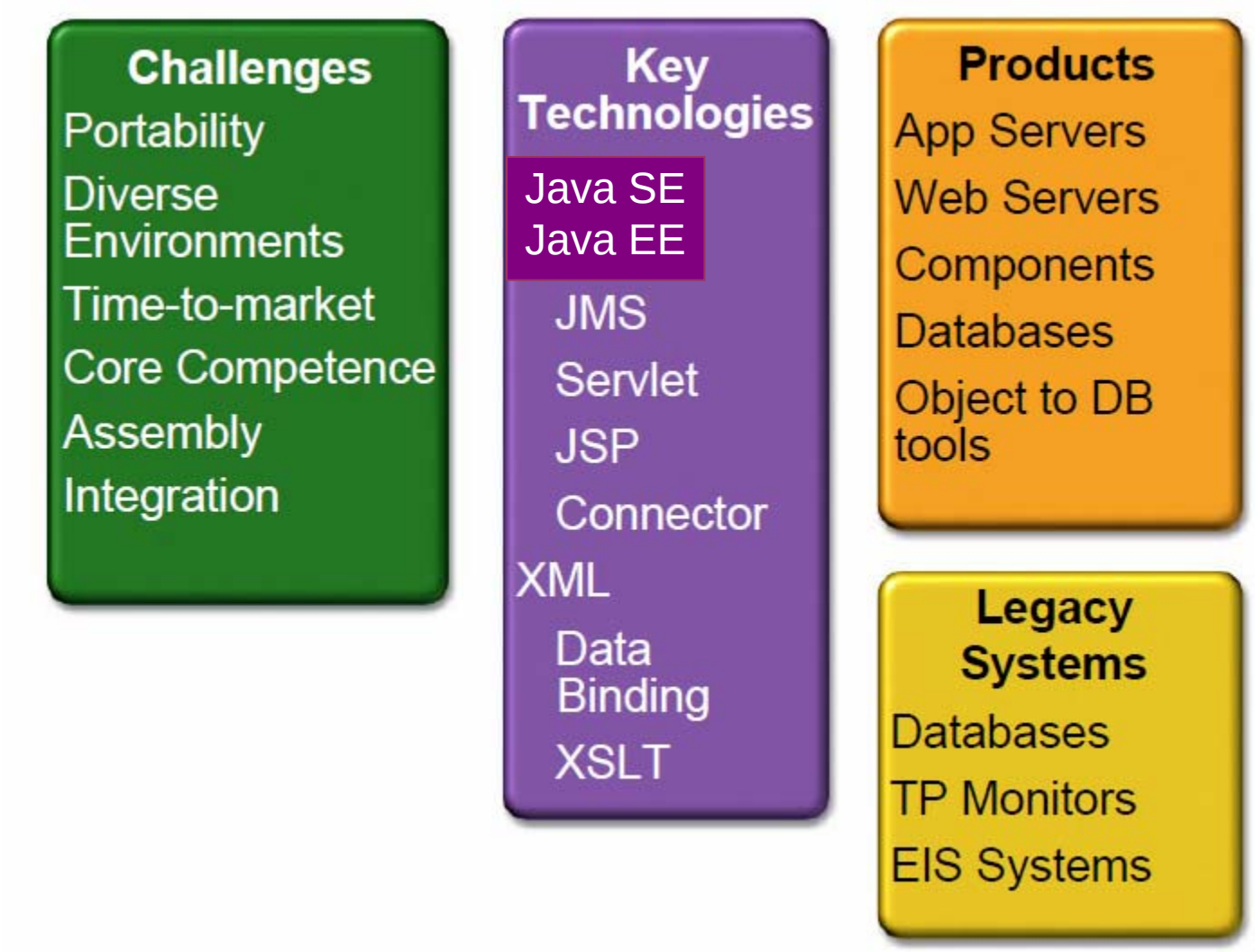
Introduction to Java EE

Agenda

- What is Java EE?
- Evolution of Enterprise Application Development Frameworks
- Why Java EE?
- Java EE Platform Architecture
- Java EE APIs and Technologies
- BluePrints

