Large Scale Software Engineering

Mike Helmick
University of Cincinnati
CS6028
Spring 2014

Details

Details

- Monday and Wednesday - 11:15 - 12:40
- Zimmer 302, Baldwin 649, Offsite
    - All via video conference, must go to room for your section
- **All recording of class (audio/video/still image) is strictly prohibited**

Participation

- Participation - required and expected. Please attend all class sessions.
- I hope for this to be a discussion based class
    - Preparation is essential
    - Please complete assigned readings in advance

Exams

- Two midterm exams
- One comprehensive final exam

Course Web site

- CascadeLMS
  - https://cascade.ceas.uc.edu
  - Everything will be on here, you are responsible for checking the Web site and keeping up with assignments

Me

Me

- Mike Helmick
- mike.helmick@uc.edu
- Office: Rhodes 812-A
- Office Hours: W,F 2:30-3:30pm, or by appointment
  - VC office hours available via Google Hangouts

Class Structure

- 1 Large team project, w/ several grading checkpoints
  - Project journals as well
- 3 exams
- An extra paper for graduate students
- Maybe some quizzes

Why Me?

- What am I teaching this class
- Over 10 years of industry experience
- I’ve worked on some large software
  - Sprint billing / metering system
  - Kroger labor management system
  - Amazon SimpleDB
  - Google+

A class in 2 parts

- **Software architecture**
  - Emphasis on distributed systems
- **The tools to build large software**
  - Emphasis on open source tools
  - All of the large systems that I've worked on use custom software to achieve scale, but we don't have access to it

What this class isn't

- A class on software engineering methodology
  - For me this simply isn't a question, we go agile or we don't go at all :)

Virtual Box

- I recommend getting virtual box installed this week
  - http://www.virtualbox.org
- I will be providing you a Linux image that has a bunch of items installed
- For your project, you will actually be turning in a virtual machine image with your project on it
  - **this is experimental, it may not work, let's try it.**

Today

Today

- What is application architecture?
- What makes an enterprise application?
- Toolset Introduction and setup

What is (software|application) architecture?

Any Ideas?

Software

Software

- Large software systems are difficult to build
  - There are many pieces
  - There are many team members
  - There are many areas of specialty
  - It is not uncommon to only be able to understand one part of a system at a time
    - It is extremely uncommon for any single team member to understand the entire system

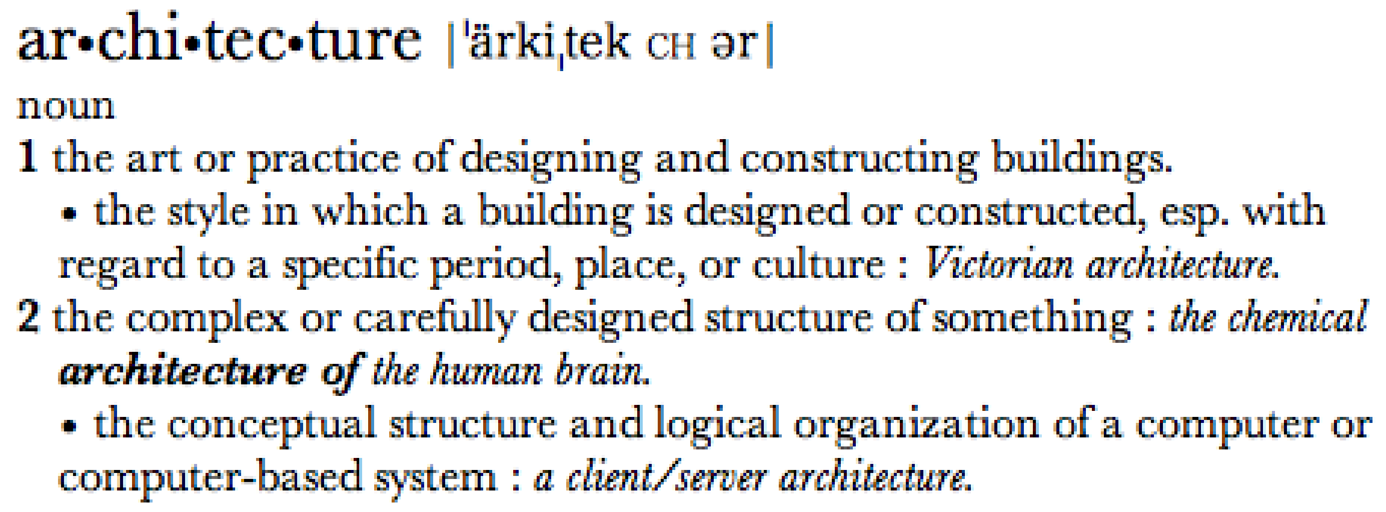
Architecture

Architecture

- What is it?

Architecture

- What is it?



ar•chi•tec•ture |ˈärkiˌtek CH ər|
noun
1 the art or practice of designing and constructing buildings.
• the style in which a building is designed or constructed, esp. with regard to a specific period, place, or culture : *Victorian architecture.*
2 the complex or carefully designed structure of something : *the chemical* ***architecture of*** *the human brain.*
• the conceptual structure and logical organization of a computer or computer-based system : *a client/server architecture.*

