IBM

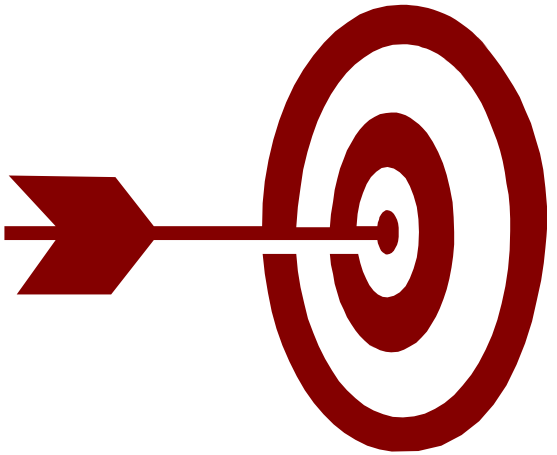
Tendances Logicielles

L'architecture pour répondre aux besoins métier

WebSphere 7.0

WAS v7 Programming Model Currency Goals

- One word …… Simplify
  - Simplify the programming model
  - Simplify application coding
  - Simplify application assembly / packaging
  - Simplify testing
  - Simplify Object Relational Mapping
  - Simplify Web Services Development



Leveraging Benefits of SOA Programming Model

WAS V7 Keeps Pace with Industry Advancements

Feature Packs enable you to selectively take advantage of new standards and features while maintaining a more stable internal release cycle.

1. Choose the application server technology you need.
2. Install additional functionality on core WAS 7
3. Build the Application Server you want without waiting for new releases.

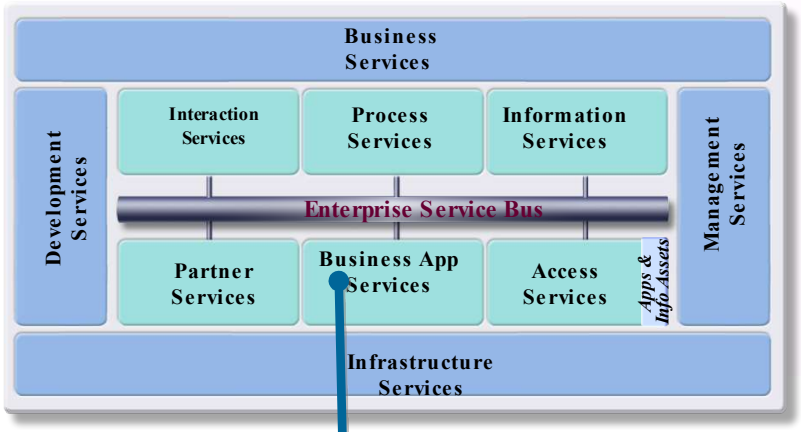
Goal: Leverage Benefits of SOA Programming Model

You want to…

- Build and deploy new flexible, agile applications/ services
- Reuse, extend existing applications
- Leverage programming standards to enable service composition

IBM Solution

- Core JEE 5 programming model
- EJB 3.0, JPA standards for simplified programming
- Support for latest Web services profiles
- Support for Java portlets



WebSphere Application Server

Supporting the strongest, most com- prehensive SOA programming model

Goal: Leverage Benefits of SOA Programming Model

You want to…

- Develop EJBs that have interfaces and inheritance hierarchies that reflect you business rather than the framework within which you are programming.
- Enable Java SE developers to quickly learn Java EE and develop enterprise applicationsg applications

IBM Solution

- Interfaces no longer have to implement the EJBObject interface.
- EJB methods no longer have to throw RemoteException.
- EJB implementations no longer required to implement EJB life cycle methods. Use callback annotations instead

All implemented as Plain Old Java Objects (POJOs)

EJB 2.1

```
public interface ShoppingCart
    extends EJBObject {
  public int
    someShoppingMethod()
    throws RemoteException;
}
```

EJB 3.0

```
public interface ShoppingCart
    {
    public int
    someShoppingMethod();
}
```

Goal: Simplify application coding

You want to…

• Write application code focused on your business logic.
•Avoid cluttering you application with plumbing code that interacts with the framework and runtime.

IBM Solution

- Inversion of control aka … 'Don't call us, we'll call you'.
- Uses dependency Injection.
  - Application code simply declares variables and annotates them to indicate what they want.
  - The container **injects** the specified object or resource references.
- Uses JSE 5.0 Annotations to place configuration metadate directly in the code.

