

Comparing JavaBeans and OSGi

Towards an Integration of Two Complementary
Component Models

HUMBERTO CERVANTES

JEAN-MARIE FAVRE

09/02



Who I Am

- ◆ Humberto Cervantes
- ◆ 3d year PhD at Adèle team, LSR, Grenoble
 - ◆ www-adele.imag.fr
- ◆ Collaborators:
 - ◆ Jean-Marie Favre : PhD supervisor
 - ◆ Jacky Estublier : Team director
 - ◆ Richard S. Hall : Visitor
- ◆ Work area:
 - ◆ Applying Component-Based Technologies to build Software Environments



Comparing OSGi and JavaBeans

- Two component technologies:
 - JavaBeans: visual assembly of applications
 - Open Services Gateway Initiative (OSGi): service deployment in home gateways.
 - Richard S. Hall, implementor of OSCAR
- How do these two technologies support a set features that characterize component models?
- Since some aspects are complementary, how can they be integrated?



Points of Comparison

- ◆ Components :
 - ◆ Have a clear and explicit boundary
 - ◆ Well specified interface and explicit dependencies
 - ◆ Can be customized
 - ◆ Can be assembled
 - ◆ Are reusable
 - ◆ Are units of substitution
 - ◆ Are units of delivery and deployment
 - ◆ Have certified properties*
- ◆ Component Framework
 - ◆ Runtime services to support the model

JAVABEANS



JavaBeans

- ◆ Sun's technology oriented towards visual assembly of applications
 - ◆ Introduced in 1996
 - ◆ Client-side
 - ◆ Relatively simple compared to EJB



JavaBeans in a Builder Tool

Be:am

The screenshot shows the 'Bean Builder Tutorial' interface. On the left, a dialog box titled 'Add:' contains a text field with 'Entry #4' and a list with 'Entry #1', 'Entry #2', and 'Entry #3'. Below the list is a 'Remove All' button. The main area is a design canvas with a toolbar and a 'Design Mode' checkbox. A 'Selected instance' window is open, showing a table of properties for a 'javax.swing.plaf.metal.Meta[TextFieldUI'.

Property	Value
UI	javax.swing.plaf.metal.Meta[TextFieldUI
action	
actionMap	javax.swing.ActionMap
alignmentX	0.5
alignmentY	0.5
background	255,255,255
border	
bounds	X: 222 Y: 36 Width: 150 Height: 24

Properties for: javax.swing.JTextField

Selected instance properties

Assembly canvas

Selected instance

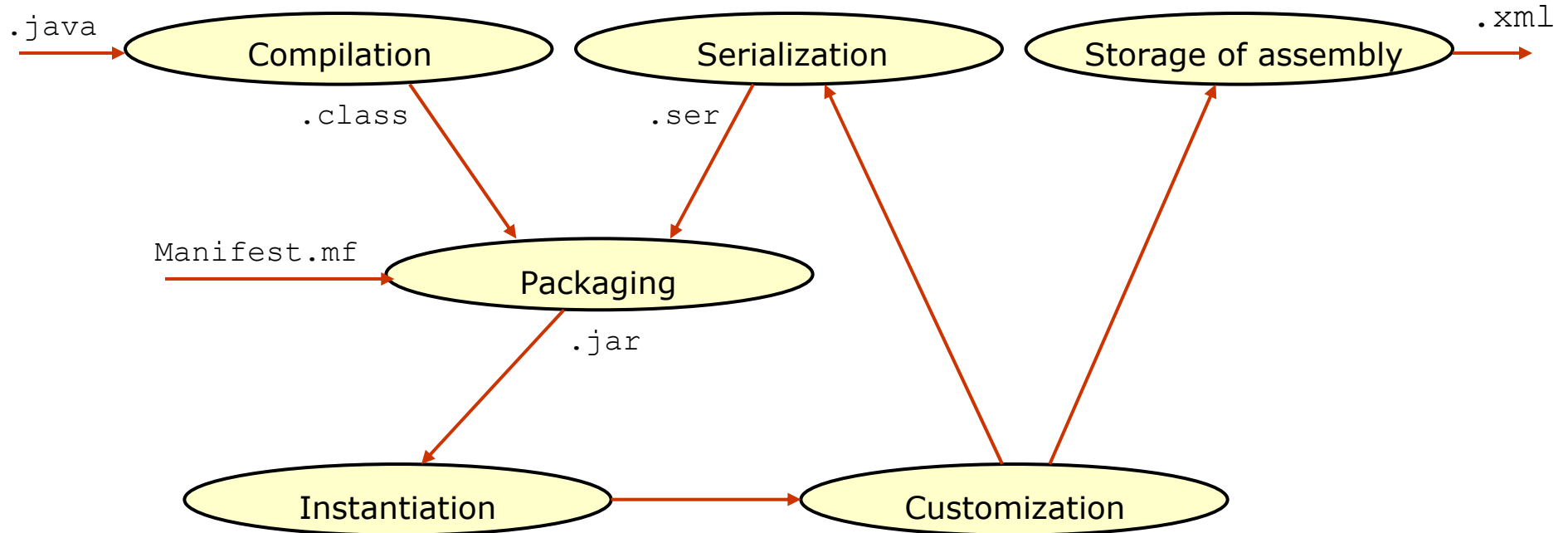


What is a JavaBean ?