Architecture

Architecture

- A word that is often overused - especially in software
- No solid definition

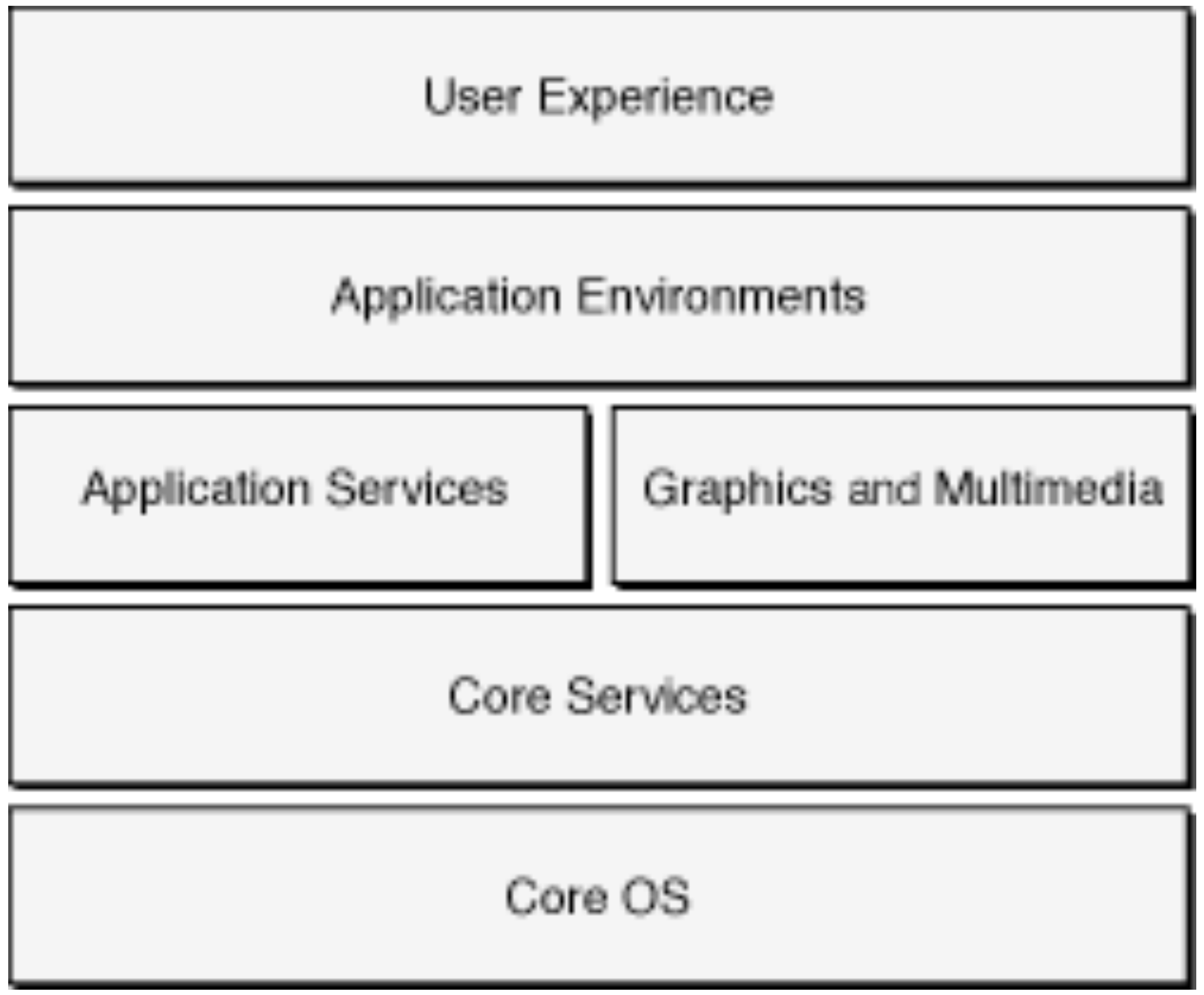
Architecture

Architecture

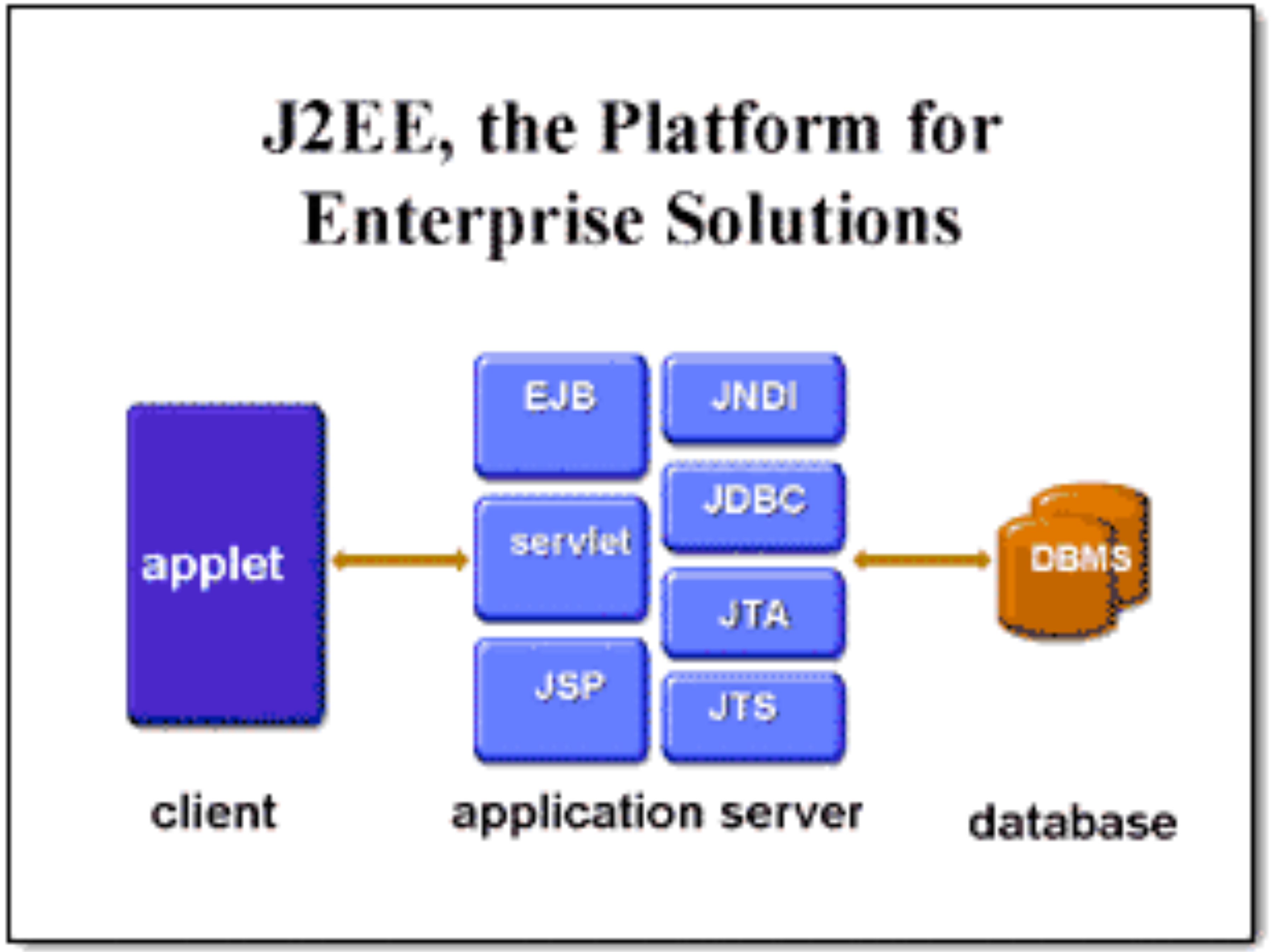
- The highest level breakdown of a system into its parts
- Decisions that are hard to change

Often summed up with pictures

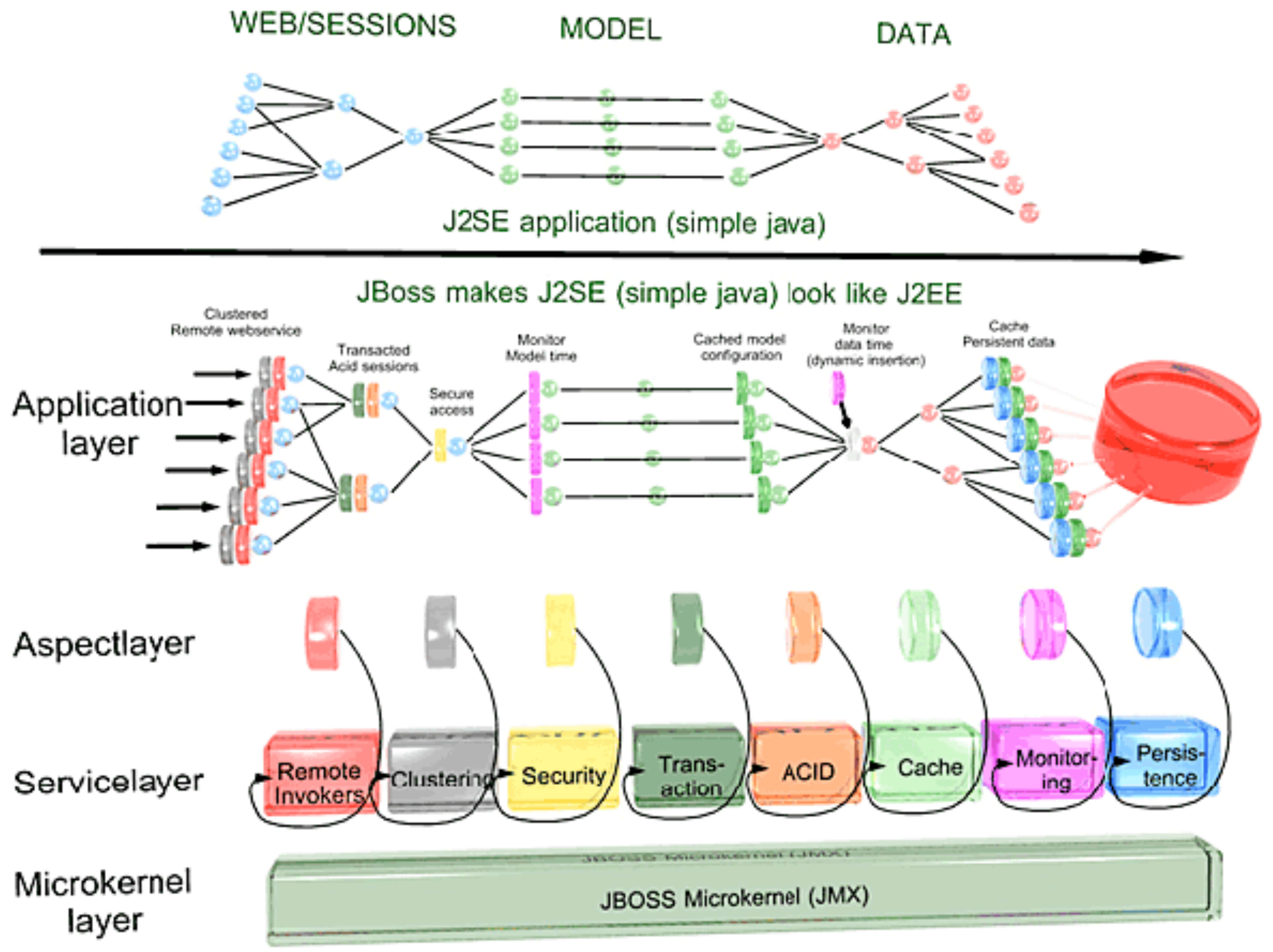
OS X


User Experience
Application Environments
Application Services
Graphics and Multimedia
Core Services
Core OS

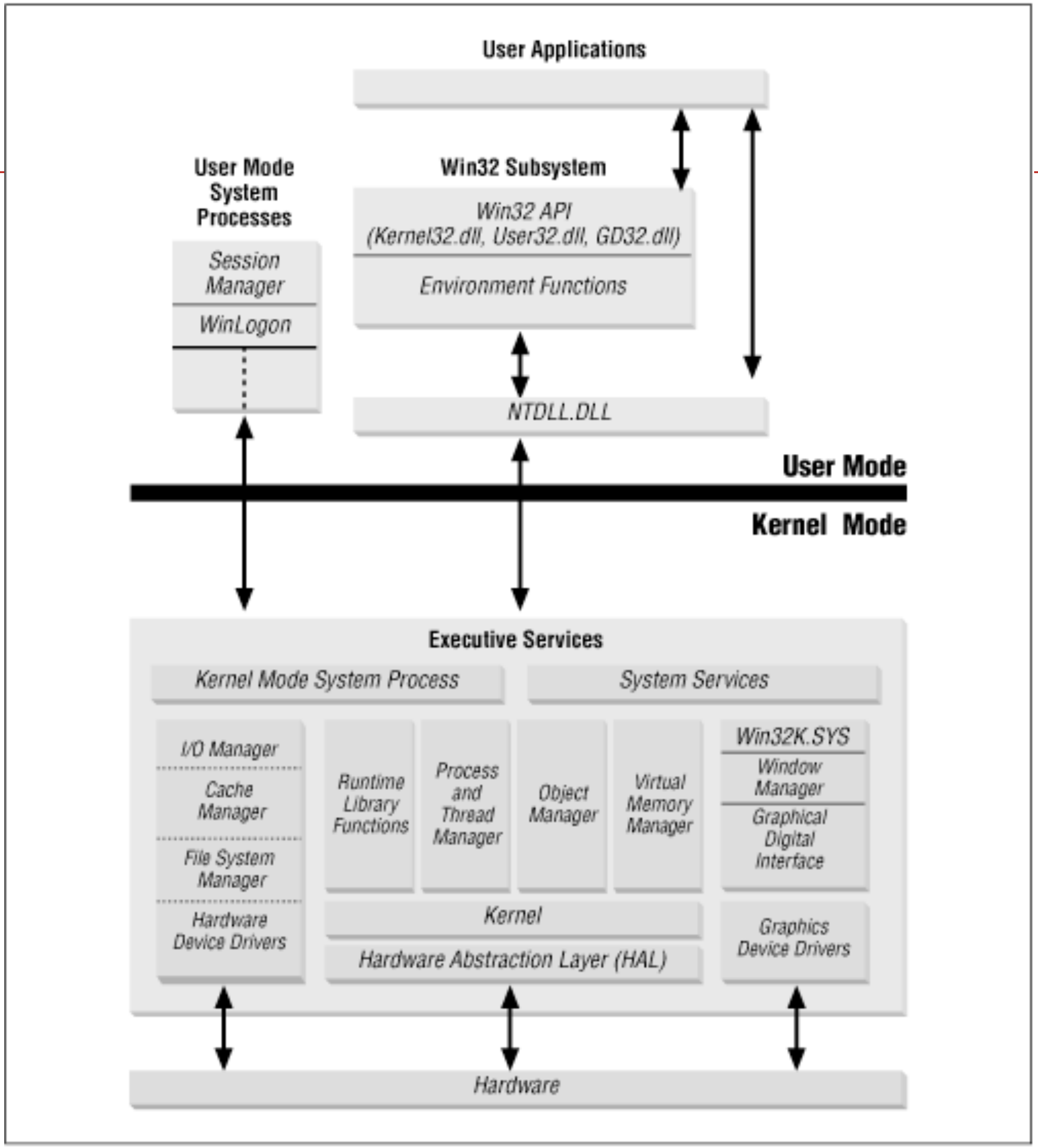
J2EE


J2EE, the Platform for Enterprise Solutions
applet
EJB
JNDI
servlet
JDBC
JTA
JSP
JTS
DBMS
client
application server
database

JBoss


WEB/SESSIONS
MODEL
DATA
J2SE application (simple java)
JBoss makes J2SE (simple java) look like J2EE
Clustered Remote webservice
Transacted Acid sessions
Secure access
Monitor Model time
Cached model configuration
Monitor data time (dynamic insertion)
Cache Persistent data
Application layer
Aspectlayer
Servicelayer
Remote Invokers
Clustering
Security
Trans-action
ACID
Cache
Monitor-ing
Persis-tence
Microkernel layer
JBOSS Microkernel (JMX)

Windows



User Applications
User Mode System Processes
Session Manager
WinLogon
Win32 Subsystem
Win32 API (Kernel32.dll, User32.dll, GD32.dll)
Environment Functions
NTDLL.DLL
User Mode
Kernel Mode
Executive Services
Kernel Mode System Process
System Services
I/O Manager
Cache Manager
File System Manager
Hardware Device Drivers
Runtime Library Functions
Process and Thread Manager
Object Manager
Virtual Memory Manager
Win32K.SYS
Window Manager
Graphical Digital Interface
Kernel
Hardware Abstraction Layer (HAL)
Graphics Device Drivers
Hardware

Architectures

Architectures

- Lots of pretty pictures, but it can be hard to decipher exactly what they mean.