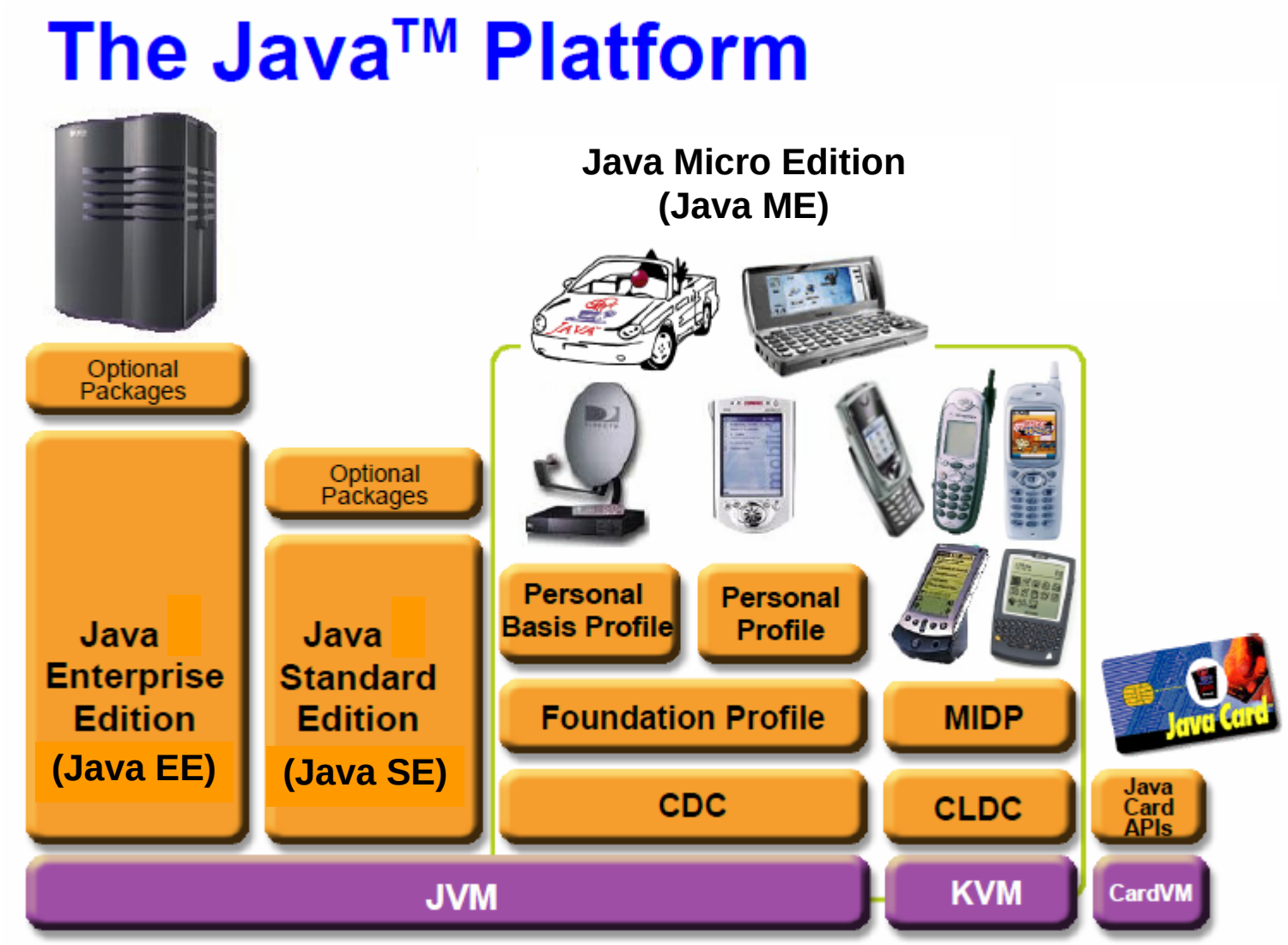
Enterprise Computing



What is the Java EE

- Java EE : Java Enterprise Edition
- Open and standard based platform for developing, deploying and managing
- n-tier, Web-enabled, server-centric, and component-based enterprise applications

Java SE, Java EE, Java ME

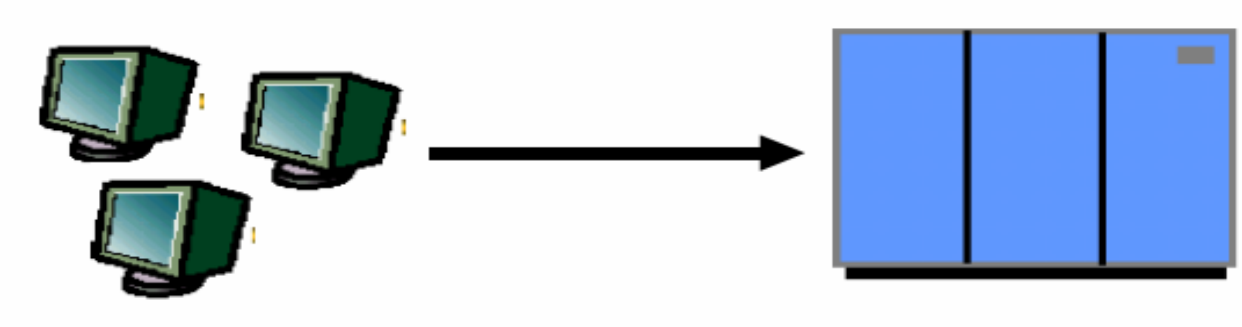


Evolution of Enterprise Application Framework

- Single tier
- Two tier
- Three tier
  - RPC based (Remote Procedure Call)
  - Remote object based
- Three tier (HTML browser and Web server)
- Proprietary and standard application server

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

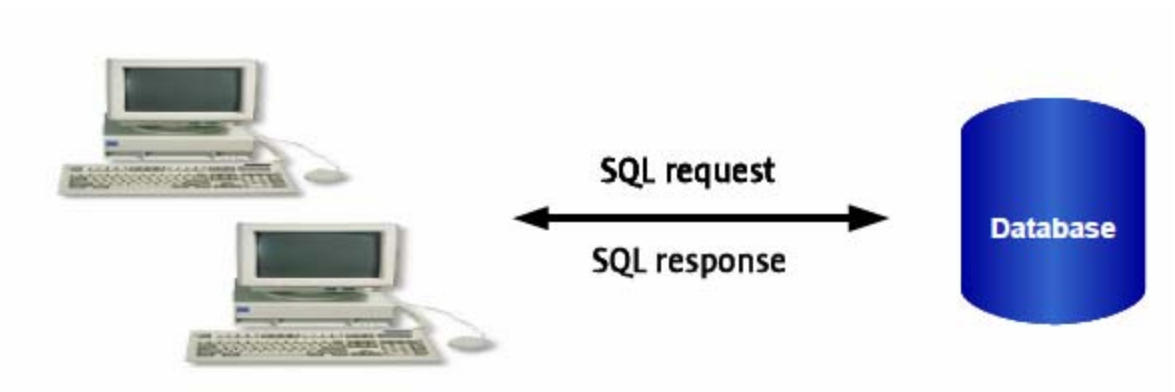


Single-Tier: Pros & Cons

- 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 : Pros & Cons

- Pros:
  - 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 break
  - Updates have to be deployed to all clients making system maintenance nightmare
  - DB connection for every client, thus difficult to scale

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, transaction, security, persistence, multiplexing, performance, etc.