EJB 2.1

```
Object obj =
    Context.lookup("java:comp/env/ejb/
    MyCartHome");
CartHome theCartHome = (CartHome)
    PortableRemoteObject.narrow(obj,
    CartHome.class);
ShoppingCart myCart =
    theCartHome.create() ;


myCart.someShoppingMethod();
```

EJB 3.0

```
@EJB
ShoppingCart myCart;

myCart.someShoppingMethod();
```

Goal: Simplify application assembly / packaging

You want to…

• Develop applications consisting of fewer artifact.
• Only work with deployment descriptors when I have something to add during assembly or deployment.

EJB 2.1

```
<ejb-jar>
  <enterprise-beans>
    <session>
      <description>Account Transfer Controller EJB</description>
      <display-name>Account Transfer Controller EJB</display-name>
      <ejb-name>AccountTransferController</ejb-name>
      <home>com.mycompany.AccountTransferControllerHome</home>
      <remote>com.mycompany.AccountTransferController</remote>
      <ejb-class>com.mycompany.AccountTransferControllerImpl</ejb-
        class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
  <assembly-descriptor>
    <container-transaction>
      <method>
        <ejb-name>AccountTransferController</ejb-name>
        <method-name>*</method-name>
      </method>
      <trans-attribute>Required</trans-attribute>
    </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```

EJB 3.0

```
@Stateless
public class AccountTransferControllerBean
      implements AccountTransferController {

      public void transfer(Account src, Account, dest, float amount)
                  throws InsuficientFundsException
      { ... }

}
```

IBM Solution

- EJB deployment descriptors are now optional.
  - Used to override information specified in the annotations.
  - Can be sparsely populated. No entries needed except to override defaults.
  - Can be used as alternative to annotations or in combination.
- Annotations assume the obvious – but allow specification of non-default values.

EJB 2.1

```
<ejb-jar>
 <enterprise-beans>
  <session>
   <description>Account Transfer Controller EJB</description>
   <display-name>Account Transfer Controller EJB</display-name>
   <ejb-name>AccountTransferController</ejb-name>
   <home>com.mycompany.AccountTransferControllerHome</home>
   <remote>com.mycompany.AccountTransferController</remote>
   <ejb-class>com.mycompany.AccountTransferControllerImpl</ejb-
     class>
   <session-type>Stateless</session-type>
   <transaction-type>Container</transaction-type>
  </session>
 </enterprise-beans>
 <assembly-descriptor>
  <container-transaction>
   <method>
    <ejb-name>AccountTransferController</ejb-name>
    <method-name>*</method-name>
   </method>
   <trans-attribute>Required</trans-attribute>
  </container-transaction>
 </assembly-descriptor>
</ejb-jar>
```

EJB 3.0

```
@Stateless
public class AccountTransferControllerBean
    implements AccountTransferController {

    public void transfer(Account src, Account, dest, float amount)
            throws InsuficientFundsException
    { ... }

