University of Tokyo Java Class September 22-26, 2003 J2EE Overview and Roadmap

Marc Hamilton

Director of Technology

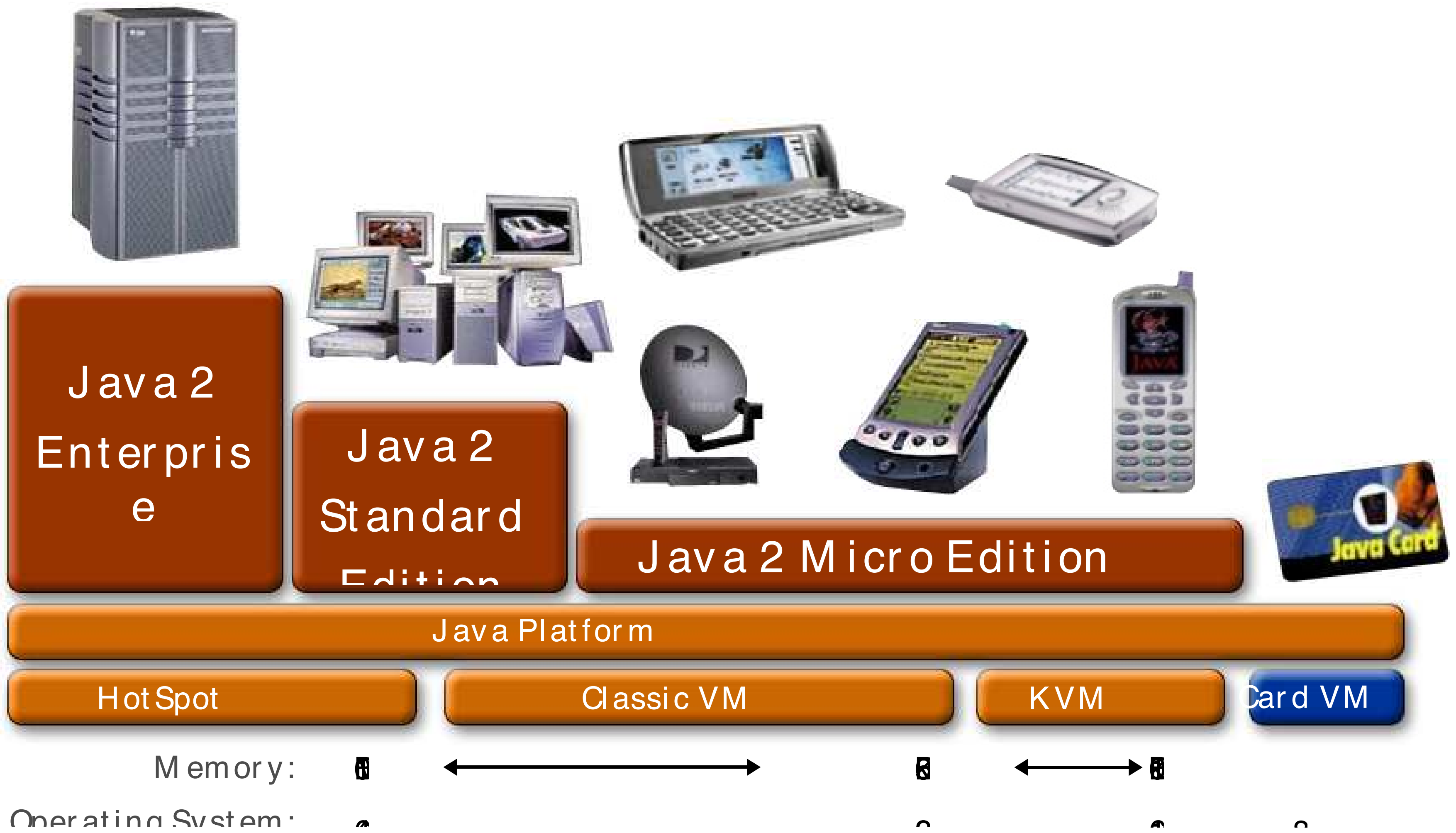
Global Education and Research

Sun Microsystems, Inc

Overall Presentation Goal

Learn how to build enterprise applications with Java™ 2 Platform, Enterprise Edition (J2EE) Technology building Enterprise JavaBeans™ (EJB) using Java Studio