Three-tier (RPC based): Pros & Cons

- Pros:
  - 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,DCOM, RMI
  - Interface language in CORBA is IDL
  - Interface language in RMI is Java interface

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

- Pros:
  - 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 (Servlet/JSP, ASP, PHP)

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

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

แนวโน้ม

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

Monolithic & 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, Transactions
  - 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

Proprietary Solution

- Use “component and container” model
  - Components captures business logic
  - Container provides system services
- The contract between components and container is defined in a well-defined but with proprietary manner
- Problem of proprietary solution: Vendor lock-in
  - Example: Tuxedo, .NET, IIS

Open and Standard Solution

- Use “component and container” model in which container provides system services in a well-defined and as industry standard
- Java EE is that standard that also provides portability of code because it is based on Java technology and standard-based Java programming APIs

Why Java EE?

- **Platform Value to Developers**
- **Platform Value to Vendors**
- **Platform Value to Business Customers**

Platform Value to Developers

- Can use any Java EE implementation for development and deployment
  - Use production-quality standard implementation which is free for development/deployment
  - Use high-end commercial Java EE products for scalability and fault-tolerance
- Vast amount of Java EE community resources
  - Many Java EE 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 have 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
  - Best of breed of applications and platforms
- Large developer pool

Java EE5 Technologies

--http://java.sun.com/javaee/technologies/

Java EE 6 Technologies

At a Glance | Java EE 5 | Web Services | Web App | Enterprise App | Management

Learn more about the technologies that comprise the Java EE 6 platform using the specifications, and then apply them with the Java EE 6 SDK.

Specification downloads are the final releases. Please check the individual JSR pages for download updates such as maintenance releases.

Java EE 6 Technologies Technologies	JSR	Download
Java Platform, Enterprise Edition 6 (Java EE 6) (includes Managed Beans 1.0)	JSR 316	Download spec
Web Services Technologies » Read more		
Java API for RESTful Web Services (JAX-RS) 1.1	JSR 311	Download spec
Implementing Enterprise Web Services 1.3	JSR 109	Download spec
Java API for XML-Based Web Services (JAX-WS) 2.2	JSR 224	Download spec
Java Architecture for XML Binding (JAXB) 2.2	JSR 222	Download spec
Web Services Metadata for the Java Platform	JSR 181	Download spec
Java API for XML-Based RPC (JAX-RPC) 1.1	JSR 101	Download spec
Java APIs for XML Messaging 1.3	JSR 67	Download spec

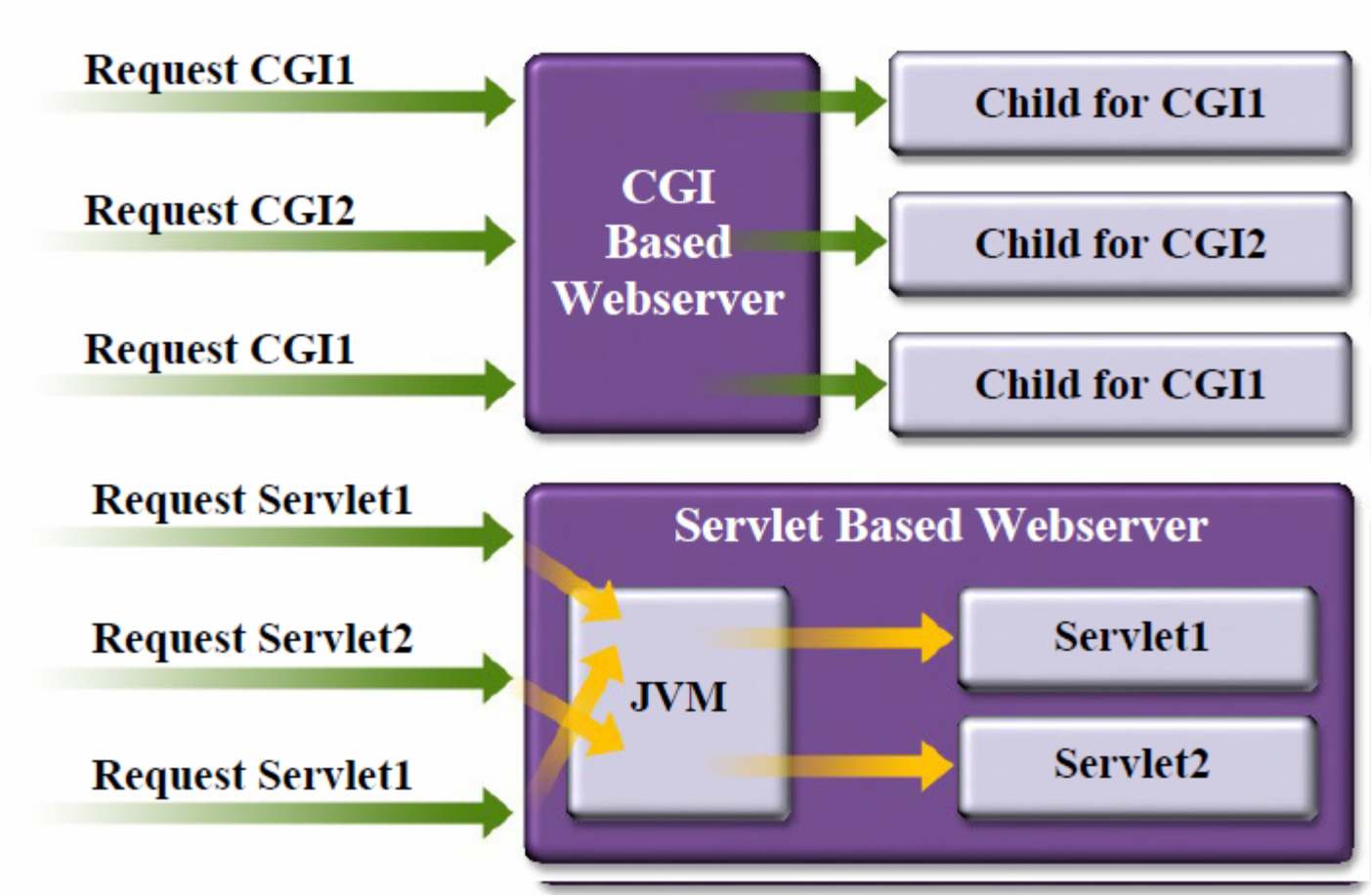
Web Application Technologies » Read more		
Java Servlet 3.0	JSR 315	Download spec
JavaServer Faces 2.0	JSR 314	Download spec
JavaServer Pages 2.2/Expression Language 2.2	JSR 245	Download spec
Standard Tag Library for JavaServer Pages (JSTL) 1.2	JSR 52	Download spec
Debugging Support for Other Languages 1.0	JSR 45	Download spec
Enterprise Application Technologies » Read more		
Contexts and Dependency Injection for Java (Web Beans 1.0)	JSR 299	Download spec
Dependency Injection for Java 1.0		
Bean Validation 1.0		
Enterprise JavaBeans 3.1 (includes Interceptors 1.1)		

