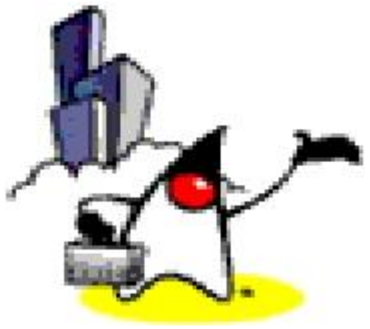
Introduction to Java EE (J2EE)

Objectives

- Understanding J2EE
- Understanding J2EE architecture and platform
- Understanding why J2EE is a great platform for development and deployment of web services

Contents

What is J2EE?



What Is the J2EE?

- Open and standard based platform for developing, deploying and managing n-tier, Web-based, server-centric, and component-based enterprise applications

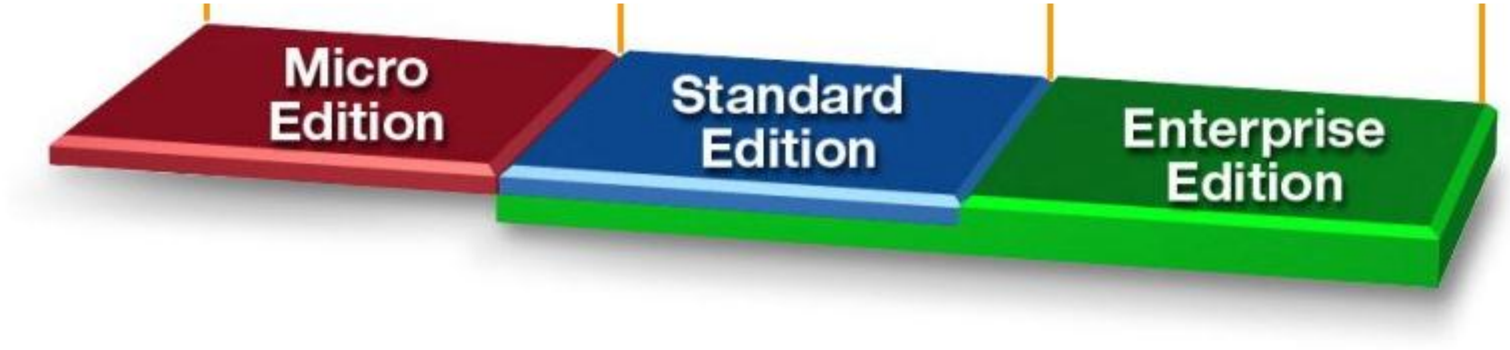
The Java™ Platform

Java Technology Enabled Devices

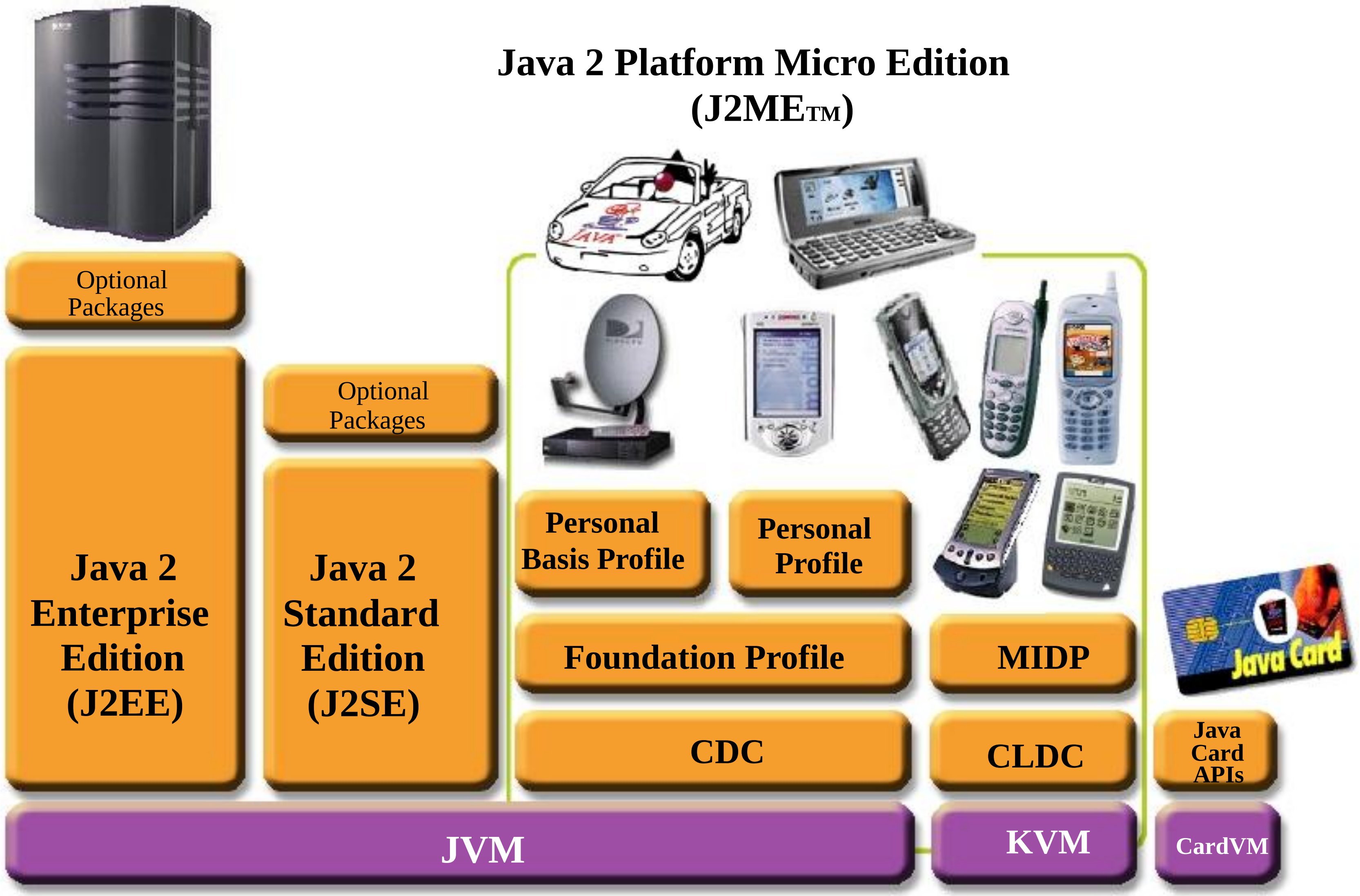
Java Technology Enabled Desktop

Workgroup Server

High-End Server



The Java TM Platform



What Makes Up J2EE?