What do I think?

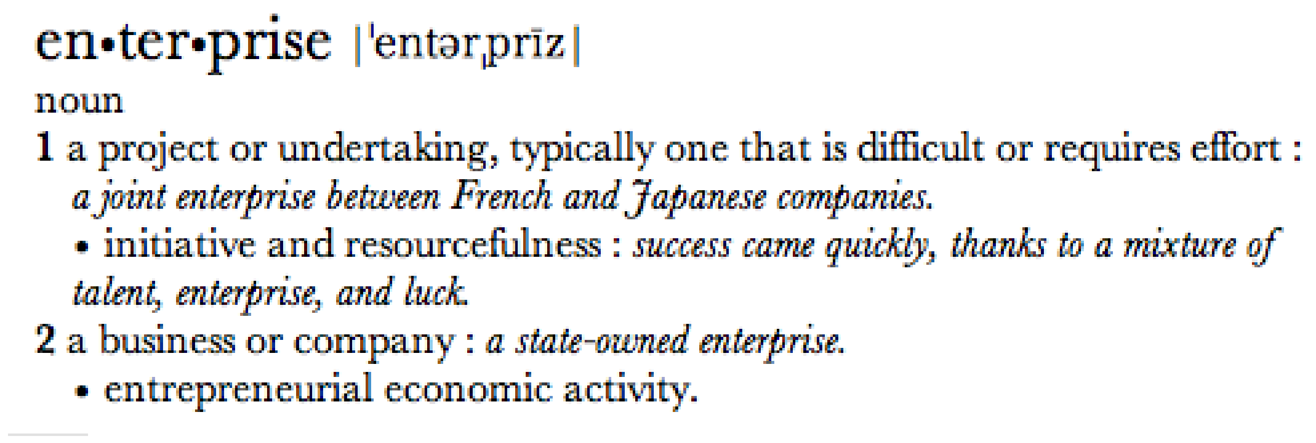
What do I think?

- “Decisions that are hard to change” is a good rule of thumb
- Example
  - Decision to use Oracle or Mysql - can be pretty easily changed during development
  - Design of your database tables/relationships - very difficult to change
  - - or -
    - Relational vs NoSQL solution, nearly impossible to change after you’re up and running

Enterprise

What are enterprise applications?

Enterprise



en•ter•prise |ˈentərˌprīz|
noun
1 a project or undertaking, typically one that is difficult or requires effort : *a joint enterprise between French and Japanese companies.*
• initiative and resourcefulness : *success came quickly, thanks to a mixture of talent, enterprise, and luck.*
2 a business or company : *a state-owned enterprise.*
• entrepreneurial economic activity.

Enterprise Applications

Enterprise Applications

- Other terms
  - Information Systems
  - Data Processing
- No precise definition

Examples

Examples

- Payroll
- Patient Records
- Shipping tracking
- cost analysis
- credit scoring
- insurance
- supply chain
- accounting
- customer service
- foreign exchange trading

Counter-examples

Counter-examples

- automobile fuel injection
- word processors
- elevator controllers
- chemical plan controllers
- telephone switches
- operating systems
- compilers
- games

Consumer Systems

- Large consumer systems
  - Social networks
  - Blogs
  - Shopping, etc…
- Closely model enterprise systems
  - Similar architecture
  - Much large scaling concerns (much large user base)

Persistent Data

Persistent Data

- Enterprise applications almost always involve persistent data
  - Needs to be around between runs of the program
  - Days - weeks - months - years - forever
- The servers for these programs are often “always up”
  - Uptime is measured, with goals of 99.95% (and higher) per year

Availability

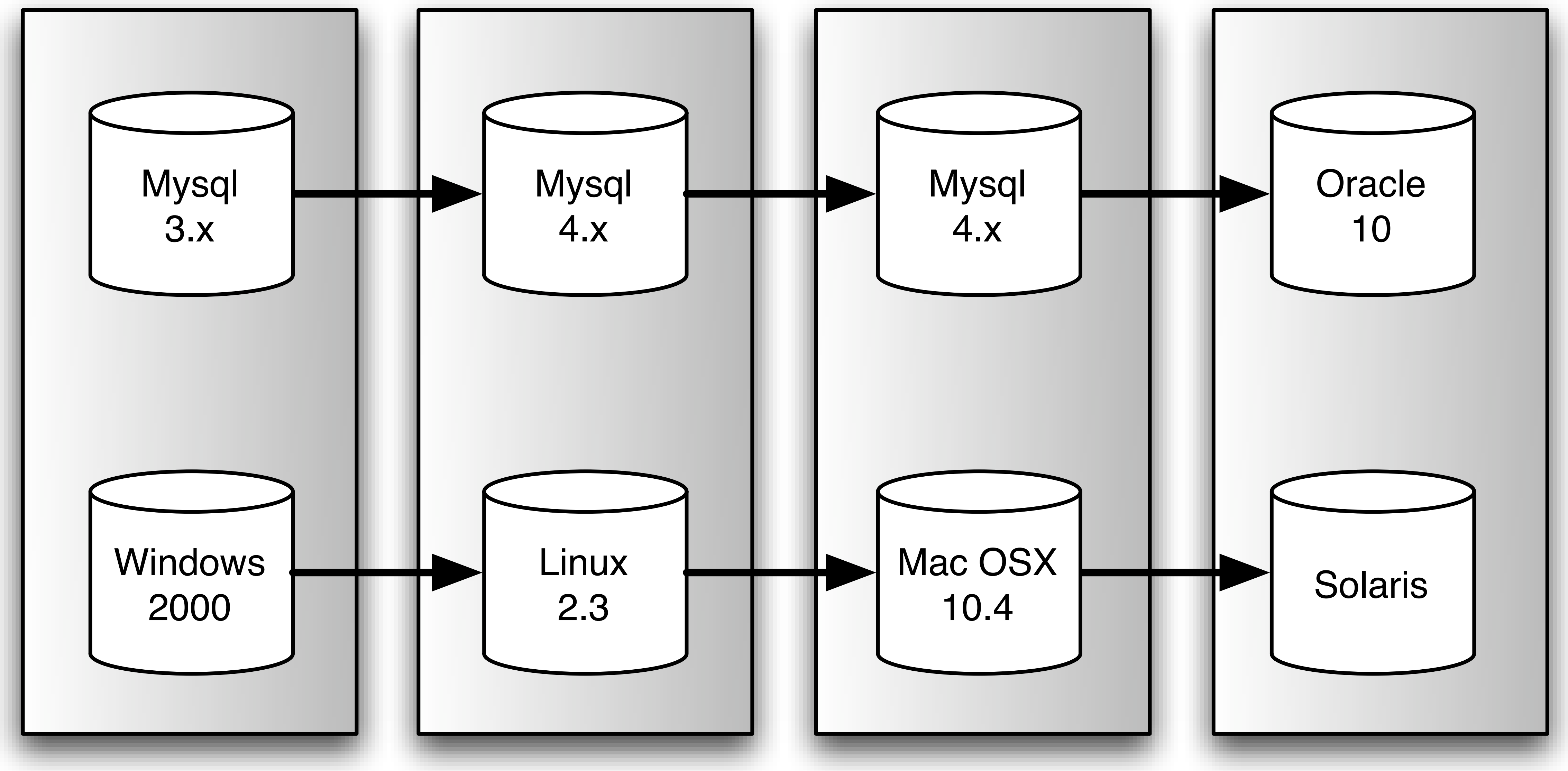
Availability	Yearly Downtime (Minutes)
99.900	525.60
99.950	262.80
99.990	52.56
99.999	5.26

Persistent

Persistent

- The data outlasts
    - database products / versions
    - operating systems
    - hardware
- And
    - The structure of the data itself

For Example


Mysql 3.x
Mysql 4.x
Mysql 4.x
Oracle 10
Windows 2000
Linux 2.3
Mac OSX 10.4
Solaris

Data

Data

- Large amounts
  - Giga-bytes to Terra-bytes, to petabytes
- For instance
  - 500 million users in a social network
  - Each with 1kb of profile date
  - Each with 5kb of connection data
    - That’s just under 3TB of total data!