}
```

Simplifying testing

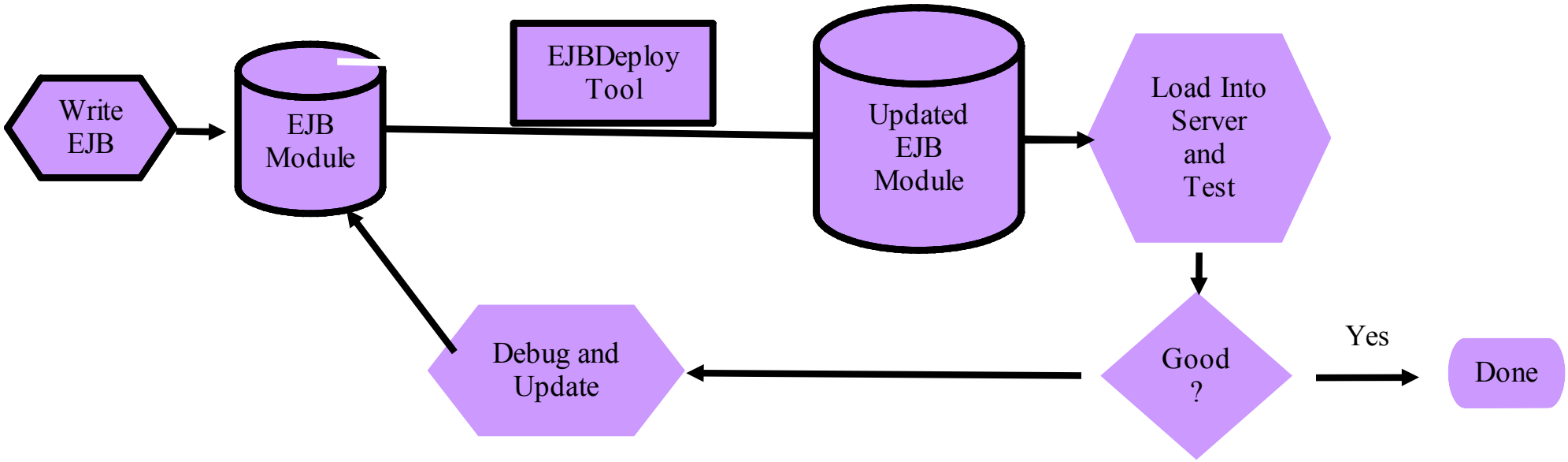
WAS v7 allows you to tightly iterate on code-test-debug cycle…

Put Away that EJBDeploy Tool!

Old: Process all your EJBs through the EJBDeploy tool before use

New: *Just-In-Time Deployment* for EJB 3.0 Modules

BONUS: It's Actually *Faster* at Runtime Than Before!

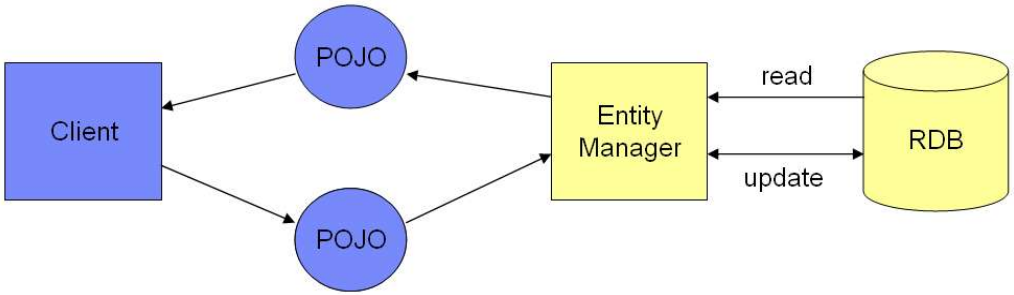


Quicker update-compile-test cycle, Improved developer productivity, Le

Goal: Simplify Object Relational Mapping

You want to…

- Move away from the complex and somewhat limited EJB 2.1 programming model for Persistence.
- Use simplified POJO based model similar to those that arose in response to this complexity.
- Use persistence programming model that's supported by standards including O/R mapping.



IBM Solution

- Java Persistence API (JPA)
  - O/R mapping in either annotations or XML deployment descriptor.
- Standard support for a richer semantic for O/R mapping.
  - Both object and native queries
  - RDB key generation
  - Inheritance
  - Optimistic locking

Entities are just POJOs

- They don't need to be 'deployed'
- The can be used as Data Transfer Object as well.

