Web Anwendungen mit Java, Tomcat und MySQL – ein Überblick –

Rainer Schuler

Ulm, AOI-Systeme
Schulung, Beratung, Entwicklung

November 2010

Übersicht

1 Web-Anwendung

Übersicht

1. Web-Anwendung
2. Entwicklungsumgebung

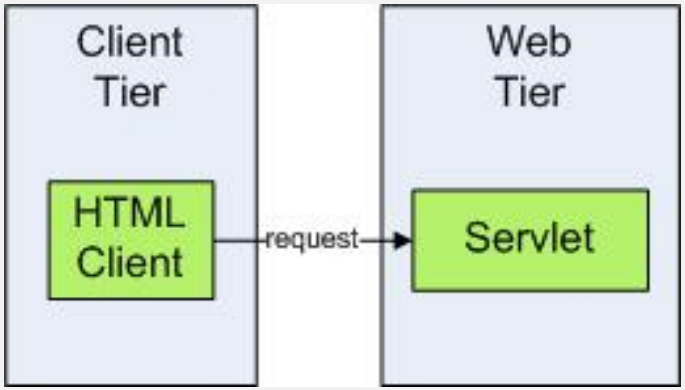
Übersicht

1. Web-Anwendung
2. Entwicklungsumgebung
3. Web Applikation

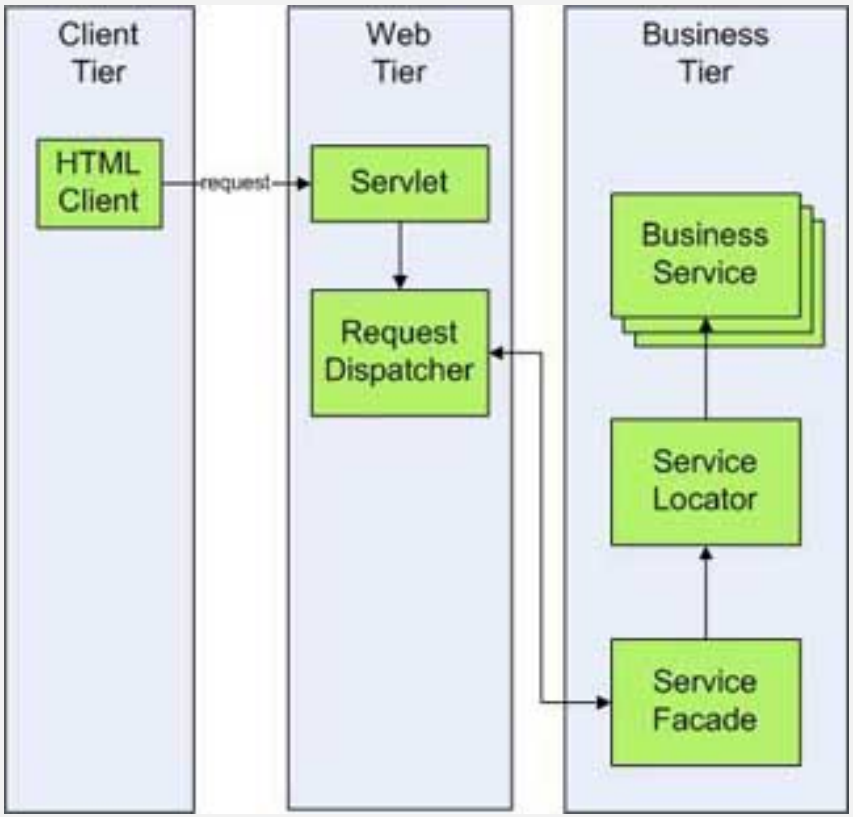
Übersicht

1. Web-Anwendung
   - Multitiered Applications
   - Web (Application) Server
   - Server Installation
2. Entwicklungsumgebung
3. Web Applikation