Enterprise Application Technologies » Read more		
Contexts and Dependency Injection for Java (Web Beans 1.0)	JSR 299	Download spec
Dependency Injection for Java 1.0	JSR 330	Download spec
Bean Validation 1.0	JSR 303	Download spec
Enterprise JavaBeans 3.1 (includes Interceptors 1.1)	JSR 318	Download spec
Java EE Connector Architecture 1.6	JSR 322	Download spec
Java Persistence 2.0	JSR 317	Download spec
Common Annotations for the Java Platform 1.1	JSR 250	Download spec
Java Message Service API 1.1	JSR 914	Download spec
Java Transaction API (JTA) 1.1	JSR 907	Download spec
JavaMail 1.4	JSR 919	Download spec
Management and Security Technologies » Read more		
Java Authentication Service Provider Interface for Containers	JSR 196	Download spec
Java Authorization Contract for Containers 1.3	JSR 115	Download spec
Java EE Application Deployment 1.2	JSR 88	Download spec

What is a Servlet?

- Java™ objects which extend the functionality of a HTTP server
- Dynamic contents generation
- Better alternative to CGI, NSAPI, ISAPI, etc.
  - Efficient
  - Platform and server independent
  - Session management
  - Java-based

Servlet vs. CGI



What is JSP Technology?

- Enables separation of business logic from presentation
  - Presentation is in the form of HTML or XML/XSLT
  - Business logic is implemented as Java Beans or custom tags
  - Better maintainability, reusability
- Extensible via custom tags
- Builds on Servlet technology