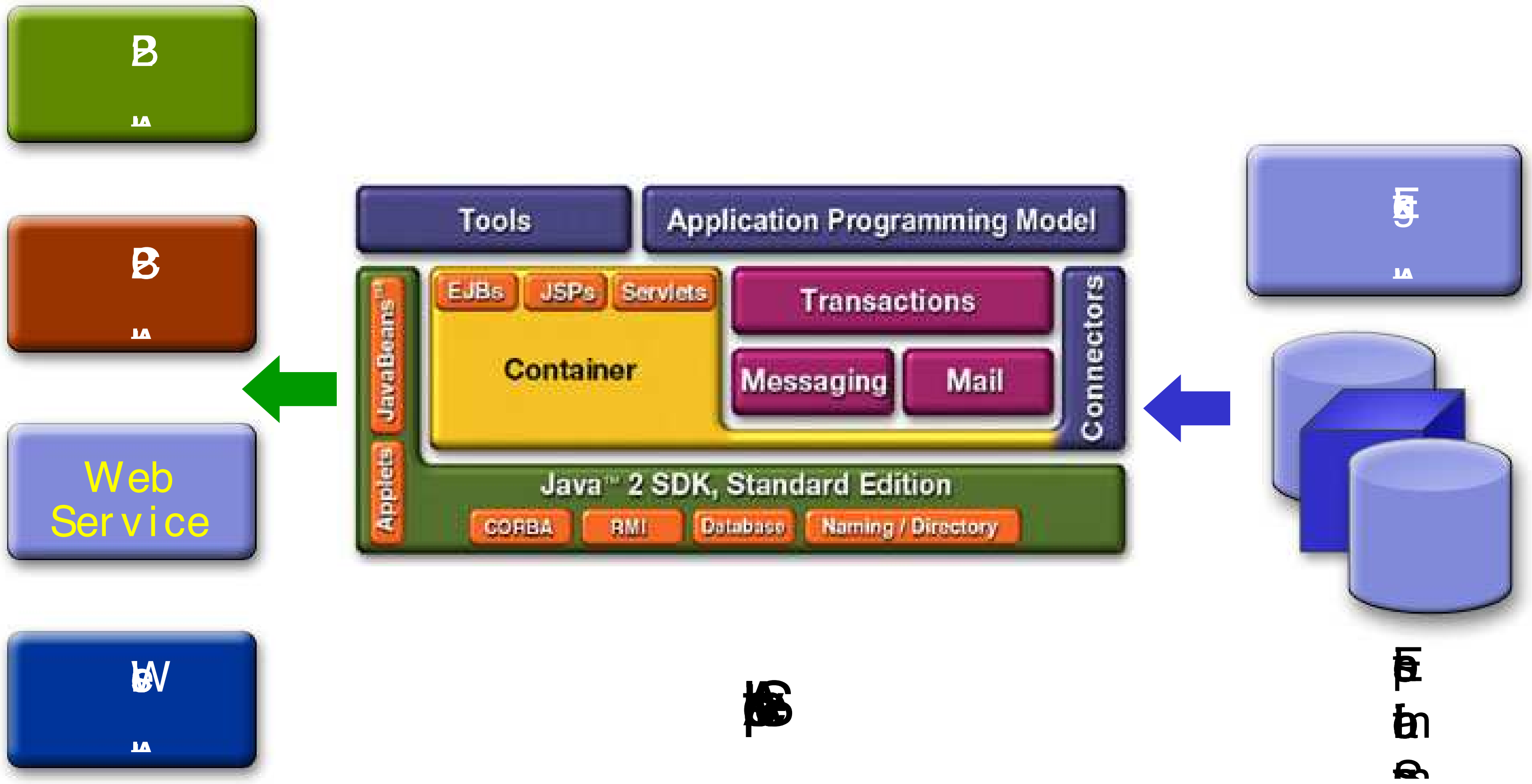
Java 2 Platform



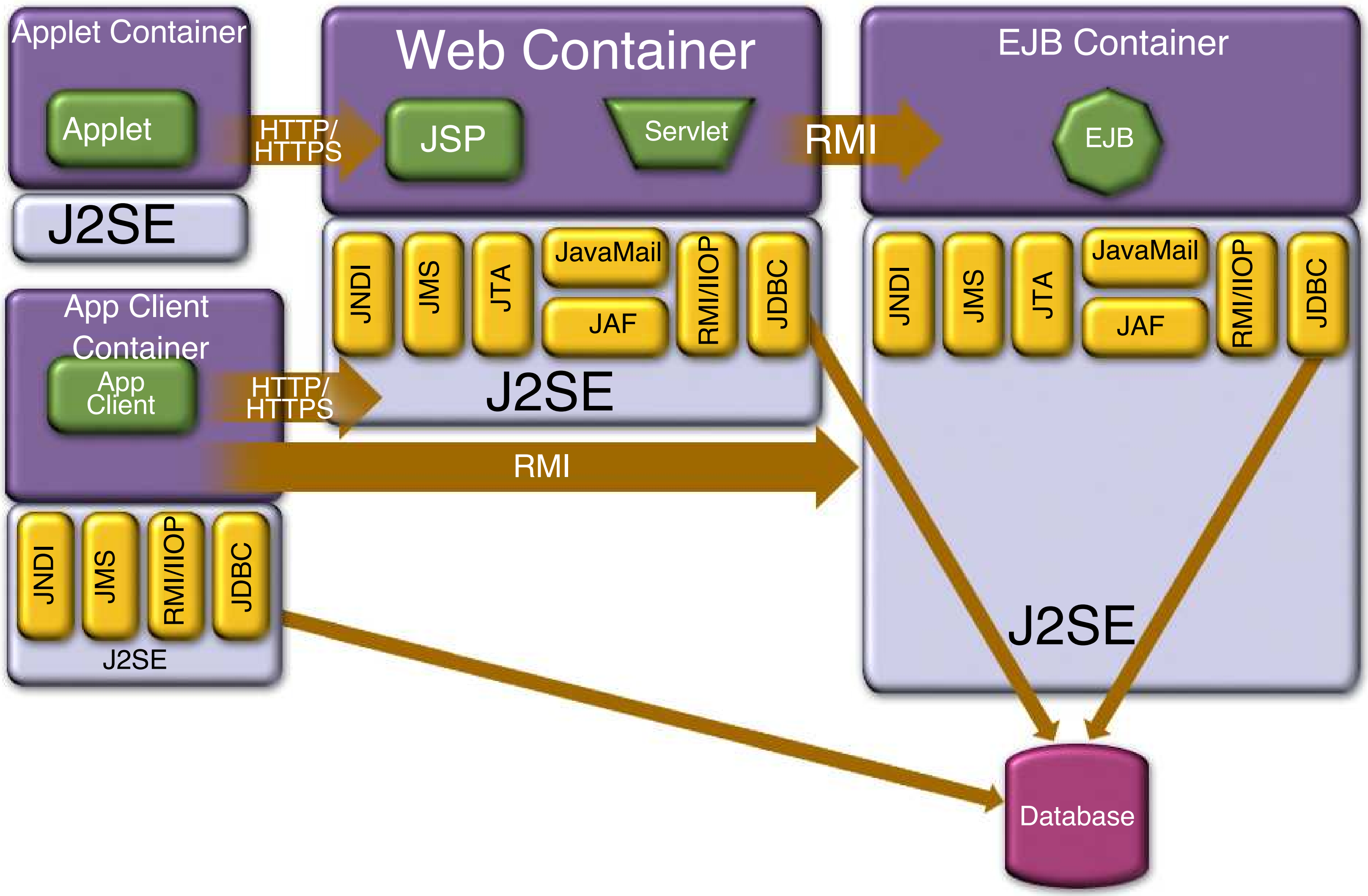
J2EE™ Platform

- The J2EE platform brings the benefits of component-based development to enterprise applications
- Components are:
  - Simpler to develop
  - Portable
  - Reusable

The J2EE Platform Architecture



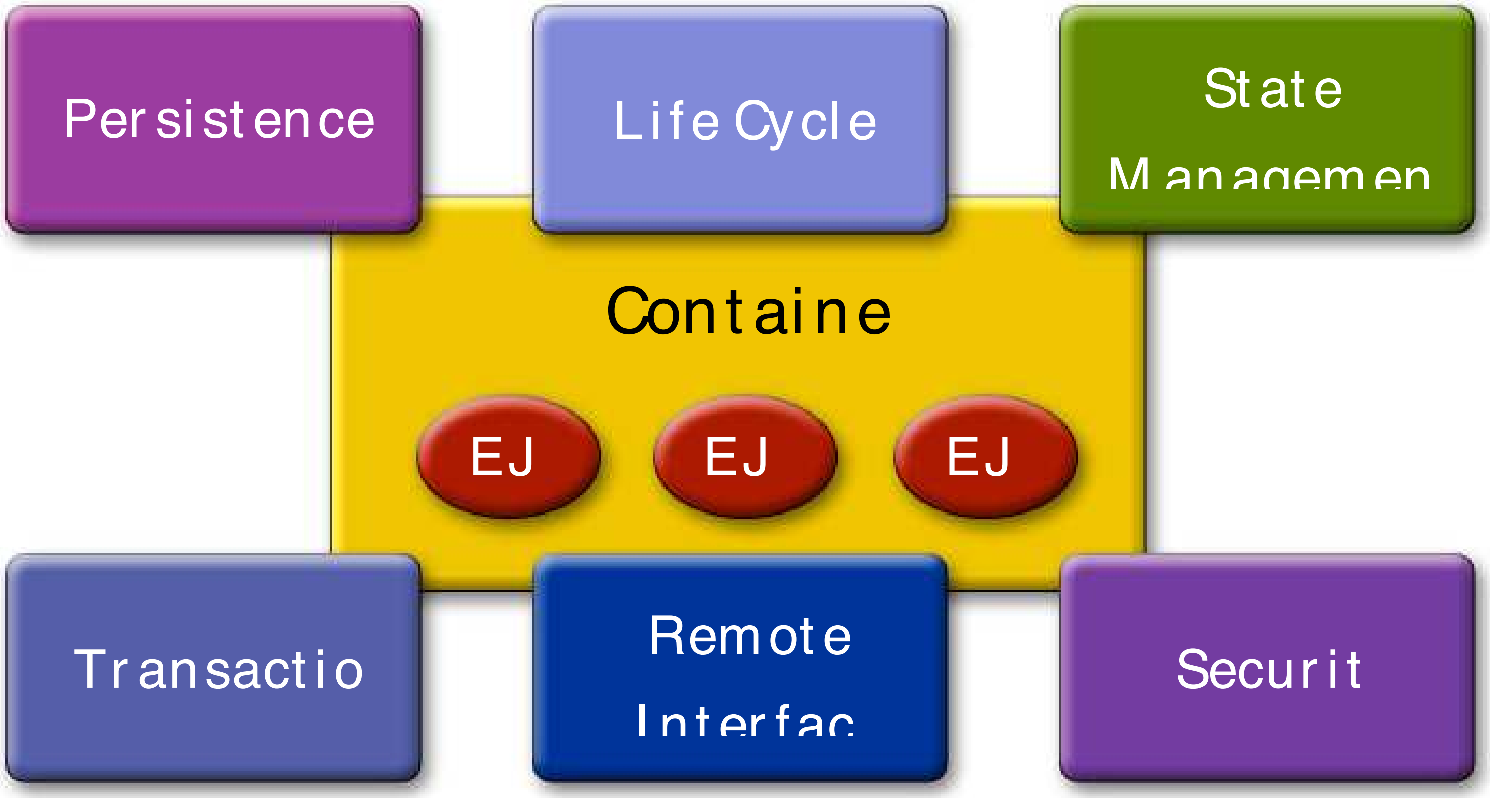
The J2EE Platform Containers and Components



J2EE Components

- Server-side component technologies
  - Enterprise JavaBeans™ (EJB)
  - Java™ Servlet API and JavaServer Pages™
- Client-side components
  - Application client
- Configured via deployment descriptors
- Deployed into containers

Container Responsibility



Advantages of Container

- Advantages of letting the container take care of security, persistence, and transactions:
  - Less programming, easier
  - Security, transactions: configurable when assembling application components
  - Transactions, persistence: container can give better performance

Enterprise JavaBeans (EJBs) Overview

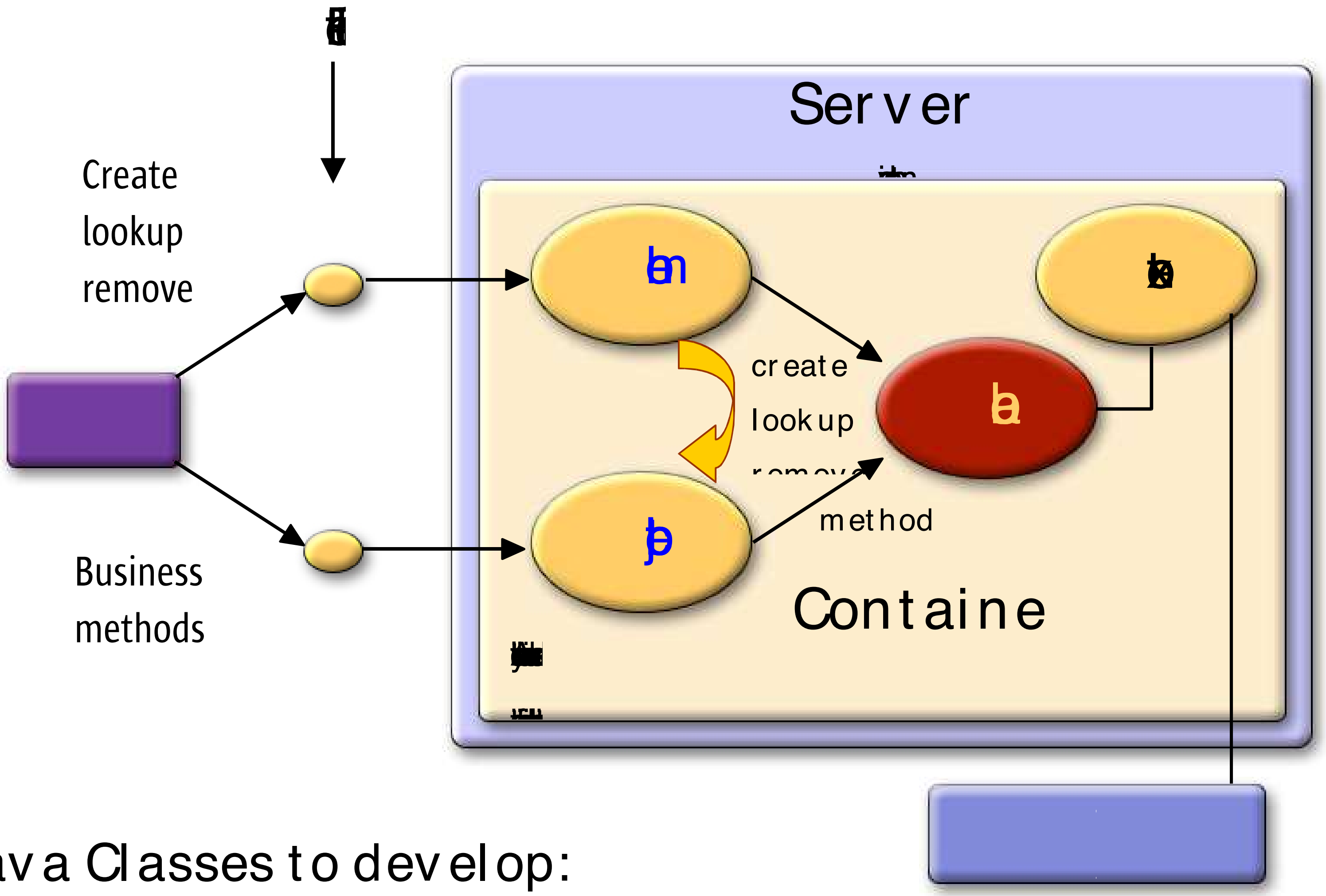
What Is EJB Technology?

