolle von EJB Komponenten

Enterprise JavaBeans (EJB) sind standardisierte Komponenten innerhalb eines Java-EE-Servers (Java Enterprise Edition). Sie vereinfachen die Entwicklung komplexer mehrschichtiger verteilter Softwaresysteme mittels Java. Mit Enterprise JavaBeans können wichtige Konzepte für Unternehmensanwendungen, z. B. Transaktions-, Namens- oder Sicherheitsdienste, umgesetzt werden, die für die Geschäftslogik einer Anwendung nötig sind.

4.2 Die Rolle vom EJB Container

Der Container managed den Lebenszyklus der instantiierten EJBs.

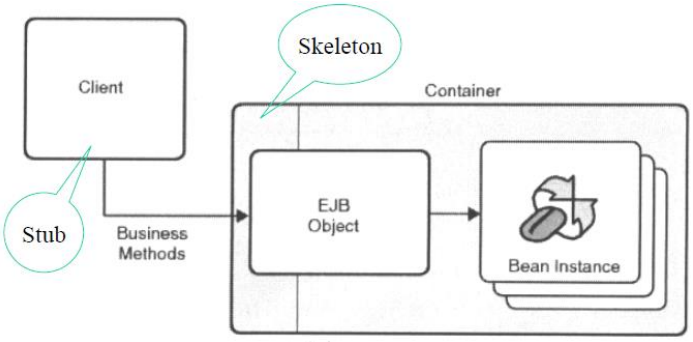


- *Der Container kapselt die Zugriffe zu externen Ressourcen.*
- *Der Container managed den Lebenszyklus der instantiierten EJBs*
- *Der Container isoliert die Klasse mit der Implementation von ihrem Client (Aufrufer). Clients müssen einen Aufruf durch OJBObject in den Container machen.*
- *Der Container stellt Timer Zeit Services zur Verfügung die es erlauben eine Methode zu einer bestimmten Zeit aufzurufen.*
- *Für Message-Driven Beans überwacht der Container eine Message Queue im Auftrage der EJB Komponente.*

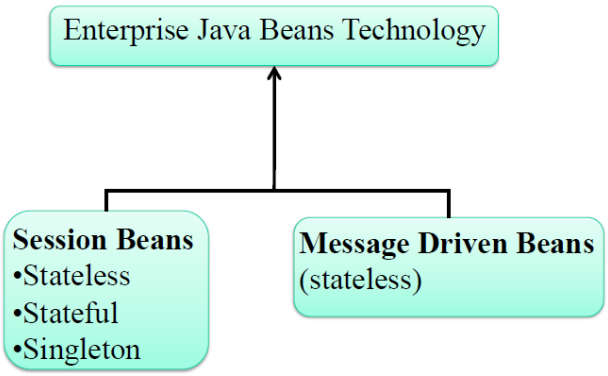
- *Remote Client Communication*
- *Dependency Injection*
- *State Management*
- *Pooling*
- *Component Life Cycle*
- *Messaging*
- *Transaction Management*
- *Security*
- *Concurrency Support*
- *Interceptors*
- *Asynchronous Method Invocation*

4.3 Proxy Modell

Wenn eine EJB Client-Komponente eine Session Komponente aufruft, gibt der Container eine Referenz zum Komponenten EJBObject zurück.



4.4 Typen von EJBs und deren Einsatz.



Was sind Enterprise Beans?

- *Serverseitige Komponenten in Java geschrieben die Business Logik zusammenfassen.*
- *Die Business Logik ist der Code der die Aufgabe der Applikation erfüllt.*
- *Wenn die Applikation eine Flugbuchungssystem ist, dann implementiert das Enterprise Bean Methoden wie zB. bookFlight oder payFlight. Wenn diese Methoden aufgerufen werden so können Clients Flight Services verwenden die von der Applikation zur Verfügung gestellt werden.*

Vorteile von Java Enterprise Beans

EB vereinfachen die Entwicklung von grossen verteilten Applikationen.

Differences between Message-Driven Beans and Stateless Session EJBs

In several ways, the dynamic creation and allocation of message-driven bean instances mimics the behavior of stateless session EJB instances. However, message-driven beans are different from stateless session EJBs (and other types of EJBs) in several significant ways:

- *Message-driven beans process multiple JMS messages asynchronously, rather than processing a serialized sequence of method calls.*
- *Message-driven beans have no home or remote interface, and therefore cannot be directly accessed by internal or external clients. Clients interact with message-driven beans only indirectly, by sending a message to a JMS Queue or Topic.*

4.5 Interaktionen von Beans in der geschichteten Architektur

Ein Client kann ein Session Bean nur über die im Business Interface definierten Methoden zugreifen (remote).

4.6 Client Zugriffe auf EJBs

Ein Client kann ein Session Bean nur über die im Business Interface definierten Methoden zugreifen (remote).
Gut geplante Interfaces vereinfachen den Unterhalt von Java EE Applikationen.
Die Typen eines Client Zugriffs sind:

- *local*
- *remote*
- *web service*

4.7 Methoden Typen

Lebenszyklus- oder Call Back Methoden

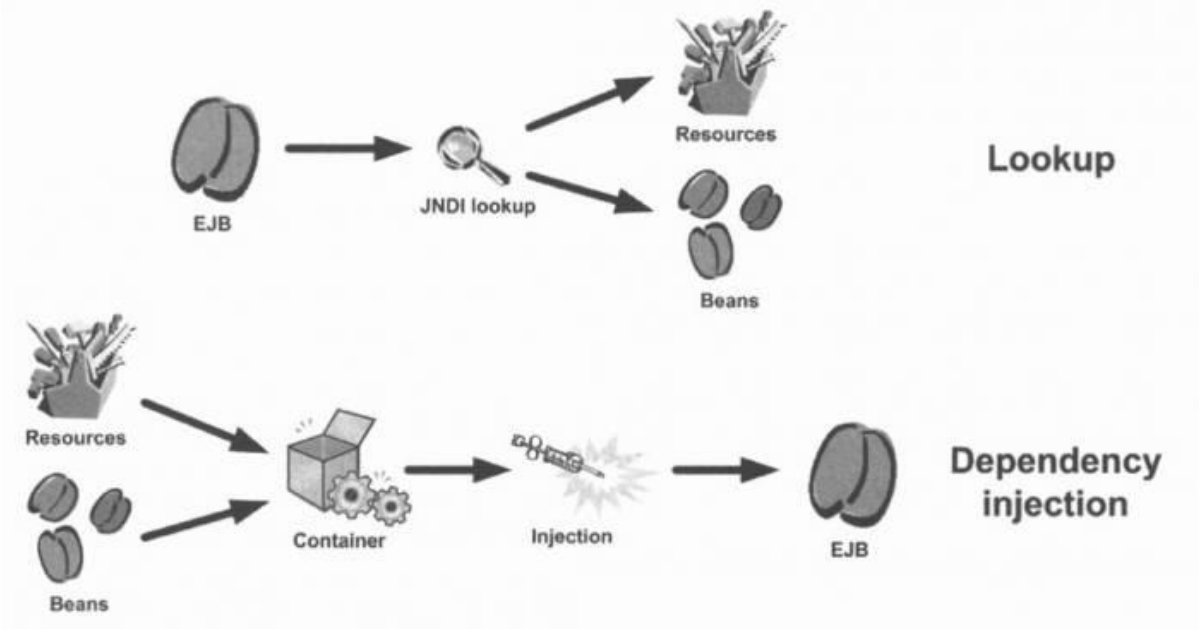
- *Werden vom Container aufgerufen*
- *Ein EJB Komponenten Entwickler kann diese Methoden implementieren*
  Beispiel: @PreDestroy oder @PostConstruct (z.B. leerer Konstuktor und alle Abhängigkeiten (@Inject) aufgelöst sind)

Business Methoden

- *Typischerweise definiert in einem Business Interface. Die Bean Klasse implementiert diese Methoden.*

4.8 Dependency Injection und deren Auswirkungen

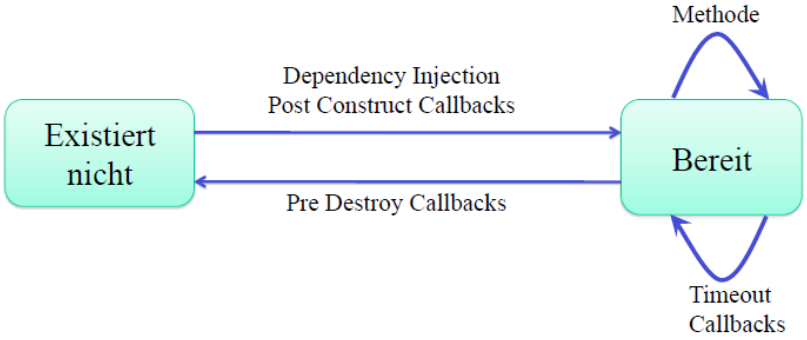
Für Session EJBs, der EJB Client verwendet Dependency Incection oder JNDI Lookups um Referenz zu erhalten.
Beide machen das Gleiche. Dependency Incection ist eleganter und koppelt lose. Lookups haben eigene try/catch Blocks und halten sich an Namenskonventionen, können also gewisse Voraussetzungen durchsetzen.



4.9 Lebenszyklus von Beans

Lebenszyklus Stateless Session Bean

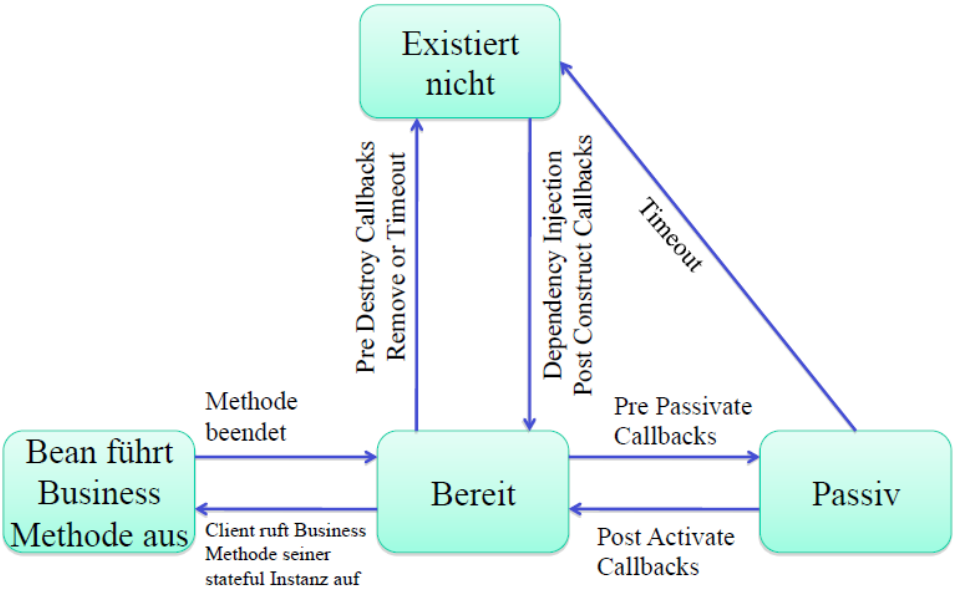
→ Ein stateless Bean behält den Status nicht länger als der Methodenaufruf eines Clients dauert.



Vorteile stateless Beans:

- ***Bean Pooling***
  Stateless Bean Methoden die momentan nicht aufgerufen wurden stehen dem EJB Container zur Verfügung jegliche Client Anfragen zu bedienen.
- ***Scalability***
  Stateless Beans bedienen mehrere Clients und haben kleineren Footprint. Das erlaubt viel mehr zu erzeugen als ihr Stateful Pendant.
- ***Performance***
  EJB Container lagern keine stateless Beans vom RAM auf die Festplatte aus - somit sind sie performanter.