Users

Users

- With something like Word - you only have 1 user at a time
- Enterprise applications will usually have many users (possibly thousands)
    - And many users at the same time (concurrency)

Concurrency

Concurrency

- Internet applications have many thousands of concurrent users
  - Think amazon.com

Screens

Screens

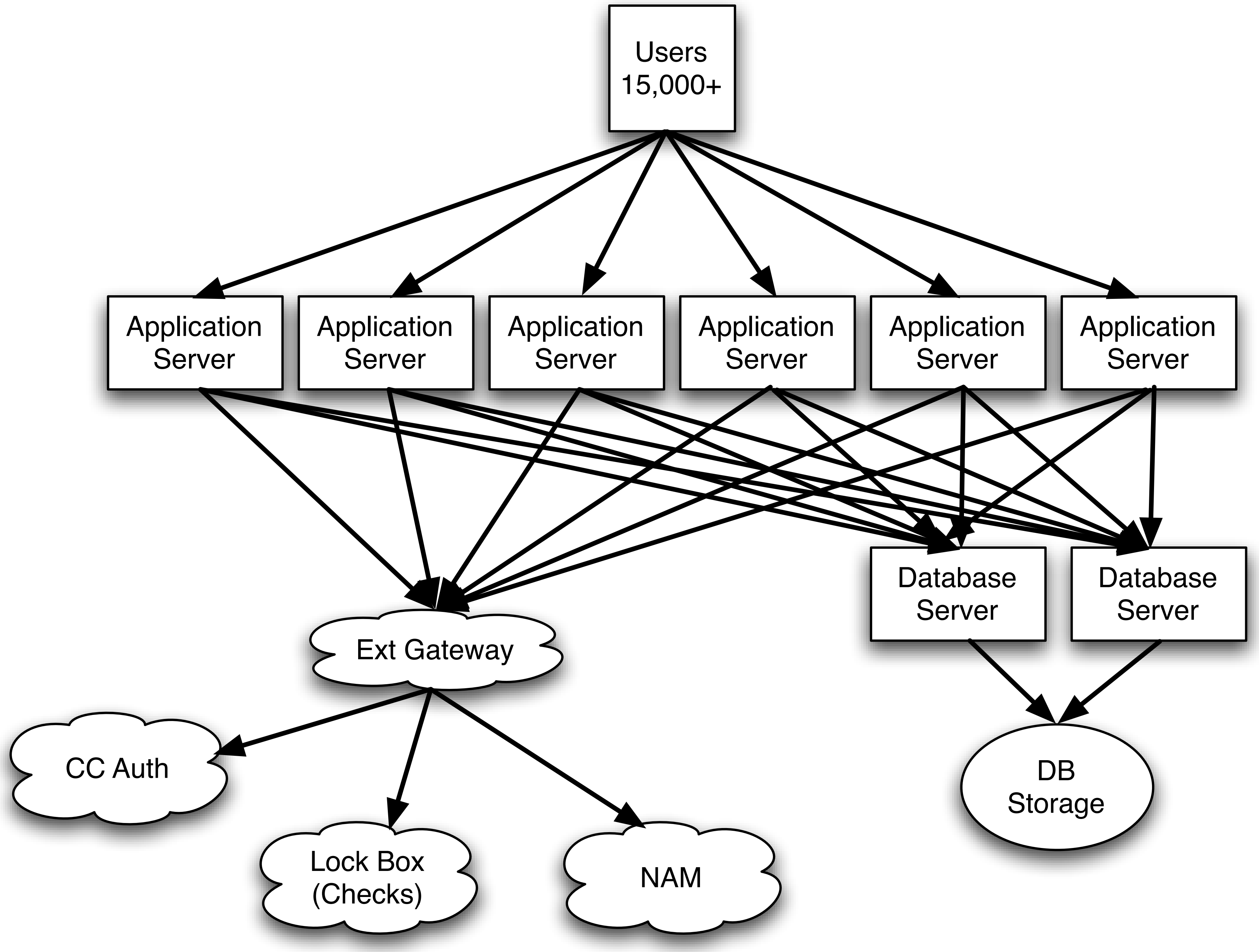
- Very very large number of interface screens
  - hundreds - possibly thousands

Integration

Integration

- Enterprise Applications almost always interact with many external systems

Integration



The catch: all systems interact in different ways

The catch: all systems interact in different ways

- Files
- ISAM
- Database
- CORBA
- Tuxedo
- RMI
- Web Services
    - XMLRPC
    - SOAP
- Custom
- HTTP
- etc...

Conceptual Dissonance

Conceptual Dissonance

- Suppose a large company is able to unify its information systems
  - Still have the problem on interpretation
  - What does the data mean?

Business Logic

Business Logic

- The process for doing business
  - Customer acquisition process
    - capture email
      - is valid address?
      - has it been previous registered?
      - is it on a blacklist?
      - etc...

Kinds of Enterprise Applications

Different Kinds

Different Kinds

- There are many different kinds of enterprise applications
- There are no “one size fits all” solutions

B2C

B2C

- “Business to consumer”
- Online shopping, banking, ticketing...
- Characteristics:
    - Many concurrent users
    - Simple domain logic
    - web presentation
    - database backed

B2B

B2B

- “Business to Business”
- Loan processing is a classic example
- Characteristics:
    - fewer concurrent users
    - much more complicated domain logic
    - “rules”

Internal

Internal

- Possibly smaller apps
- Less constraints - simple focus
- Often focused on the ability for quick development and deployment

Performance

Performance

- Almost impossible to determine ahead of time what should be done for performance
- Can be dependent on your host software / hardware
- Some simple guidelines can be made, but
    - best to measure once you get the system working (and then make optimizations)

Response Times

Response Times

- Amount of time it takes for an action to complete
- business logic may be quick 1 or 2 ms
- maybe database is 15 ms
- but an inefficient view render take 1.5 seconds
- Need to look at the whole picture

Responsiveness

Responsiveness

- How quickly processing begins for a request for service
- Say a request as a response time of .5s, but it takes 5 seconds for the request to start processing
- Can be an issue in queuing systems or server side processing systems (Web!)

Latency

Latency

- Minimum time to get a response
- Can usually be measured by creating a button / form / API call that does nothing
- A developer can do nothing to improve latency
    - i.e. hardware / network / os / server dependent

Throughput

Throughput

- How much X you can do in Y amount of time
    - 10 requests per second

Performance

Performance

- When people talk about an applications performance the are usually talking about
  - Throughput
  - Response Time

Load

Load

- How much stress the system is under
  - i.e. how many users currently logged on
- Used in the context of other measurements
  - with 100 users, response time is .5s
  - with 1000 users, response time is 1.5s

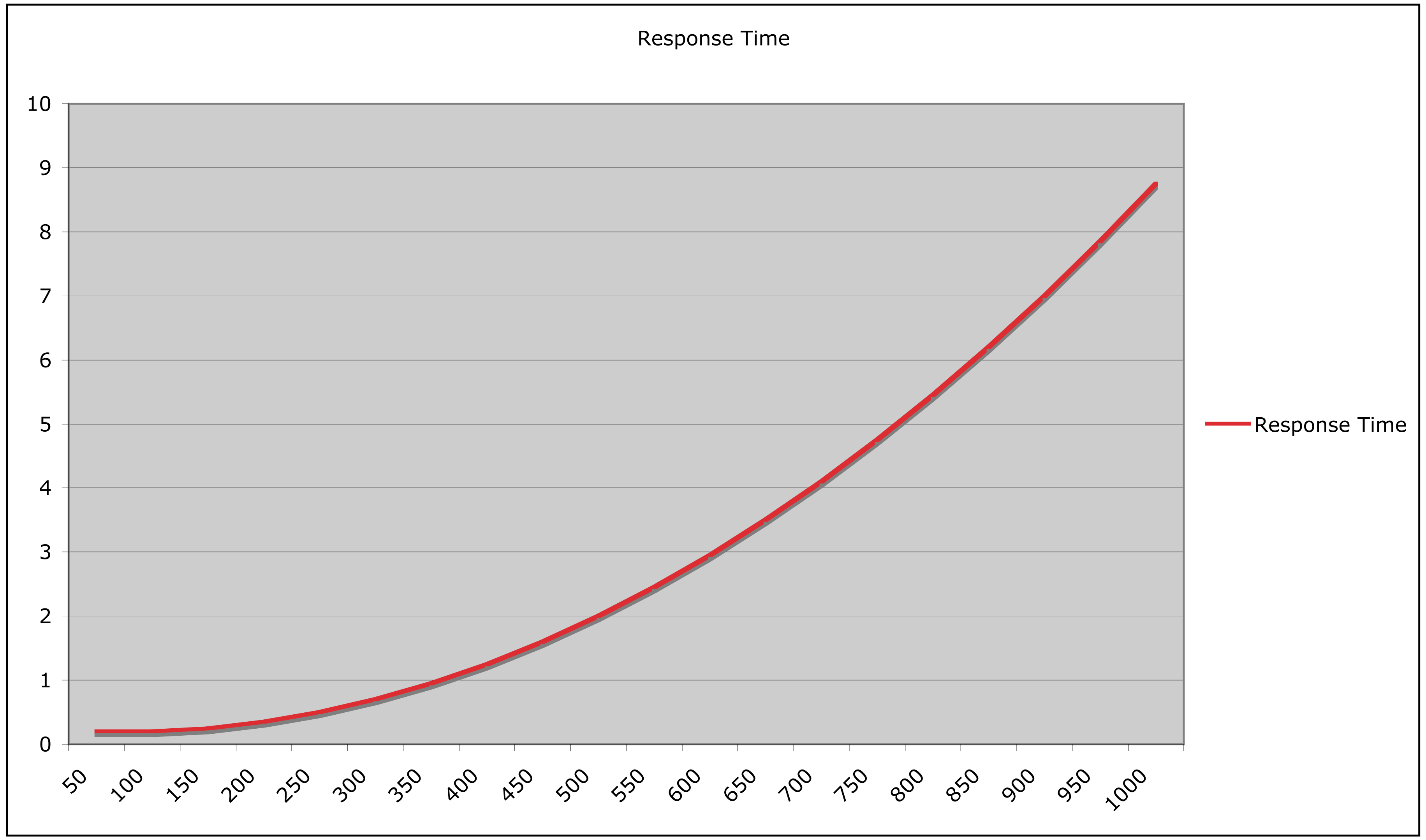
Load Sensitivity

Load Sensitivity

- How sensitive system performance is to changes in the load

Load Sensitivity

- How sensitive system performance is to changes in the load



Efficiency

Efficiency

- How efficiency the resources are used
- Which is more efficient?

Efficiency

- How efficiency the resources are used
- Which is more efficient?

CPUs	Transactions/Sec
8	1000
4	600

Capacity

Capacity

- Maximum throughput
    - Actually throughput, not theoretical

Scalability

Scalability

- Ability of application to gain capacity by adding resources
  - Vertical Scalability: increasing power of a single server
  - Horizontal Scalability: adding additional servers

Building

Building

- Should we build for capacity or scalability?