- API and Technology specifications
- Development and Deployment Platform
- Standard implementation & Compatibility Test Suite (CTS)
- J2EE brand
- J2EE documentations & Sample codes

Evolution of Enterprise Application Frameworks

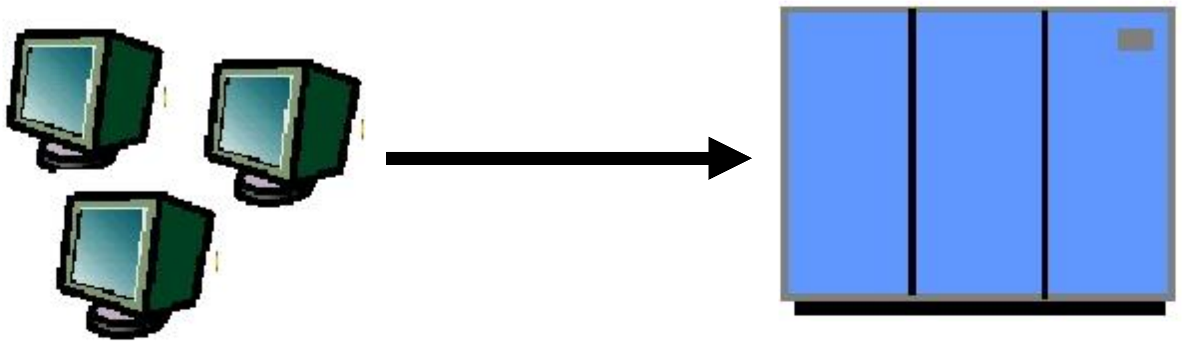
Evolution of Enterprise Application Framework

- Single tier
- Two tier
- Three tier
  - RPC based
  - Remote object based
- Three tier (HTML browser and Web server)
- “Component and container" model

About Enterprise Applications

- Things that make up an enterprise application
  - Presentation logic
  - Business logic
  - Data access logic (and data model)
  - System services
- The evolution of enterprise application framework reflects
  - How flexibly you want to make changes
  - Where the system services are coming from

Single Tier (Mainframe-based)

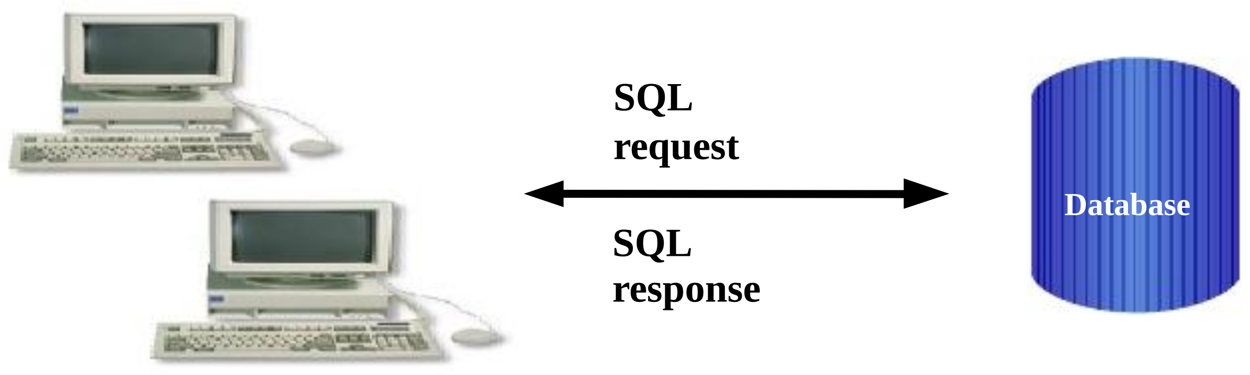


- Dumb terminals are directly connected to mainframe
- Centralized model (as opposed distributed model)
- Presentation, business logic, and data access are intertwined in one monolithic mainframe application

Single-Tier: Pros & Cons

- Pros:
  - No client side management is required
  - Data consistency is easy to achieve
- Cons:
  - Functionality (presentation, data model, business logic) intertwined, difficult for updates and maintenance and code reuse

Two-Tier

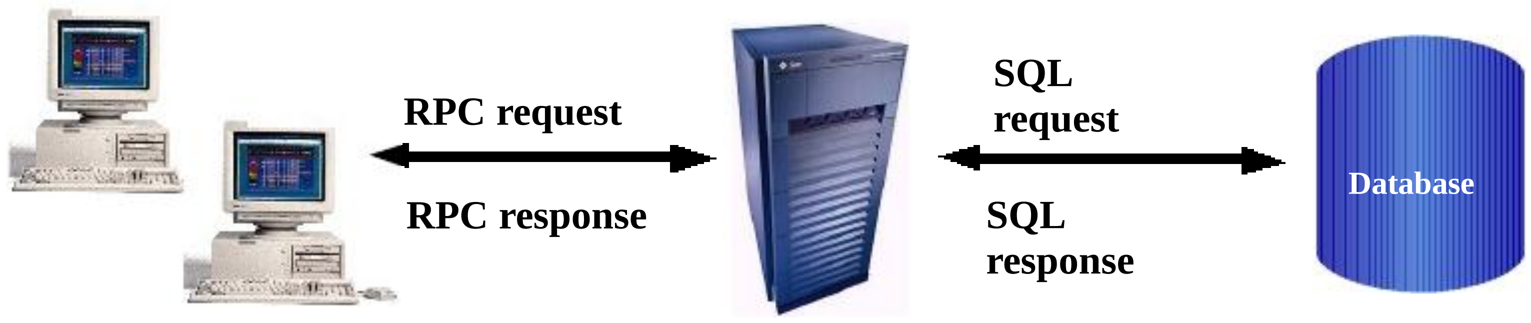


- Fat clients talking to back end database
  - SQL queries sent, raw data returned
- Presentation,Business logic and Data Model processing logic in client application