JDBC

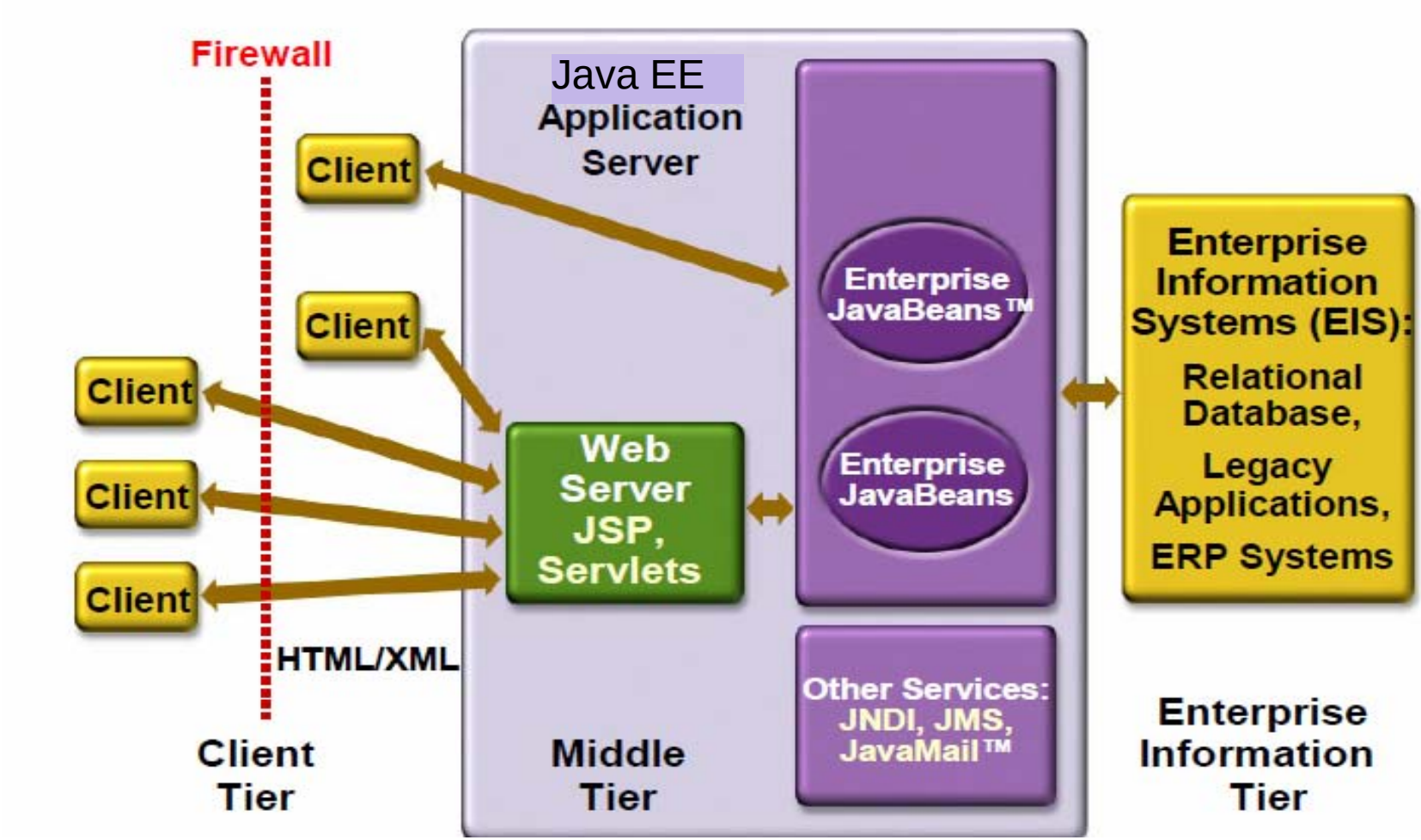
- Provides standard Java programming

API to relational database

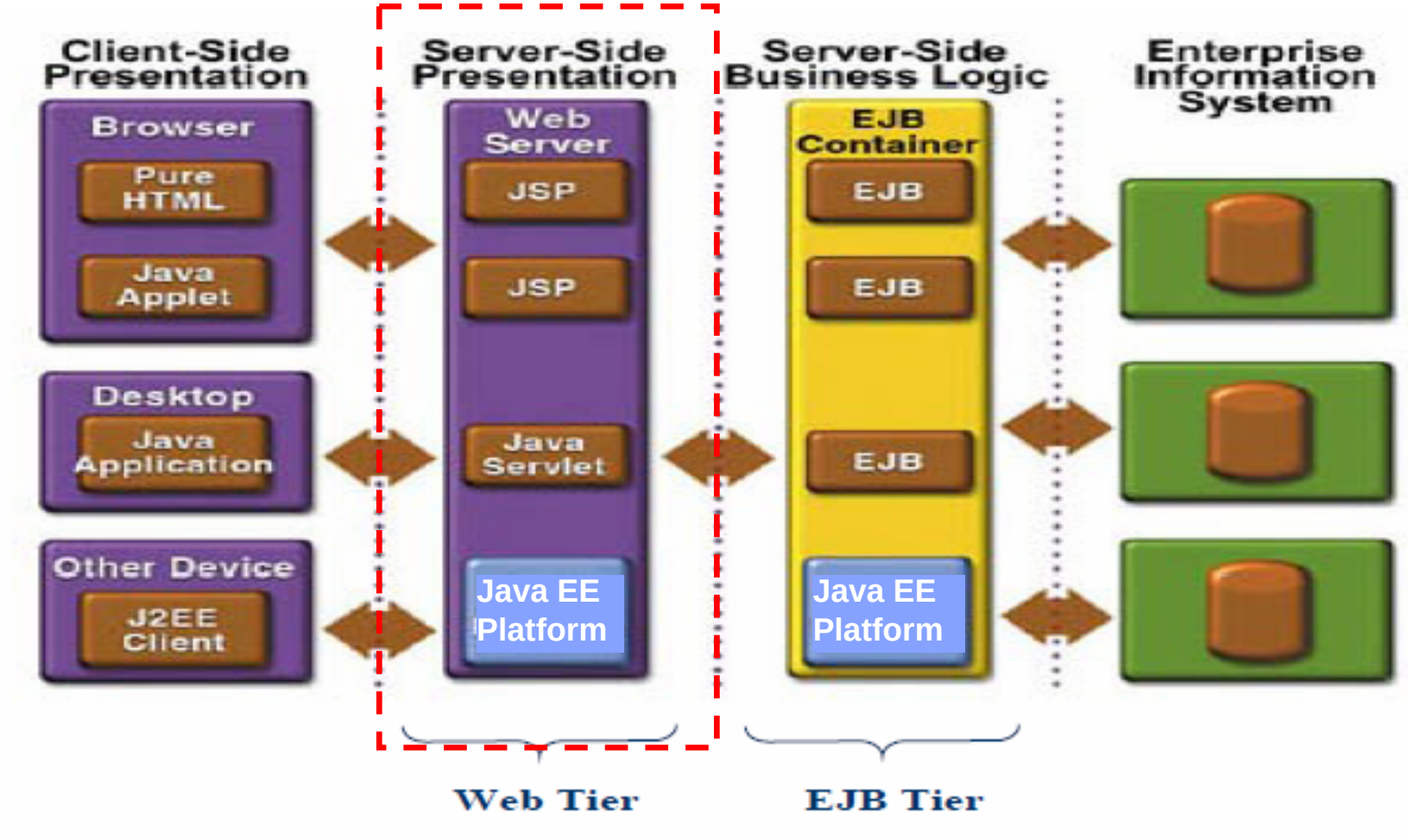
  - Uses SQL

- Vendors provide JDBC compliant driver which can be invoked via standard Java programming API