Lebenszyklus Stateful Session Bean



4.10 Die Rolle der Interfaces beim JEE Bean

- *Stellt ein Zugriffspunkt und Typ für Clients zur Verfügung*
- *Spezifiziert dem Server wie er das EJB zu konstruieren hat.*
- *Dient als Framework für das Verhalten einer EJB Komponente.*

Bean Interfaces

- *Remote*
  Methoden Parameter sind serialisierbar (values).
- *Local*
  Methoden Parameter sind Referenzen.
- *No-Interface*
  Gleich wie "local", aber ohne Business Interface.

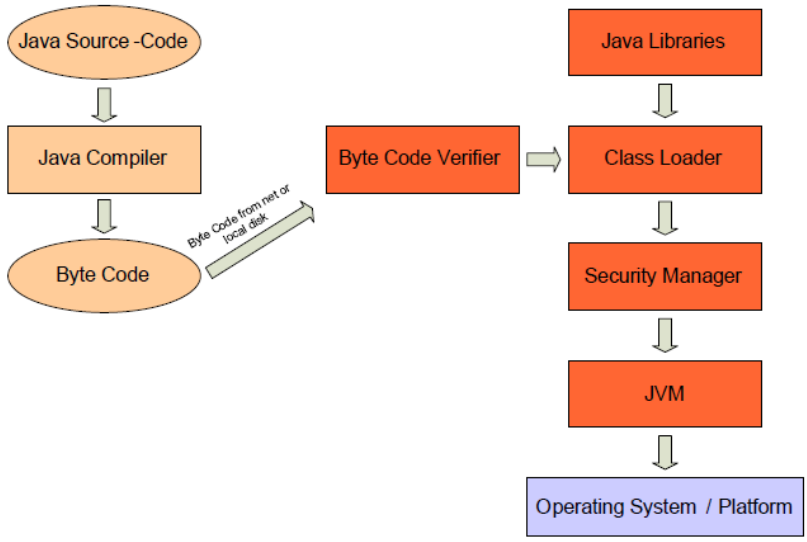
5 Enterprise Security

5.1 Java Plattform Security

Java wurde entwickelt mit dem Ziel Anwendungen eine sichere Plattform zu bieten.

- *Keine Pointer sondern Objektreferenzen*
- *Virtuelle Maschine - überwacht Programmausführung und Rechte*
- *Strenge Typisierung*
- *private, protected, package, public (Sichtbarkeiten)*
- *Exceptions - definierte und kontrollierte Programmabbrüche*
- *Strenge Arraygrenzen - kein Zugriff auf benachbarte Entitäten*

Java Security Flow



Byte Code Verifier

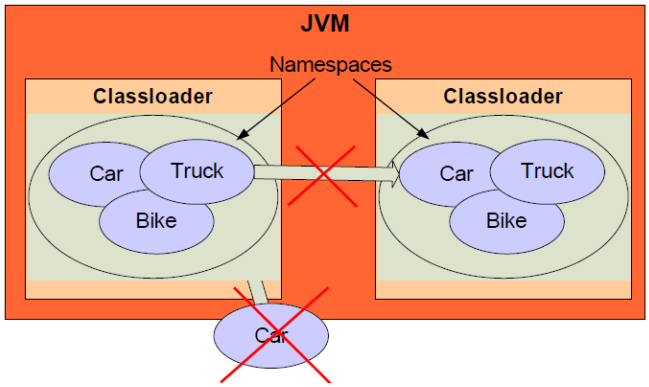
Unveränderter Bytecode nach Kompilation

Pass 1 – Format	Java Interpreter überprüft Class File (See `0xCAFEBABE` in .class)
Pass 2 – Java-Konzepte	Vererbungshierarchie, gültige Objekt-Verweise
Pass 3 – Programmausführung	Datenflussanalyse der Methoden während dem Linken
Pass 4 – Referenzen	Verweise auf fremde Klassen, auf Methoden und Attribute während dem dynamischen Linken

5.2 Welche Rolle spielt der Class Loader im Security Flow von Java?

3 Arten für die Sicherheit beim Laden von Klassen:

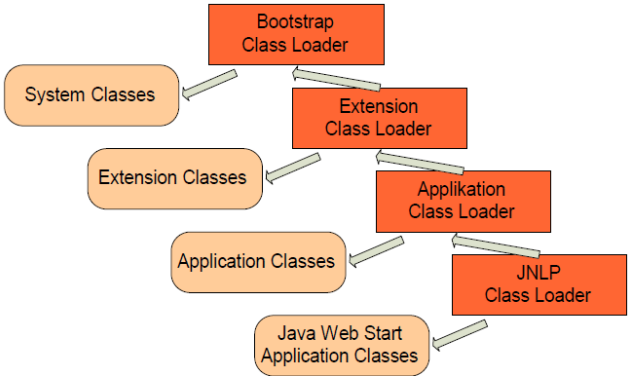
- *Class Loader und Namespaces*
- *Trusted Class Libraries*
- *Protection Domains*



Der Class Loader stellt die zentrale Stütze der Sicherheitsarchitektur dar. Er lädt den Byte-Code eines ankommenden Datenstroms und ruft den 'Byte Code Verifier' auf, um überprüfen zu lassen, ob dieser fehlerfrei ist. Danach startet der Class Loader den 'Security Manager' und 'Access Controller', um die Zugriffsrechte des JavaProgramms auf Systemressourcen zu bestätigen.