Two-Tier

- Pro:
  - DB product independence (compared to single-tier model)
- Cons:
  - Presentation, data model, business logic are intertwined (at client side), difficult for updates and maintenance
  - Data Model is “tightly coupled” to every client: If DB Schema changes, all clients need to be updated.
  - Updates have to be deployed to all clients making System maintenance nightmare
  - DB connection for every client, thus difficult to scale
  - Raw data transferred to client for processing causes high network traffic

Three-Tier (RPC based)

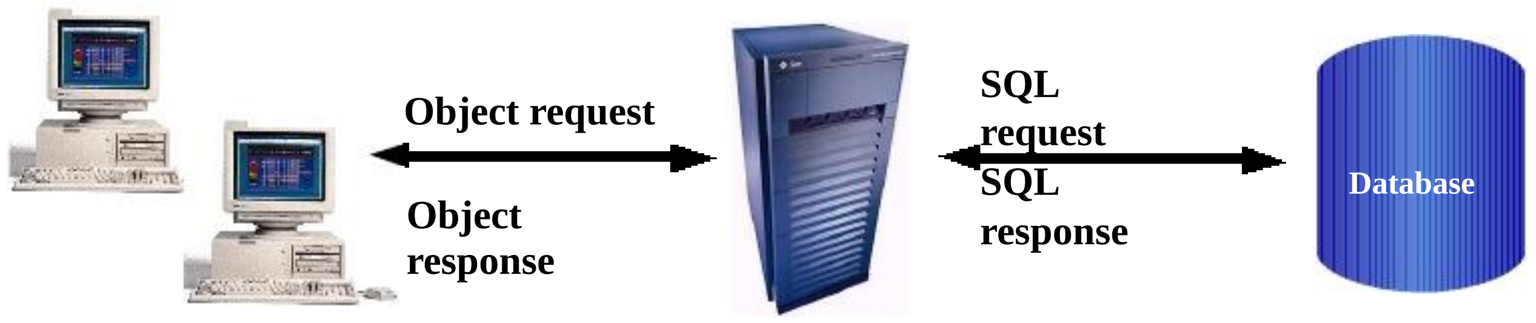


- Thinner client: business & data model separated from presentation
  - Business logic and data access logic reside in middle tier server while client handles presentation
- Middle tier server is now required to handle system services
  - Concurrency control, threading, security, persistence, performance, etc.

Three-tier (RPC based): Pros & Cons

- Pro:
  - Business logic can change more flexibly than 2-tier model
    - Most business logic reside in the middle-tier server
- Cons:
  - Complexity is introduced in the middle-tier server
  - Client and middle-tier server is more tightly-coupled (than the three-tier object based model)
  - Code is not really reusable (compared to object model based)

Three-Tier (Remote Object based)

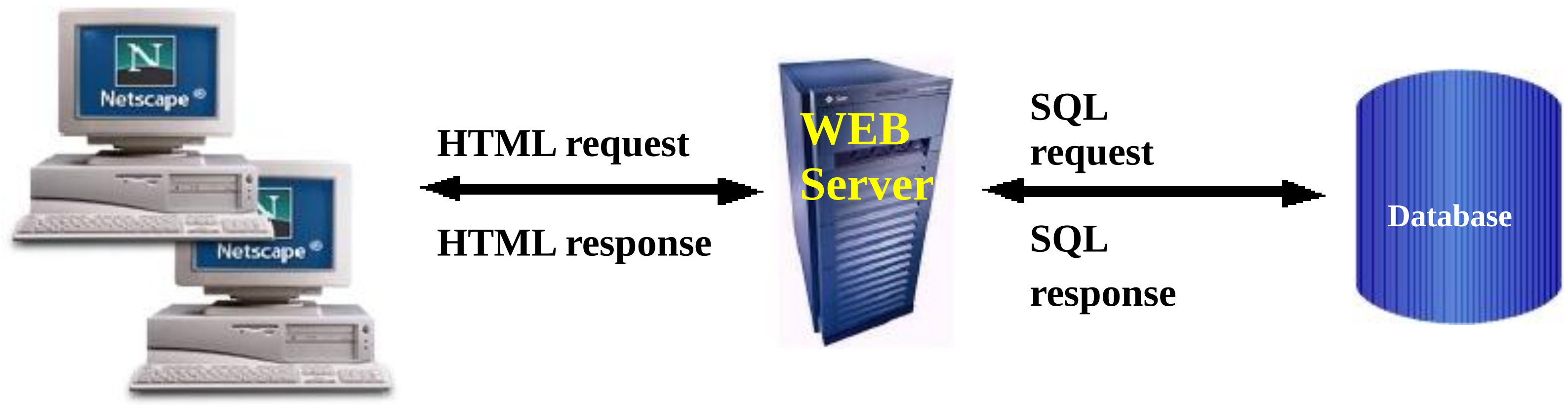


- Business logic and data model captured in objects
  - Business logic and data model are now described in "abstraction" (interface language)
- Object models used: CORBA, RMI, DCOM
  - Interface language in CORBA is IDL
  - Interface language in RMI is Java interface

Three-tier (Remote Object based): Pros & Cons

- Pro:
  - More loosely coupled than RPC model
  - Code could be more reusable
- Cons:
  - Complexity in the middle-tier still need to be addressed

Three-Tier (Web Server)



- Browser handles presentation logic
- Browser talks Web server via HTTP protocol
- Business logic and data model are handled by "dynamic contents generation" technologies (CGI, Servlet/JSP, ASP)