- Enterprise JavaBeans (EJB) is the cornerstone of J2EE
- A standard-based, server-side component technology for building distributed, object-oriented applications
- Easy development and deployment of Java technology-based application that are:
  - Transactional, distributed, multi-tier, portable, scalable, secure, ...

Why EJB™ Technology?

- Leverages the benefits of component-model on the server side
- Separates business logic from system code
- Provides framework for portable components
  - Over different J2EE-compliant servers
  - Over different operational environments
- Enables deployment-time configuration
  - Deployment descriptor

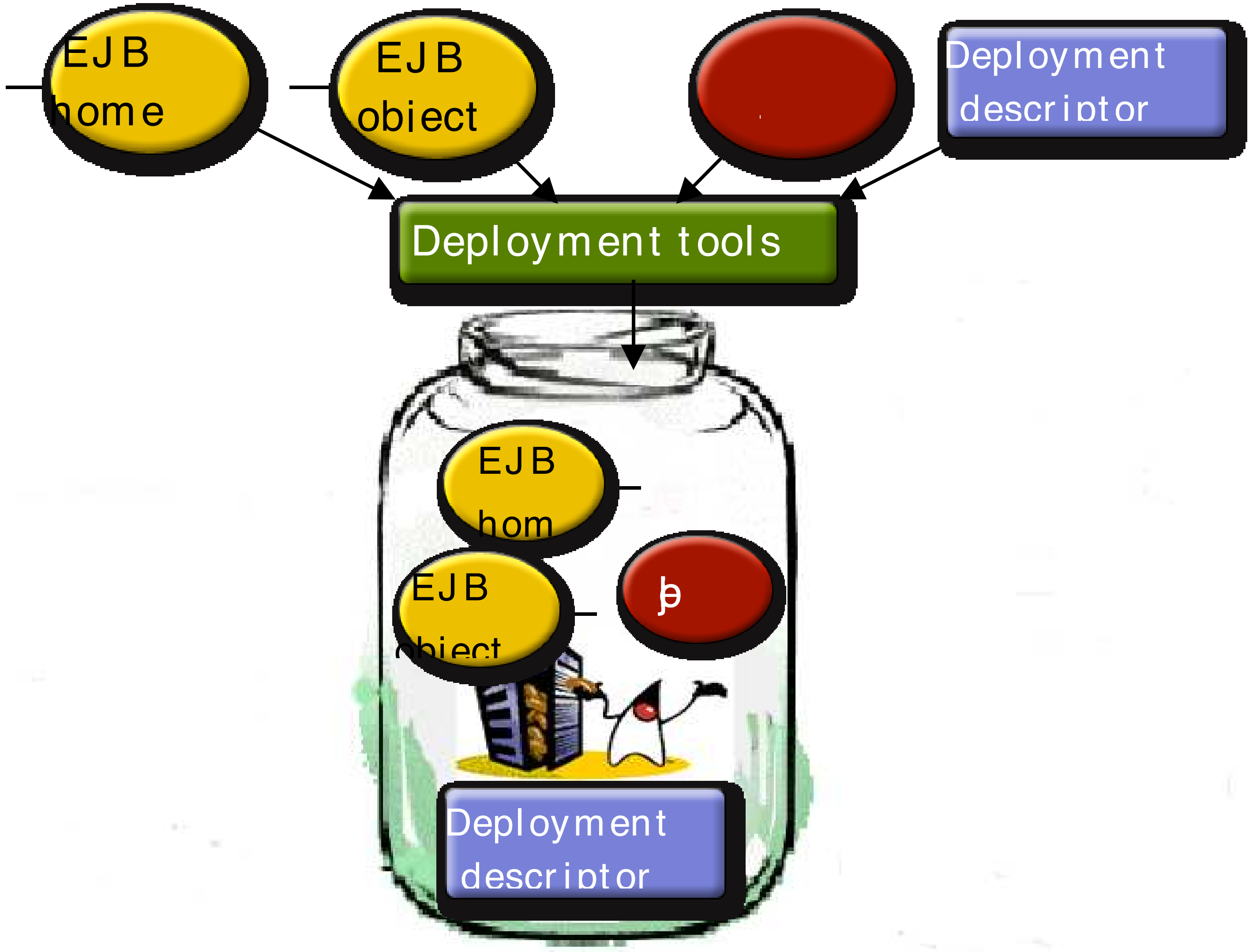
EJB™ Architecture



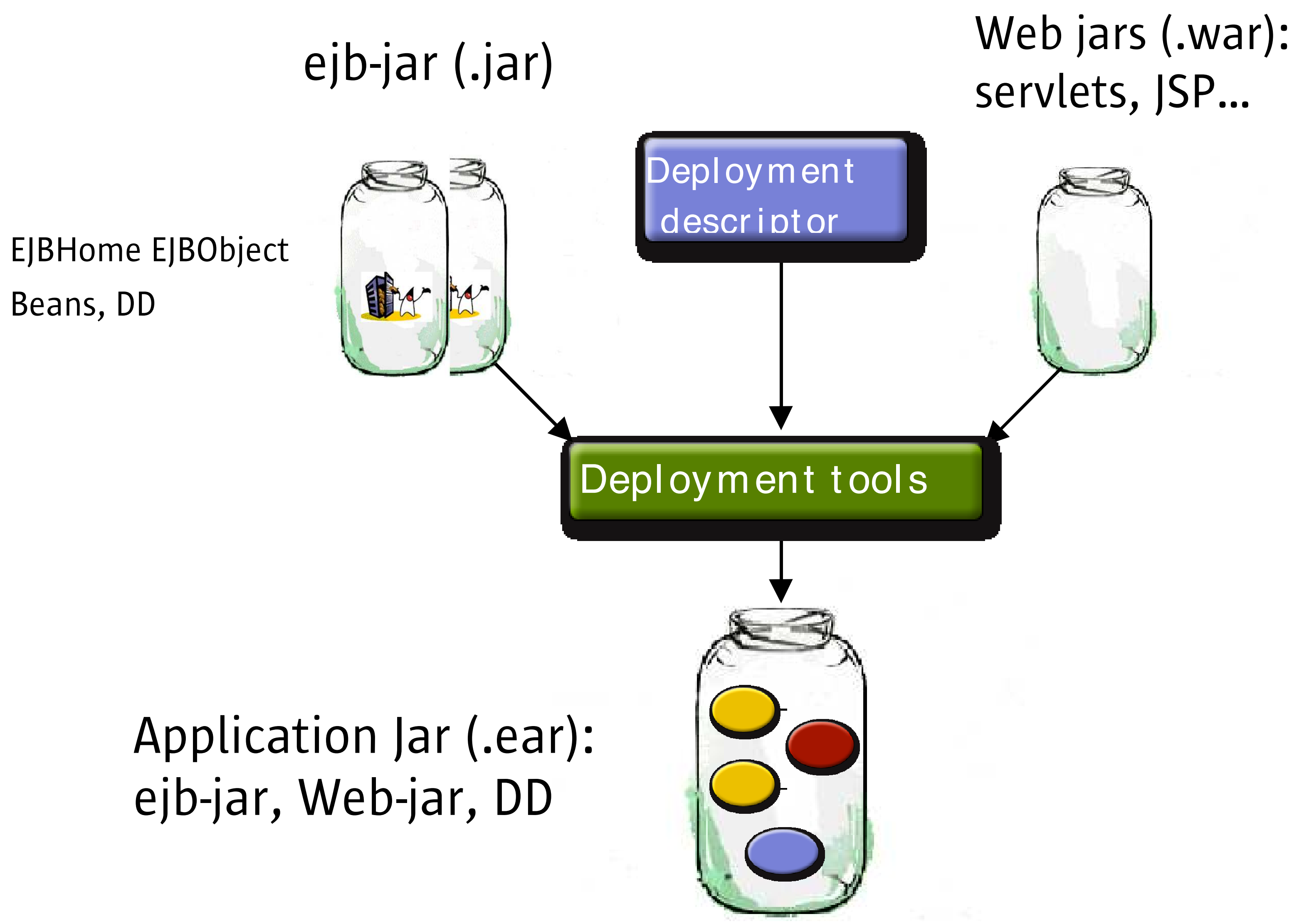
Java Classes to develop:

Two interfaces and One implementation

EJB Packaging



EAR Packaging



Deployment Tools

- GUI-based Tools
- Packaging, assembly and Deployment of applications for the J2EE platform
- Wizard for creation of J2EE platform-based modules
- Simplify deployment descriptor generation
- Application update functionality

Types of EJBs

- Entity Bean
  - For Persistent Data Management
  - BMP: Bean Managed Persistence
  - CMP: Container Managed Persistence
- Session Bean
  - Statefull: Maintains the client session
  - Stateless: Fast and less overhead
- Message Driven Bean
  - Asynchronous communication with EJBs

Entity Bean

- Model business concepts, often called domain classes
  - Abstractions of real-world entities that can be expressed as nouns (i.e., Customer inventory item)
- Object view of business entity stored in persistent storage
  - In memory view and manipulation of data
- Entity beans support shared access from multiple users
- Entity beans can be re-instantiated from attributes stored in database:
  - “Lives” as long as the data in the database

Entity Bean Types

- Bean-Managed Persistence

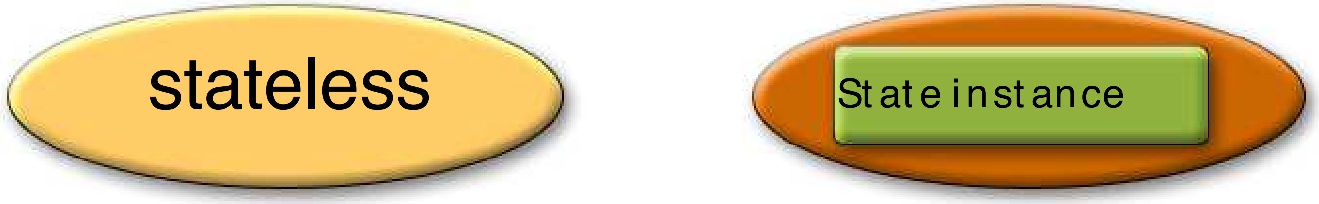
  Advantages:
  - Developer has full control

  Disadvantages:
  - More complex coding
  - May be less portable
- Container-Managed Persistence

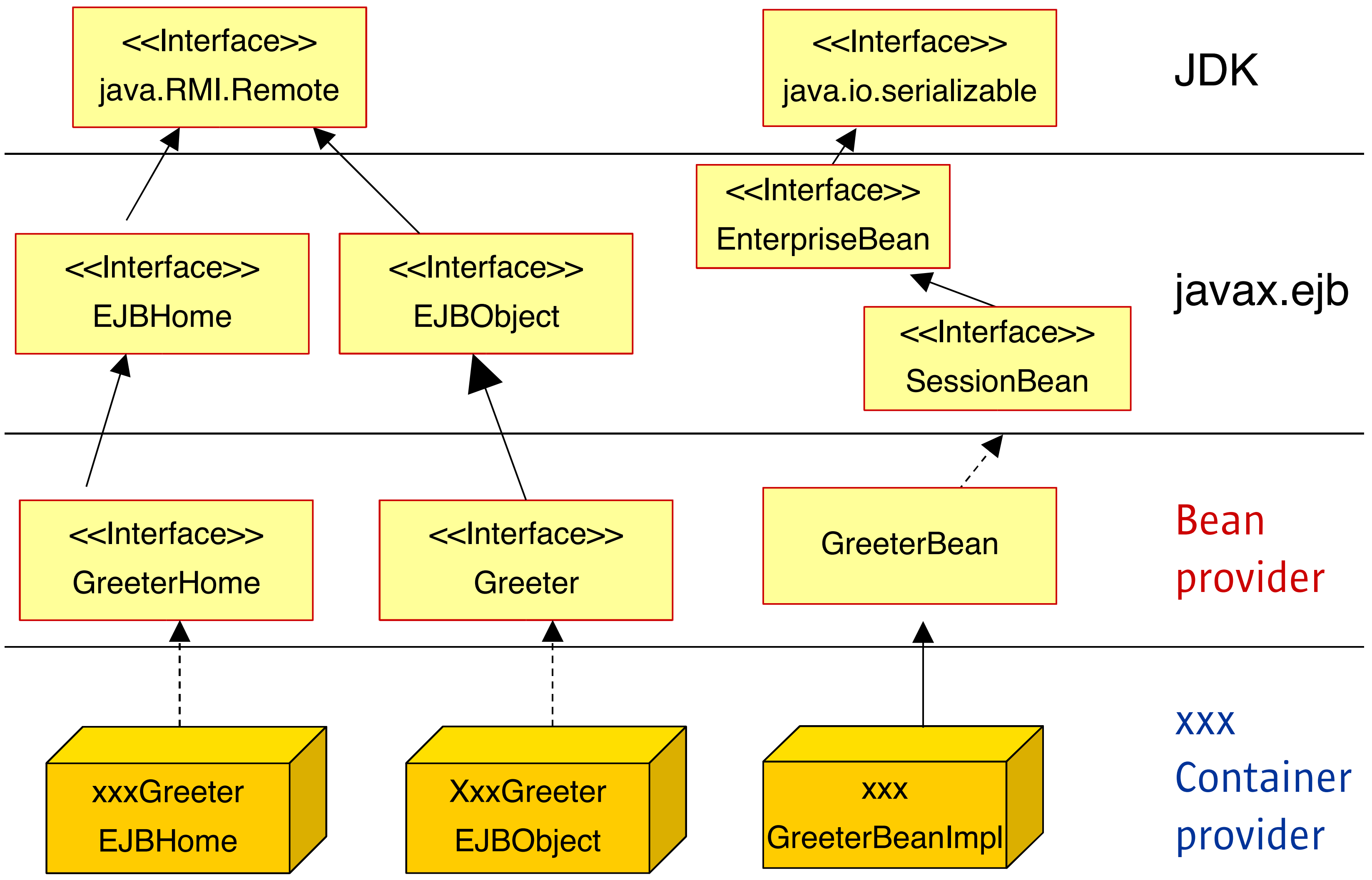
  Advantages:
  - Vendor does the work, better caching, performance
  - Changes are implemented in deployment descriptors
  - Generated at deploy time, more portability

Two Types of Session Beans

- Stateless
  - An object that represents a stateless service
  - Provides responses to requests without storing client specific information
  - Transient
  - Temporary piece of business logic needed by a specific client for a limited time span
- Stateful
  - Maintains client specific state



EJB API



<<Interface>>
java.RMI.Remote
<<Interface>>
java.io.serializable
JDK
<<Interface>>
EJBHome
<<Interface>>
EJBObject
<<Interface>>
EnterpriseBean
<<Interface>>
SessionBean
javax.ejb
<<Interface>>
GreeterHome
<<Interface>>
Greeter
GreeterBean
Bean provider
xxxGreeter EJBHome
XxxGreeter EJBObject
xxx GreeterBeanImpl
xxx Container provider