Quelle: http://berrendorf.inf.fh-bonn-rhein-sieg.de/lehre/ss03/vups1/Seminar/5_JavaClassLoader.pdf

5.3 Was sind trusted Libraries?



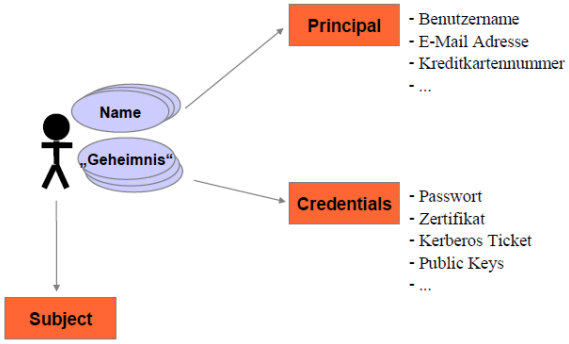
5.4 JAAS

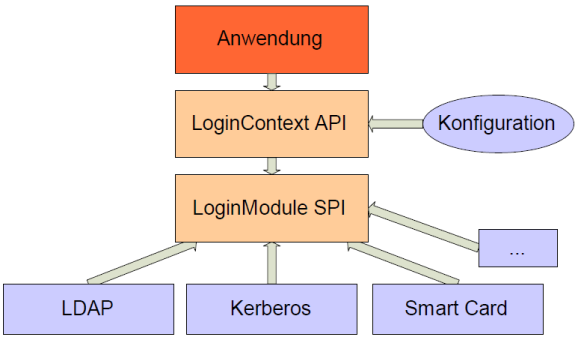
Java Authentication and Authorization Service

- *Java Platform Security – Zugriffskontrolle nach Herkunft des Codes (Codezentriertes Sicherheitsmodell)*
- *Erweiterung der Java Platform Security mit JAAS (Benutzerzentriertes Sicherheitsmodell - vgl. Betriebssystem)*

- *JAAS ist für Multiuser-Applikationen wichtig*
- *JAAS ist die Standardimplementierung des JEE-deklarativen Sicherheitsmodells*

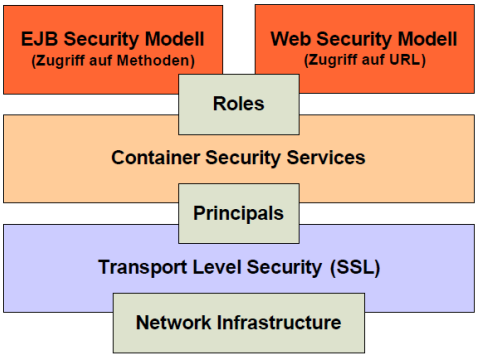
Benutzer sind entweder Personen oder Dienste (Subject) und können mehrere Identitäten (Principal) und Berechtigungen (Credential) besitzen.

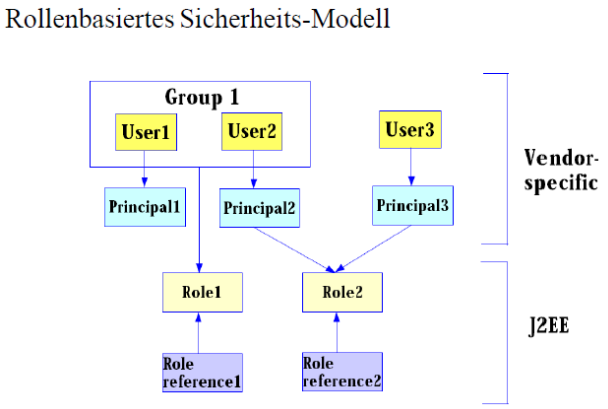




5.5 Java EE Security Architektur

- *Portabilität*
  „Write Once, Run Anywhere“
- *Transparenz*
  Applikationsentwickler müssen keine tiefgründigen Kenntnisse über Security haben um ihre Applikationen zu entwickeln
- *Isolation/ Abstraktion*
  Trennung von Business und Security – Der Deployer kann die Sicherheit hinzufügen
- *Erweiterbarkeit*
  Die Portabilität der Applikation wird durch die Einbindung von Sicherheits-Services nicht behindert
- *Unabhängigkeit*
  Die Einbindung von verschiedenen Sicherheitstechnologien soll möglich sein





Deklarativ vs. Programmatisch

- ***Deklarative Security***
  Im DD wird definiert wer auf welche EJB und/oder welche Methoden zugreifen kann oder welche URL aufrufen darf.
- ***Programmatische Security***
  Innerhalb der Bean-Implementation kann auf die Identity zugegriffen werden, damit Context-spezifische Security angewendet werden kann

- *Deklarative and Programmatische Security werden typischerweise zusammen verwendet*

Realm?

- *komplette „Datenbank“ mit Usern und Gruppen (Webapplikation)*
- *identifiziert valide User einer Applikation*

6 Messaging Services

6.1 Java Messaging Service

Messaging: Kommunikation zwischen Software Komponenten

Vorteile von JMS:

- *Einfache Integration*
- *Asynchrone Kommunikation*
- *One-to-many Kommunikation*
- *Garantierte Lieferung*
- *Transaktionale Messaging*

6.2 Was sind administrierte Objekte?

Administrierte Objekte sind Provider-spezifische Implementationen der JMS Interfaces. (http://www.joller-voss.ch/ndkjava/notes/jms/javamessagingservices.pdf)

Administrative Objekte werden in den JNDI Namespace platziert.

Administrative Objekte != Programmierte Objekte

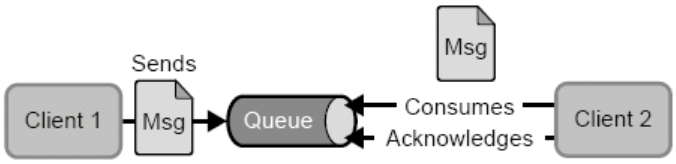
JMS definiert zwei administrierte Objekte:

- *Connection Factory*
- *Queue / Topic Destination*

6.3 Welche JMS Architekturen gibt es?

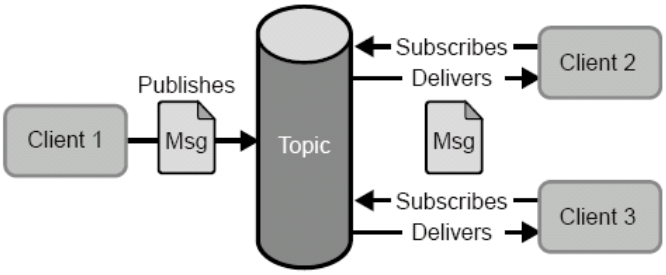
- ***Point to Point Architektur***

  Die Message-Queue kann mehrere Consumer haben. Wenn einer der Consumer die Message von der Queue konsumiert (liest), dann wird diese gelöscht.