Simplifying Object Relational Mapping

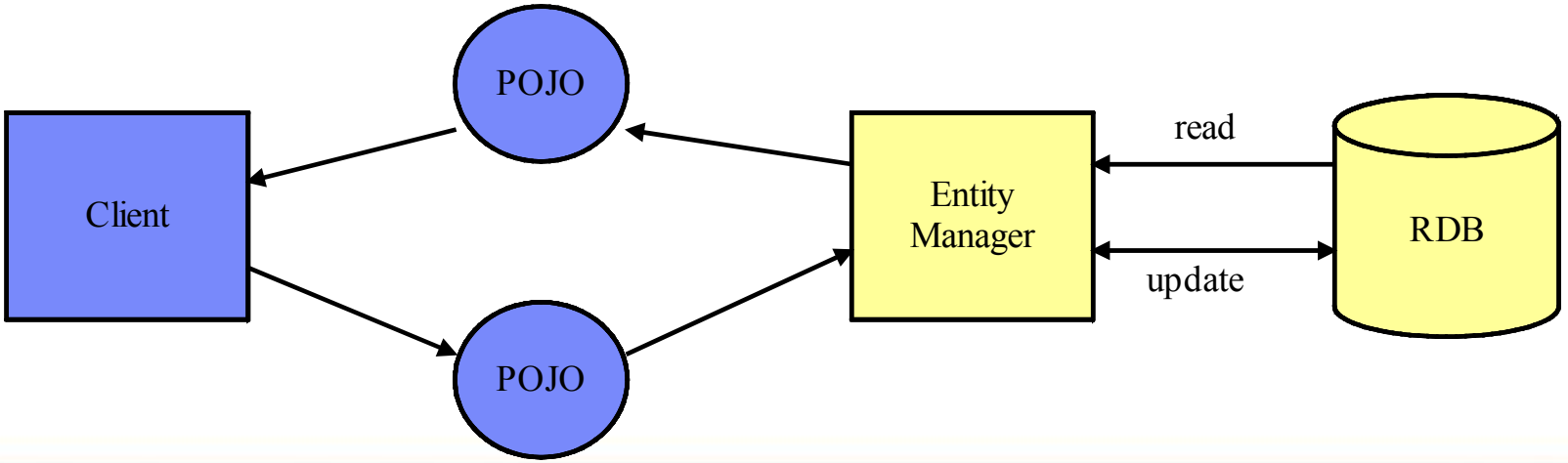
WAS v7 provides an O/R mapping model that is supported by standards…

- Standardized O/R mapping metadata (unlike EJB 2.1)
  - Metadata is specified with Java annotations or in XML deployment descriptors
- Entities are POJOs
  - No deployment code or abstract classes
  - No interfaces required
- An application server is not required
  - JPA is usable in both Java SE and Java EE environments

Simplifying Object Relational Mapping

WAS v7 leverages POJO based progamming and the Entity Manager API to simplify object persistence.

- Simplified object persistence
  - POJO-based persistence
  - Modeled on successful patterns
  - Simplifies JDBC access patterns
- Container stays out of the way



Simplifying Object Relational Mapping

With JPA WAS v7 introduces standardized support for inheritence in its O/R mapping model.

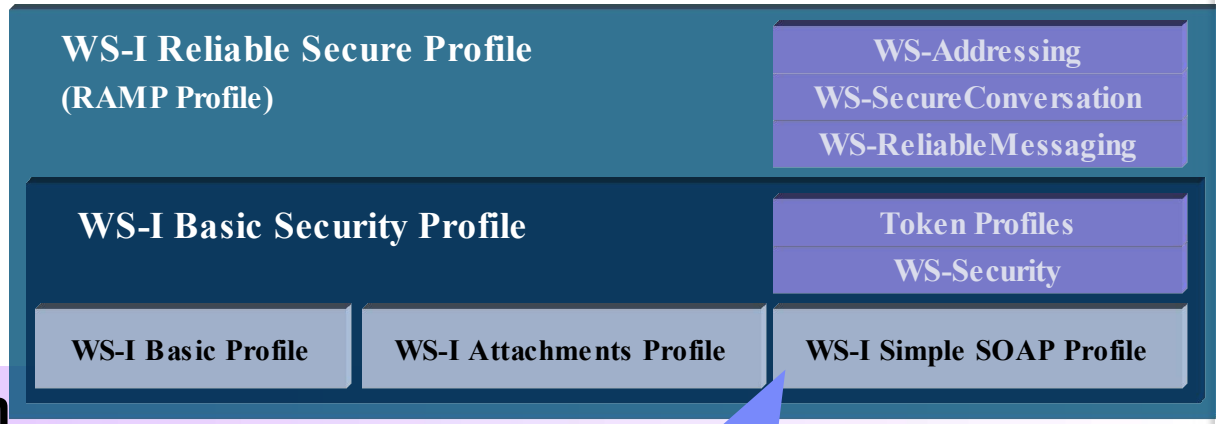
```
@Entity
@Table(name="EMP")
@Inheritance(strategy=JOINED)
public abstract class Employee {
    @Id protected Integer empId;
    @Version protected Integer version;
    @ManyToOne protected Address address;
}
```