- ◆ JavaBean: Standard Java class that follows certain naming conventions
 - ◆ Express properties, receptacles, events.
- ◆ JAR: Package file for JavaBeans
 - ◆ Along with an optional class that provides information, the `BeanInfo` class
- ◆ Difference between design-time and run-time



Life-cycle

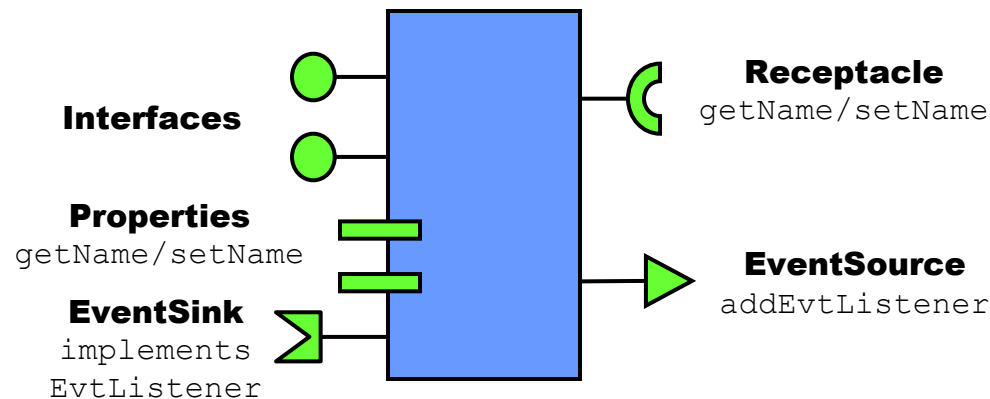


- Serialized beans become *prototypes*
- Instantiation occurs either inside builder tools or in 'real' applications.



Boundary and Dependencies

- ◆ Interface is well defined



- ◆ Dependencies
 - ◆ With respect to the runtime and imported packages: not explicit
 - ◆ Towards other beans, classes or resources: can be described if they occur in the same JAR file.



Customization

- ◆ Properties are persistent attributes
- ◆ Changing value of properties
 - ◆ Build-time activity
 - ◆ Calls to setter methods
 - ◆ Can be done at the component or instance level



Assembly

- ◆ Assembly: graph of inter-connected instances.
 - ◆ References are transient
- ◆ An assembly cannot directly become itself a Bean.
 - ◆ Storage and retrieval of graphs is available.
 - ◆ XML Encoder / Decoder



Reusability and Substitution

- ◆ JavaBeans promote reuse
 - ◆ However, all the dependencies of the Bean must be known.
- ◆ A JavaBean can be substituted by another if it provides the exact same interfaces.



Delivery and Deployment

- ◆ JAR file is the unit of delivery and deployment for JavaBeans.
 - ◆ A manifest file inside the JAR provides simple information about the JavaBeans it contains
- ◆ There is no component registry, Beans are loaded from the classpath
 - ◆ Only one version can be available