Java EE is End-to-End solution

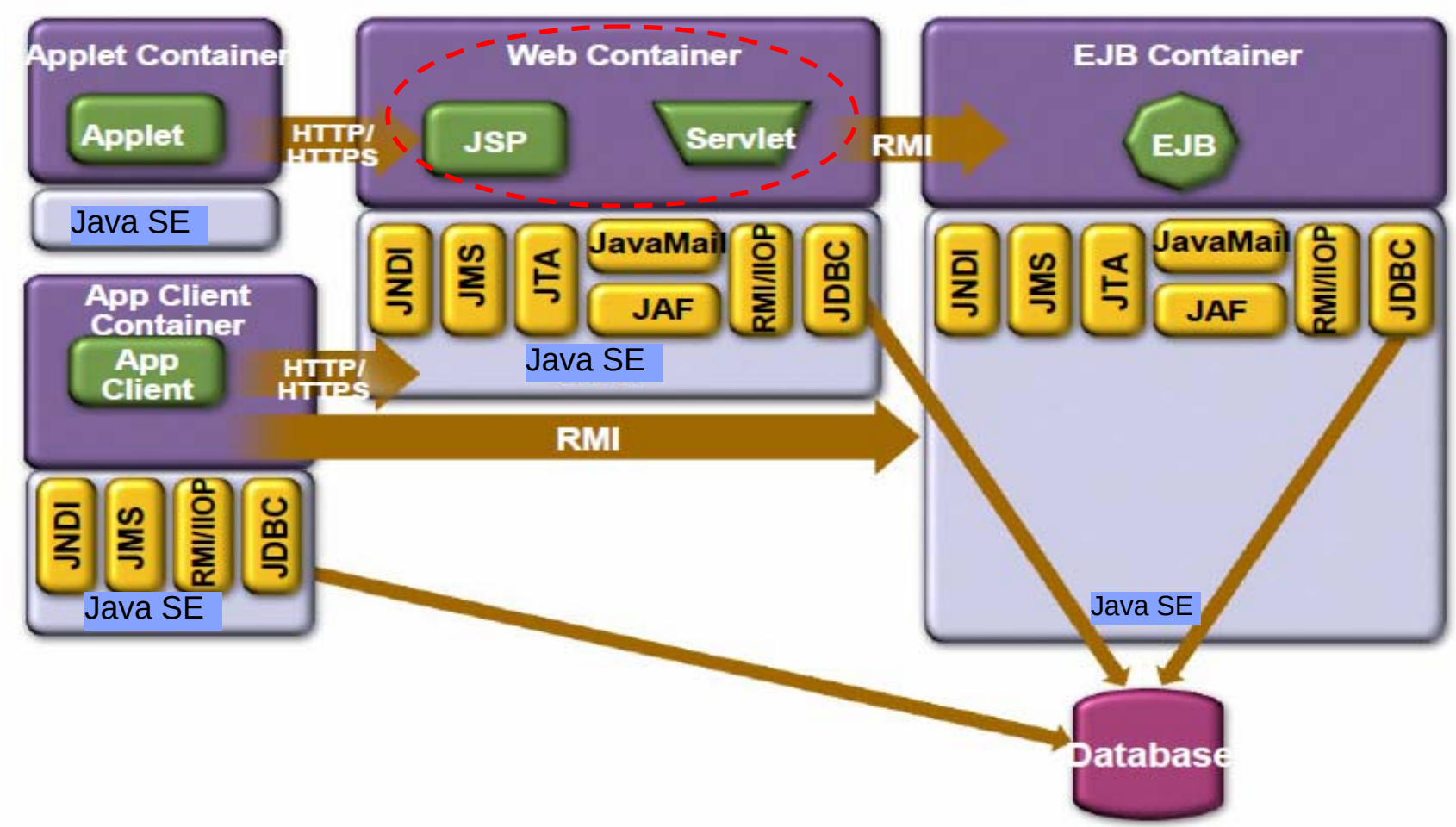


N-tier Java EE Architecture

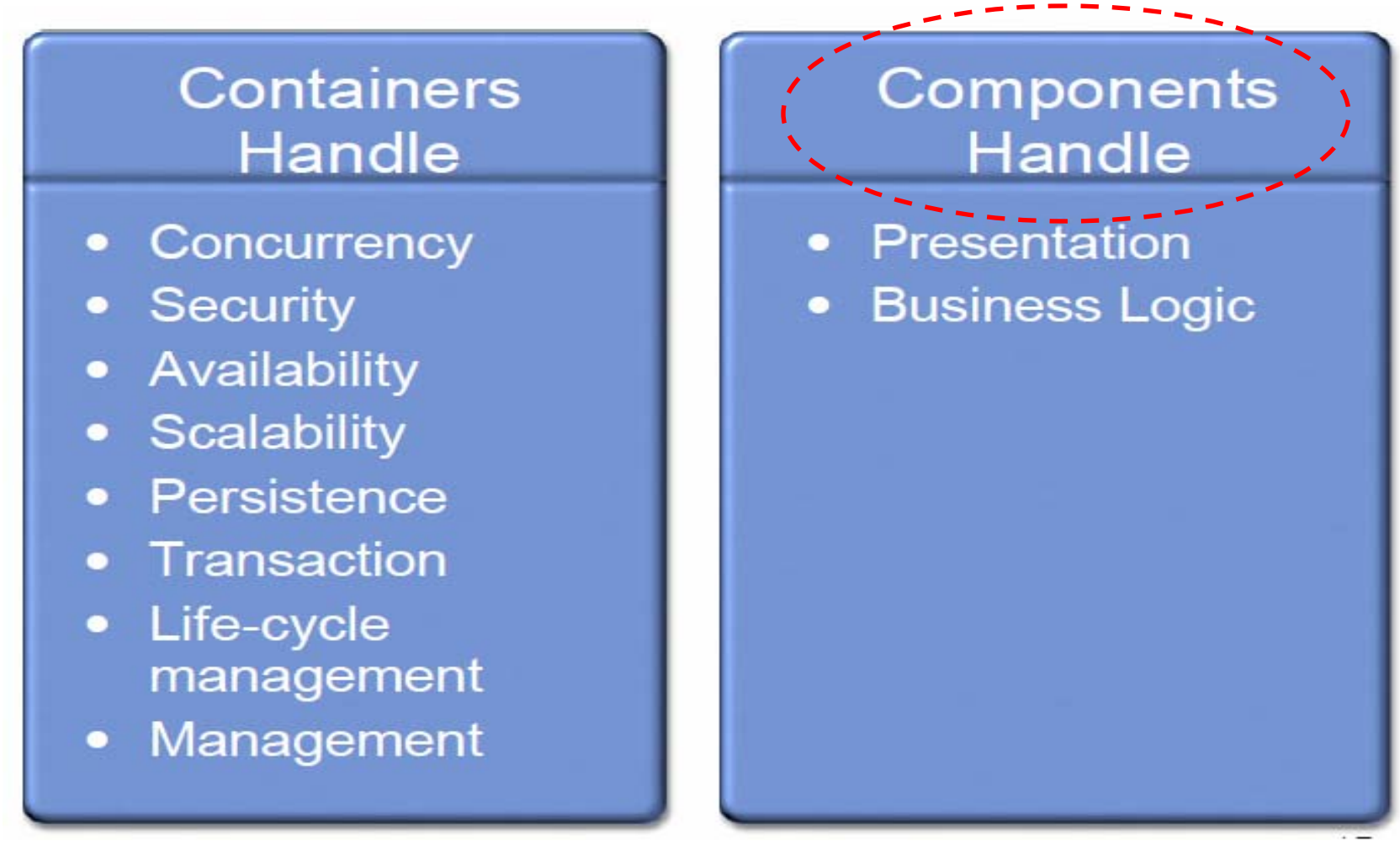


Java EE Component & Container Architecture

Web Components & Container



Containers/Components Handle



Web Components & Container

- Web components are in the form of either Servlet or JSP (along with JavaBean's and custom tags)
- Web components run in a Web container
  - Tomcat is a popular web containers