```
@Entity
@Table(name="FT_EMP")
@DiscriminatorValue("FT")
@PrimaryKeyJoinColumn(name="FT_EMPID")
public class FullTimeEmployee extends Employee {
    // Inherit empId, but mapped in this class to FT_EMP.FT_EMPID
    // Inherit version mapped to EMP.VERSION
    // Inherit address mapped to EMP.ADDRESS foreign key
    // Defaults to FT_EMP.SALARY
    protected Integer salary;
}
```

Goal: Simplify Web Services Development

You want to…

- Interoperable, Reliable Web services
- Easy-to-implement
- Consumable and Extensible



IBM Solution

- JCP-based programming model
  - JAX-WS 2.1,
  - JAXB 2.1
  - SAAJ 1.3
  - StAX 1.0

Policy Sets allow these to be specified administratively

Simplifying Web Services Development

With JAX-WS WAS v7 improves and simplifies Java Web Services .

- JAX-WS is a follow on to the JAX-RPC 1.1 specification
  - The evolution of JAX-RPC
  - More than JAX-RPC 2.0
- JAX-RPC defined conventions for developing remote procedure call (RPC) based Web Services
  - JAX-WS deals with both RPC and message based services
- Core specification for a newly redesigned API stack for Web services
  - This “Integrated Stack” includes JAX-WS 2.0, JAXB 2.0, and SAAJ 1.3

Leveraging Benefits of SOA Programming Model

WAS V7 Supports Most Important Web Services for SOA Traction …

- WS-I
  - Basic Profile 1.2, 2.0, Reliable Secure Profile 1.0, Basic Security Profile 1.1
- OASIS
  - WS-ReliableMessaging (WS-RM), WS-SecureExchange (WS-Trust/WS-SecureConversation), WS-DistributedManagement (WSDM), WS-Policy
- W3C
  - SOAP 1.2, MTOM, XOP, WS-Security 1.1, WS-Addressing (WSDL)

Simplifying Web Services Development

With JAX-WS WAS v7 improves and simplifies Java Web Services .

- API for developing Web services with Java™ and XML
- A JAX-WS implementation integrated with the Web services Engine
- Support for Web services annotations
  - Incorporates JSR-181 and defines Java SE 5 annotations
- Tools for creating portable artifacts for the client
  - WSImport for top down development
    - Create a Web service form a WSDL
    - JAX-WS equivalent of WSDL2Java
  - WSGen can be used for bottom up development
    - Create a Web service from Java code
    - JAX-WS equivalent of Java2WSDL

Simplifying Web Services Development

With JAX-WS WAS v7 simplifies Web Services code using annotations

JAX-RPC 1.1 Code