Greeter EJB Example

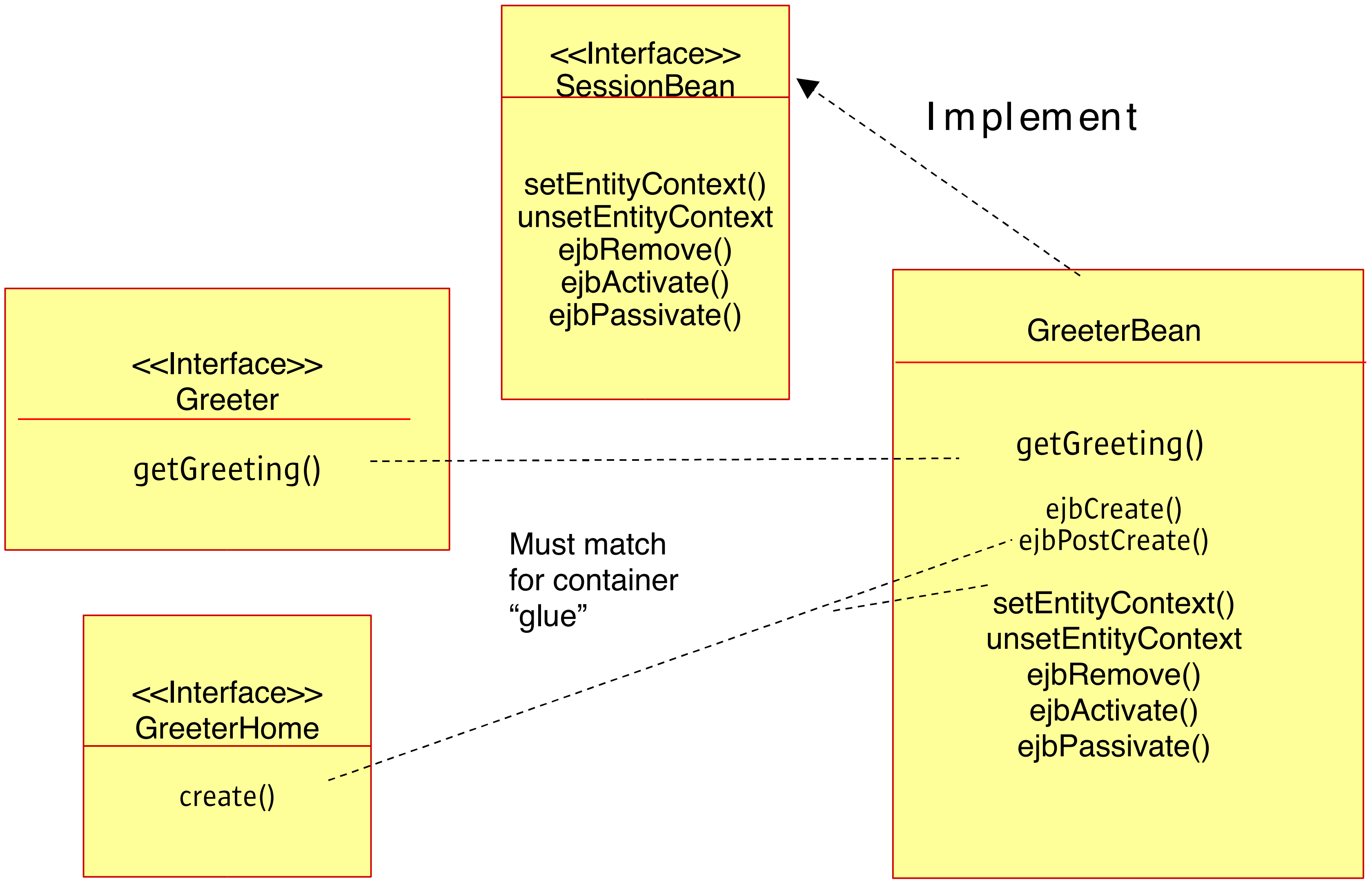
- Remote Interface code:

```
package ejb;
import javax.ejb.*;
public interface Greeter extends javax.ejb.EJBObject {
   public java.lang.String getGreeting() throws
                                java.rmi.RemoteException;
}
```

- Home Interface code:

```
public interface GreeterHome extends javax.ejb.EJBHome {
public ejb.Greeter create()
 throws javax.ejb.CreateException, java.rmi.RemoteException;
}
```

Greeter EJB Implementation



Deployment Descriptor

```
[illegible]
[illegible]
[illegible]
 [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
 [illegible]
[illegible]
[illegible]
```

EJB Transactions

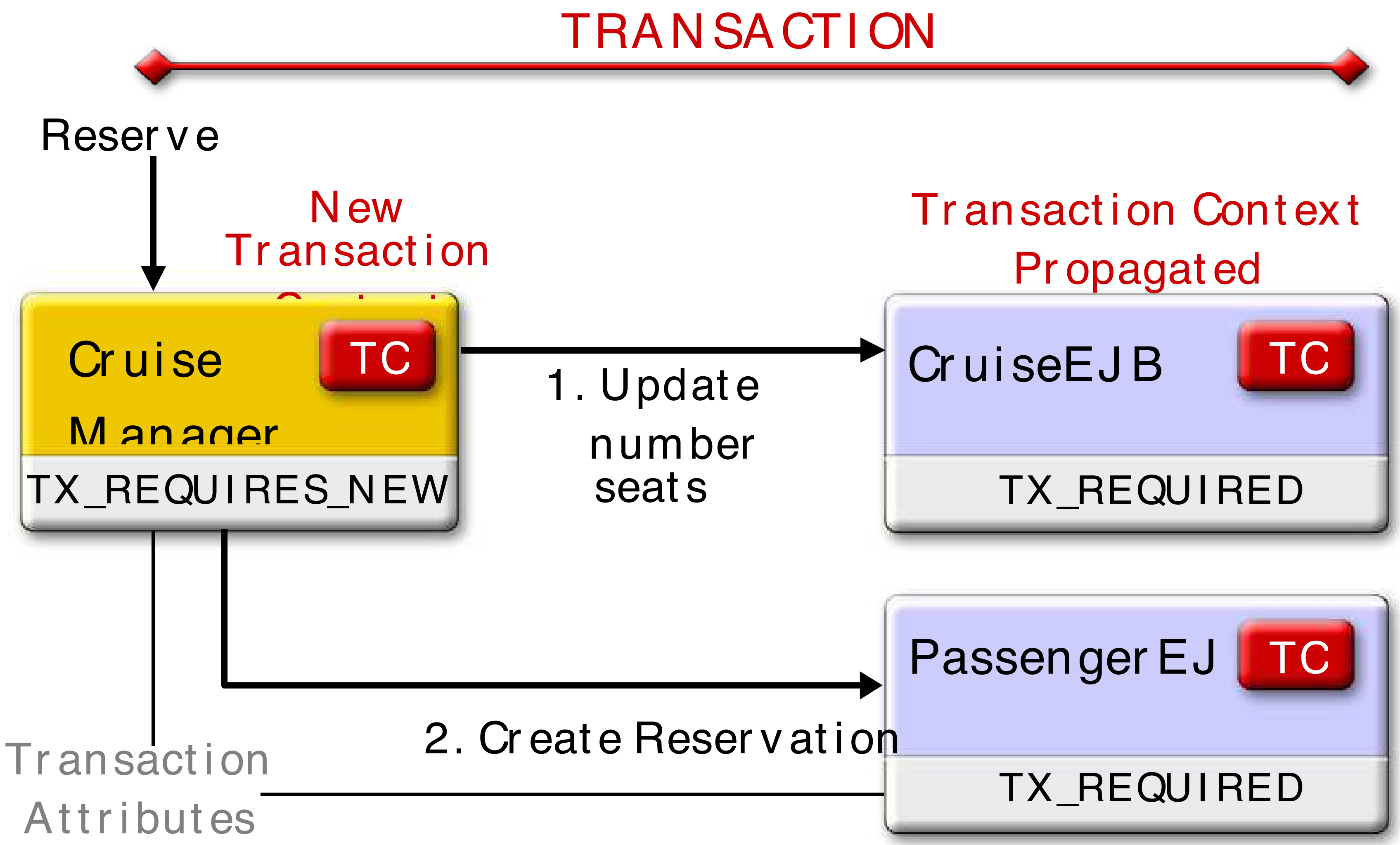
- EJB transactions in two ways:

  Container-managed transactions: Depend on the declarative transactions specified in deployment descriptor

  - The EJB container controls the integrity of your transactions

  Bean-managed transactions: Use the user transaction API (JTA) to explicitly drive transactions

Container Managed Transaction



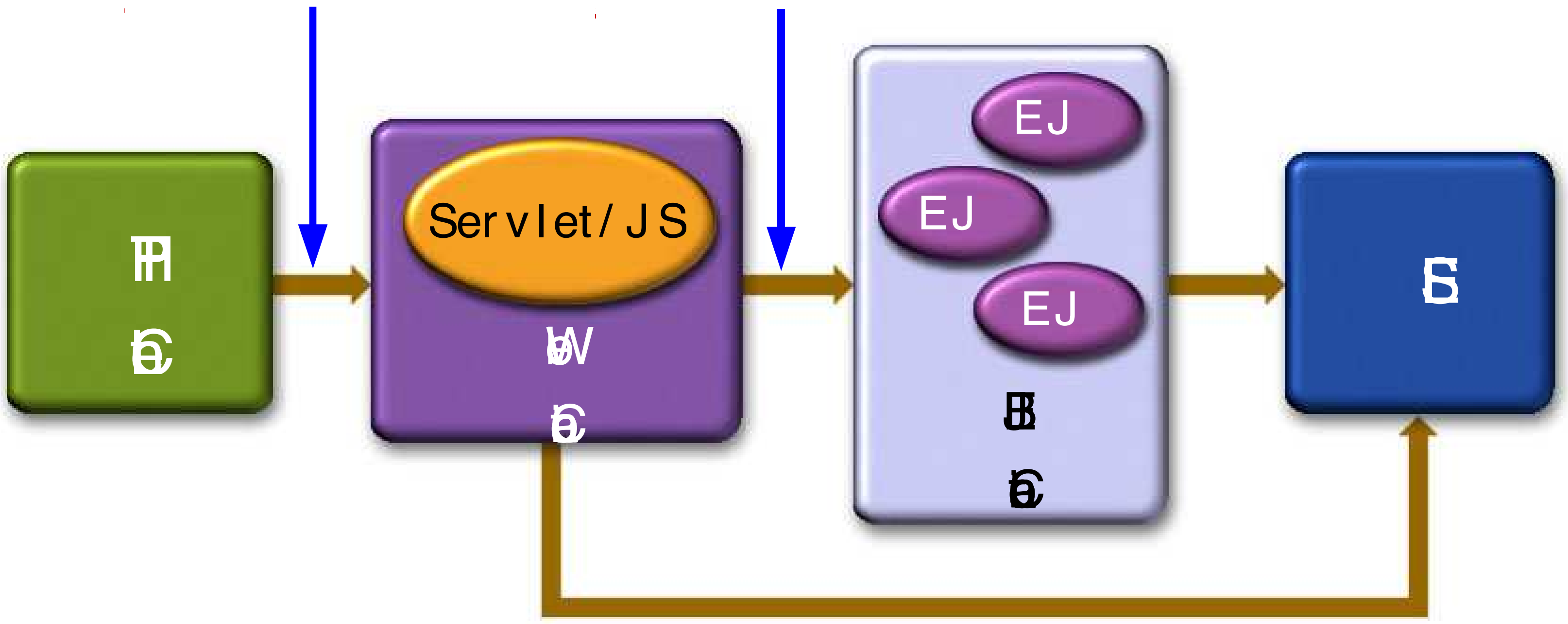
Declarative Transaction Management

- The following transaction attributes can be specified in the deployment descriptor to declare what type of transaction support the bean or bean method requires:
  - TX_NOT_SUPPORTED
  - TX_SUPPORTS
  - TX_REQUIRED
  - TX_REQUIRES_NEW
  - TX_BEAN_MANAGED
  - TX_MANDATORY