  - All Java EE compliant app servers (GlassFish App Server) provide web containers
- Web container provides system services to Web components
  - Request dispatching, security, and life cycle management

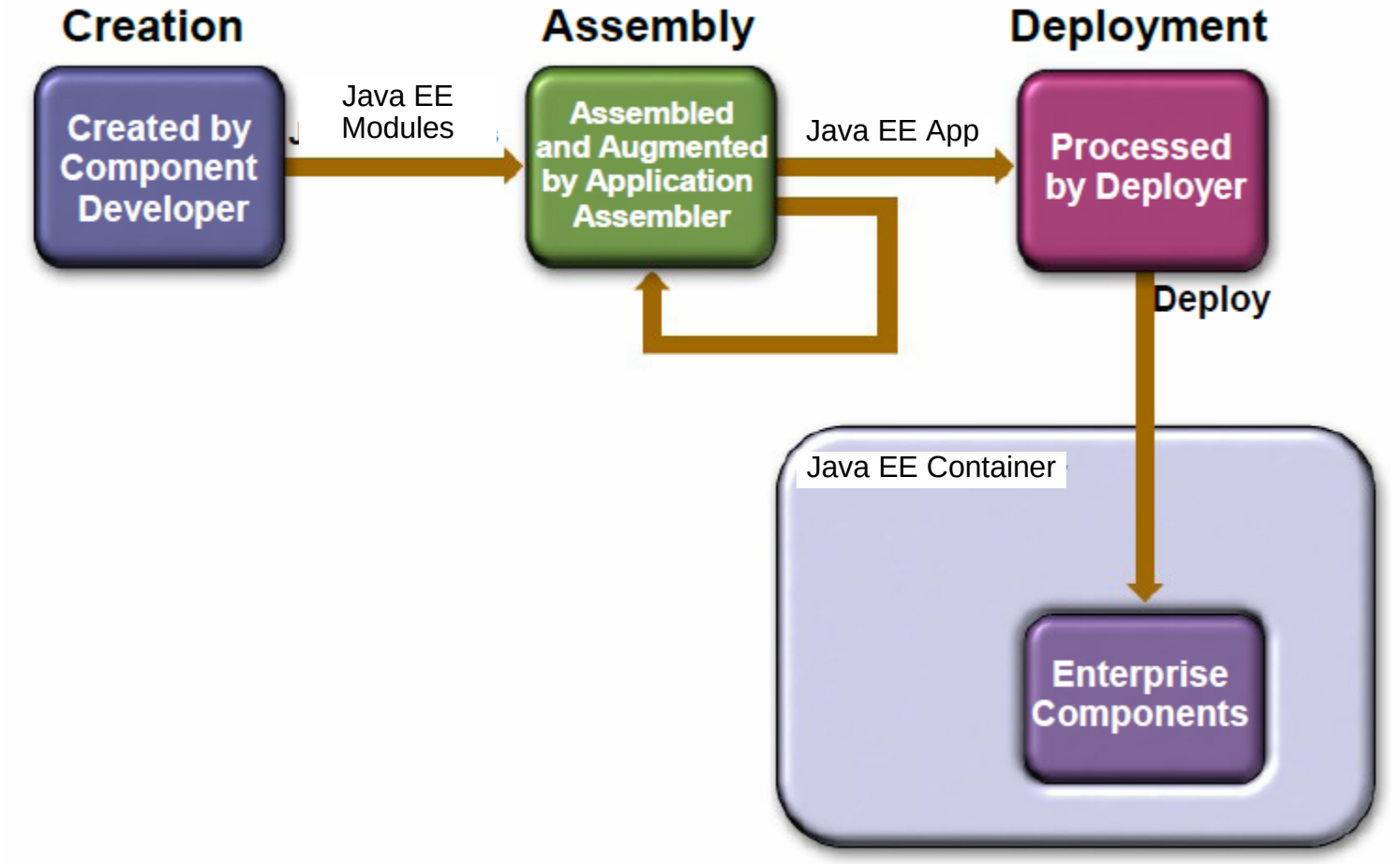
Java EE Application Development Lifecycle

- Write and compile component code
  - Servlet, JSP, EJB
- Write deployment descriptors for components
- Assemble components into ready to deployable package
- Deploy the package on a server