```
public interface StockQuote extends
    Remote {
  public float getQuote(String sym)
    throws RemoteException;
}
public class QuoteBean implements {
    public float getQuote(String sym) { …
    }
}
```

JAX-WS 2.0 Code

```
@WebService public interface StockQuote
    {
   public float getQuote(String sym);
}
@WebService public class QuoteBean
    implements StockQuote {
    @WebMethod
    public float getQuote(String sym)
     { ... }
}
```

Web 2.0 support in WAS 7

Web 2.0 Feature Pack highlights include:

- Web 2.0-to-SOA connectivity for enabling connectivity from AJAX clients and mashups to external Web services, internal SOA services, and Java EE assets. This connectivity also extends enterprise data to customers and partners through Atom Syndication Format (ATOM) and Really Simple Syndication (RSS) Web feeds.
- AJAX Messaging for connecting AJAX clients to real-time updated data such as stock quotes or instant messaging.
- AJAX Development Toolkit for WebSphere Application Server based on Dojo with IBM extensions.

Web 2.0 support is enabled by the WebSphere Application Server Feature Pack for Web 2.0,an optional product extension for WebSphere Applications Server V7.0.

Leveraging Benefits of SOA Programming Model

Spring Certified WAS as a Deployment Platform

- The Spring model offers ease of development, like EJB3. Spring IoC container provides some abstraction between Java SE/EE – good for unit test.
- Significant collaboration between IBM® and Interface21 in 2007 tested Spring with WAS on all Base WAS platforms and under load.
  - Framework integration points added to WAS and exploited by Spring 2.5
    - Early availability of these delivered in 6.1.0.9 and 6.0.2.19.
  - Spring certified WAS as a deployment platform for the Spring framework
    - http://www.springsource.com/pressreleases/2007/ibmwebsphere062007
  - WAS/Spring integration testing focused on core Spring capabilities.
  - Spring distribution is not shipped with WAS, nor does WAS provide support for the Spring framework itself.