J2EE Security

- Declarative and programmatic security
- Realm administration
  - LDAP, certificate, database, file, Solaris-based realms
- Pluggable authentication via JAAS
  - You can add custom realm
- Single sign-on (value-add)
  - Same authenticate state shared among multiple J2EE applications

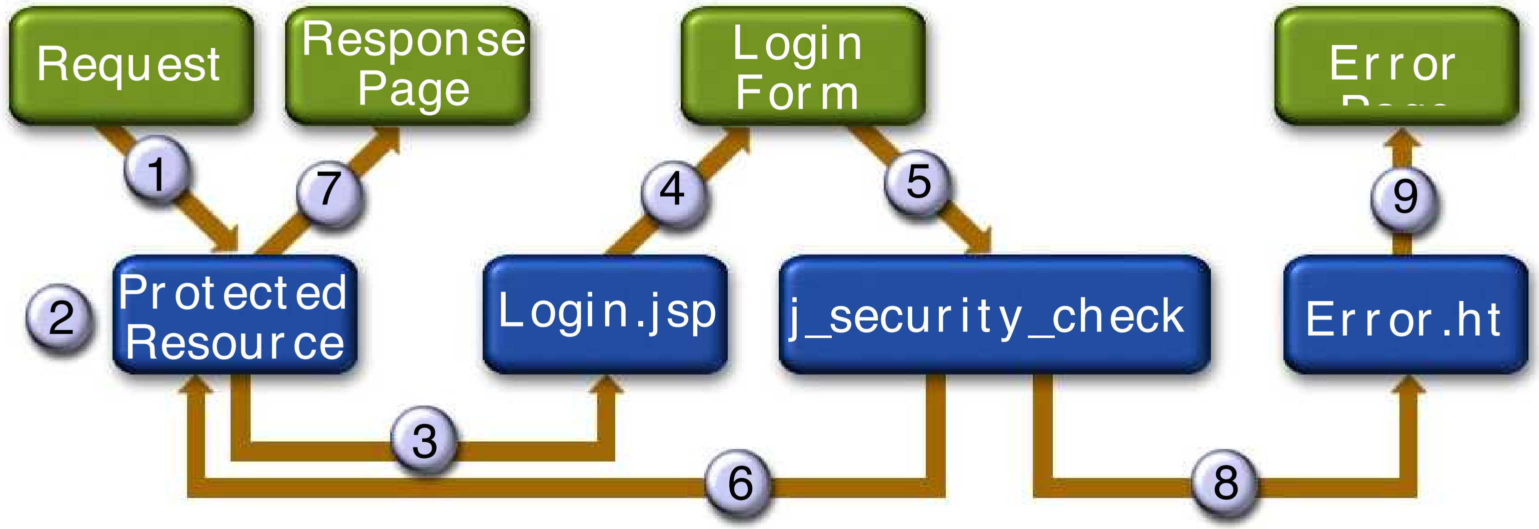
Authentication Framework of J2EE



Web-Tier Authentication

- Authentication Mechanisms by which browser supplies user identity information (logging-in) to web container
  - HTTP basic authentication (with or without SSL)
  - Form-based authentication (with or without SSL)
  - Certificate authentication
- Web container then performs actual authentication
  - By checking it against "backend user identity information" (Realms)
    - Database, LDAP server, Flat-file, etc.

Form-Based Login in Detail



1[illegible]

2[illegible]

3[illegible]

4[illegible]

5[illegible]

6Authentication [illegible]

7Authorization [illegible]

8[illegible]

9[illegible]

Development and Deployment of EJBs Using Java Studio

J2EE Application Development

- Design and develop components
  - Create Java source
  - Create deployment descriptor
- Assemble components into application
  - Create deployment descriptor
  - EJB modules are packaged as JAR files
  - Web modules are packaged as JAR files with a .war (Web ARchive) extension
- Deploy the enterprise applications
  - Deployment time configurations

Java Studio 5, Enterprise Edition

- Full Featured Development Environment for J2EE application development
- Control deployment of enterprise application to: Sun™ ONE Application Server, BEA, RI
- Flexibility: Core (foundation) and plug-in modules
- Advanced Source Code Editor
- Auto-complete, color coding, Source Code Control etc.

Java Studio 5 and Application Server 7.0

[illegible]	[illegible]
[illegible]	- [illegible] - [illegible] - [illegible]
[illegible]	- [illegible] - [illegible] - [illegible]
[illegible]	- [illegible] - [illegible]

Java Studio 5 and Application Server 7.0

[illegible]	[illegible]
[illegible]	- [illegible]
[illegible]	- [illegible] - [illegible]

J2EE Roadmap

J2EE 1.4 Content

- JAX-RPC 1.1 (JSR-101)
- SAAJ 1.1
- Web Services (JSR-109)
- Management (JSR-77)
- Deployment (JSR-88)
- Connectors 1.5
- JMX 1.1
- JMS 1.1
- JTA 1.0
- Servlet 2.4
- JSP 2.0
- EJB 2.1
- JAXR 1.0 (JSR-93)
- JACC (JSR-115)
- JAXP 1.2
- JavaMail 1.3
- JAF 1.0

J2EE Ease-of-Development (1)

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- [illegible]
  - [illegible]

J2EE Ease-of-Development (2)

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]

EL Example:

Before (in JSP 1.2):

```
<% Map m = (Map)pageContext.get(“myMap”);
   FooBean f = ((FooBean)m.get( "key" ));
   if( f != null ) {%><%= f.getBar() %><%} %>
```

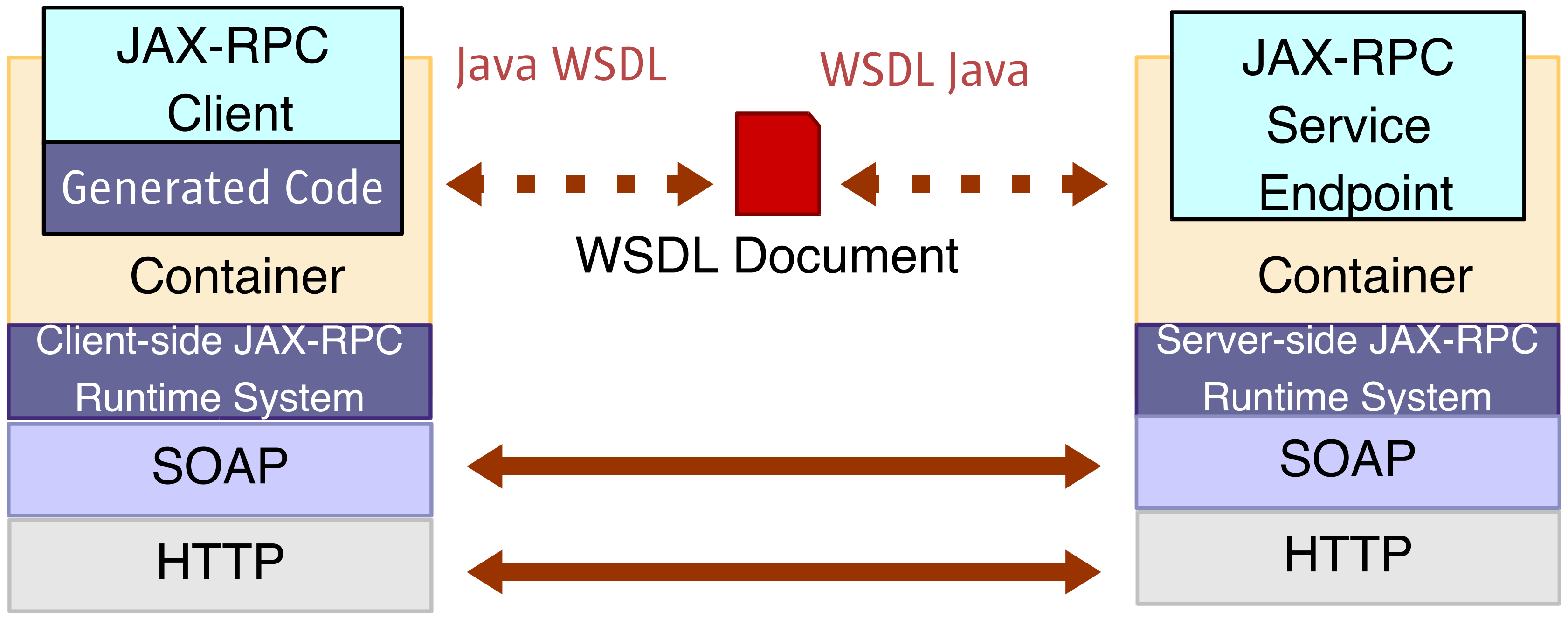
After (in JSP 2.0):

```
${myMap["key"].bar}
```