Textbooks

Textbooks

- Patterns are
  - not comprehensive
  - a starting point

Enterprise Programming

Enterprise Apps

Microsoft .NET

Enterprise Apps

- Written in many programming languages
    - C/C++, COBOL, Fortrain, Java, Ruby, Smalltalk, Objective-C
- Written with many frameworks
    - .NET, J2EE/JEE, Rails

Principles

Principles

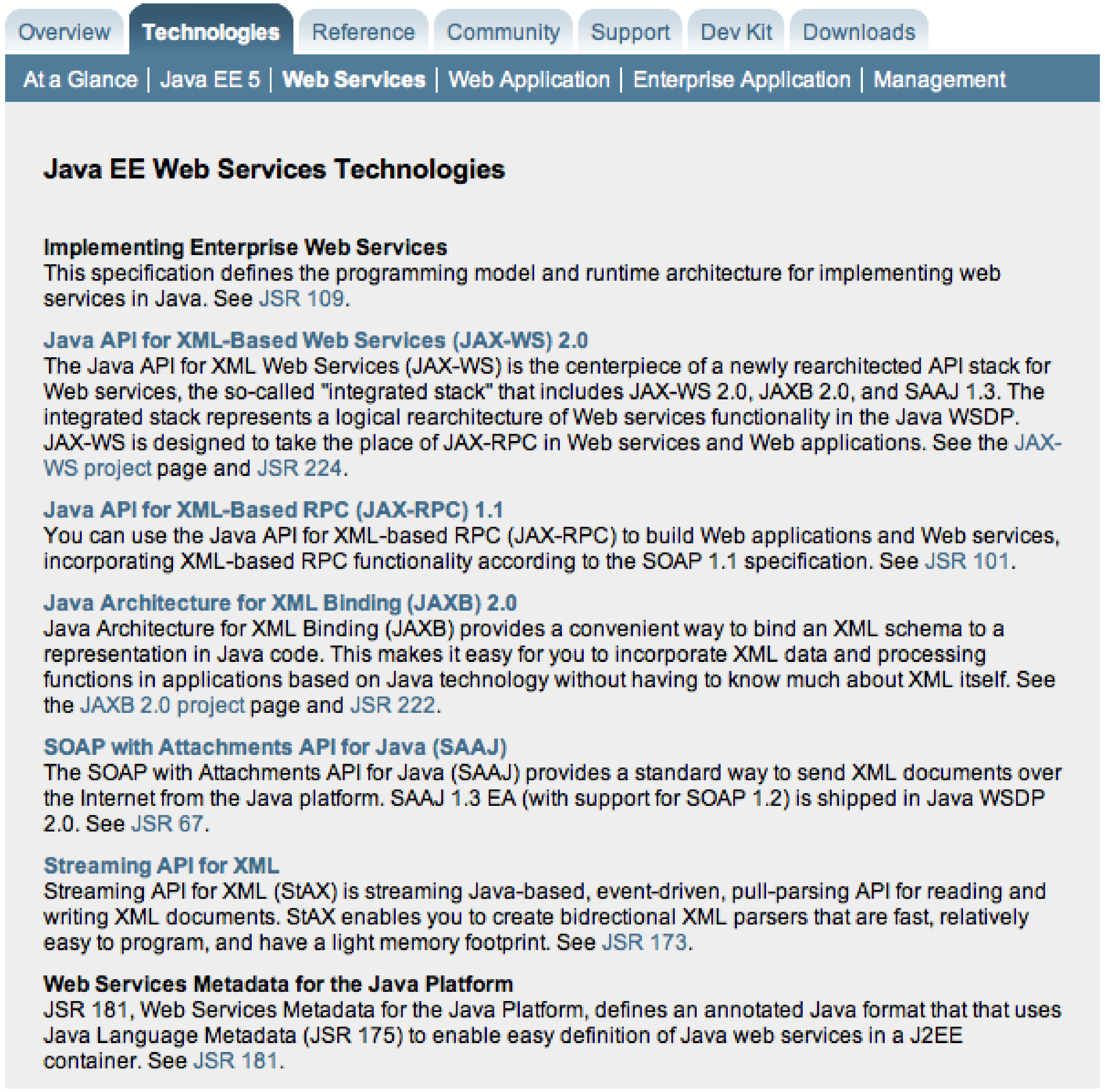
- The principles we study can be translated from one language / framework to another
- We try to point out best practices and commonalities, while implementing a project using Java

J(2)EE Landscape

J(2)EE Landscape

- Java Enterprise Edition is made up of many frameworks / specifications
- Some of the framework code comes from ~~Sun~~ Oracle - some does not!
  - very different from the Microsoft model
  - Open source community has many frameworks

Web Services


Overview
Technologies
Reference
Community
Support
Dev Kit
Downloads
At a Glance | Java EE 5 | Web Services | Web Application | Enterprise Application | Management

Java EE Web Services Technologies

Implementing Enterprise Web Services
This specification defines the programming model and runtime architecture for implementing web services in Java. See JSR 109.

Java API for XML-Based Web Services (JAX-WS) 2.0
The Java API for XML Web Services (JAX-WS) is the centerpiece of a newly rearchitected API stack for Web services, the so-called "integrated stack" that includes JAX-WS 2.0, JAXB 2.0, and SAAJ 1.3. The integrated stack represents a logical rearchitecture of Web services functionality in the Java WSDP. JAX-WS is designed to take the place of JAX-RPC in Web services and Web applications. See the JAX-WS project page and JSR 224.

Java API for XML-Based RPC (JAX-RPC) 1.1
You can use the Java API for XML-based RPC (JAX-RPC) to build Web applications and Web services, incorporating XML-based RPC functionality according to the SOAP 1.1 specification. See JSR 101.

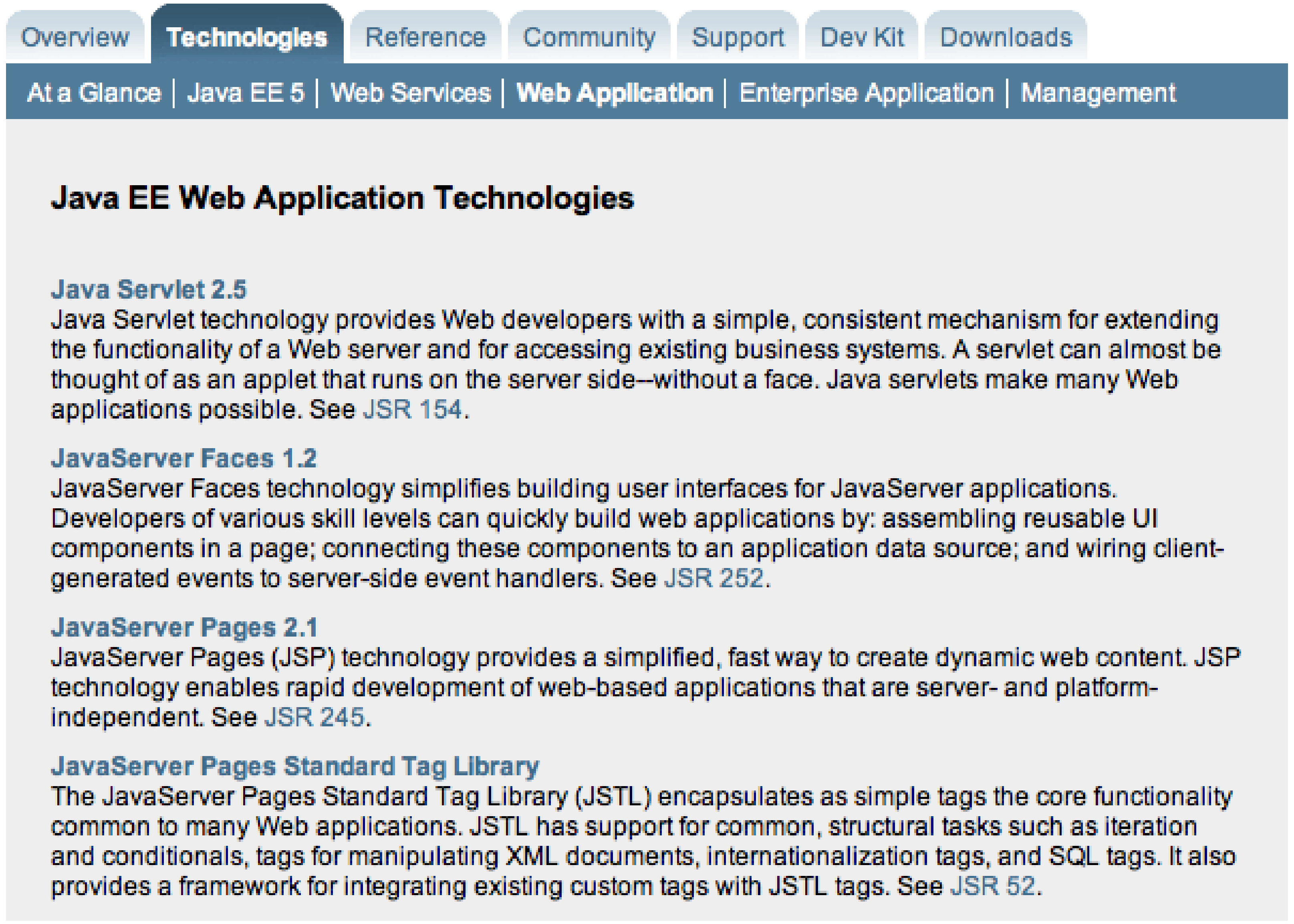
Java Architecture for XML Binding (JAXB) 2.0
Java Architecture for XML Binding (JAXB) provides a convenient way to bind an XML schema to a representation in Java code. This makes it easy for you to incorporate XML data and processing functions in applications based on Java technology without having to know much about XML itself. See the JAXB 2.0 project page and JSR 222.

SOAP with Attachments API for Java (SAAJ)
The SOAP with Attachments API for Java (SAAJ) provides a standard way to send XML documents over the Internet from the Java platform. SAAJ 1.3 EA (with support for SOAP 1.2) is shipped in Java WSDP 2.0. See JSR 67.

Streaming API for XML
Streaming API for XML (StAX) is streaming Java-based, event-driven, pull-parsing API for reading and writing XML documents. StAX enables you to create bidirectional XML parsers that are fast, relatively easy to program, and have a light memory footprint. See JSR 173.

Web Services Metadata for the Java Platform
JSR 181, Web Services Metadata for the Java Platform, defines an annotated Java format that that uses Java Language Metadata (JSR 175) to enable easy definition of Java web services in a J2EE container. See JSR 181.