The Deployment Descriptor (web.xml)

- Gives the container instructions on how to manage and control behaviors of the Java EE components
  - Transaction
  - Security
  - Persistence
- Allows declarative customization (as opposed to programming customization)
  - XML file
- Enables portability of code

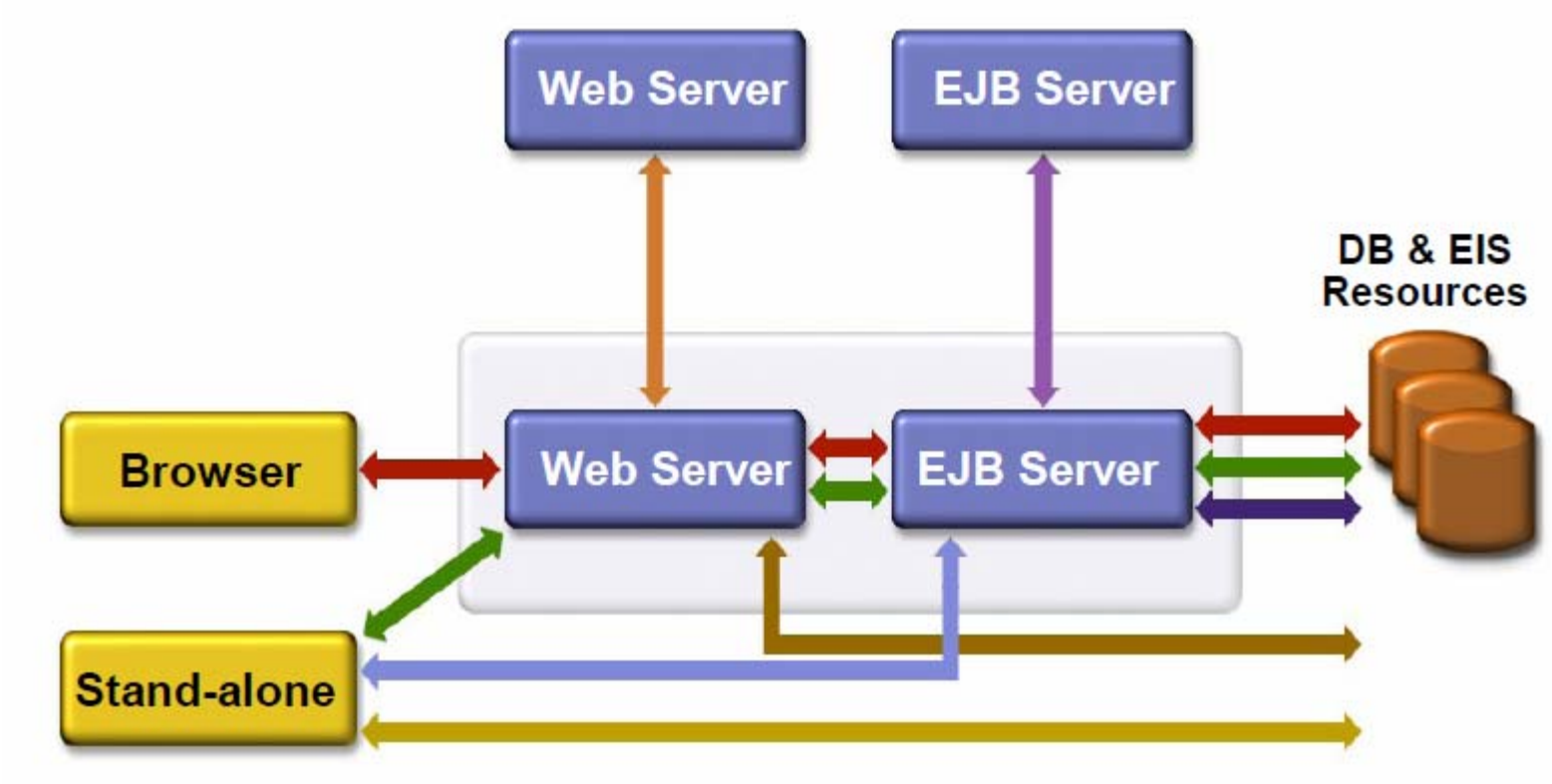
Java EE Application Development Life-cycle



Java EE Development Roles

- Component provider
  - Bean provider
- Application assembler
- Deployer
- Platform provider
  - Container provider
- Tools provider
- System administrator

Java EE Application Anatomies



Java EE Application Anatomies

- 4-tier Java EE applications
  - HTML client, JSP/Servlets, EJB, JDBC/Connector
- 3-tier Java EE applications
  - HTML client, JSP/Servlets, JDBC
- 3-tier Java EE applications
  - EJB standalone applications, EJB, JDBC/Connector
- B2B Enterprise applications
  - Java EE platform to Java EE platform through the exchange of JMS or XML-based messages: Web Services

Which One to Use?

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

Compatible Products for the Java EE Platform (Brand)

Java EE Blueprint

- Best practice guidelines, design patterns and design principles
  - MVC pattern
- Covers all tiers
  - Client tier
  - Web tier
  - Business logic (EJB) tier
  - Database access tier
- http://www.oracle.com/technetwork/java/blueprints-141945.html
  - Java Pet Store, Adventure builder

Summary

- Java EE is the platform of choice for development and deployment of n-tier, web-based, transactional, componentbased enterprise applications
- Java EE is standard-based architecture
- Java EE is all about community
- Java EE evolves according to the needs of the industry
- Resources:
  - http://www.oracle.com/technetwork/java/javaee/overview/index.html
  - http://www.netbeans.org

Acknowledgement

- Contents are borrowed from the presentation slides of Sang Shin, Java™ Technology Evangelist, Sun Microsystems, Inc.
- www.javapassion.com