- ***Publish / Subscriber Architektur***

  Die Topic-Queue hält alle Messages bis sie von allen Subscribern konsumiert wurden. Fällt ein Subscriber aus, behält die Topic-Queue die Message bis der Consumer zurückkommt.



6.4 Mesage Driven Beans, wann und wie werden sie eingesetzt?

- *Erlaubt Java EE Applikationen die Messages asynchron zu kommunizieren.*
- *Ähnlich einem Event Listener*
- *Messages können von jeder Java EE Komponente gesendet und empfangen werden.MDBs sind nicht Teil von EJB light Profil. Sie können nicht in einem „war" File deployed werden.*
- *MDBs werden mit @javax.ejb.MessageDriven annotiert.*

Unterschied zwischen MDBs und Session Beans

- *Clients greifen nicht über Interfaces auf die MDBs zu.*
- *Sind ähnlich zu Stateless Beans.*

- *MDBs halten keine Daten oder Status für einen Client.*
- *Alle Instanzen eines MDBs sind gleichwertig. Der Container kann die Nachricht an jedes MDB weitergeben.*
- *Ein einziges MDB kann Nachrichten für mehrere Clients verarbeiten.*

MDB Characteristics

- *Sie beginnen zu laufen, nachdem sie eine Nachricht erhalten haben.*
- *Sie werden asynchron ausgeführt.*
- *Sie sind kurzlebig.*
- *Sie repräsentieren direkt keine Daten von der Datenbank, aber sie können auf diese zugreifen und editieren.*
- *Sie können transaktionsgesteuert sein.*
- *Sie sind zustandslos, gleich wie Stateless Beans.*

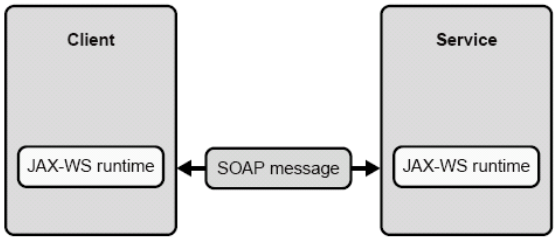
6.5 Erklären sie synchrone und asynchrone Message Services.

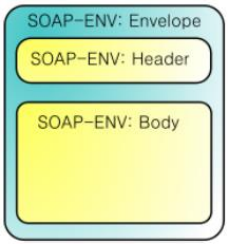
- *EJBs operieren synchron, sie werden aktiv nach Clientaufruf.*
- *MDBs operieren asynchron. Der Entwickler schreibt onMessage();*

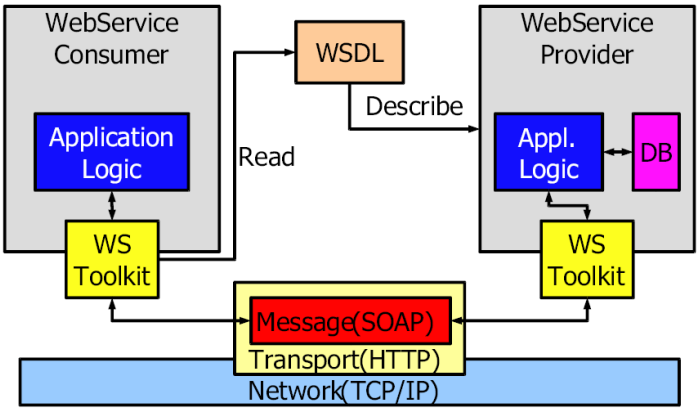
6.6 Java EE Web Services, erklären sie die essentiellen Komponenten

Java EE Web Service (JAX-WS)

Java API for XML Web Services







WSDL: Web Service Description Language

Die Web Services Description Language (WSDL) ist eine plattform-, programmiersprachen- und protokollunabhängige Beschreibungssprache für Netzwerkdienste (Webservices) zum Austausch von Nachrichten auf Basis von XML. WSDL ist ein industrieller Standard des World Wide Web Consortiums (W3C).

SOAP: Simple Object Access Protocol

SOAP ist ein Netzwerkprotokoll, mit dessen Hilfe Daten (XML) zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können. SOAP ist ein industrieller Standard des World Wide Web Consortiums (W3C).

7 REST

REST ist kein Standard – es ist ein Design Pattern! (Wiki sagt: Programmierparadigma)

Representational State Transfer (abgekürzt REST, seltener auch ReST) bezeichnet ein Programmierparadigma für Webanwendungen. REST ist eine Abstraktion der Struktur und des Verhaltens des World Wide Web. REST fordert, dass eine Web-Adresse (URI) genau einen Seiteninhalt repräsentiert, und dass ein Web-/REST-Server auf mehrfache Anfragen mit demselben URI auch mit demselben Webseiteninhalt antwortet.

Der Client referenziert eine Web Ressource unter Verwendung einer URL.

*Eine **Repräsentation** der Ressource wird zurückgegeben (in diesem Falle als HTML Dokument).*

*The Repräsentation (Boeing747.html) platziert den Client in einen neuen **Status**.*

Wenn der Client einen Hyperlink auf der Seite „Boeing747.html" selektiert, greift er auf eine andere Ressource zu.

*Diese neue Repräsentation **transferiert** die Client-Applikation in einen neuen Status.*

Also wechselt der Status mit jeder Ressourcen Repräsentation.

7.1 Design Pattern vs. architektureller Stil

?

7.2 Fundamentale Web Komponenten

Web Komponenten operieren auf Informationen vom HTTP Header!

Das Web ist zusammengesetzt aus Komponenten:

- *Firewalls: welche HTTP Nachrichten hinaus und hinein gehen.*
  Diese Komponenten setzen die Web Security durch.
- *Routers: Diese Komponenten entscheiden wohin mit den HTTP Nachrichten*
  Diese Komponenten verwalten das Web Scaling.
- *Caches: Diese Komponenten entscheiden ob eine gespeicherte Kopie verwendet werden darf.*
  Diese Komponenten erhöhen den Web Speed.