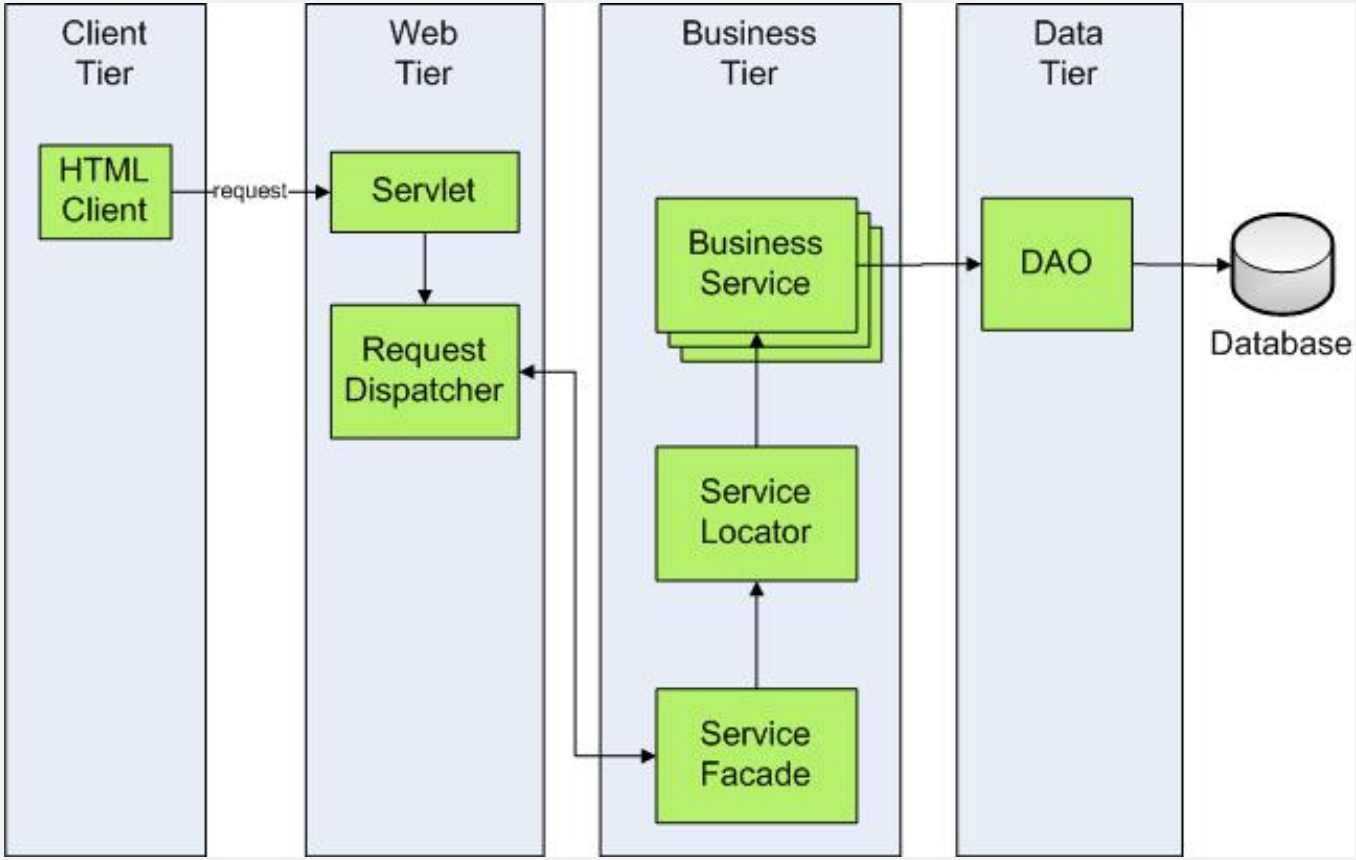
Multitiered Applications



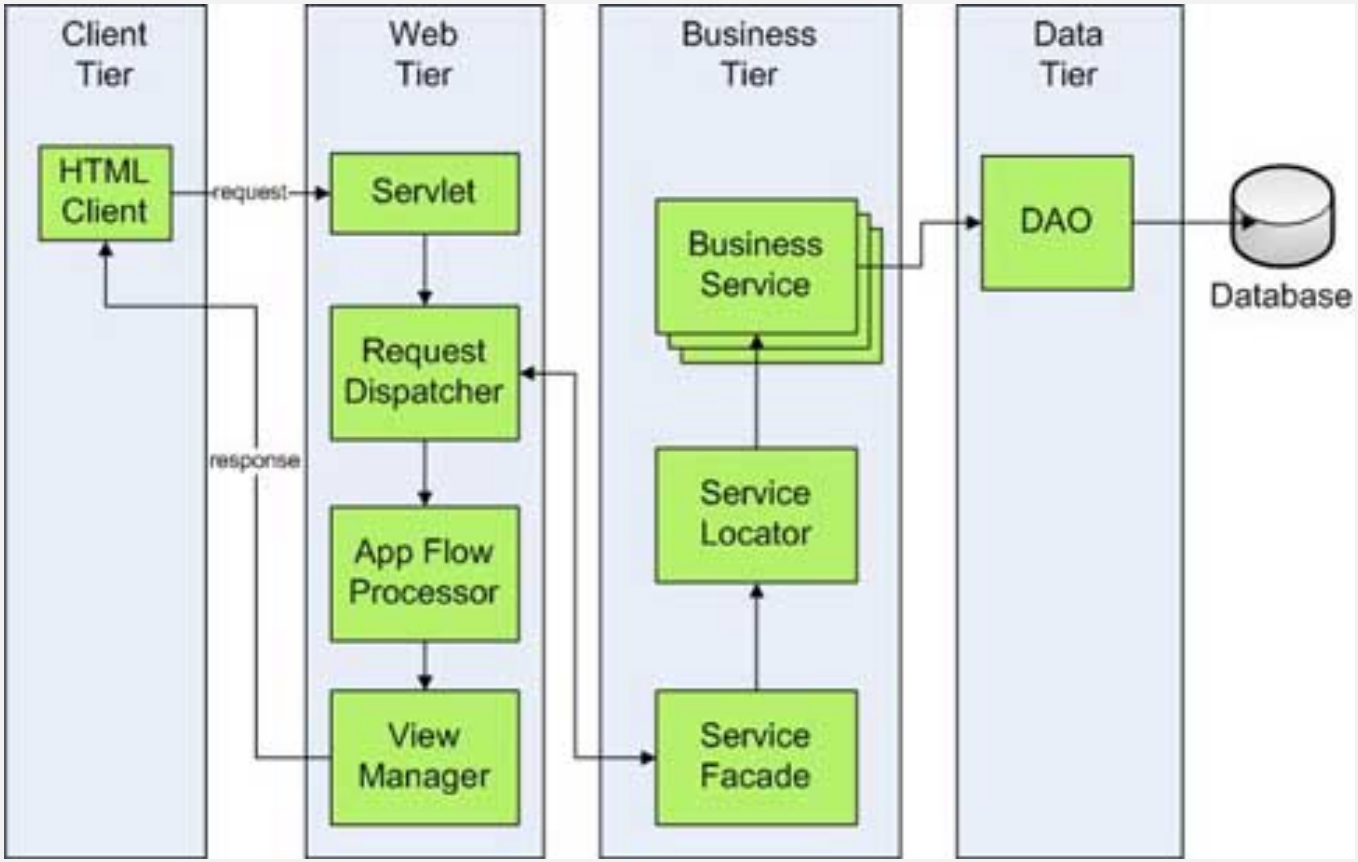
Multitiered Applications



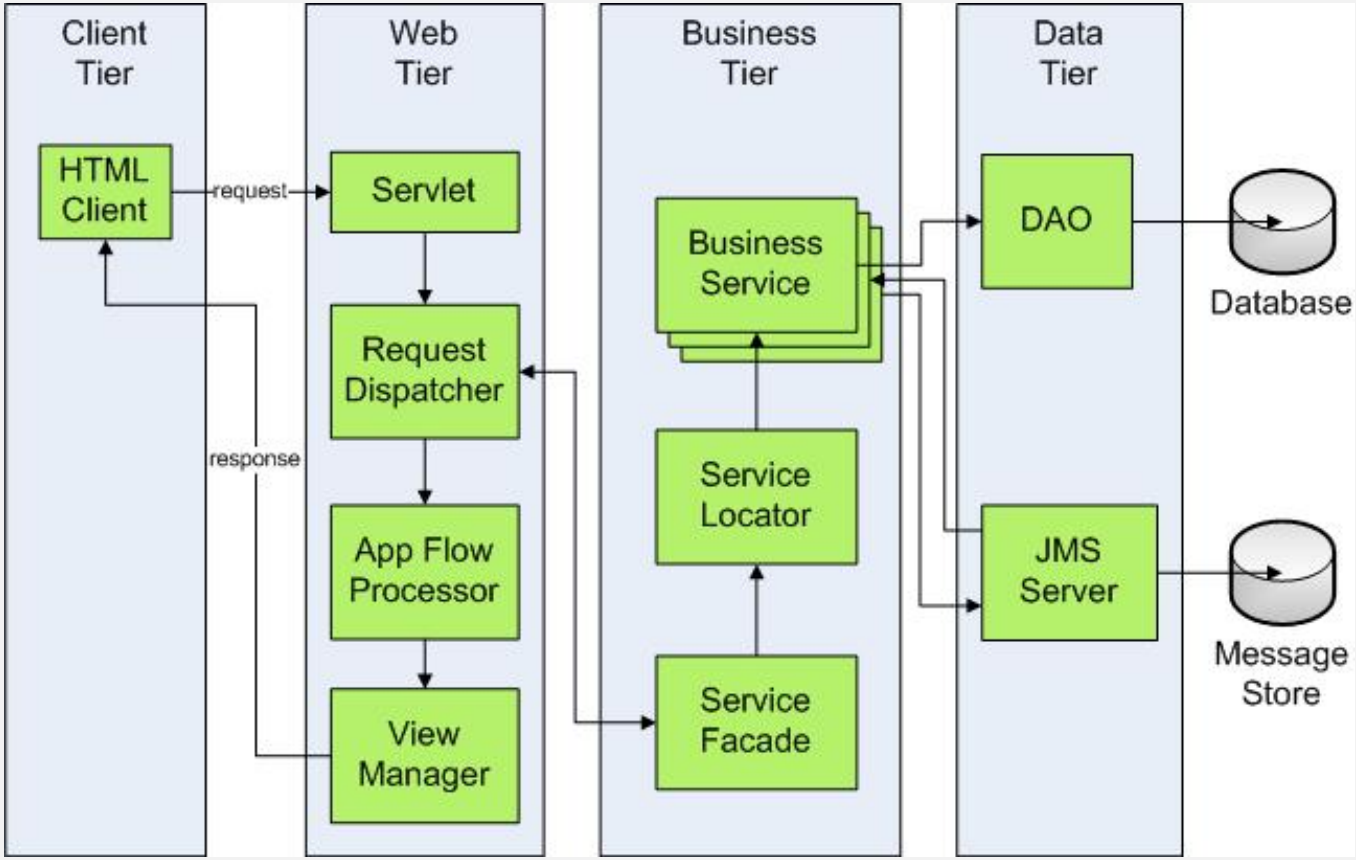
Multitiered Applications



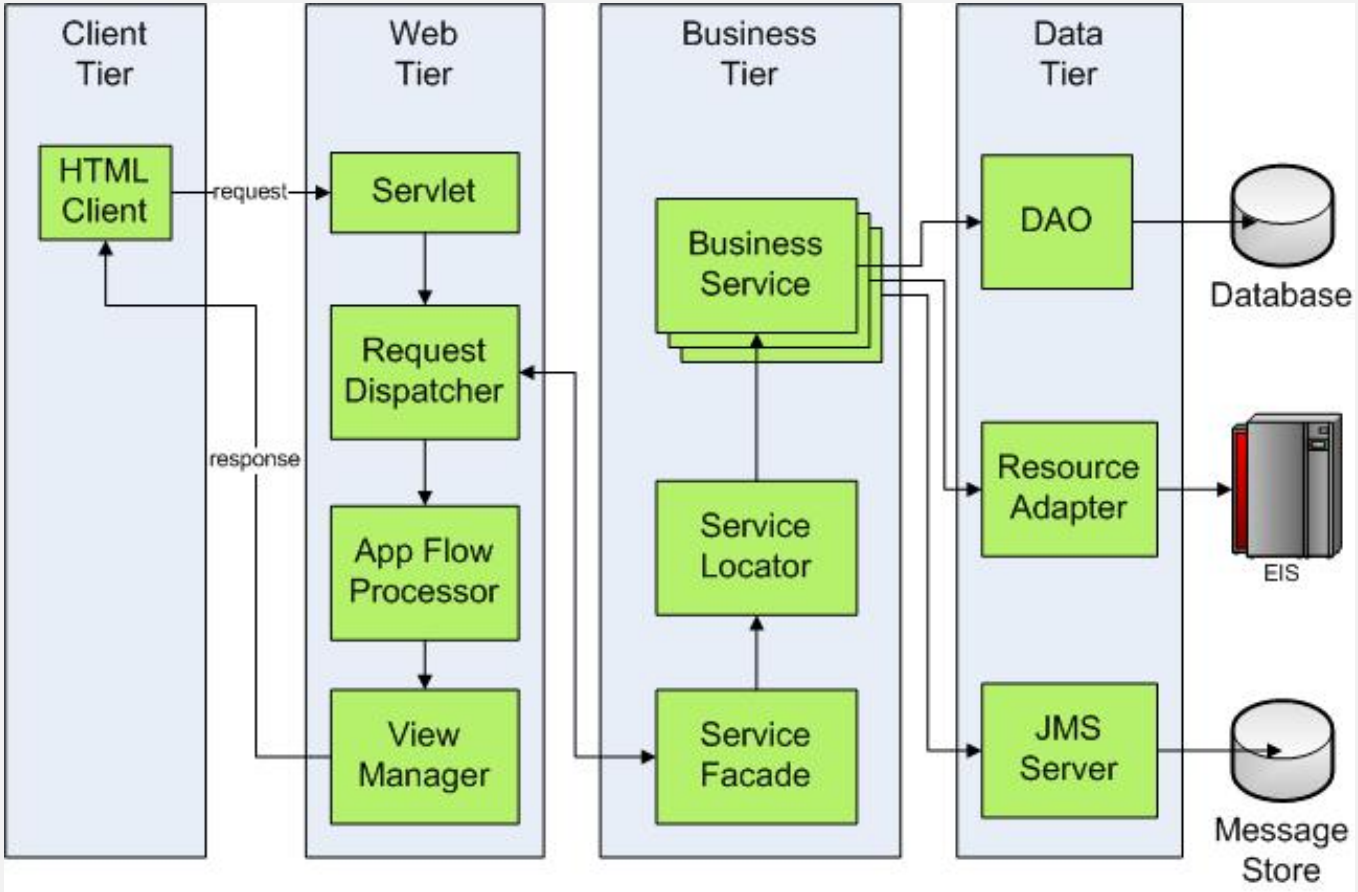