OSGI



OSGi

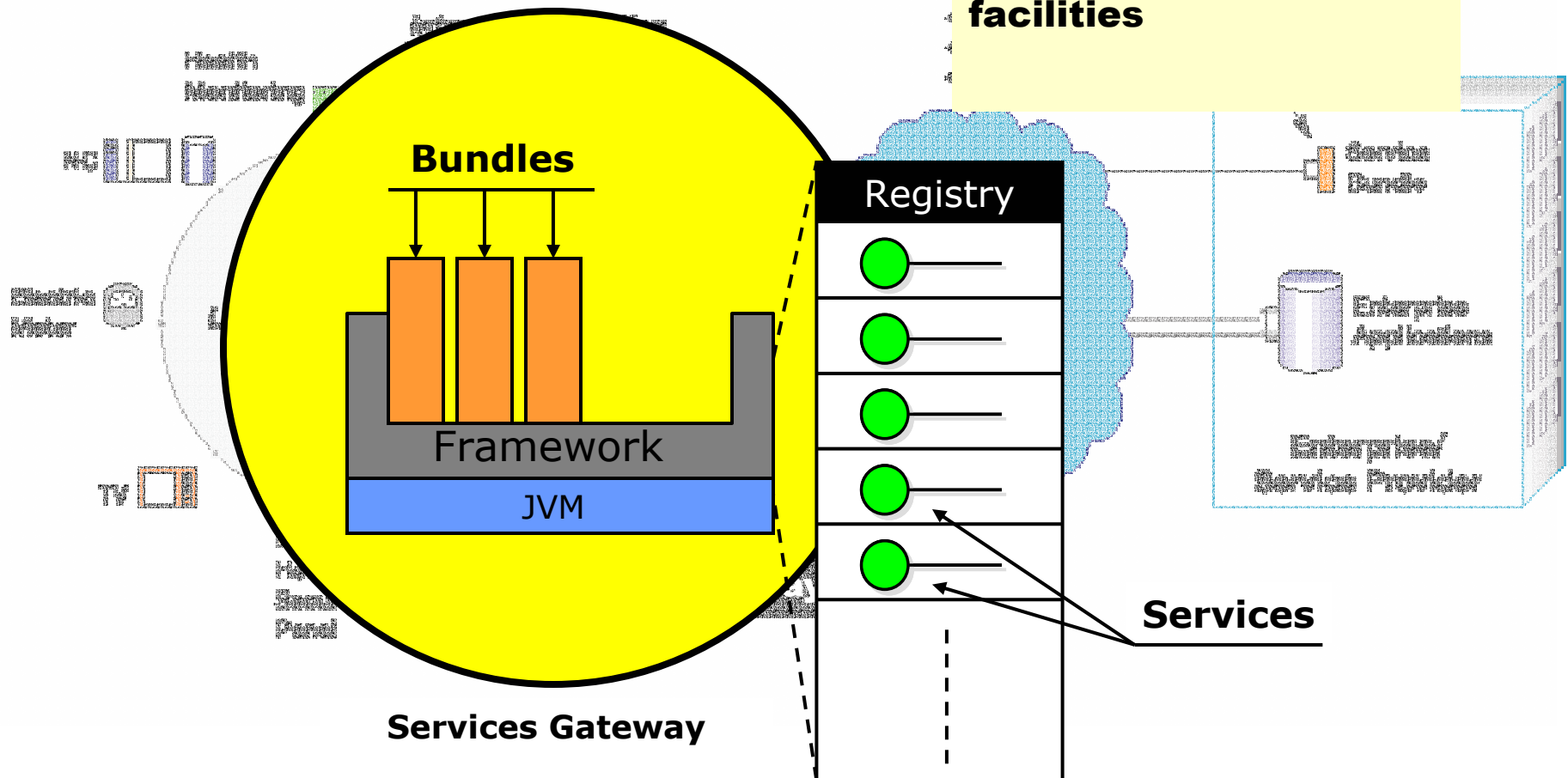
- ◆ Open Services Gateway Initiative
 - ◆ Founded in 1999 (more than 70 companies)
 - ◆ Goal is to define a set of Java APIs that specify a services gateway architecture .
- ◆ OSGi specification:
 - ◆ 1st rel: May 2000, 2d rel: October 2001
 - ◆ A service platform that includes:
 - ◆ A minimal component-like model
 - ◆ A lightweight framework to manage the components.



Global View

OSGi Framework :

- Service registry and request facilities
- Bundle management facilities





Services and Bundles

- ◆ Service:
 - ◆ A Java interface
 - ◆ Registered in the framework along with an object that implements it and a set of properties of type name,value
- ◆ Bundle:
 - ◆ Delivery and deployment units (JAR file)
 - ◆ Contains services: interfaces, implementations and resources
 - ◆ Manifest file contains meta-data



Boundary and Dependencies

- ◆ Services have a clearly defined interface
 - ◆ No dependencies at service level
- ◆ Bundle
 - ◆ Bundle-to-package dependencies
 - ◆ Code inside the bundle imports packages exported by other bundles
 - ◆ Managed by the framework
 - ◆ Bundle-to-Service dependencies
 - ◆ Code inside the bundle uses services
 - ◆ Unmanaged but can be described
 - ◆ Runtime dependencies
 - ◆ Unmanaged but can be described



Customization

- ◆ Services that implement a particular interface can be customized.
 - ◆ Through getter/setter methods
 - ◆ Services are shared, a change in the state affects all of the clients.



Assembly

- ◆ No concept of assembly
- ◆ Applications are dynamically created by bundles that connect to services
- ◆ A service request includes the name of the service plus a filter