Alle Entscheidungen dieser Komponenten basieren auf Informationen im HTTP Header.: „Stecke deine Nase nie in den HTTP Payload!"

7.3 Ist REST synchron oder asynchron?

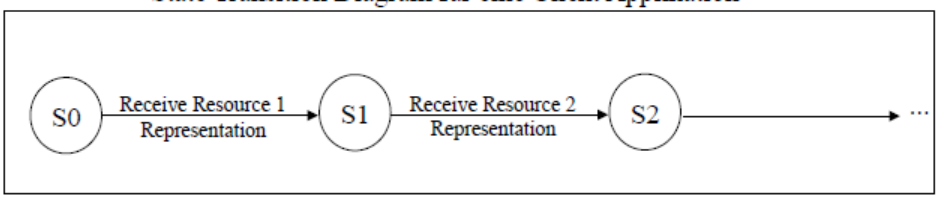
Synchron

7.4 Wie wird der Status von REST hinterlegt?

Der Client transferiert von einem Status in den Nächsten weil er im Antwort Dokument die alternativen URLs prüft und eine darunter auswählt.

Nehmen wir an eine Client Applikation ist als State Maschine ausgelegt. Jede Ressource wechselt bei erhalt einer weiteren Ressource Repräsentation ihren Status.

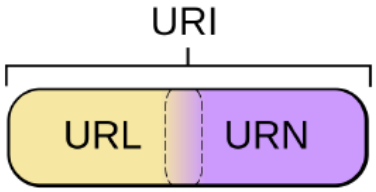
State Transition Diagram für eine Client Applikation



7.5 Ressource und Ressource Identifikation

- ***Ressourcen***
  Jede unterscheidbare Entität ist eine Ressource. Eine Ressource kann eine Web Seite, eine HTML Page, ein XML Dokument, ein Web Service, ein physikalisches Device, etc. sein.
- ***URLs Identifizieren Ressourcen***
  *Jede Ressource wird einzigartig identifiziert durch eine URL. Das ist Tim Berners-Lee's Web Design Axiom 0 *.*

7.6 Well formed URI's



- ***URL: Uniform Resource Locator***
  ist vom Typ URI und ist eine Referenz auf eine Web Ressource. Um ein Buch zu finden braucht man eine Lokation; z.B: file:///home/username/books
- ***URN: Uniform Resource Name***
  Sag nichts aus über die Verfügbarkeit einer Ressource. ZB: eine ISBN Nummer ist eine URN.
- ***URI: Uniform Resource Identifier***
  folgt einem URI Schema (RFC 396) - können klassifiziert werden als Namen, Lokator oder beides.

7.7 Vorteile von REST

Vorteile von Rest:

- *Skalierbarkeit*
- *Anbindung von Fremdsystemen*
- *Unabhängig installierbare Komponenten*
- *Komposition von Diensten*

Quelle: http://www.oio.de/public/xml/rest-webservices.htm

REST Paradigma:

- *Für jeden Service wird eine Ressource erzeugt.*
- *Identifiziere jede Ressource unter Verwendung einer URL.*
- *Die Daten die von einem Web Service retourniert werden sollten Links zu anderen Daten enthalten.*

8 AJAX

8.1 AJAX Komponenten mit Erklärungen im Detail

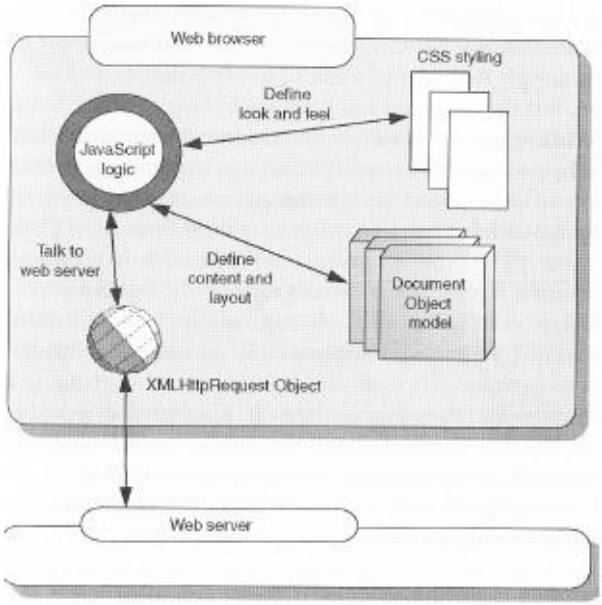
AJAX (Asynchronous Java Script and XML)

- *ClientseitigeControl Engine (JavaScript)*
- *Verwendung von XHTML*
- *Serverseitige Event Processing*

