Introduction to OSGI

Kamal Govindraj
SpringPeople Technologies

About me

- Programming for 13 Years
- Co Founder / Architect @ TenXperts
- Trainer / Consultant @ Spring People
- Building enterprise apps using open source frameworks (Spring / Hibernate / GWT / Grails / jBPM)
- Key contributor to InfraRED & Grails jBPM plugin

About SpringPeople

- Master Certified training partner of SpringSource in India
- Conduct public & onsite trainings
  - Core Spring
  - Enterprise Integration with Spring
  - OSGI & dmServer
  - ...
- Consulting
- More info @ http://springpeople.com

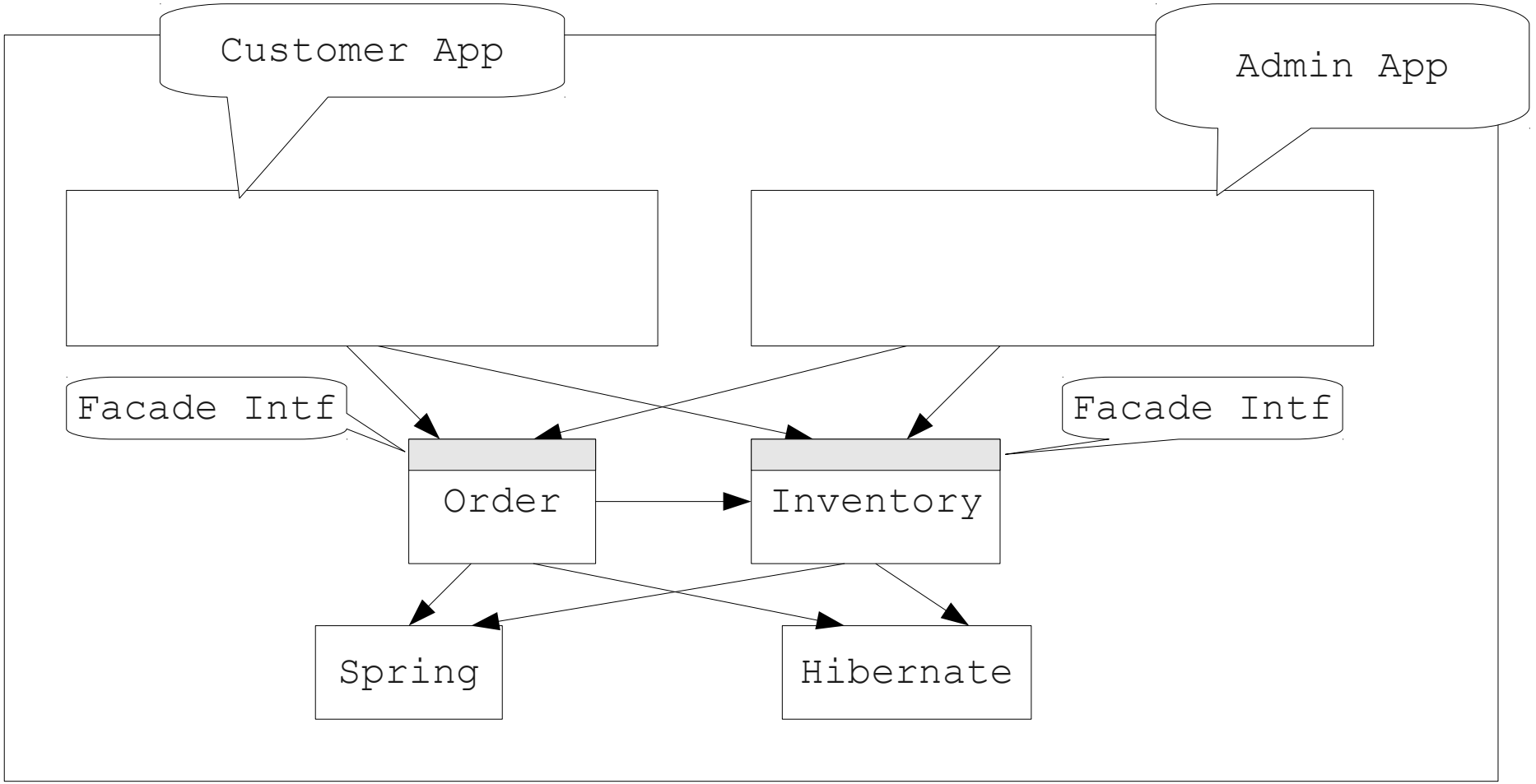
Agenda

- Need for OSGI?
- Versioning, Bundle Lifecycle
- Services
- Spring DM
- Demo of building a simple OSGI enabled app.
- Limitations