Multitiered Applications



Multitiered Applications



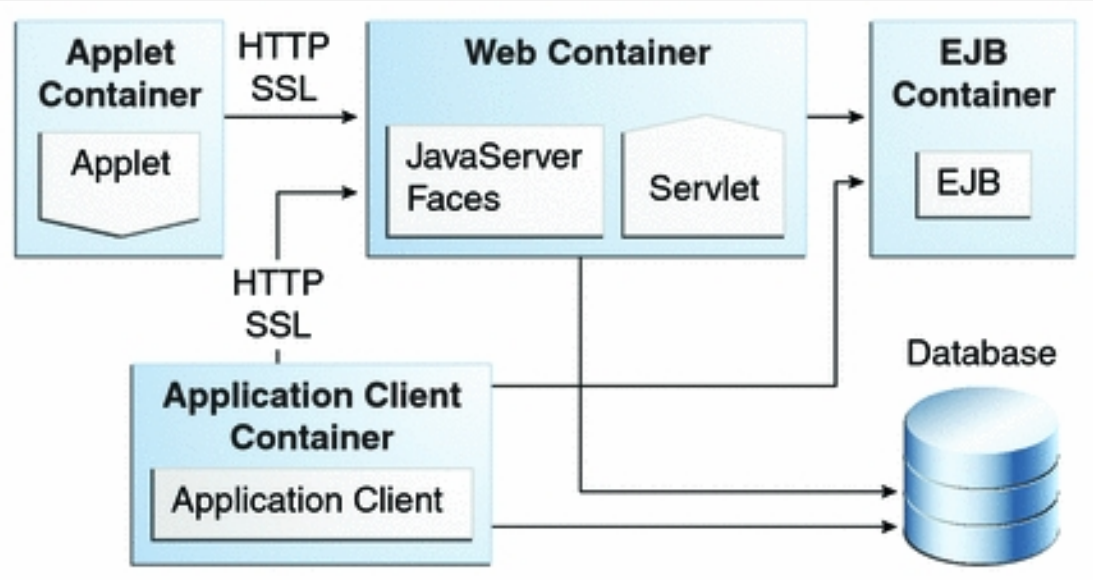
Multitiered Applications



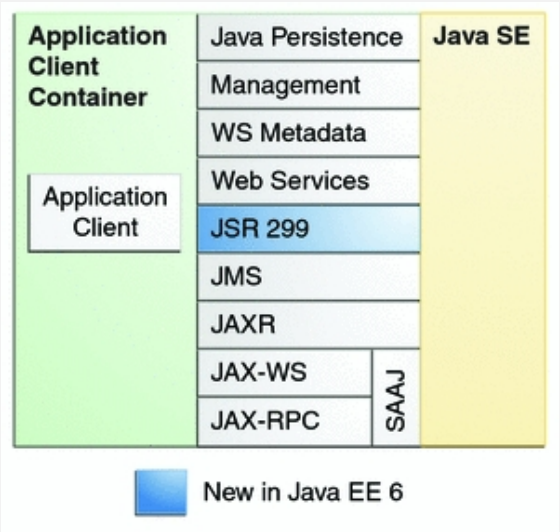
Inversion of control (IoC), Dependency Injection, Container