Three-tier (Web Server based): Pros & Cons

- Pro:
  - Ubiquitous client types
  - Zero client management
  - Support various client devices
    - J2ME-enabled cell-phones
- Cons:
  - Complexity in the middle-tier still need to be addressed

Trends

- Moving from single-tier or two-tier to multi-tier architecture
- Moving from monolithic model to object-based application model
- Moving from application-based client to HTML-based client

Single-tier vs. Multi-tier

Single tier

- No separation among presentation, business logic, database
- Hard to maintain

Multi-tier

- Separation among presentation, business logic, database
- More flexible to change, i.e. presentation can change without affecting other tiers

Monolithic vs. Object-based

Monolithic

- 1 Binary file
- Recompiled, relinked, redeployed every time there is a change

Object-based

- Pluggable parts
- Reusable
- Enables better design
- Easier update
- Implementation can be separated from interface
- Only interface is published

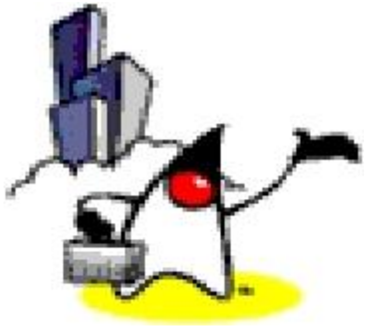
Outstanding Issues & Solution

- Complexity at the middle tier server still remains
- Duplicate system services still need to be provided for the majority of enterprise applications
  - Concurrency control
  - Load-balancing, Security
  - Resource management, Connection pooling
- How to solve this problem?
  - Commonly shared container that handles the above system services
  - Proprietary versus Open-standard based

Component and Container model

- Use
  - Components captures business logic
  - Container provides system services
- The contract between components and container is defined in a well-defined manner
- J2EE is that standard that also provides portability of code because it is based on Java technology and standard-based Java programming APIs

Why J2EE?



Platform Value to Developers

- Can use any J2EE implementation for development and deployment
  - Use production-quality standard implementation which is free for development/deployment
  - Use high-end commercial J2EE products for scalability and fault-tolerance
- Vast amount of J2EE community resources
  - Many J2EE related books, articles, tutorials, quality code you can use, best practice guidelines, design patterns etc.
- Can use off-the-shelf 3rd-party business components

Platform Value to Vendors

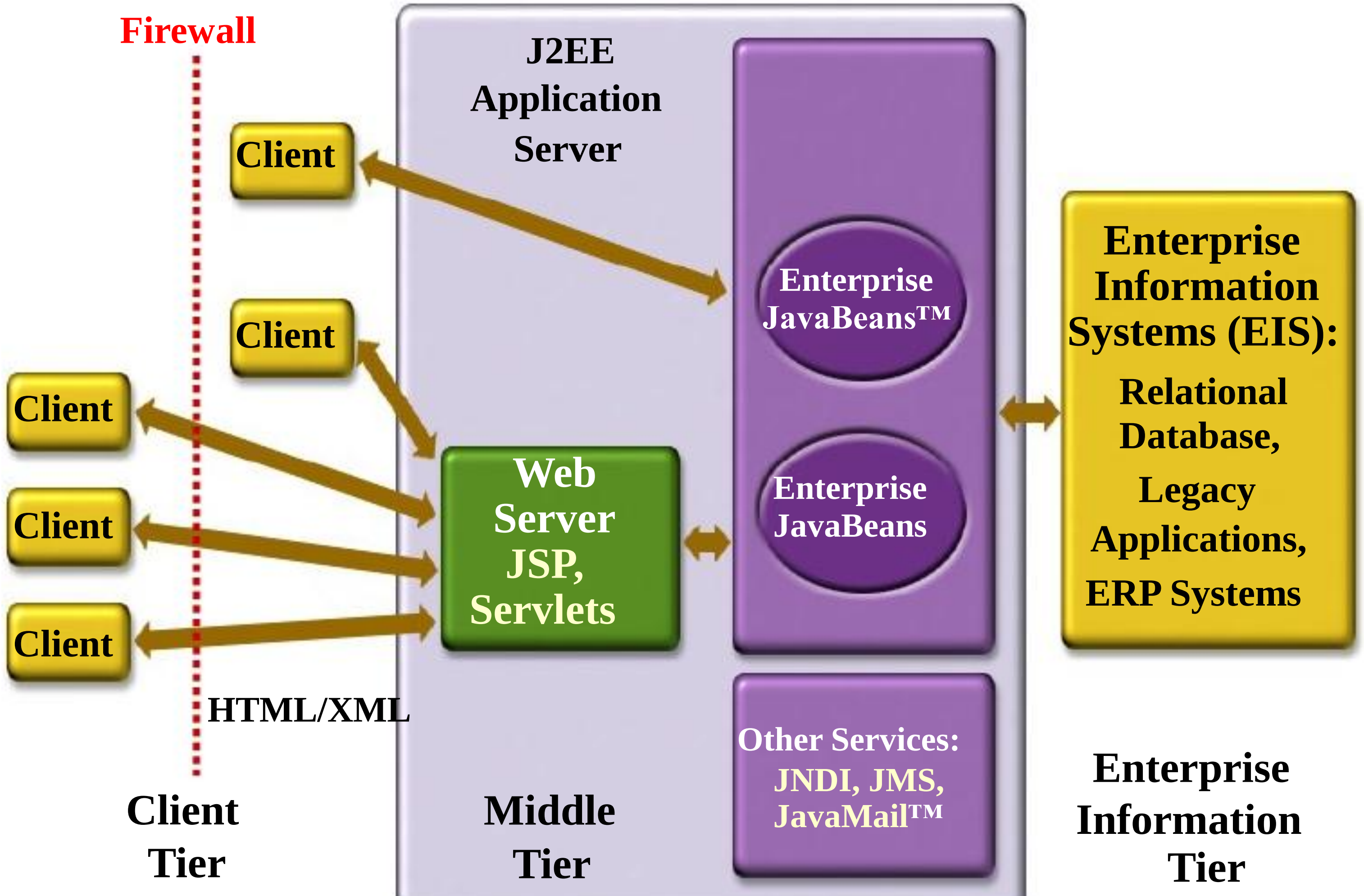
- Vendors work together on specifications and then compete in implementations
  - In the areas of Scalability, Performance, Reliability, Availability, Management and development tools, and so on
- Freedom to innovate while maintaining the portability of applications
- Do not need create/maintain their own proprietary APIs