J2EE Web Services

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- [illegible]
  - [illegible]

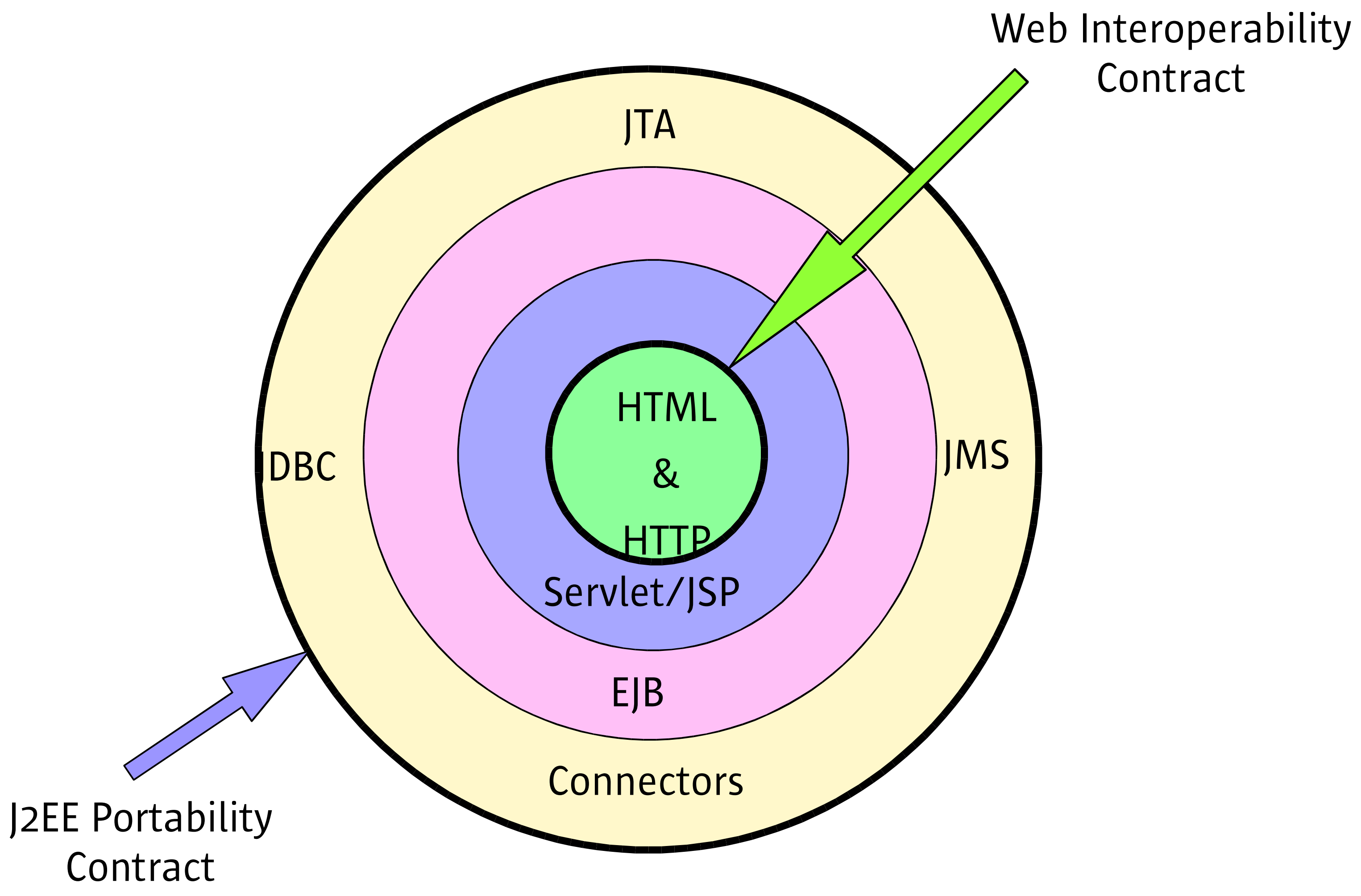
JAX-RPC Architecture



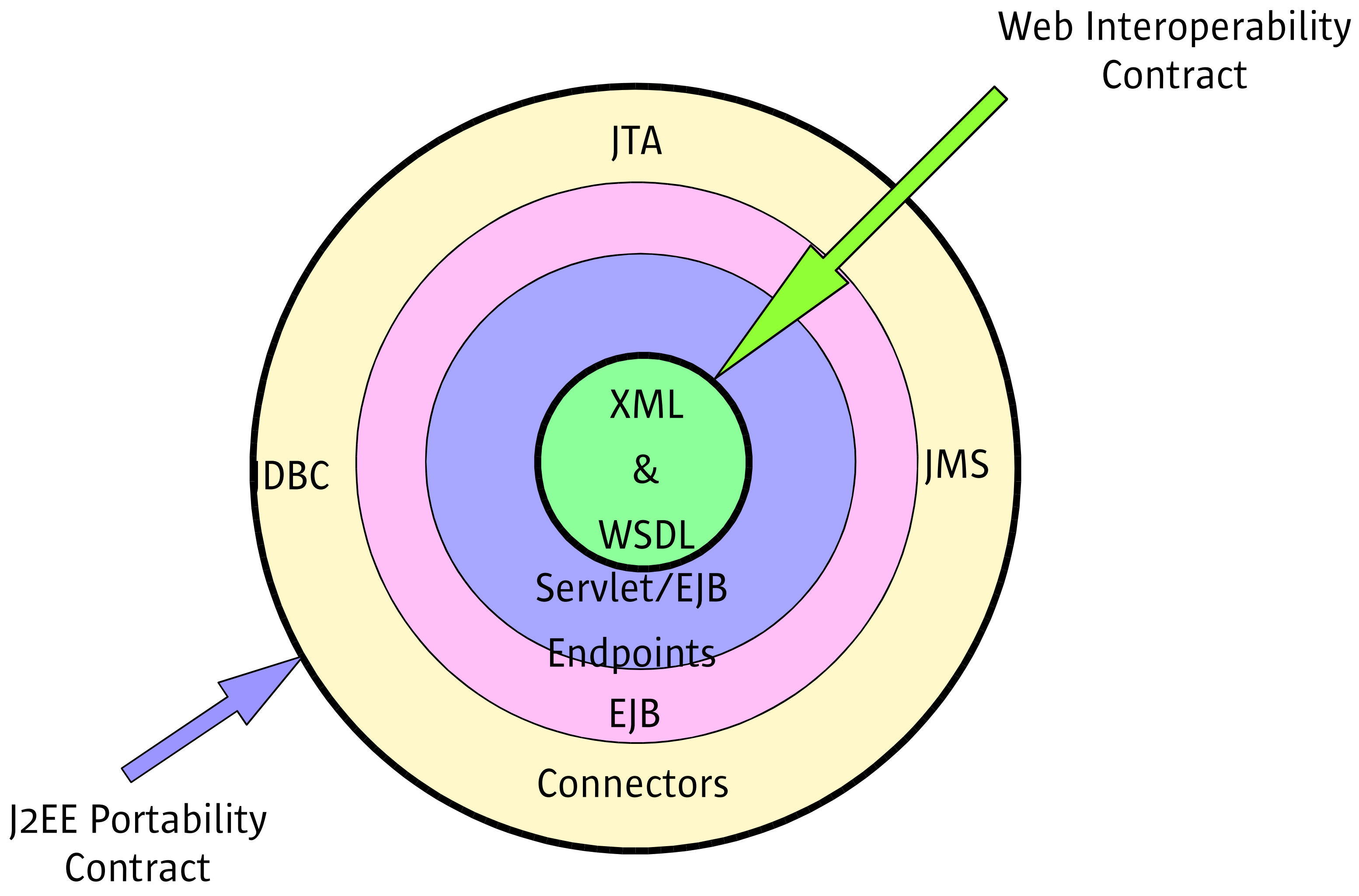
Web Standards Drive Service Oriented Architectures

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- [illegible]
  - [illegible]
  - [illegible]

J2EE Web Application



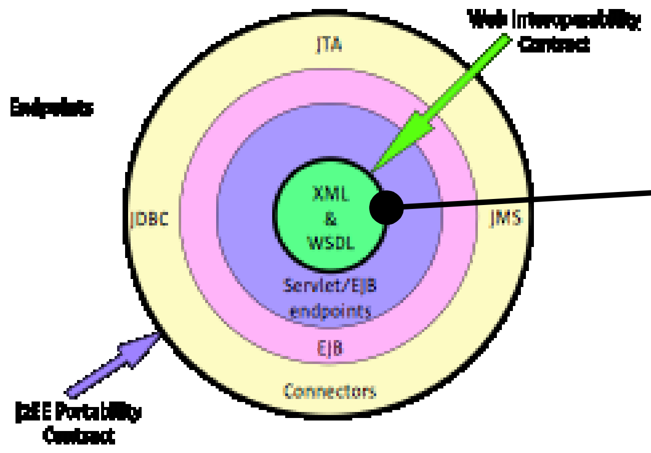
J2EE 1.4 Web Service



XML Message Exchange Patterns

- [illegible]
    - [illegible]
    - [illegible]
    - [illegible]
- [illegible]
    - [illegible]
    - [illegible]