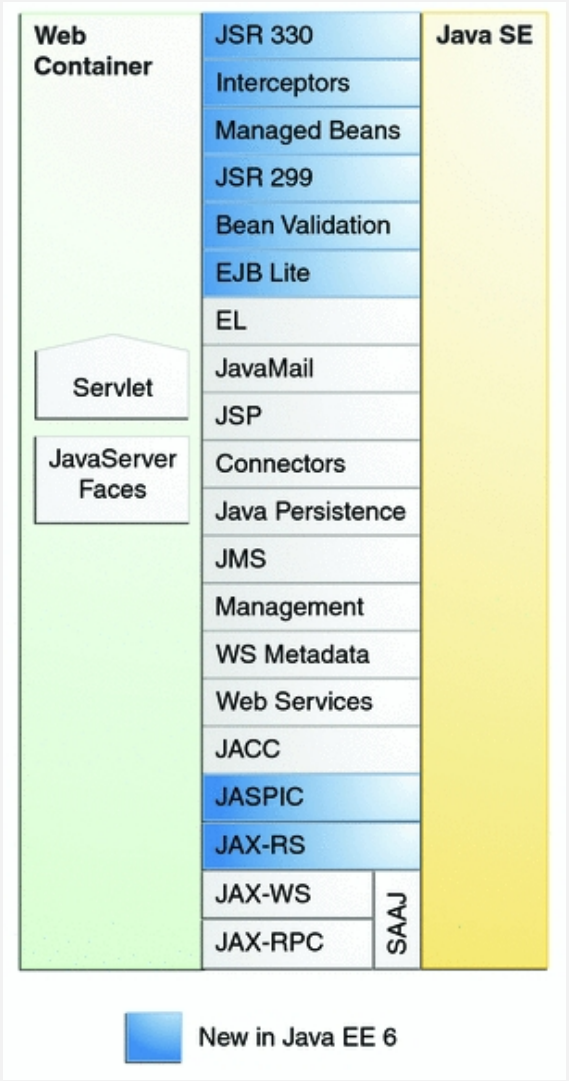
Hollywood Principle: Don't call me, I'll call you.

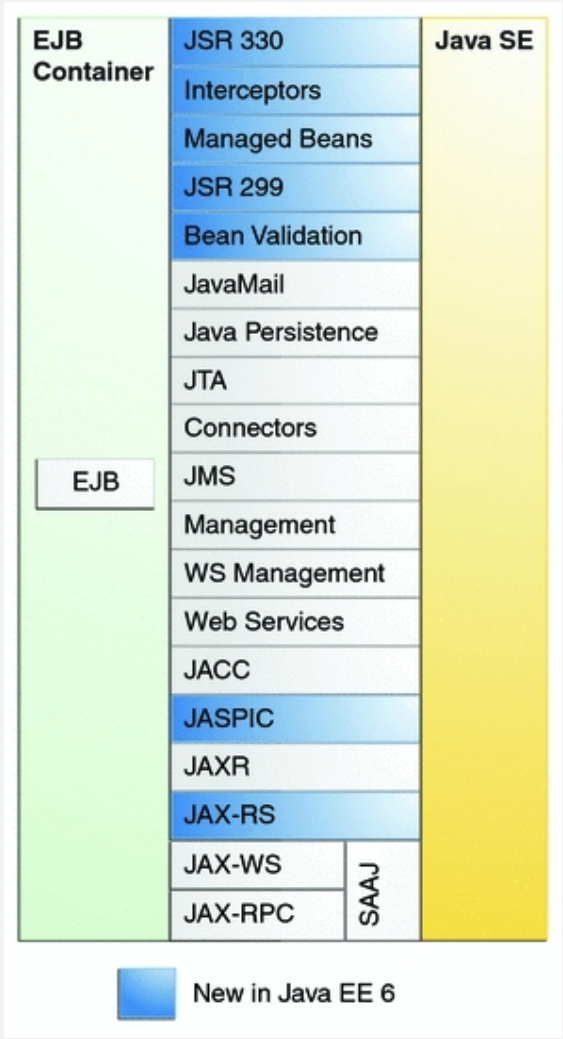
Instead of application code using framework APIs to resolve dependencies such as configuration parameters and collaborating objects, application classes expose their dependencies through methods or constructors that the framework can call with the appropriate values at runtime, based on configuration.



Java EE APIs der Web Container







Web (Application) Server Tomcat

Features

Tomcat supports many features considered to be standard in a Java Web application environment.

- JNDI resources,
- JDBC data sources,
- JSP (JSF) support,
- session replication, virtual hosting support, clustering support
- enhanced JMX-based management and monitoring.
- asynchronous HTTP request handling via Comet,
- thread pool sharing, non-blocking connectors,
- Servlet 3.0, JSP 2.1. JSF 2.0

Web (Application) Server Tomcat

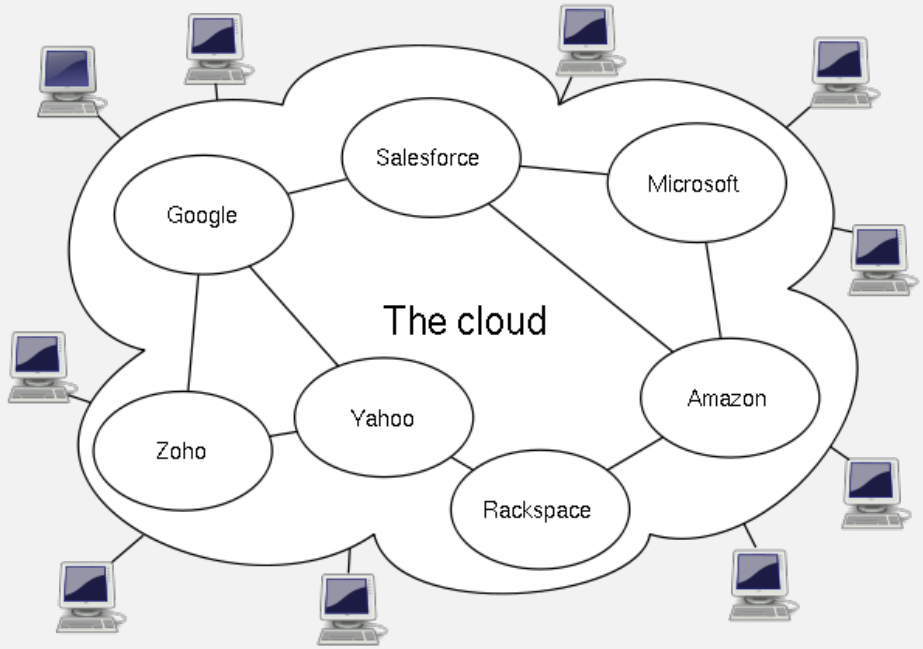
Missing Components

Tomcat does not support the entire Java EE stack, e.g.

- distributed transactions,
- Enterprise Java Beans (EJBs)
- Java Message Service (JMS)

Cloud computing

Projects can be hosted in the cloud, e.g. Google App Engine (Database is a datastore which is not a relational database)



Java Enterprise Edition Application Server

Use Tomcat for

- simpler Web applications.

Tomcat is a fast (runtime performance), reliable and lightweight (ease of use) solution that supports many EE features.

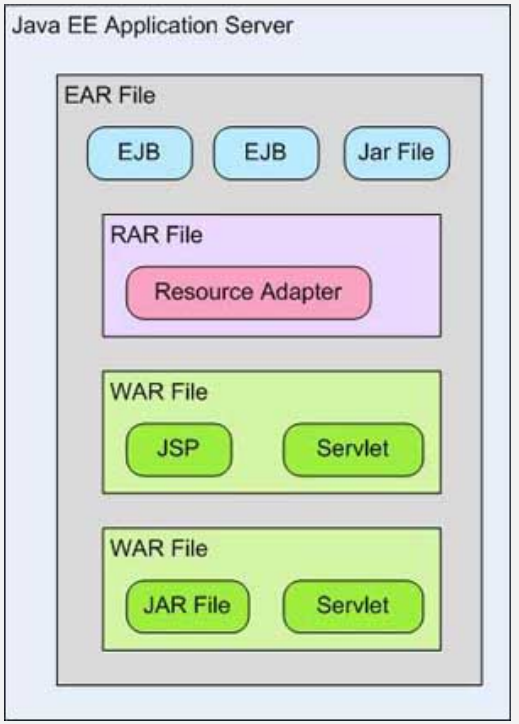
Use Java EE application server for

- SOAs
- highly scalable Web applications
- more complex enterprise applications

Many Java EE application servers actually use Tomcat as their Web container (e.g. should run umodified on Glassfish).