Platform Value to Business Customers

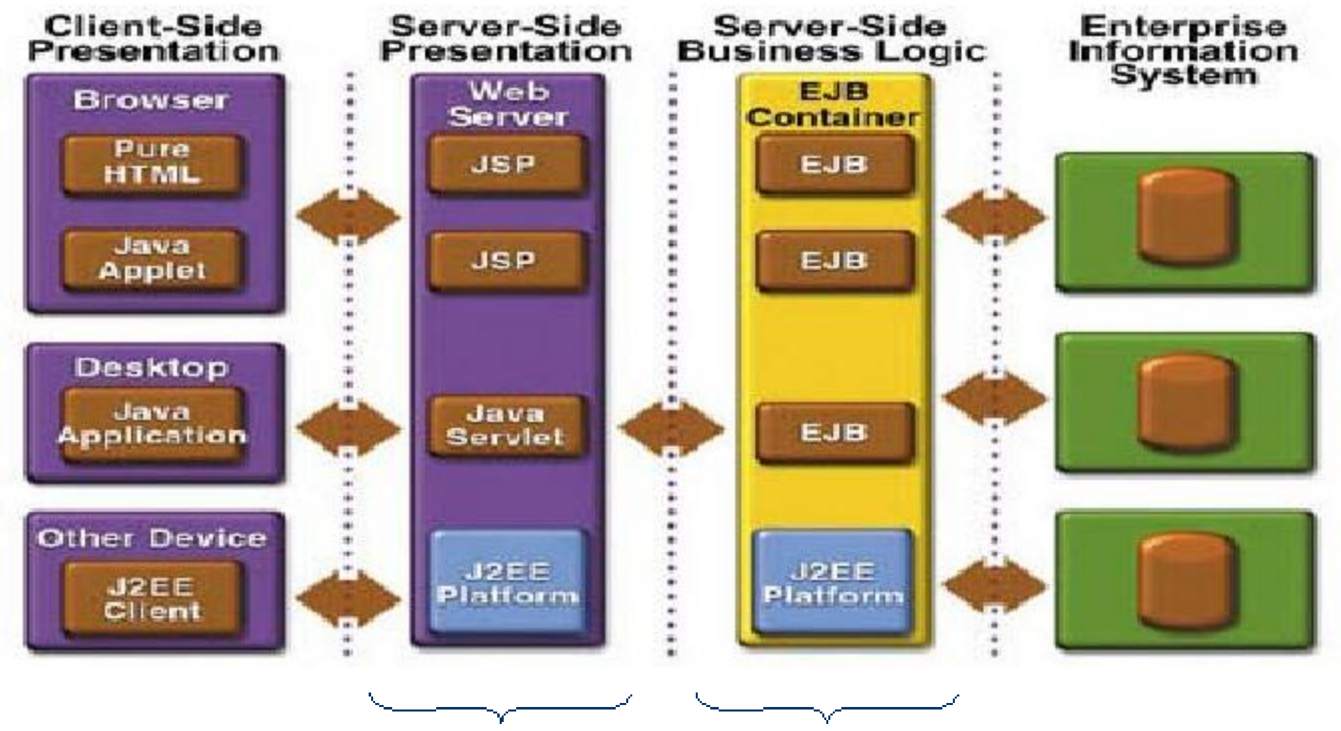
- Application portability
- Many implementation choices are possible based on various requirements
  - Price (free to high-end), scalability (single CPU to clustered model), reliability, performance, tools, and more
  - applications and platforms
- Large developer pool