JAX-RPC - JAXB - JAXP Core Web Service APIs



JAX-RPC - The Java^{TM} WSDL MEP API

JAXB - The Java^{TM} XML Binding API

JAXP - The Java^{TM} XML Parsing API

Basic But Powerful

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
    - [illegible]
    - [illegible]
    - [illegible]

J2EE Integration

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- [illegible]
- [illegible]
  - [illegible]
  - [illegible]

J2EE Management, Deployment and Authorization

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- [illegible]
  - [illegible]
- [illegible]

J2EE Scalable Business Logic

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- [illegible]
  - [illegible]

Ease-of-Development in J2EE

- Main thrust for J2EE Platform going forward: Ease-of-Development
- J2EE Platform is designed to serve the needs of every developer:
  - Enterprise Developer
  - Corporate (Workgroup) Developer
  - Content (Web) Developer
- J2EE is becoming a ubiquitous platform for every type of application
  - Not just the Enterprise

Example: EJB Creation

Currently, to create an EJB:

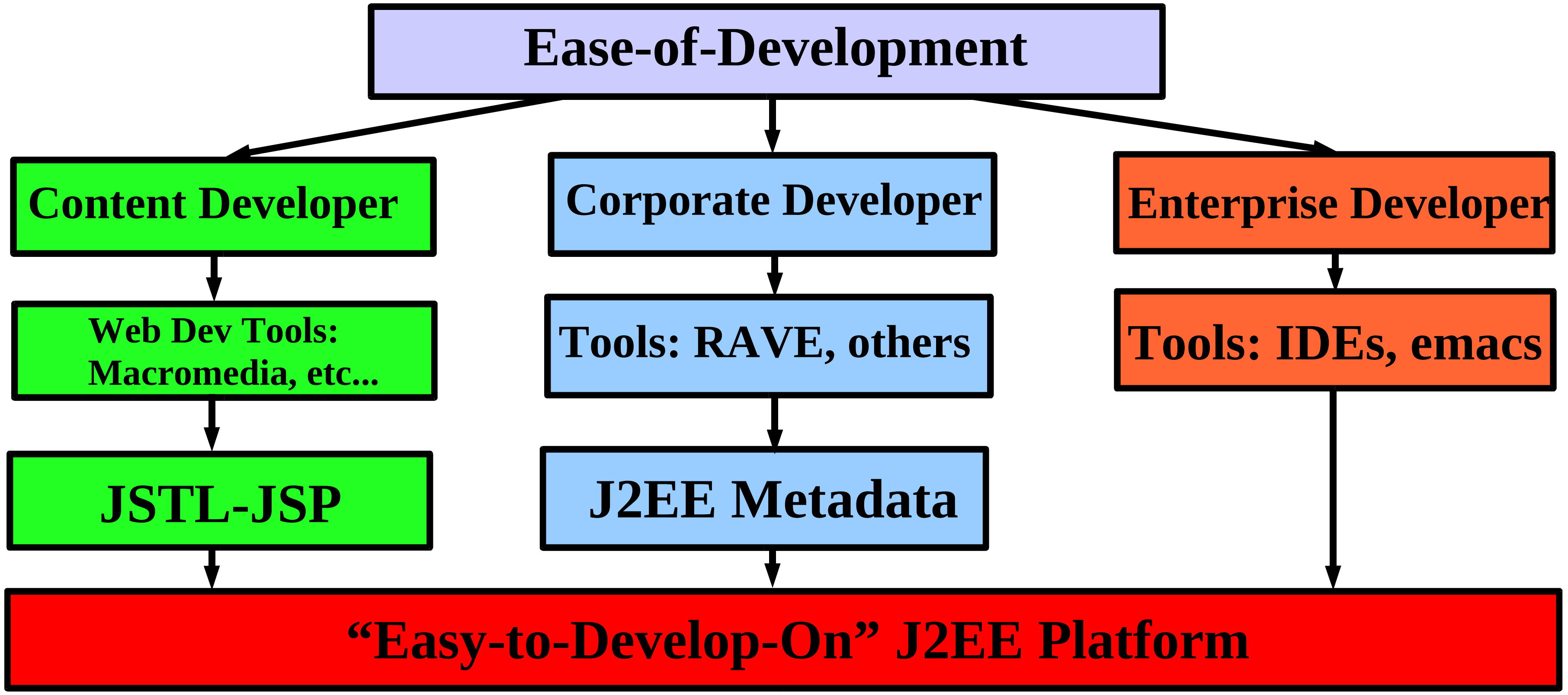
```
    Context initial = new InitialContext();
    Object objref = 
initial.lookup(java:comp/env/ejb/SimpleFoo”);
FooHome home = (FooHome)
        PortableRemoteObject.narrow(objref,FooHome.class);
Foo myFoo = home.create();
```

Wouldn't It be Nice, instead:

```
              private @create Foo myFoo;
```

What is the Ease-of-Development?

- Tools are important, but...
- Code should be easy to write, understand and maintain as well



Metadata in Java™ Language

- Metadata facility for the Java™ programming language (JSR 175)
- Allows to define custom attributes
- Allows to annotate fields, methods and classes with attribute-value pairs
- Do not affect the semantics of a program
  - Stored by the compiler
  - Development and deployment tools can read and process the annotated program elements

JAX-RPC 2.0

- Proposed extension to JAX-RPC (JSR 224)
- Major focus on Ease-of-Development
  - To simplify the most common development scenarios for Web Services clients and servers.
  - Build on J2SE Metadata facility (JSR 175)
  - Align with JSR 181 (Metadata in WS)
- WSDL to Java^{TM} binding migrates to JAXB 2.0
- Improvements in handler processing framework
  - Choice of handler models
  - Improve the declarative model for handlers

EJB 3.0

- Proposed extension to EJB (JSR 224)
- Focus on Ease-of-Development
  - Define metadata attributes to annotate EJBs
  - Target to simplify/eliminate EJB deployment descriptors for developers
  - Automatic generation of component and home interfaces
  - Programmatic defaults for the common, expected behaviors of EJB container
  - Introduce simplified EJB component that more closely resembles a plain Java class.
  - Other improvements

JAXB 2.0

- Proposed extension to JAXB (JSR 222)
- Full W3C XML Schema support
- Will implement WSDL to Java^{TM} databinding for JAX-RPC 2.0
- Bi-directional XML Schema to Java^{TM} mapping
    - Java^{TM} to XML Schema mapping will be added
- Ease-of-Development feature
    - Use of annotations and metadata in bindings

JDBC 4.0

- Proposed extension to JDBC (JSR 221)
- Focus on Ease-of-Development
  - Management of JDBC Drivers
    - provide utility classes to improve the JDBC driver registration and unload mechanisms
  - Standard set of tags to manipulate and manage active connections
    - Using metadata and annotations
  - Align various persistence and update mechanisms
  - Support of JDBC RowSet data model

PHP Scripting and J2EE

- Will enable the development of portable Java classes that can be invoked from a page written in an scripting language (JSR 223)
  - including details on security, resources and class loader contexts
- Will work with PHP, ECMAScript, others...
- Ease-of-Development features in a Servlet container
  - Packaging scripting pages and Java^{TM} classes, into a single WAR file

Summary

- J2EE 1.4 fully implements Web Services protocols
- J2EE 1.4 fully supports WS-I
- J2EE introduces more Ease-of-Development and Web Services Features

If You Only Remember One Thing...

[illegible]

Summary

- J2EE is a proven platform for building flexible, scalable, reliable, maintainable enterprise applications
- Java Studio is a great tool for developing J2EE Solutions
- Sun's Applications Server is a 1st-class platform for developing and deploying scalable, robust and secure Enterprise Services

Development Resources

- Java™ 2 Platform, Enterprise Edition Developer Portal: java.sun.com/j2ee
- Download the Java Studio and Sun Application Server: http://www.sun.com/edu/edusoft/

References

- Enterprise JavaBeans specification JSR-153

  [illegible]

- Java 2 Platform, Standard Edition Specification

  [illegible]

- Implementing Enterprise Web Services JSR-109

  [illegible]

- Java APIs for XML based RPC JSR-101

  [illegible]

J2EE-Related JSRs

- Web services support for J2EE
  - JSR-109 (Web Services)
  - JSR-101 (JAX-RPC)
  - JSR-93 (JAXR)
- The following provide new capabilities to 1.4:
  - JSR-77 (Management)
  - JSR-88 (Deployment API)
  - JSR-115 (J2EE Authorization SPI)
  - JSR-56 (JNLP)

J2EE-Related JSRs (Cont.)

- The following JSRs enhance APIs:
  - JSR-112 (J2EE Connector Architecture 2.0)
  - JSR-152 (JSP 1.3)
  - JSR-154 (Servlet 2.4)
  - JSR-153 (EJB 2.1)
  - JSR-9XX (JAXP 1.2—XML Schema support)
  - JSR-9XX (JMS 1.1—Queue/topic unification)
- J2EE Client Provisioning (JSR 124)
- The J2EE Connector Architecture (JSR 016)

Marc Hamilton

marc.hamilton@sun.com

Sun Microsystems, Inc.

Sun microsystems

We make the net work.