Web Applications


Overview
Technologies
Reference
Community
Support
Dev Kit
Downloads
At a Glance | Java EE 5 | Web Services | Web Application | Enterprise Application | Management

Java EE Web Application Technologies

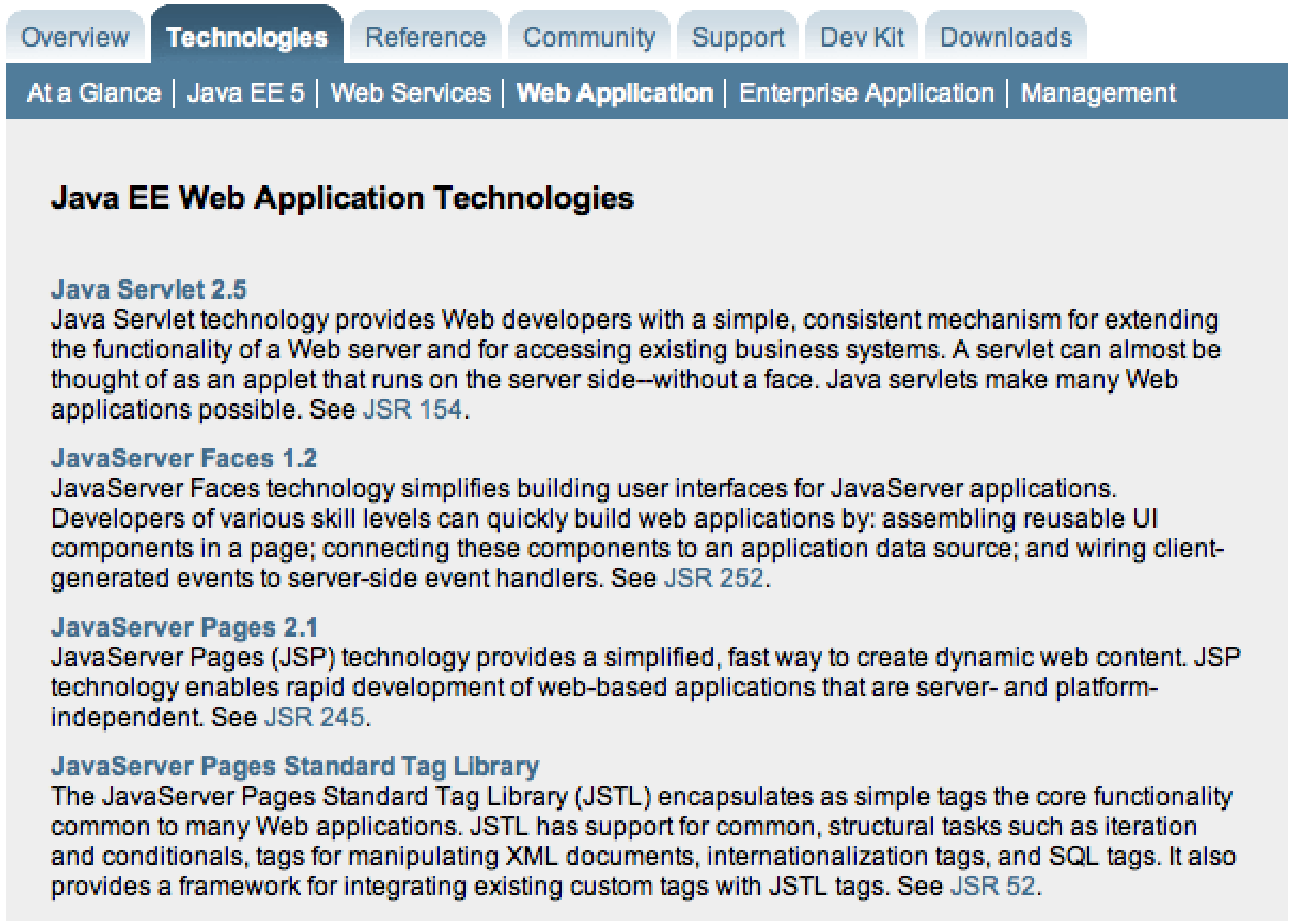
Java Servlet 2.5
Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side--without a face. Java servlets make many Web applications possible. See JSR 154.

JavaServer Faces 1.2
JavaServer Faces technology simplifies building user interfaces for JavaServer applications. Developers of various skill levels can quickly build web applications by: assembling reusable UI components in a page; connecting these components to an application data source; and wiring client-generated events to server-side event handlers. See JSR 252.

JavaServer Pages 2.1
JavaServer Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server- and platform-independent. See JSR 245.

JavaServer Pages Standard Tag Library
The JavaServer Pages Standard Tag Library (JSTL) encapsulates as simple tags the core functionality common to many Web applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags. See JSR 52.

Web Applications


Overview
Technologies
Reference
Community
Support
Dev Kit
Downloads
At a Glance | Java EE 5 | Web Services | Web Application | Enterprise Application | Management

Java EE Web Application Technologies

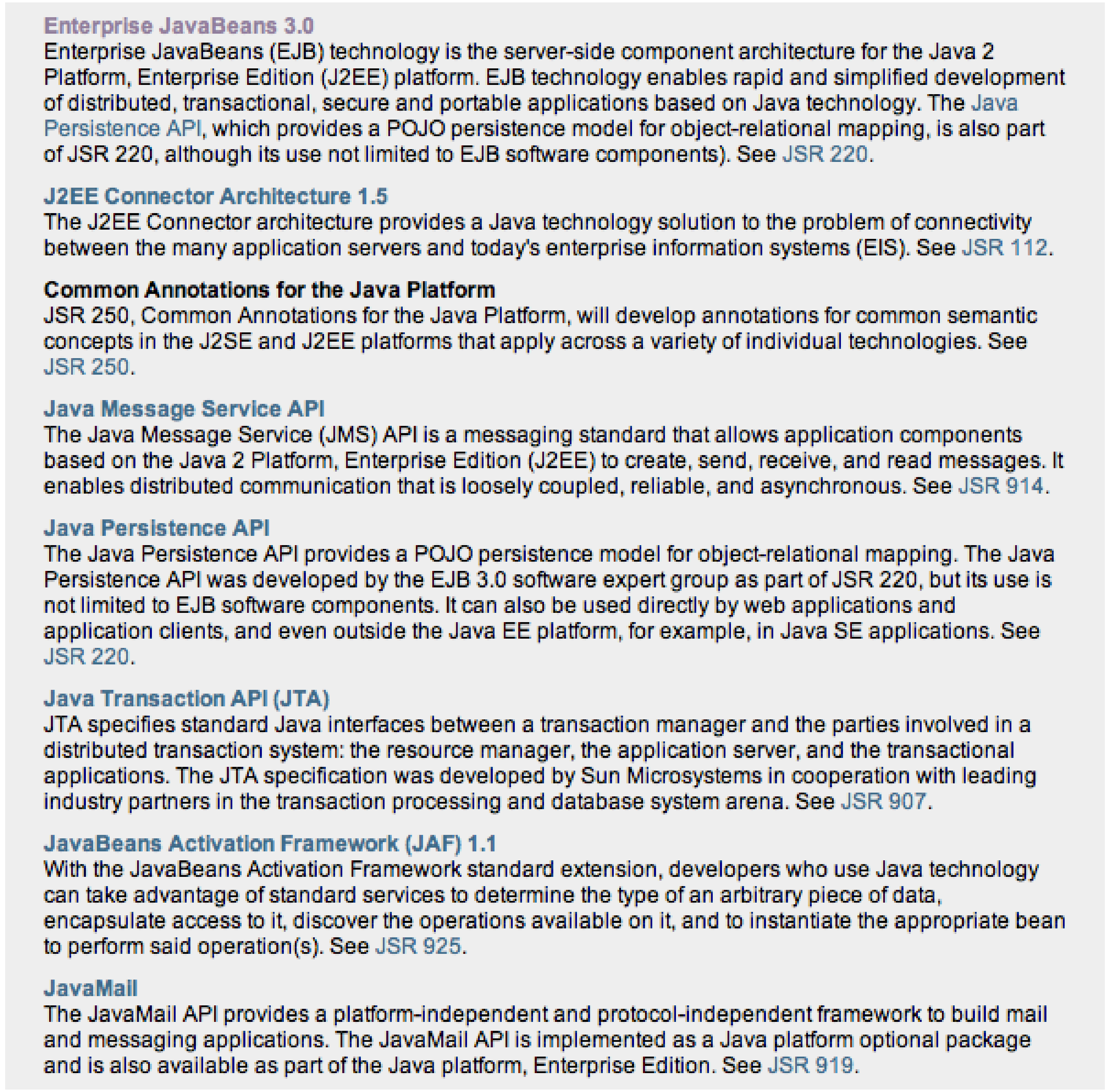
Java Servlet 2.5
Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side--without a face. Java servlets make many Web applications possible. See JSR 154.

JavaServer Faces 1.2
JavaServer Faces technology simplifies building user interfaces for JavaServer applications. Developers of various skill levels can quickly build web applications by: assembling reusable UI components in a page; connecting these components to an application data source; and wiring client-generated events to server-side event handlers. See JSR 252.

JavaServer Pages 2.1
JavaServer Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server- and platform-independent. See JSR 245.

JavaServer Pages Standard Tag Library
The JavaServer Pages Standard Tag Library (JSTL) encapsulates as simple tags the core functionality common to many Web applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags. See JSR 52.

Enterprise Apps



Enterprise JavaBeans 3.0
Enterprise JavaBeans (EJB) technology is the server-side component architecture for the Java 2 Platform, Enterprise Edition (J2EE) platform. EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology. The Java Persistence API, which provides a POJO persistence model for object-relational mapping, is also part of JSR 220, although its use not limited to EJB software components). See JSR 220.

J2EE Connector Architecture 1.5
The J2EE Connector architecture provides a Java technology solution to the problem of connectivity between the many application servers and today's enterprise information systems (EIS). See JSR 112.

Common Annotations for the Java Platform
JSR 250, Common Annotations for the Java Platform, will develop annotations for common semantic concepts in the J2SE and J2EE platforms that apply across a variety of individual technologies. See JSR 250.

Java Message Service API
The Java Message Service (JMS) API is a messaging standard that allows application components based on the Java 2 Platform, Enterprise Edition (J2EE) to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous. See JSR 914.

Java Persistence API
The Java Persistence API provides a POJO persistence model for object-relational mapping. The Java Persistence API was developed by the EJB 3.0 software expert group as part of JSR 220, but its use is not limited to EJB software components. It can also be used directly by web applications and application clients, and even outside the Java EE platform, for example, in Java SE applications. See JSR 220.

Java Transaction API (JTA)
JTA specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications. The JTA specification was developed by Sun Microsystems in cooperation with leading industry partners in the transaction processing and database system arena. See JSR 907.