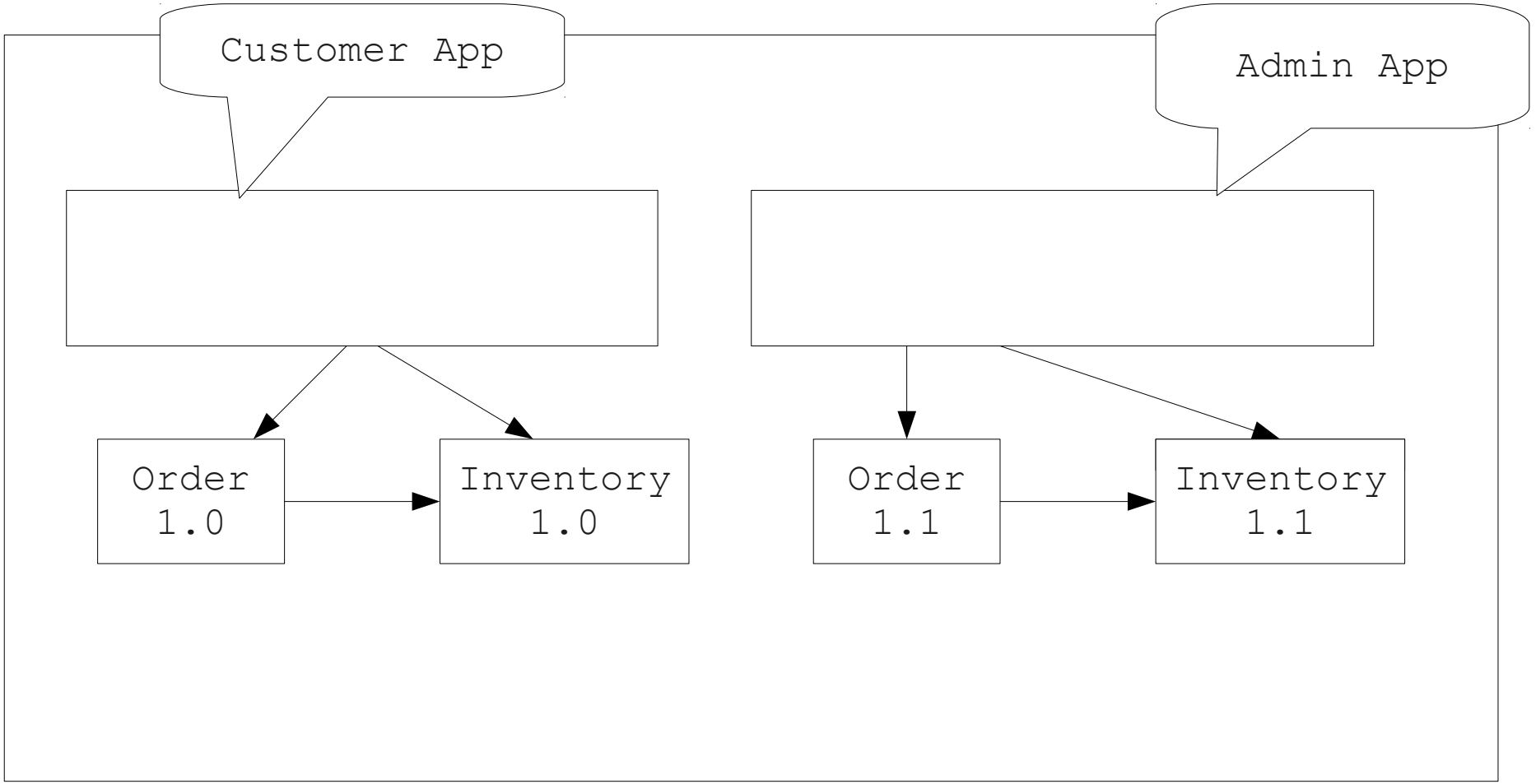
Why OSGI?