EAR, WAR and JAR files

- A JAR file can contain Java class files, XML descriptor files, auxiliary resources, static HTML files, and other files
- A WAR file is a specialized JAR file containing Web application components such as servlets, JSP files, HTML files, deployment descriptors, utility JAR files, etc. A WAR file can be deployed to a Web server such as Tomcat.
- An EAR file is a specialized JAR file containing Java EE application components such as Web applications (WAR files), EJBs, resource adapters, etc



Deployment

Java Web Application

A standard Java Web application is deployed in a Web Application Archive (WAR) file, which is a JAR file with the extension of .war.

Java EE Application

A standard Java EE application is deployed in an Enterprise Application Archive (EAR) file, which is a JAR file with an extension of .ear.

Server Installation

System Konfiguration

1. Raid, LVM (Logical Volume Management)
2. Ubuntu Server 10.04LTS (Long Term Support)
3. Package Management apt-get, apt-cache

- Raid Systeme: Hardware-, BIOS-Software-, Linux-Software-
- Raid 1: 2 Platten, Daten gespiegelt, Kapazität 50%, schnelle Wiederherstellung nach Crash.
- Raid 5: 3 Platten, 3. Platte speichert parity Bit, Kapazität 66%
- LTS vermeidet Neuinstallation für 3 bzw. 5 Jahre da Sicherheits-Updates garantiert werden.

https://help.ubuntu.com/10.04/serverguide/C/advanced-installation.html

Server Dienste

Web-Dienste

1. JDK1.6 (bereits installiert)
2. SSH
3. Tomcat, Web-(Application)-Server
4. MySQL, Datenbank

Installation

2. `sudo apt-get install openssh-server`
   https://help.ubuntu.com/10.04/serverguide/C/openssh-server.html
   ssh-keygen, /etc/ssh/sshd_config, /etc/hosts.allow, .deny
3. `sudo apt-get install tomcat6`
   https://help.ubuntu.com/10.04/serverguide/C/tomcat.html
4. `sudo apt-get install mysql-server`
   https://help.ubuntu.com/10.04/serverguide/C/mysql.html

Versionsverwaltung SVN

Repository Layout

Für jedes Projekt wird ein Projekt-Root-Verzeichnis mit den Unter-Verzeichnissen `trunk`, `tags` und `branches` angelegt.

1. `trunk` Hauptentwicklung
2. `tags` mit einer Versionsnummer versehene Entwicklungsstände (nicht veränderbar).
3. `branches` abgeleitete Entwicklungen (in jeweils einem eigenen Unterverzeichnis).

Es kann ein (gemeinsames) Repository für mehrere Projekte angelegt werden. Dazu müssen die Projekt-Root-Verzeichnis in einem gemeinsamen Verzeichnis (gruppiert in Unterverzeichnissen) angelegt werden.
http://svnbook.red-bean.com/nightly/de/svn-book.html

Administration

Sicherheit

1. Firewall
2. PPTP (point to point tunneling protocol), VPN

Backup

1. mysqldump, mylvmbackup (Backup MySQL)
2. tar, rsync, rdiff,
   https://help.ubuntu.com/8.04/serverguide/C/backup-shellscripts.html
3. Tatarus (backup via ftp),
   http://www.html-forum.de/serverdienste/6520-tutorial-tartarus-backup.html

Was läuft?

Shell Kommandos

1. IP-Ports und deren Aliase `/etc/services`
2. `netstat -atube` (Aktive Ports)
3. `netstat -tueb` (Aktive TCP, UDP Ports)
4. `lsof -i udp` (Welche Programme nutzen Protokoll UDP)
5. `lsof -i :22` (Welche Programme nutzen Port 22)
6. `nmap -v -A <hostname>` Portscan: welche Dienste laufen in welcher Version

list open files, network statistics, network mapper (Zenmap)

Übersicht

1. Web-Anwendung
2. Entwicklungsumgebung
   - Eclipse IDE
   - Frameworks
     - Hibernate
     - Spring
     - Spring Security
3. Web Applikation

Entwicklungstools

Eclipse IDE

1. IDE: Eclipse IDE for Java EE Developers (Helios)
2. Versionsverwaltung: SVN (Subversion)
3. Einrichten/Bauen: Maven, Ant

Laufzeit

1. Tomcat, Web-Server
2. MySQL (HSQLDB), Datenbank
3. IE (Firefox, Safari, Opera), Browser
4. Word, Adobe Flash, Quicktime

Frameworks für Entwicklung und/oder Laufzeit

Java-Packages

(i) Logging, XML Parser, Drucken, e.g. log4j, DOM (SAX), iText
(ii) SOAP, RESTful (Web Services)
(iii) Hibernate/Spring (Datenbank)
(iv) JSF (User Interface)
(v) Anwendungssoftware (Routing, Optimierung, Logistik)

Frameworks werden als .jar-files eingebunden. Oft gibt es für ein (standardisiertes) Framework unterschiedliche Implementierungen, e.g. die JSF Implemetierung von MyFaces und Sun.

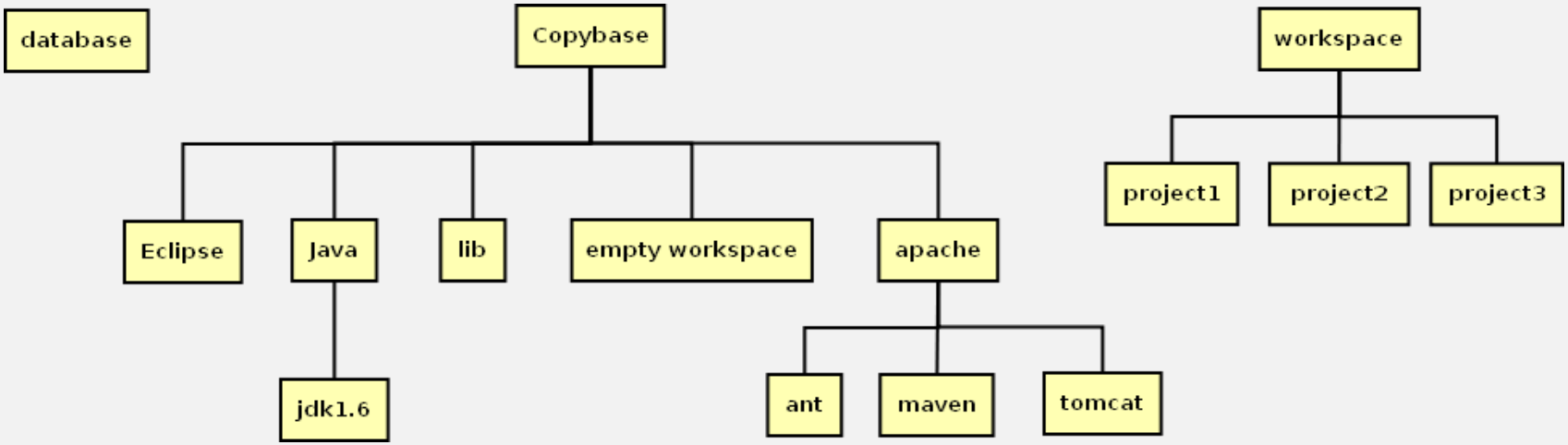
Eclipse IDE

Frameworks (reusable sets of libraries or classes) help devlopers with to implement a standard structure of an application. In many cases the devlopment process is supported by IDE plugins. Eclipse employs plugins in order to provide all of its functionality on top of (and including) the runtime system.