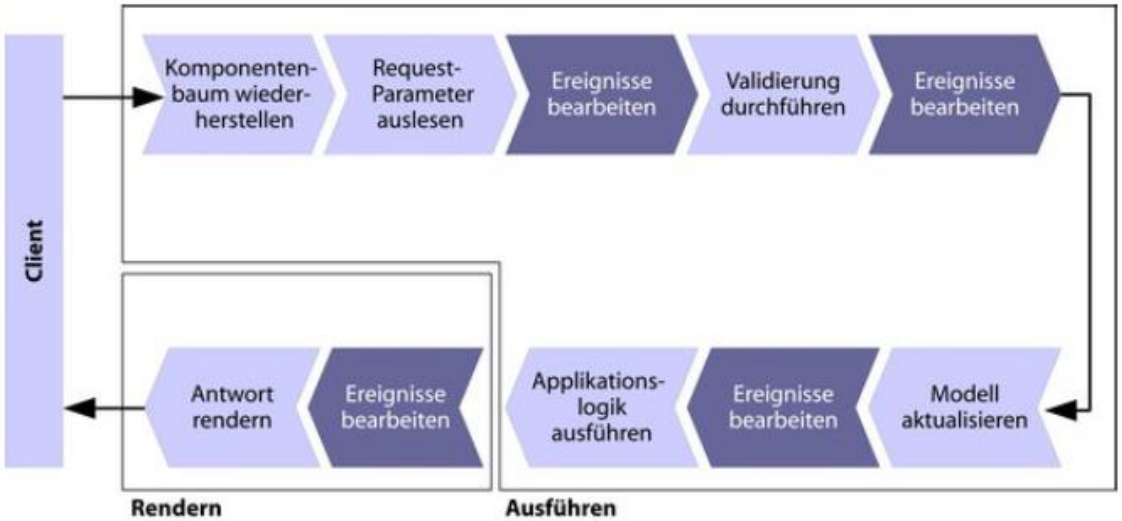
AJAX Komponenten

Java Script definiert Business Regeln und Programm Ablauf Das Document Object Model und das Cascading Style Sheet erlauben der Applikation ihr Aussehen aufgrund der neuen Daten, die im Hintergrund per XMLHttpRequest Object vom Server geholt wurden, zu verändern.

Die 4 Hauptkomponenten von AJAX



8.2 AJAX Lebenszyklus



8.3 Attribute vom f:ajax Tag

- ***event:***
  Name des Ereignisses, das die Ajax-Anfrage auslöst. Mögliche Werte sind valueChange für Eingabekomponenten, action für Steuerkomponenten.
- ***execute:***
  Eine durch Leerzeichen separierte Liste der IDs jener Komponenten, die beim Bearbeiten der Ajax-Anfrage durch JSF im Lebenszyklus ausgeführt werden sollen.
- ***render:***
  Eine durch Leerzeichen separierte Liste der IDs jener Komponenten, die beim Bearbeiten der Ajax-Anfrage durch JSF im Lebenszyklus gerendert werden sollen.
- ***onevent:***
  Erlaubt das Registrieren einer JavaScript-Callback-Funktion für Ajax-Ereignisse.
- ***onerror:***
  Erlaubt das Registrieren einer JavaScript-Callback-Funktion für Fehler, die beim Bearbeiten der Ajax-Anfrage auftreten.
- ***disabled:***
  Das Ajax-Verhalten wird "abgeschaltet", wenn dieses Attribut auf true gesetzt ist.

8.4 Java Script API

- ***jsf.ajax.request(source, event, options):***
  Diese Methode sendet eine Ajax-Anfrage an den Server.
- ***jsf.ajax.response(request, context):***
  Diese Methode bearbeitet die Anwort des Servers auf die Ajax-Anfrage und ist für Endanwender nicht relevant.
- ***jsf.ajax.addOnError(callback):***
  Diese Methode registriert eine Callback-Funktion zum Behandeln von Fehlern, die während der Bearbeitung der Ajax-Anfrage aufgetreten sind.
- ***jsf.ajax.addOnEvent(callback):***
  Diese Methode registriert eine Callback-Funktion zum Behandeln von Ajax-Ereignissen.

8.5 JSON uns XML Notationen

JSON (Java Script Object Notation)

- *Dient zur Übertragung von Daten bei Ajax.*
- *Dient als Ersatz für XML*
- *Ist auch eine Beschreibungssprache*
- *Definition selber ist ein gültiges Java Script -> kann daher mit eval() in ein Java Script Objekt umgewandelt werden.*
- *Ein Parser wird aber empfohlen (schädliche Programmanweisungen werden ausgeführt).*

JSON

```
{ "Herausgeber": „UBS",
   "Nummer": "1234-5678",
   "Deckung": 2e+6, "Währung":
   "EURO",
   "Inhaber": {
        "Name": "Mustermann",
        "Vorname": "Max",
        "männlich": true,
        "Hobby": [ "Reiten“,
                   „Golfen“,
                   „Lesen“],
        "Alter": 42
   }
```

XML

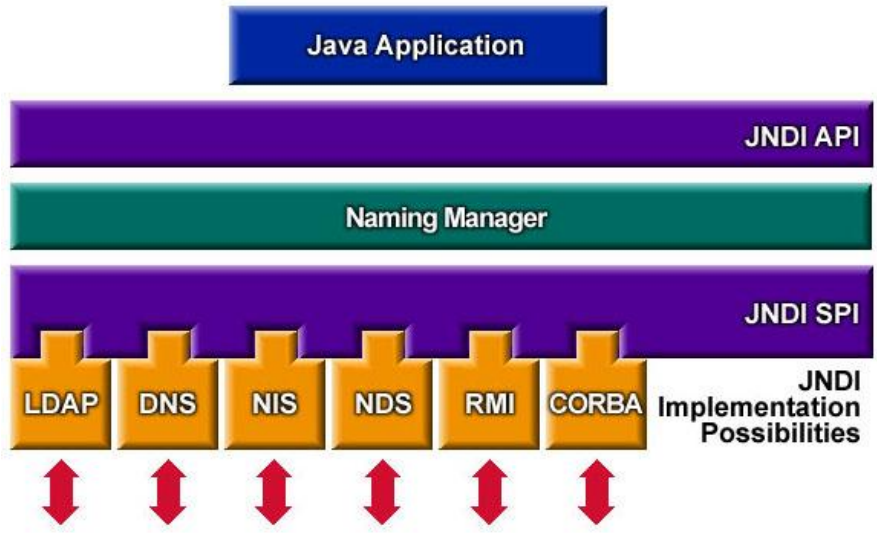
```
<Kreditkarte  Herausgeber="UBS"
   Nummer="1234-5678" Deckung="2e
   +6" Währung="EURO">
  <Inhaber
        Name="Mustermann"
        Vorname="Max"
        männlich="true"
        Alter="42">
    <Hobby>Reiten<Hobby>
    <Hobby>Golfen<Hobby>
    <Hobby>Lesen</Hobby>
  </Inhaber>
</Kreditkarte>
```