- Java Platform lacks good support for modularity (especially at Runtime)
  - Linear classpath
  - War / Ear as deployment units are too monolithic / coarse grained
  - Complex to share common components / modules across multiple applications
  - No support for multiple versions of class / component in the same vm
  - No support for hot deploying modules
  - No support for light weight service discovery / late binding

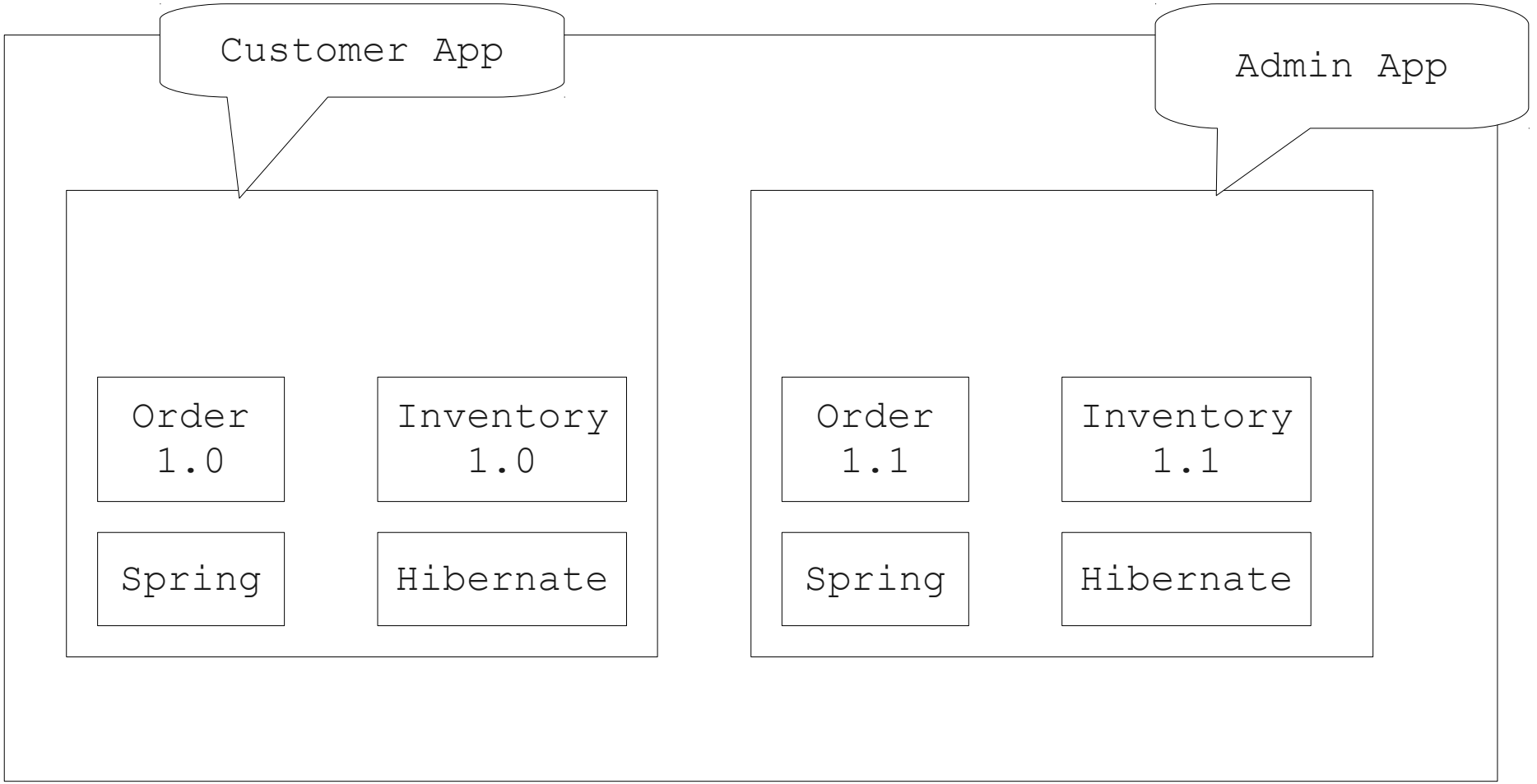
What we want


Customer App
Admin App
Facade Intf
Facade Intf
Order
Inventory
Spring
Hibernate

What we want


Customer App
Admin App
Order
1.0
Inventory
1.0
Order
1.1
Inventory
1.1

What is possible today



How does OSGI Help?