J2EE is an End-to-End Architecture

J2EE is End-to-End Solution

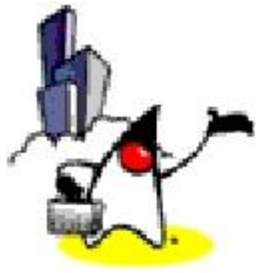


N-tier J2EE Architecture

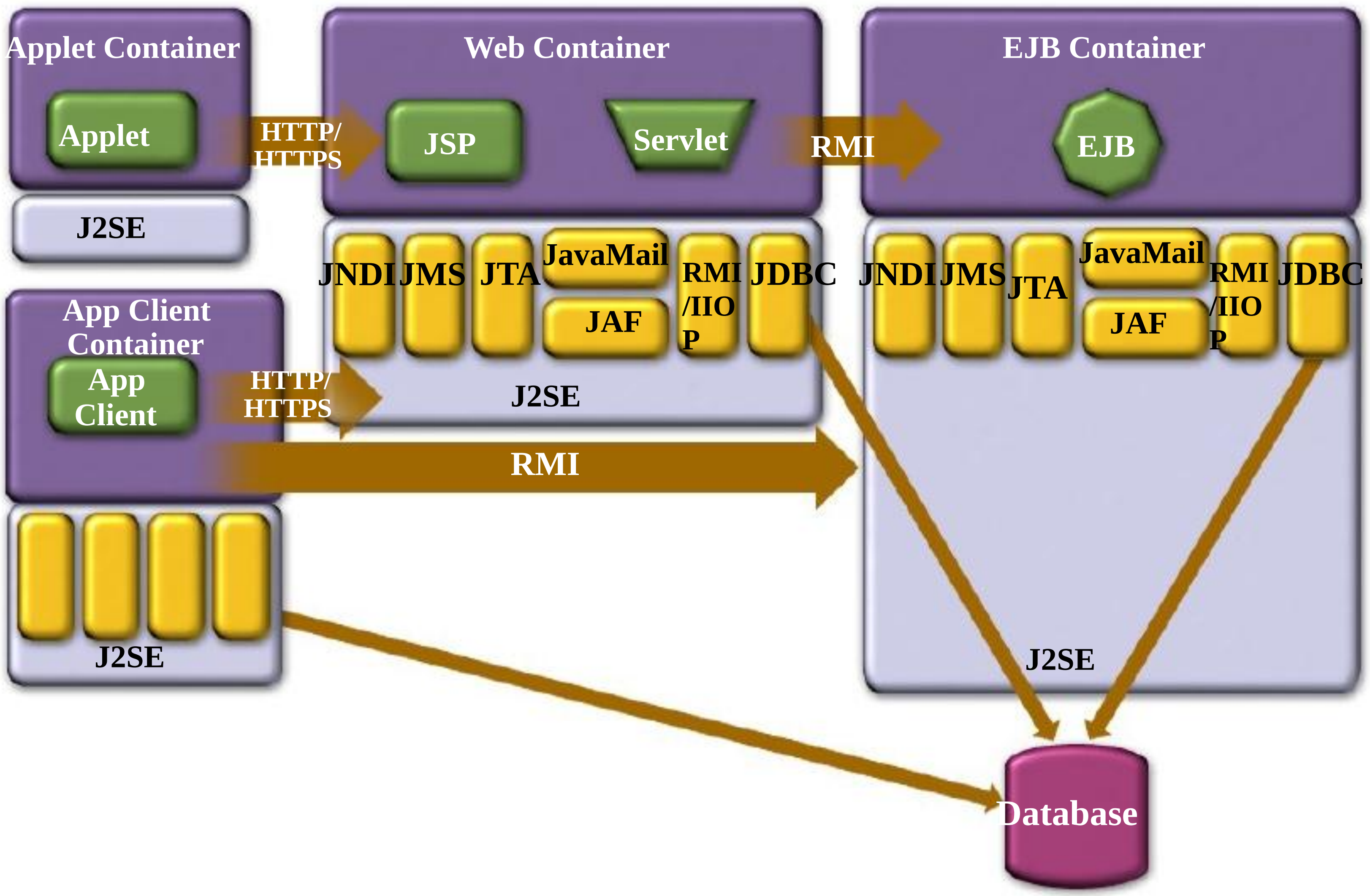


Web Tier EJB Tier

J2EE
Component & Container Architecture



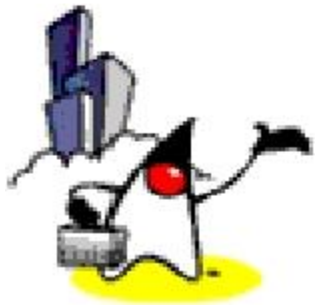
J2EE Containers & Components



Containers and Components

Containers Handle	Components Handle
• Concurrency • Security • Availability • Scalability • Persistence • Transaction • Life-cycle management • Management	• Presentation • Business Logic

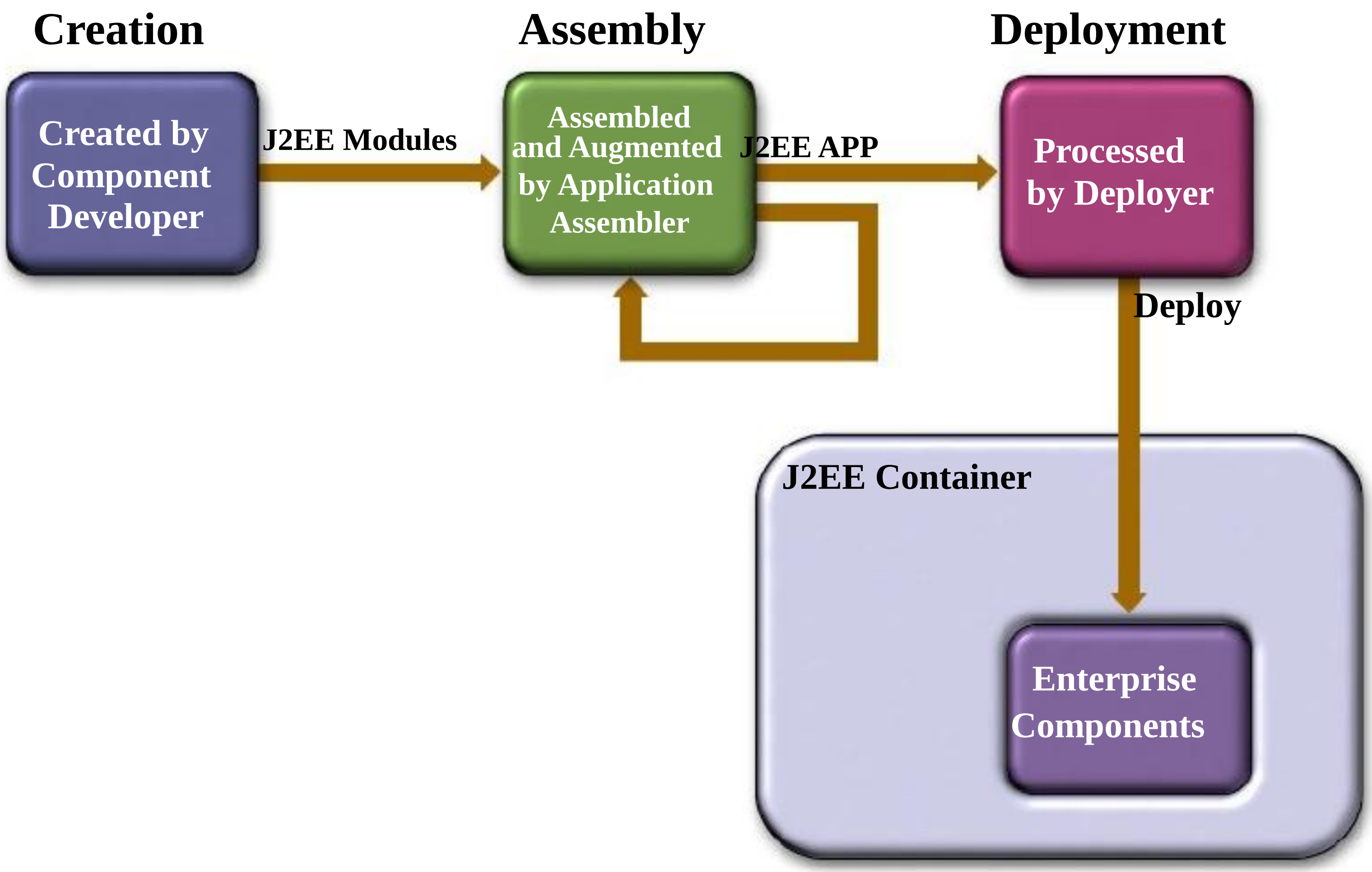
J2EE Application Development & Deployment Life Cycle