JavaBeans Activation Framework (JAF) 1.1
With the JavaBeans Activation Framework standard extension, developers who use Java technology can take advantage of standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and to instantiate the appropriate bean to perform said operation(s). See JSR 925.

JavaMail
The JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications. The JavaMail API is implemented as a Java platform optional package and is also available as part of the Java platform, Enterprise Edition. See JSR 919.

History

History

- In 1999 - Java & its extensions were repackaged into 3 editions
  - Java 2 Micro Edition
  - Java 2 Standard Edition
  - Java 2 Enterprise Edition
- Everything is now on Java 7 versions (8 upcoming)

Components

Components

- JDBC (Java DataBase Connectivity)
- Java Servlets
- Java Server Pages
- JNDI (Java Naming and Directory Interface)
- RMI (Remote Method Invocation)
- Java Mail

Components

Components

- EJB (Enterprise JavaBeans)
- JTA (Java Transaction API)
- JMS (Java Messaging Service)

Deployment

Deployment

- JAR - standard Java Archive
- WAR - Web Java Archive
- EAR - Enterprise Java Archive

- Standard for deployment on all servers

JEE Vendors

JEE Vendors

- Sun Microsystems
  - Java developer
  - Still controls the official specifications
    - Java Community Process (JCP)

JEE Vendors

JEE Vendors

- JDBC Drivers
  - Provided by various database vendors or open source projects
  - Oracle, MySQL, HSQLDB, SQLServer, etc...

JEE Vendors

JEE Vendors

- JDBC Drivers
  - Provided by various database vendors or open source projects
  - Oracle, MySQL, HSQLDB, SQLServer, etc...
  - http://developers.sun.com/product/jdbc/drivers/browse_all.jsp

JEE Vendors

JEE Vendors

- Application Servers
  - BEA Weblogic
  - IBM WebSphere
  - iPlanet
  - JBoss (Open Source)

JEE Vendors

JEE Vendors

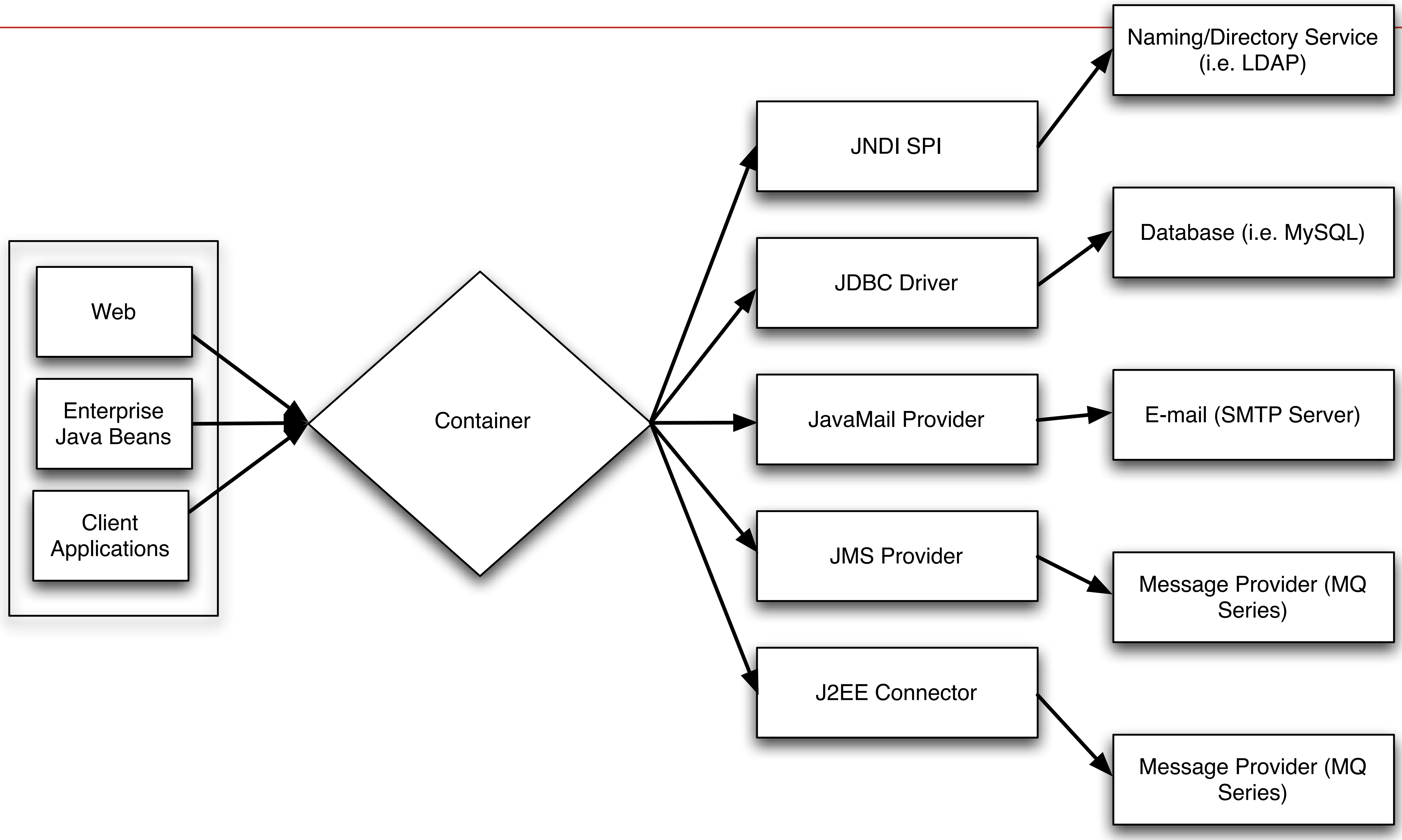
- Web Servers (These are all open source)
    - Apache Tomcat
    - Jetty
    - Resin

JEE Architecture

JEE Architecture

- Based on:
  - Components
  - Archives
  - Containers
  - Connectors

JEE


Web
Enterprise Java Beans
Client Applications
Container
JNDI SPI
JDBC Driver
JavaMail Provider
JMS Provider
J2EE Connector
Naming/Directory Service (i.e. LDAP)
Database (i.e. MySQL)
E-mail (SMTP Server)
Message Provider (MQ Series)
Message Provider (MQ Series)

Database

Database

- RDBMS (Relational DataBase Management System)
- With the tools we have available now - you don’t have to be a database expert to use one

SQL

SQL

- Database systems typically use the Structured Query Language (SQL) as their programming language
- Many early systems would directly embed SQL statements in their program
  - Reduces portability (everyone has different flavors of SQL)
  - Makes database changes hard to track in your code

JDBC

JDBC

- Let to a common way to connect to databases
- A common way to execute commands
- but...
  - SQL Still embedded in code

Removing SQL

Removing SQL

- Several solutions have been introduced
    - EJB with CMP
    - Hibernate
- We'll talk about patterns for this

Removing SQL - A Different Way

mongoDB

Cassandra

Amazon SimpleDB (beta)

Removing SQL - A Different Way

- "NoSQL" solutions are more and more popular
  - Typically these systems are more scalable, more robust (higher availability)
  - But, at the cost of maybe a less power query language, inability to do foreign key constraints, limited transaction scopes, etc…

mongoDB

Cassandra

Amazon SimpleDB (beta)

RMI

RMI

- In the past (late mid to late 90s) - distributed computing were handled by
  - CORBA (Common Object Request Broker Architecture)
  - COM (Component Object Model - MS)

RMI

RMI

- Java can talk both CORBA and COM
- but also includes Java RMI
  - Remote Method Invocation
  - Both CORBA and RMI use the Internet Inter-ORB Protocol (IIOP)
    - i.e. they can talk to each other

RMI

RMI

- Used for (pre 3.0) Enterprise Java Beans
- Can also be used standalone
- Advantage to RMI
    - Looks just like a function call locally, but it goes over the network and executes somewhere else

Now…

Now...

- Pretty much everyone uses Web Services for external distribution
- Internally:
    - Web Services
    - JSON-RPC
    - CORBA / RMI / COM
    - Custom
    - X over HTTP

E-Mail

E-Mail

- Applications need to be able to send E-Mail
    - think about all the registration confirmation, order, etc... e-mails that you get from web sites every day

Receive Side

- Receiving emails often needs to trigger processing as well

EJB

EJB

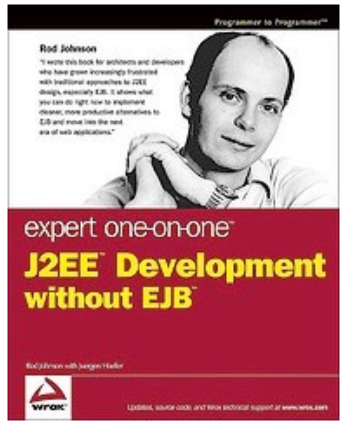
- Enterprise JavaBeans
    - The “traditional” approach for enterprise class Java systems
    - Proved to be overly complicated
        - The community has come up with better ways

So...

So...