Frameworks

.NET Framework von Microsoft, ACE, Adobe Flex, Application Kit, CakePHP, Django (Framework), Drupal, Eclipse Rich Client Platform, Grails, Hamlets, Horde (Framework), JavaServer Faces, JBoss Seam, JHotDraw, MFC - Application Framework von Microsoft, Ruby on Rails, Spring (Framework), Struts, Symfony, Visual Component Library (VCL) - Komponentenbibliothek von Borland, YAML (Framework) CSS-Framework, Zend Framework

Plugins and Frameworks



Plugins werden im plugin-Verzeichnis von Eclipse abgelegt.
Frameworks (.jar-files), und andere Resourcen werden z.T Projekt bezogen verlinkt und beim bauen geladen.
Daher ist eine einheitliche Struktur der Entwicklungsumgebung wichtig.

Hibernate, Framework und Plugin

Hibernate maps the Java classes to the database tables.

Object relational mapping

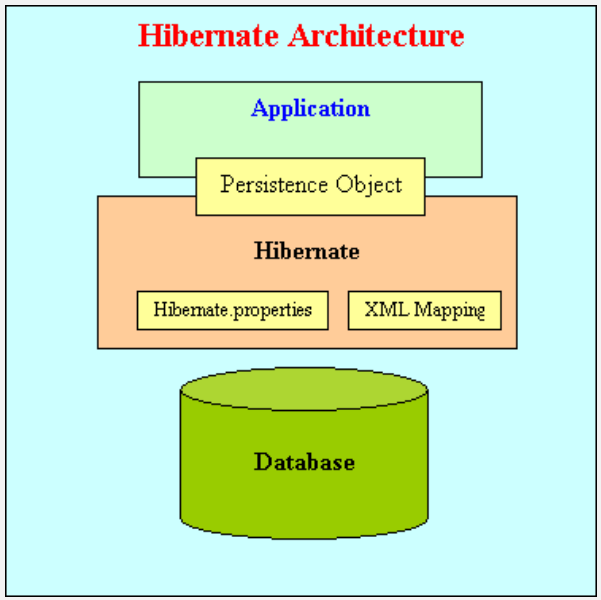
Provides mapping of the data from an object model to a relational data model.

Transaction management

Allows application to execute more than one database statement at a time.

Connection Management

Provides efficient management of the database connections.



http://www.jboss.com/products/hibernate/

hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
    <property name="dialect">org.hibernate.dialect.HSQLDialect</property>
    <property name="connection.driver_class">org.hsqldb.jdbcDriver</property>
    <property name="connection.url">jdbc:hsqldb:/copybase/database/bookcase</property>
    <property name="connection.username">sa</property>
    <property name="connection.password"></property>
    <property name="connection.shutdown">true</property>
        <!-- JDBC connection pool (use the built-in one) -->
    <property name="connection.pool_size">1</property>
        <!-- Enable Hibernate's automatic session context management -->
    <property name="current_session_context_class">thread</property>
        <!-- Disable the second-level cache  -->
    <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>
        <!-- Echo all executed SQL to stdout -->
    <property name="show_sql">true</property>
        <!-- List all the mapping documents we're using -->
    <mapping resource="de/botzenhart/bookcase/data/Book.hbm.xml"/>
    <mapping resource="de/botzenhart/bookcase/data/Author.hbm.xml"/>
    <mapping resource="de/botzenhart/bookcase/data/Keyword.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

Die Konfigurationsdatei beschreibt den Zugriff auf die Datenbank und listet die Mapping-Files auf.

Keyword.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="de.botzenhart.bookcase.data">
  <class name="Keyword">
  <id name="id" type="int" column="KEYWORD_ID">
      <meta attribute="scope-set">protected</meta>
      <generator class="native"/>
    </id>

    <property name="bezeichnung" type="string">
      <meta attribute="use-in-tostring">true</meta>
      <column name="BEZEICHNUNG" not-null="true" index="KEYWORD_BEZEICHNUNG"/>
    </property>
  </class>
</hibernate-mapping>
```

Das Mapping-File erlaubt die Zuordnung Tabellen zu Javaklassen. Hibernate erzeugt hieraus die Javaklassen und das Datenbankschema.

Book.hbm.xml

```
<hibernate-mapping package="de.botzenhart.bookcase.data">
  <class name="Book" table="BOOK">
    <meta attribute="class-description">Buecher des Verlags</meta>
    <id name="id" type="int" column="BOOK_ID">
      <meta attribute="scope-set">protected</meta>
      <generator class="native"/>
    </id>
    <property name="title" type="string" not-null="true">
      <meta attribute="use-in-tostring">true</meta>
      <column name="TITLE" not-null="true" index="BOOK_TITLE"/>
    </property>
    <property name="klappenText" type="string" column="KLAPPENTEXT"/>
    <property name="playTime" type="time" column="PLAYTIME">
      <meta attribute="field-description">Laufzeit Hoerbuch</meta>
    </property>
    <set name="author" table="BOOK_AUTHORS">
      <key column="AUTHOR_ID"/>
      <many-to-many class="de.botzenhart.bookcase.data.Book" column="BOOK_ID"/>
    </set>
    <property name="added" type="date">
      <meta attribute="field-description">Erscheinungsdatum</meta>
    </property>
  </class>
</hibernate-mapping>
```

Die many-to-many Relation wird in der Tabelle `BOOK_AUTHORS` abgebildet.

Author.hbm.xml

```
<hibernate-mapping package="de.botzenhart.bookcase.data">
  <class name="Author">
      <meta attribute="class-description">
      Authoren des Verlags. Fuer Pseudonyme ist actualAuthor
      ungleich null und verweist auf den Author.
    </meta>
   <id name="id" type="int" column="AUTHOR_ID">
     <meta attribute="scope-set">protected</meta>
     <generator class="native"/>
   </id>
   <property name="name" type="string">
    <meta attribute="use-in-tostring">true</meta>
    <column name="NAME" not-null="true" unique="true" index="AUTHOR_NAME"/>
   </property>
   <set name="books" table="BOOK_AUTHORS" inverse="true" >
     <meta attribute="field-description">Buecher des Authors</meta>
     <key column="BOOK_ID"/>
     <many-to-many class="de.botzenhart.bookcase.data.Author" column="AUTHOR_ID"/>
   </set>
   <many-to-one name="actualAuthor" class="de.botzenhart.bookcase.data.Author">
     <meta attribute="use-in-tostring">true</meta>
   </many-to-one>
  </class>
</hibernate-mapping>
```

Die many-to-many Relation muss hier als inverse markiert werden.
Änderungen (books in Author) werden nicht automatisch persistent.

Generierte Klasse Book.java

```
package de.botzenhart.bookcase.data;

// Generated 21.11.2010 12:04:31 by Hibernate Tools 3.2.4.GA

import java.util.Date;
import java.util.HashSet;
import java.util.Set;

/**
 *       Buecher des Verlags
 *
 */
public class Book implements java.io.Serializable {

private int id;
private String title;
private String klappenText;
/**
 * Laufzeit Hoerbuch
 */
private Date playTime;
private Set author = new HashSet(0);
/**
 * Erscheinungsdatum
 */
private Date added;
```

```
public Book() {
}
public Book(String title) {
this.title = title;
}
public Book(String title, String klappenText, Date playTime, Set author,
Date added) {
this.title = title;
this.klappenText = klappenText;
this.playTime = playTime;
this.author = author;
this.added = added;
}
/**     alle getters und setters  */