J2EE Application Development Lifecycle

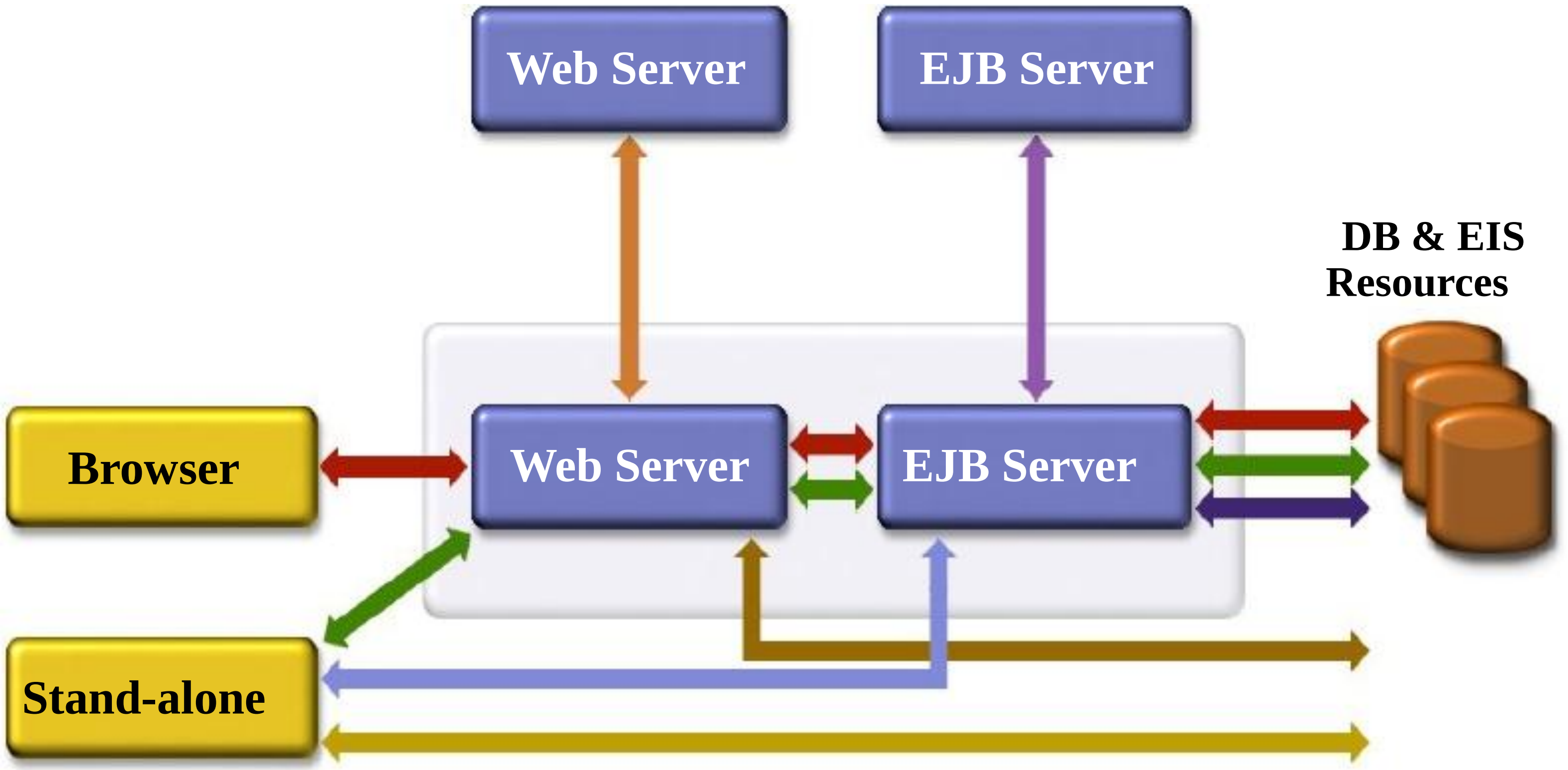
- Write and compile component code
  - Servlet, JSP, EJB
- Write deployment descriptors for components
  - From Java EE 5, you can use annotations
- Assemble components into ready-to-deployable package
- Deploy the package on a server

Life-cycle Illustration



J2EE Application Anatomies

Possible J2EE Application Anatomies



J2EE Application Anatomies

- 4-tier J2EE applications
  - HTML client, JSP/Servlets, EJB, JDBC/Connector
- 3-tier J2EE applications
  - HTML client, JSP/Servlets, JDBC
- 3-tier J2EE applications
  - EJB standalone applications, EJB, JDBC/Connector

Which One to Use?