- Recommended best practices when using Spring with WAS: http://www-128.ibm.com/developerworks/websphere/techjournal/0609_alcott/0609_alcott.html

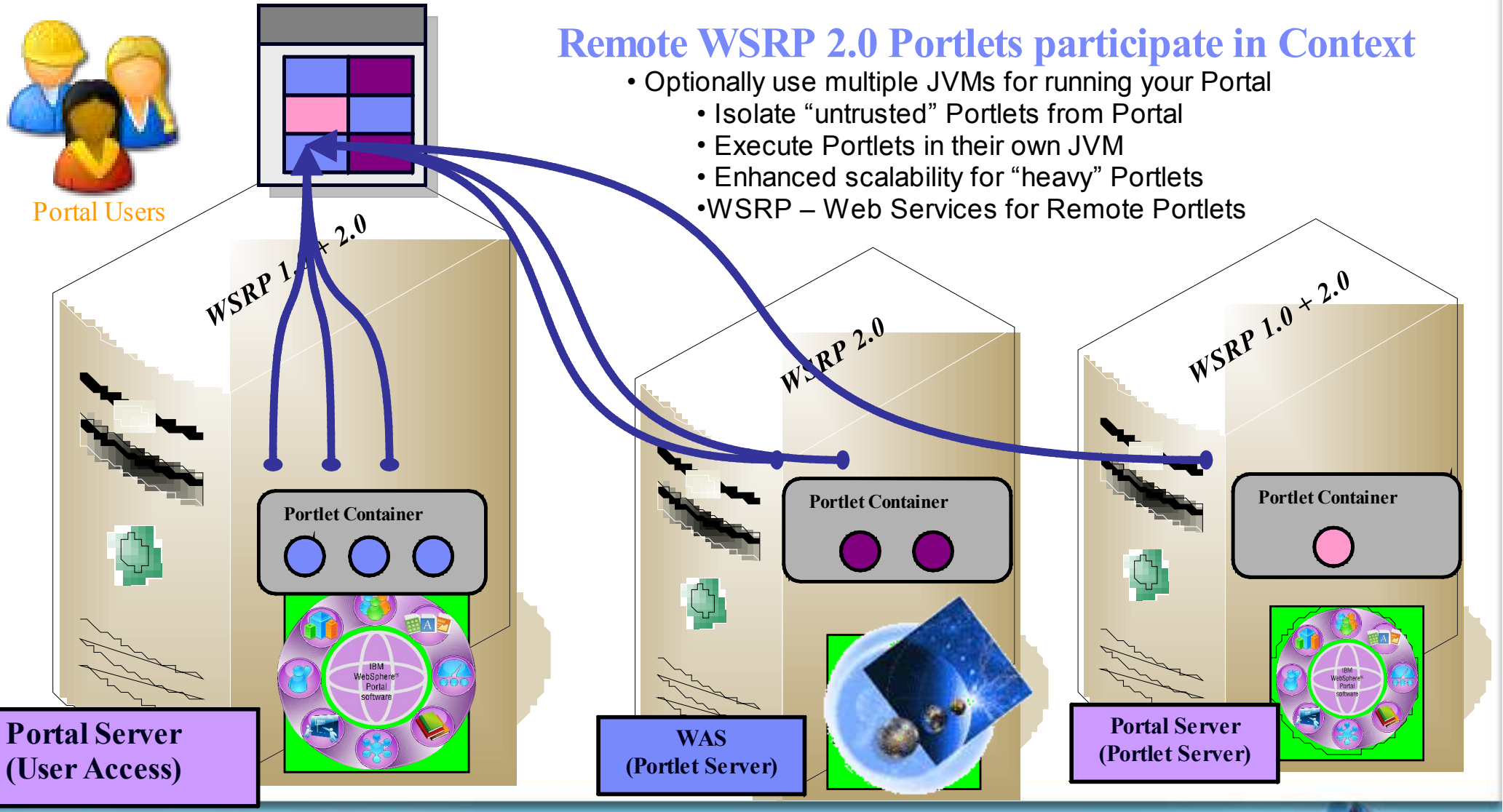
Leveraging Benefits of SOA Programming Model

WAS V7 Java Standard Edition (SE) Support Brings Productivity …

- Developer productivity
  - Java compiler API – the ability to invoke the compiler from with the Java Virtual Machine (JVM)
  - Java scripting support – gives scripts the ability to access APIs in the JVM
- Upgrades to existing components
  - JMX – MBean event generation at thresholds
- Added components
  - Web services standards: JAX-WS, JAXB, StAX
  - Java Data Base Connectivity (JDBC) 4.0

Leveraging Benefits of SOA Programming Model

WAS V7 Strengthens the Portlet Programming Model



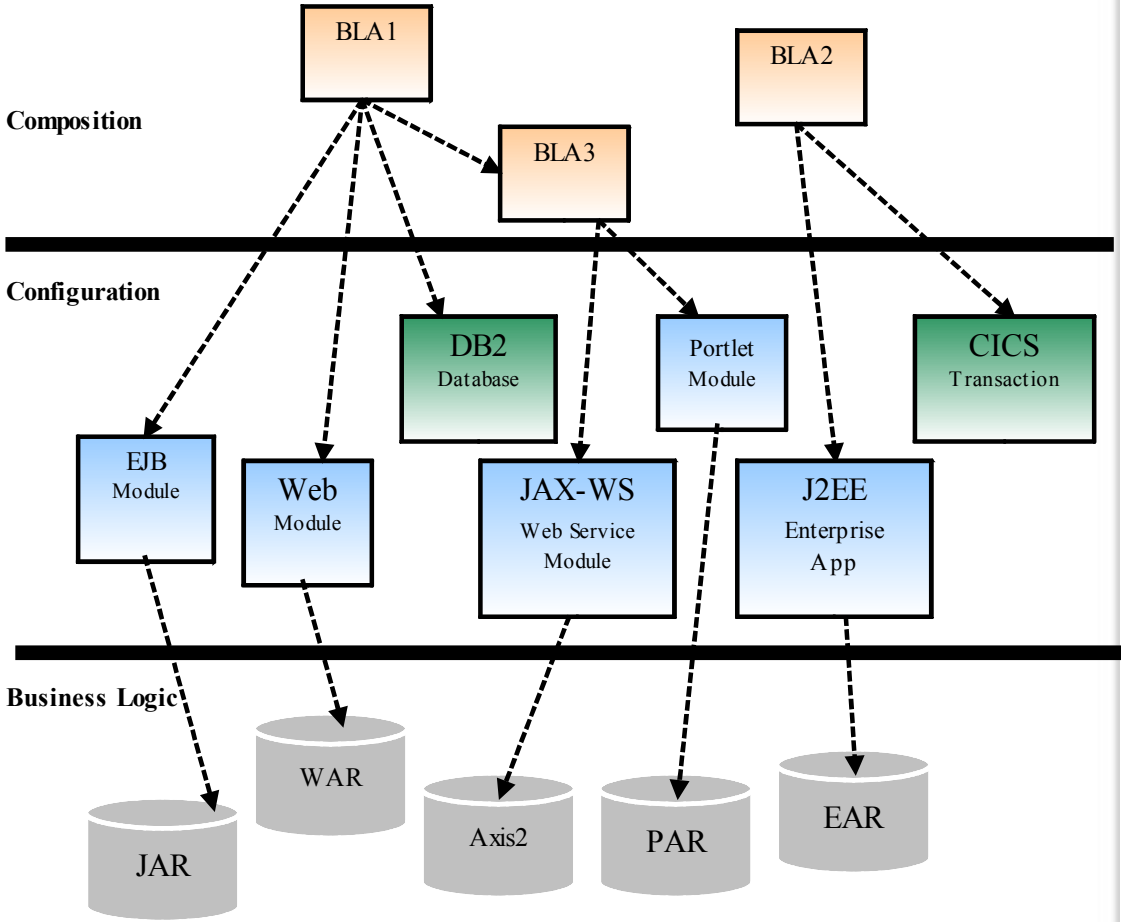
Remote WSRP 2.0 Portlets participate in Context

- Optionally use multiple JVMs for running your Portal
  - Isolate “untrusted” Portlets from Portal
  - Execute Portlets in their own JVM
  - Enhanced scalability for “heavy” Portlets
  - WSRP – Web Services for Remote Portlets

Using Technology Innovation to Meet Evolving Needs

WAS V7 Expands Support through Business Level Applications

- Expanding the notion of “Application” beyond J2EE
- Extensible deployment logic framework
- Supports more than Application Server deployment target runtimes
  - for example: Proxy Server, Web Server, CE, files, etc.
- Full lifecycle management of applications
  - Install, distribute, activate, monitor, update, remove



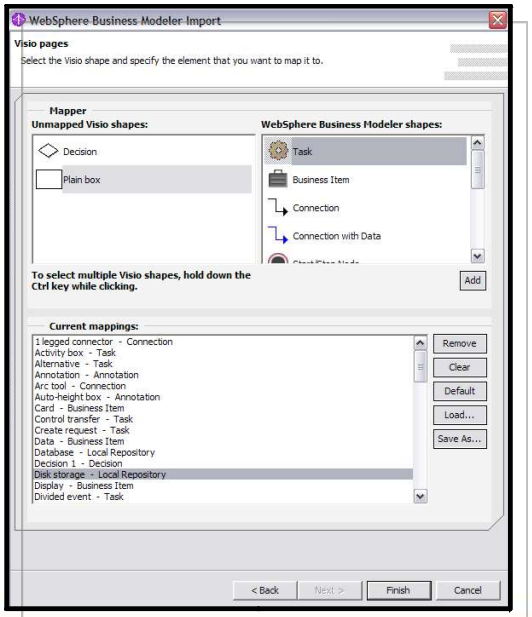
Goal: Provide Powerful, but Simplified Rational Application Developer (RAD)

You want to…

- Build and deploy flexible, agile applications/ services
- Take advantage of modern IDEs for rapid development/deployment
- Leverage programming standards to enable service composition

IBM Solution

- Rapid Assembly and Deployment Module for WebSphere
- Advanced tooling
  - J2EE 1.4 support
  - Java EE 5 support
  - Java EE 5 XML form-based DD editors
  - SIP tools
- Visual editing
- Adapters for simplified, enhanced integration