/**
 * toString
 * @return String
 */
public String toString() {
StringBuffer buffer = new StringBuffer();

buffer.append(getClass().getName()).append("@")
.append(Integer.toHexString(hashCode())).append(" [");
buffer.append("title").append("='").append(getTitle()).append("' ");
buffer.append("]");

return buffer.toString();
}
}
```

Database Schema (SQL-Statements)

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE MEMORY TABLE AUTHOR(AUTHOR_ID INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT NULL PRIMA
CREATE INDEX AUTHOR_NAME ON AUTHOR(NAME)
CREATE MEMORY TABLE BOOK(BOOK_ID INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT NULL PRIMARY K
CREATE INDEX BOOK_TITLE ON BOOK(TITLE)
CREATE MEMORY TABLE BOOK_AUTHORS(AUTHOR_ID INTEGER NOT NULL,BOOK_ID INTEGER NOT NULL,PRIMARY KEY(AUTHOR_ID
CREATE MEMORY TABLE KEYWORD(KEYWORD_ID INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT NULL PRI
CREATE INDEX KEYWORD_BEZEICHNUNG ON KEYWORD(BEZEICHNUNG)
ALTER TABLE AUTHOR ALTER COLUMN AUTHOR_ID RESTART WITH 1
ALTER TABLE BOOK ALTER COLUMN BOOK_ID RESTART WITH 1
ALTER TABLE KEYWORD ALTER COLUMN KEYWORD_ID RESTART WITH 1
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 10
```

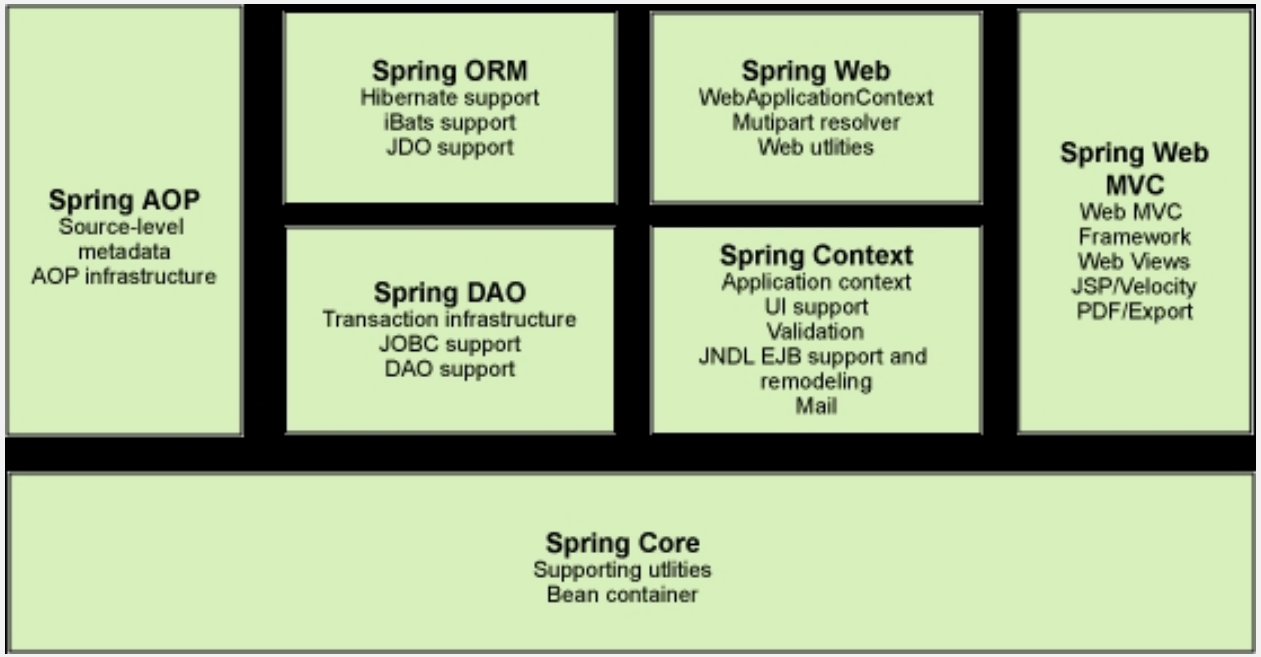
Application

```
public class TestHibernateBookcase {
   public static void main(String[] args) {
      Session session = null;
      SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
      session = sessionFactory.openSession();
      try {
         String klappenText = "Transactions also group data access operations, in fact every SQL ...";
         Author author = new Author("James");
         System.out.println("Saving " + author.toString());
         Transaction tx = session.beginTransaction();
         session.save(author);
         tx.commit();
         session.flush();
         Set<Author> authors = new HashSet<Author>();
         authors.add(author);
         Book book = new Book("Harnessing Hibernate", klappenText, null, authors, new Date());
         System.out.println("Saving " + book.toString());
         tx = session.beginTransaction();
         session.save(book);
         tx.commit();
         System.out.println("Done " + book.toString());
      } catch (Exception e) {
         System.out.println(e.getMessage());
      } finally {
         session.close();
         sessionFactory.close();
      }
   }
}
```

Database

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE MEMORY TABLE AUTHOR(AUTHOR_ID INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT NULL PRIMA
CREATE INDEX AUTHOR_NAME ON AUTHOR(NAME)
CREATE MEMORY TABLE BOOK(BOOK_ID INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT NULL PRIMARY K
CREATE INDEX BOOK_TITLE ON BOOK(TITLE)
CREATE MEMORY TABLE BOOK_AUTHORS(AUTHOR_ID INTEGER NOT NULL,BOOK_ID INTEGER NOT NULL,PRIMARY KEY(AUTHOR_ID
CREATE MEMORY TABLE KEYWORD(KEYWORD_ID INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 1) NOT NULL PRI
CREATE INDEX KEYWORD_BEZEICHNUNG ON KEYWORD(BEZEICHNUNG)
ALTER TABLE AUTHOR ALTER COLUMN AUTHOR_ID RESTART WITH 2
ALTER TABLE BOOK ALTER COLUMN BOOK_ID RESTART WITH 2
ALTER TABLE KEYWORD ALTER COLUMN KEYWORD_ID RESTART WITH 1
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 10
SET SCHEMA PUBLIC
INSERT INTO AUTHOR VALUES(1,'James',NULL)
INSERT INTO BOOK VALUES(1,'Harnessing Hibernate','Transactions also group data access operations, in fact
INSERT INTO BOOK_AUTHORS VALUES(1,1)
```

The Spring framework, IoC, dependency injection



http://www.springsource.org/
http://blog.springsource.com/2010/07/22/spring-mvc-3-showcase/

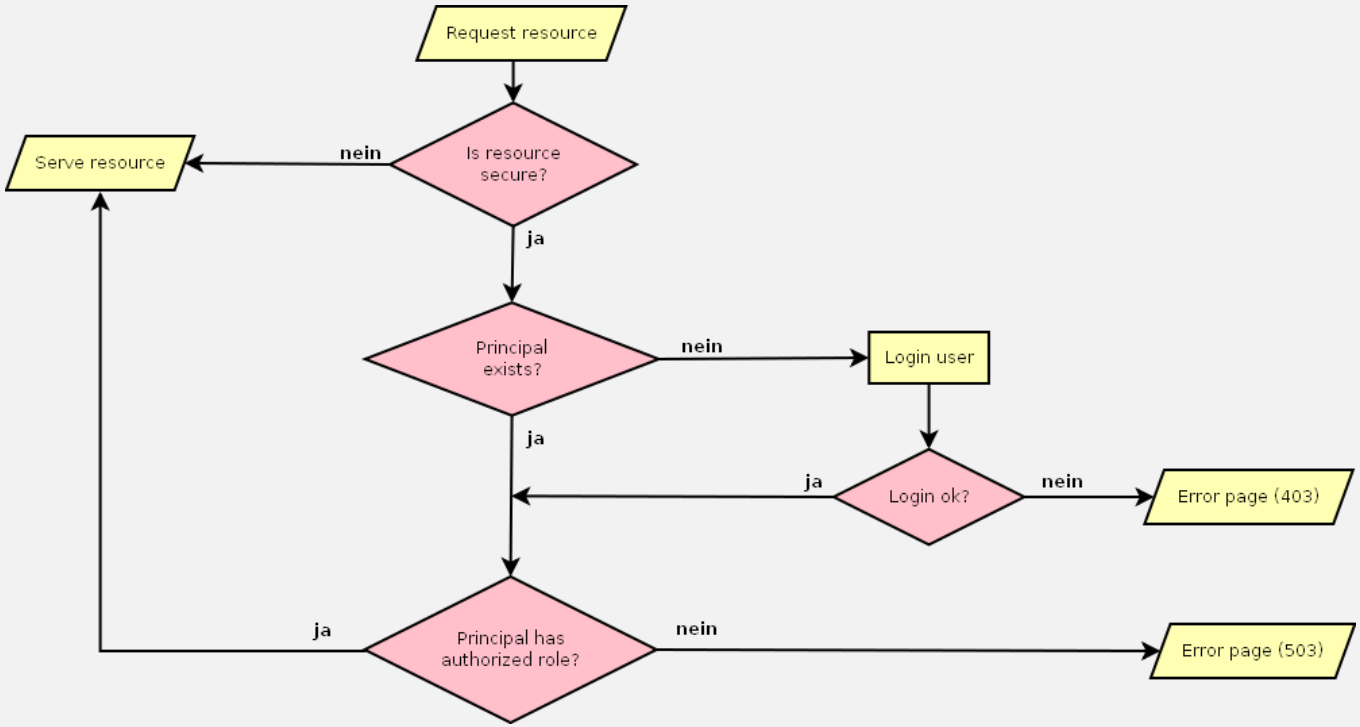
Spring depency injection