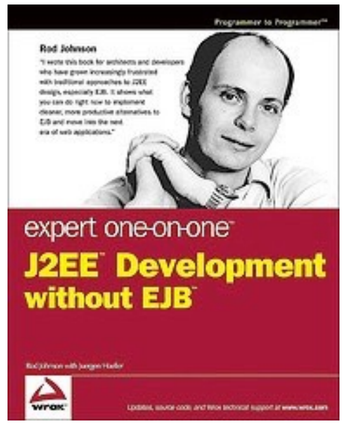
- Here are a few slides on why we’re not going to cover EJB

Complicated Source

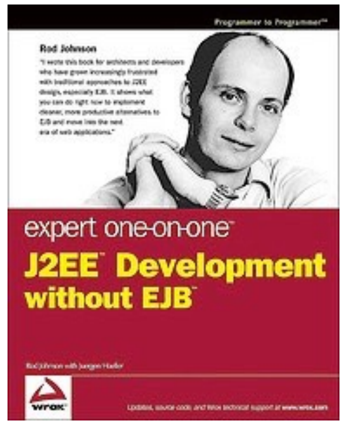


Complicated Source

- To develop an EJB you need several source files
  - EJB Home / EJB Object interfaces
  - EJB Object implementation

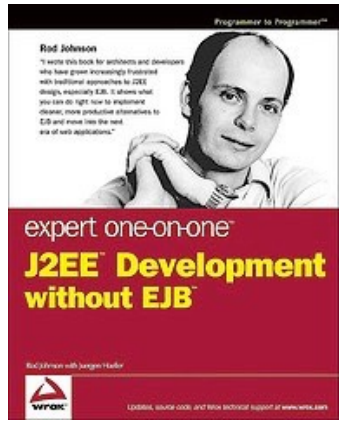


Remoting

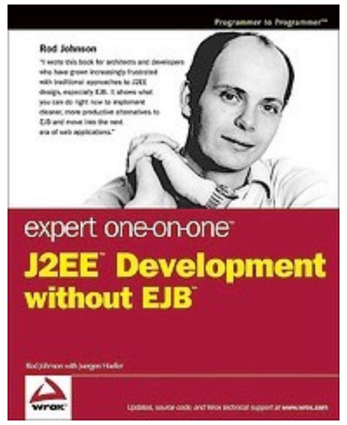

Programmer to Programmer™
Rod Johnson
expert one-on-one™
J2EE™ Development without EJB™
Rod Johnson with Juergen Hoeller
wrox

Remoting

- EJB uses RMI for remoting
- and web services are more in style now
  - why?
  - The transactions are
    - 1) All human readable
    - 2) Can be made compatible with ANYTHING

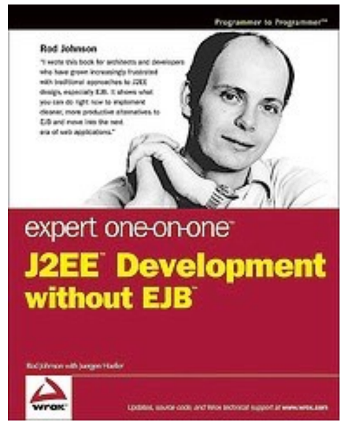


Distributed Objects

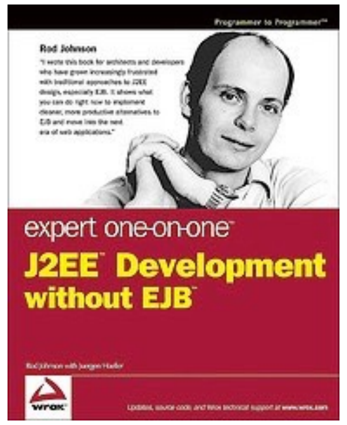


Distributed Objects

- Very good theoretical idea
- Sort of falls apart in practice
- We’ve achieved greater scalability by running more layers on the same machine and adding layers to gain capacity

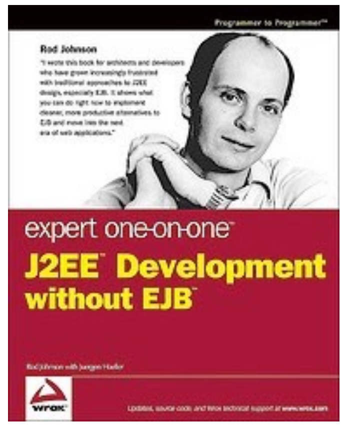


What’s used

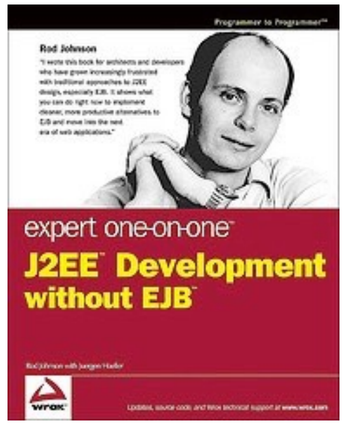

Rod Johnson
expert one-on-one
J2EE Development without EJB
wrox

What’s used

- Many applications only uses stateless session beans and message driven beans
- This requires an EJB container which introduces great overhead (dev time, cost, boot time, memory footprint)
- And these things can all be accomplished with web services

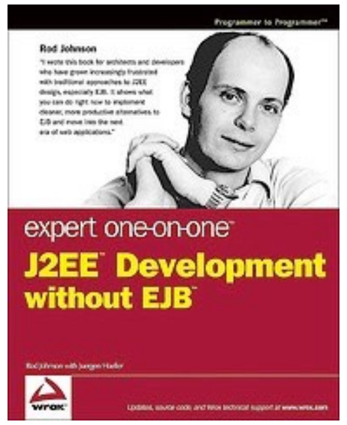


Understanding

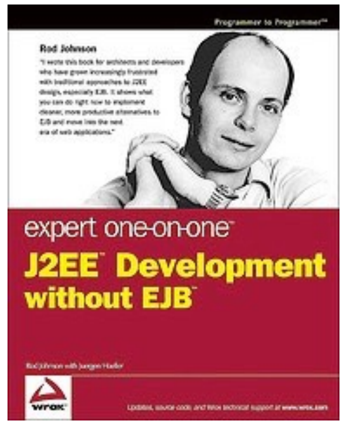


Understanding

- Many developers just don’t understand EJBs
- Leads to improper deployments / usage
    - Which leads to code which is very difficult to maintain

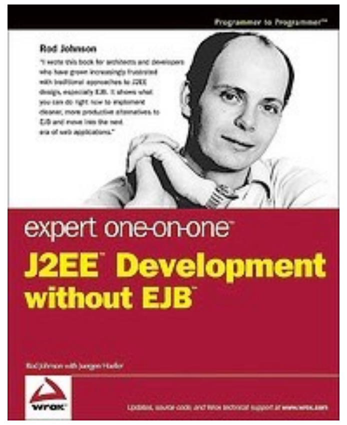


Specification

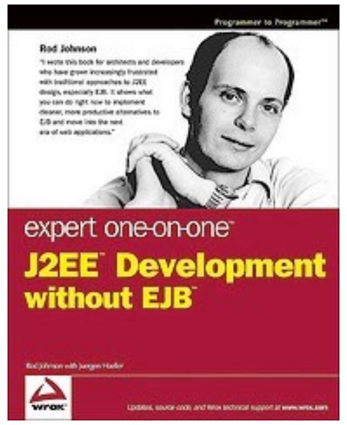


Specification

- In order to fix the problem
- The EJB specification is getting more complicated

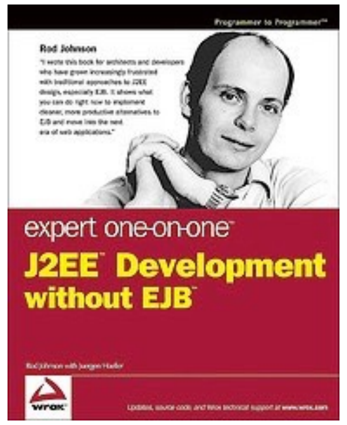


Productivity

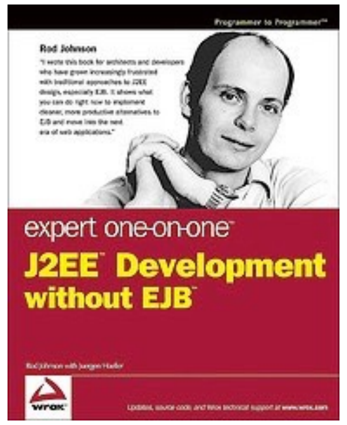


Productivity

- High complexity
- Low productivity

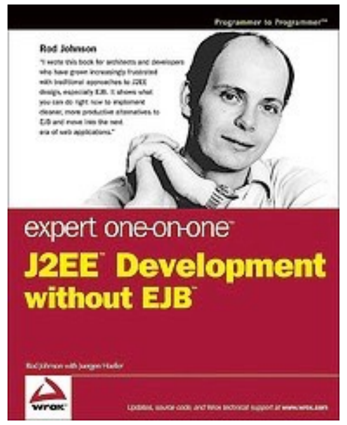


TDD

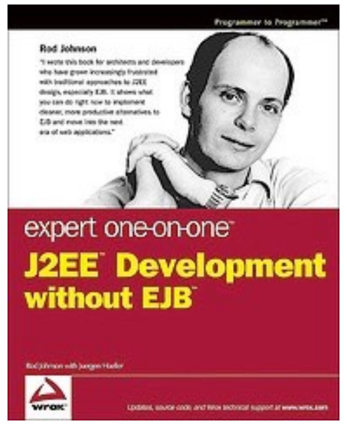


TDD

- EJBs are hard to test
- Need to boot the container, lots of dependencies to satisfy

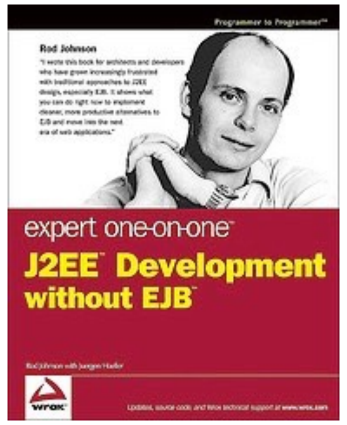


AOP

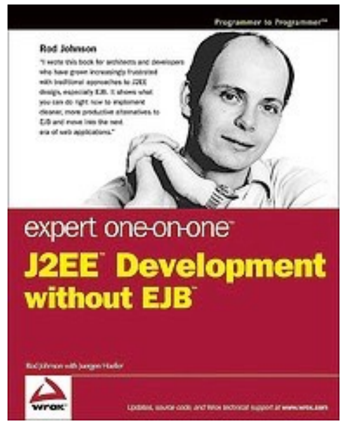

Programmer to Programmer
Rod Johnson
expert one-on-one
J2EE Development without EJB
wrox

AOP

- The emergency of Aspect Oriented Programming has given rise to more powerful approaches

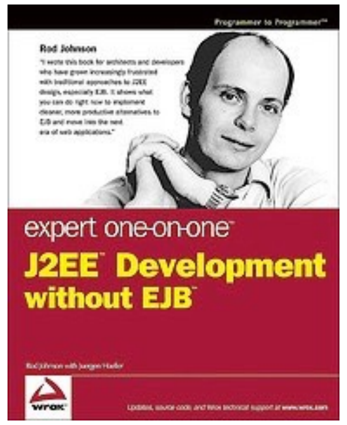


Metadata



Metadata

- .NET has made good use of medata
- Better than the approach of the huge XML files for EJB deployment



J2EE w/o EJB

J2EE w/o EJB

- You can still to J2EE without EJB
- JNDI, JTS & JTA (for transactions), JCA, JavaMail, etc...
- i.e. there are still many powerful services and building blocks that we can use to create enterprise applications

Learning EJB

Learning EJB

- If you really want to learn EJB...
  - http://search.safaribooksonline.com/0596001231
  - http://search.safaribooksonline.com/059600978X