Enabling fast, efficient development of critical applications and services

Providing Powerful, but Simplified Rational Application Developer

RAD for WebSphere Improves Programmer Productivity

- ***IBM Rational® Application Developer for WebSphere*** is generally available at the same time as WAS V7.
- A subset of *IBM Rational Application Developer for WebSphere* is included in WAS V7 and called ***IBM Rational Application Developer Assembly and Deploy Module for WebSphere***.
- *IBM Rational Application Developer Assembly and Deploy Module for WebSphere* is fully licensed and supported with the WAS license.
- Remaining content is optionally installable as a 60-day trial and becomes perpetually licensed on purchase of RAD for WebSphere, or other Rational priced products that include RAD.

Providing Powerful, but Simplified RAD

RAD for WebSphere Improves Programmer Productivity

RAD for WebSphere

Programming model tools:
- EJB3 productivity features
- Domain modeling (visual edit)
  - Java, EJB, XSD, WSDL, Data
- Web Development (visual edit)
- Web Services productivity features
- Portlet JSR
- Relational Data Tools

Miscellaneous:
- XML productivity features
- Extra Debuggers (XSLT, stored procedures…)
- WAS Test servers: v6.0, v6.1, V7.0
- WAS n-2 support
- J2C
  - CICS® and IMS Adapters

Licensed but supplied separately:
  - RequisitePro integration
  - RUP
  - IBM Support Assistant
- More…

RAD Assembly & Deploy
- Profile Management tools
- Jython Tools
- J2EE 1.4 (same level as AST 6.1)
- JEE5 XML-form based DD editors
- JEE5 application support
- WAS 7 support only
- WAS debug extensions
- Application Deployment Support (WAS 7.0)
- SIP

EIS Adapters
- SAP, Siebel, JDE, Oracle®, PeopleSoft®

Portal (Lotus®)
- Portal Tools

SCA (Post V7.5)
- SCA 1 (Public)

धन्यवाद
Hindi

多謝
Traditional Chinese

ขอบคุณ
Thai

Спасибо
Russian

Gracias
Spanish

Thank You
English

شكرا
Arabic

Merci
French

Obrigado
Brazilian Portuguese

Grazie
Italian

多谢
Simplified Chinese

Dan
German
ke

நன்றி
Tamil

Tamil

ありがとうございました
Japanese

감사합니다
Korean


Thank
You