- Depends on several factors
  - Requirements of applications
  - Availability of EJB tier
  - Availability of developer resource

J2EE 1.4
Standard Implementation, Compatibility Suite, Brand

Standard Implementation

- Under JavaEE, it is Sun GlassFish Enterprise Server.
- Free to develop and free to deploy

Compatibility Test Suite (CTS)

- Ultimate Java™ technology mission:
  - Write Once, Run Anywhere™
  - My Java-based application runs on any compatible Java virtual machines
  - My J2EE technology-based application will run on any J2EE based Compatible platforms

J2EE Application Verification Kit (J2EE AVK)

- How can I test my J2EE application portability?
  - Obtain the J2EE RI 1.3.1 and the J2EE Application Verification Kit (J2EE AVK)
- Self verification of application
  - Static verification
  - Dynamic verification
- Obtain the tests results, verify that all criteria are met

Major Investment in Compatibility by the Industry

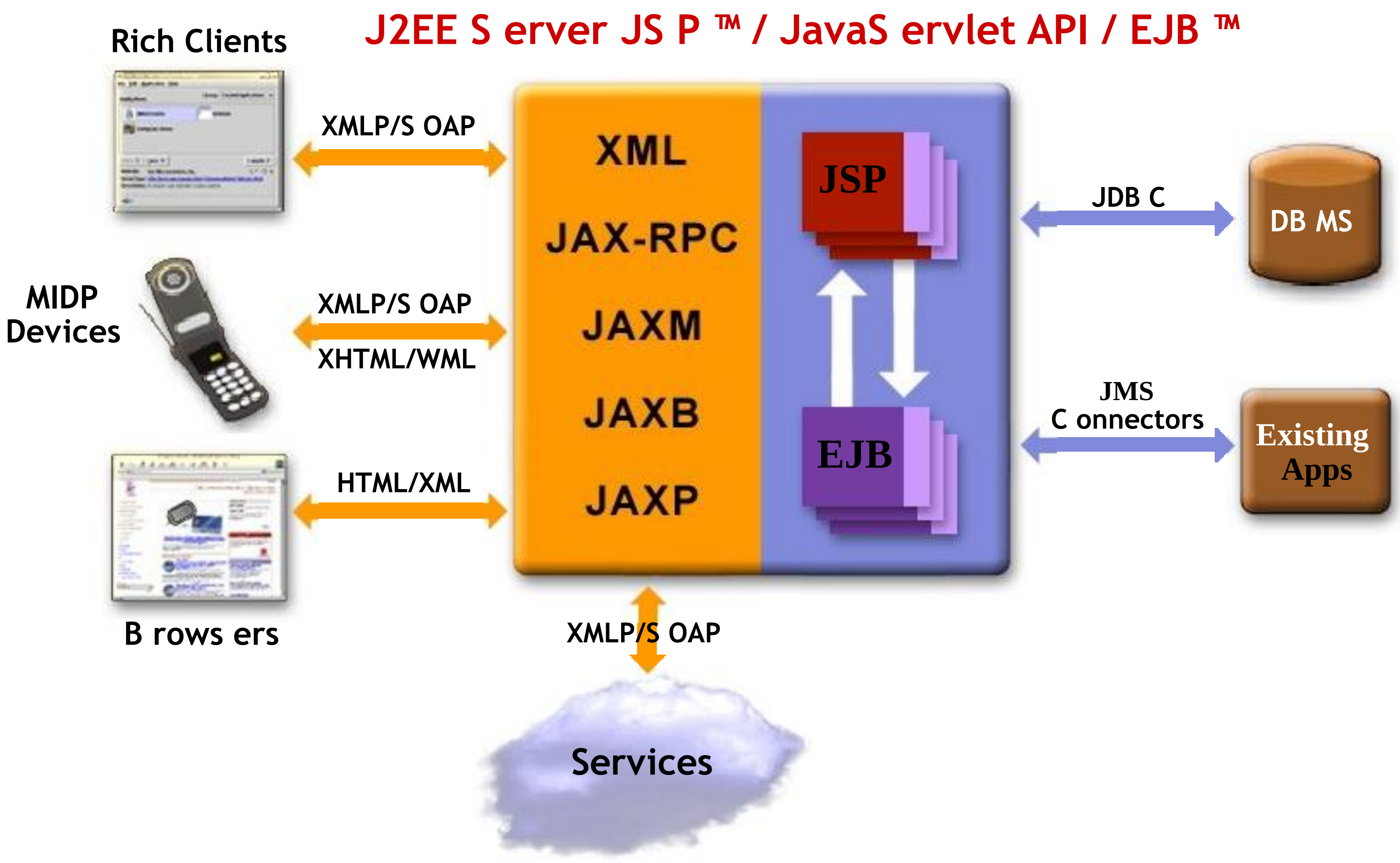
- Sun has spent scores of engineer years developing tests
- Licensees have spent scores of engineer years passing the tests
- Testing investment on top of specification investment, implementation investment, business investments
- In total, tens of millions of dollars invested in J2EE platform compatibility by the industry

Why J2EE for Web Services?

Why J2EE for Web Services?

- Web services is just one of many service delivery channels of J2EE
- Many benefits of J2EE are preserved for Web services
  - Portability, Scalability, Reliability
  - No single-vendor lock-in

Web Services Model Over J2EE



Design Goals J2EE 1.4 Web Services Framework

- Portability of Web services component
  - Over different vendor platform
  - Over different operational environment
- Leveraging existing J2EE programming models for service implementation
- Easy to program and deploy
  - High-level Java APIs
  - Use existing deployment model

Environment Configuration

- J2EE IDE
  - NetBeans IDE
  - Eclipse+MyEclipse
- Framework
  - Struts
  - Hibernate
- DB
  - Derby
  - mySQL
- Web server
  - Glassfish
  - Tomcat

Resources

- Java EE overview
  http://www.oracle.com/technetwork/java/javaee/overview/index.html
- Java EE download
  http://www.oracle.com/technetwork/java/javaee/downloads/index.html
- Java EE Tutorial

  http://download.oracle.com/javaee/6/tutorial/doc
  http://www.roseindia.net/

Resources

- Java SE API

  http://download.oracle.com/javase/1.5.0/docs/api/

- Java EE API

  http://download.oracle.com/javaee/6/api/

- NetBeans IDE

  http://www.netbeans.org

Summary

- J2EE is the platform of choice for development and deployment of n-tier, web-based, component-
  based enterprise applications
- J2EE is standard-based architecture
- J2EE is all about community
- J2EE evolves according to the needs of the industry

The End