8.6 Java Web Start

- *JRE 1.4.2 oder grösser*
- *Normalerweise ist die Web Start Software schon installiert (JDK), sonst übernimmt das JavaScript der Webseite diese.*
- *JRE registriert sich mit: application/x-java-jnlp-file mime type im Browser.*
- *Ausführen (klick Link auf Java Network Launching Protocol (JNLP) File.*
- *JNLP File ist XML Datei mit Metadaten der Applikation:*
  - *Benötigte Version JRE*
  - *Ort des Client Applikation JAR File*
- *JWS benötigt einen ACC für volle Interoperabilität mit Application Server.*
  - *Security Authentication*
  - *Naming*
  - *Annotations (Dependency Injections)*
  - *Transactional Semantics*

9 JNDI

9.1 JNDI Architektur



9.2 JNDI API

Die API enthält:

- *einen Mechanismus zur Bindung eines Objekts an einen Namen*
- *Methoden für den Abruf von Informationen anhand eines Namens*
- *ein Ereigniskonzept, über das Clients von Änderungen informiert werden*
- *spezielle Erweiterungen für LDAP-Funktionalitäten*

9.3 Namenskonzept

*Ein **Namens Service** hat die Aufgabe Menschen freundliche Namen auf Objekte abzubilden.*

- *File System mappen Filenamen mit Filehandler*
- *DNS mappen Namen mit IP Nummern*

*Das **Namenssystem** beschreibt die Syntax die befolgt werden muss.*

- *Im File System werden Objekte mit „/“ (/etc/hosts) deklariert*
- *Im DNS werden Objekte mit „.“ (www.enterpriselab.ch) deklariert.*

*Die Assoziation eines Namens mit einem Objekt wird **binding** genannt.*

9.4 Ressourcen lookup

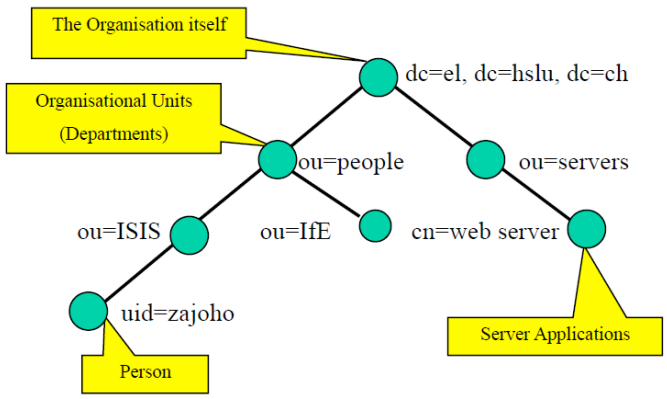
JNDI Lookup Name	Associated Reference
java:comp/env	Einträge der Applikationsumgebung
java:comp/env/jdbc	JDBC DataSourceresourcemanagerconnectionfactories
java:comp/env/ejb	EJB Referenzen
java:comp/UserTransaction	UserTransaction Referenzen
java:comp/env/mail	JavaMail Session Connection Factories
java:comp/env/url	URL Connection Factories
java:comp/env/jms	JMS Connection Factories and Destinations
java:comp/ORB	ORB Instanzen shared über die Applikationskomponenten

9.5 LDAP Protokoll

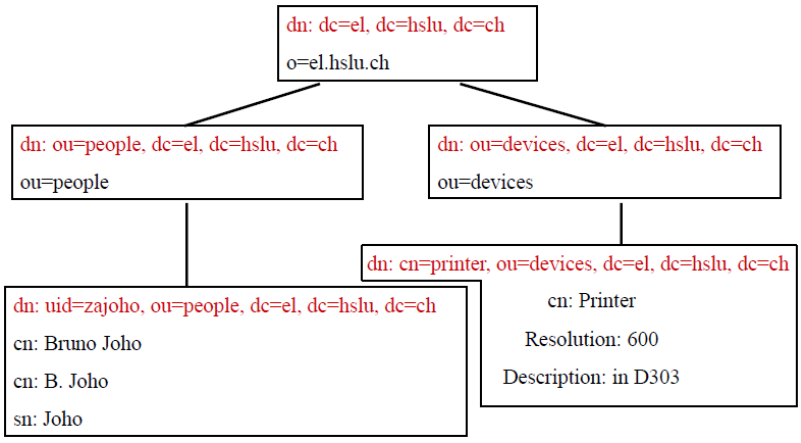
- *Lightweight Directory Access Protocol*
- *Mehr als nur ein Protokoll*
- *Informations Modell*
- *Namens Modell*
- *Funktionales Modell*
- *Sicherheits Modell*
- *APIs (C-API, JNDI, Perl-LDAP)*
- *LDIF (LDAP Data Interchange Format)*

9.6 LDAP Modelle

Das Informations Modell



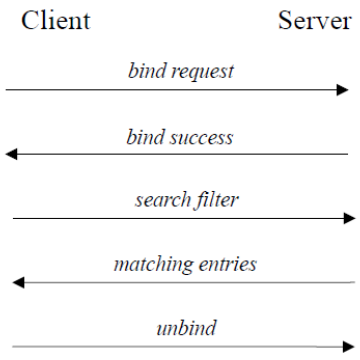
LDAP Namensmodel



Das Funktionale Modell

- *Interrogation Operations (Abfrage)*
- *Update Operations*
- *Authentication & Control Operations*

Zugriffs-Modell



Sicherheits Modell

- *Flexible Zugriffskontrolle*
  - *Entire directory information tree (DIT)*
  - *Sub-tree*
  - *Attribute level*
  - *User/Group*
  - *Authentication method*

- *Access ControlInstruction (ACI)*
  - *aci = (targetattr != 'userPassword') (allowread,search,compare)*
  - *(version 3.0; userdn='ldap:///anyone';)*

- *SSL & TLS*
  - *Beide möglich, aber nicht gleichzeitig*
  - *Benützt x.509 basierte Zertifikate*
  - *LDAP über SSL (ldaps)*

10 JEE7

10.1 Nicht prüfungsrelevant

☺