- Partition a system into a number of modules – “bundles”
- Dynamic: bundles can be installed, started, stopped, uninstalled and updated ... at runtime!
- Strict visibility rules
- Resolution process satisfies dependencies of a module
- Versioning
  - Packages & Services
  - Deploy multiple versions of a module

Bundle

- The fundamental unit of deployment and modularity in OSGi
- Just a JAR file with additional entries in META-INF/MANIFEST.MF
- Common manifest headers:
  - Bundle-SymbolicName
  - Bundle-Version
  - Bundle-Name
  - Bundle-ManifestVersion
  - Export-Package
  - Import-Package

Dependency constraints

Export-Package: com.tenxperts.order;

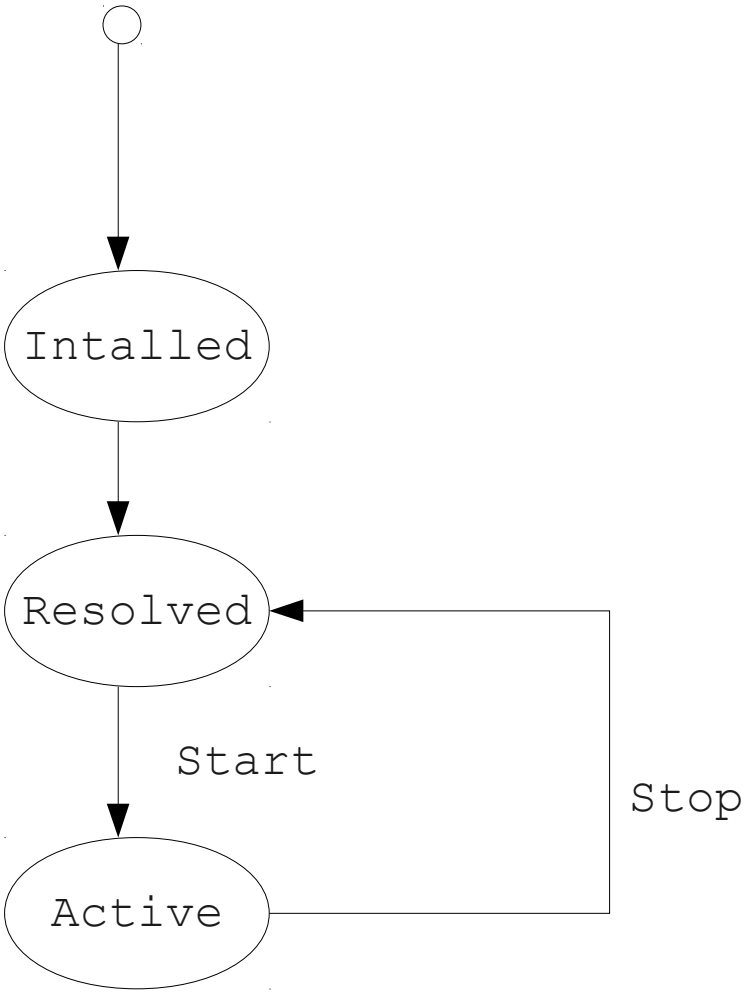
Export-Package: com.tenxperts.order;version=”1.0.0”

Import-Package: com.tenxperts.inventory;version=”1.0.0”

Import-Package: com.tenxperts.inventory;version=”[1.0.0,2.0.0)”

Import-Package: com.tenxperts.inventory;version=”[1.0.0,1.0.0]”

Bundle life cycle



Demo – building / deploying a simple bundle

Bundles register service


```
ServiceRegistration inventoryServiceRegistration =
context.registerService(
        InventoryService.class.getName(),
        new InventoryServiceImpl(),null);


    inventoryServiceRegistration.unregister();
```

Bundles can lookup services


```
ServiceReference inventoryServiceReference =
context.getServiceReference(
    InventoryService.class.getName());

InventoryService inventoryService =
    (InventoryService)context.getService(
                    inventoryServiceReference);
```

Service Demo

Spring DM

- Makes it easier to export / consume services
- Handle all the plumbing to deal with service dynamics
- Creates an spring application context
- Looks for configuration files under META-INF/spring folder

Spring DM demo

Conclusion

- OSGI bring dynamic modularity to the java platform
- SOA in a vm
- Mature standard – enterprise adoption still in early stage
- Many exists frameworks / techniques don't work well in OSGI env. Work arounds exist
- Don't use it unless your application really needs – desgin your app to take advantage of OSGI at later stage