```
public interface StringWriter {
   public void writeString(String s);
}


@Service
public class PlainWriter implements StringWriter {
   public void write(String s) {
      System.out.println("Message is  " + s);
   }
}

@Service
public class BeanWriterWithSpringDependency {
   private StringWriter writer;
   @Autowired
   public void setWriter(StringWriter writer) {
      this.writer = writer;
   }
   public void run() {
      String s = "This is a string";
      writer.write(s);
   }
}

      BeanWriterWithSpringDependency test = (BeanWriterWithSpringDependency) factory
            .getBean("BeanWriterWithSpringDependency");
      test.run();
```

setWriter(Implementierung/des/Interface StringWriter) wird ausgeführt.

Spring Security

Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications



http://static.springsource.org/spring-security/site/features.html

Übersicht

1 Web-Anwendung

2 Entwicklungsumgebung

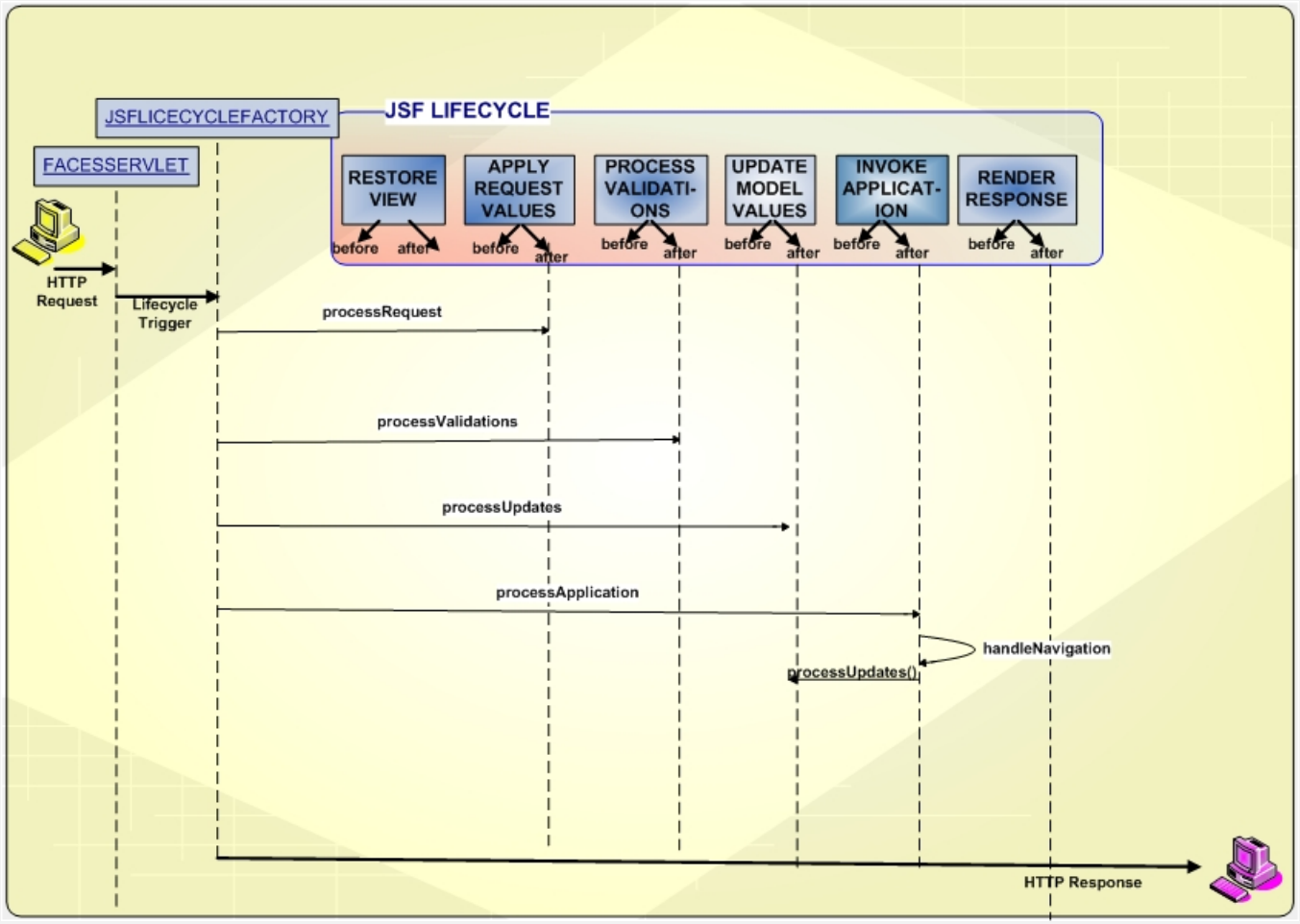
3 Web Applikation
- Architektur
- Der JSF request-processing lifecycle

Entwurfsmuster

Mode View Control

1. Das Model beschreibt die Geschäftslogik (business logic) und das Datenmodel (persitance storage). Liefert Daten, nimmt Änderungen der Daten entgegen und führt Auswertungen/Berechnungen durch.
2. Der View beschreibt die Benutzeroberfläche, d.h. Anzeige und Abfrage der Daten.
3. Der Controller beschreibt den Ablauf durch die Benutzerinteraktionen. Der Conroller steuert die (Reihenfolge der) Views und die Änderungen im Modell.

Der JSF request-processing lifecycle



Der JSF request-processing lifecycle

Vor/Nach jeder Phase kann von der Applikation in den lifecycle eingegriffen werden (listeners).

1. `Restore View`. Erzeugt Komponentenbaum der UI.
2. `Apply Request Values`. Übernimmt die Werte aller Eingabefelder der UI (als Strings).
3. `Process Validations/ Conversions`. Validiert und konvertiert Eingabefelder (als Datentyp).
4. `Update Model`. Übernimmt die Daten in die Applikation (backing bean, getters).
5. `Invoke Application`. Ruft Applikations Logik auf (action methods), bestimmt nächsten View (return value der action method).
6. `Render Response`. Erzeugt HTML, XUL, oder WML.

Modell einer Web-Applikation