

Building modular applications

11, 12 Dec. 2009

Pune, India

IndicThreads.com Conference On Java Technology

Kamal Govindraj

TenXperts Technologies

About Me

- Programming for 13 Years
- Architect @ TenXperts Technologies
- Trainer / Consultant @ SpringPeople Technologies
- Enterprise applications leveraging open source frameworks (spring,hibernate, gwt,jbpm..)
- Key contributor to InfraRED & Grails jBPM plugin (open source)



Agenda

- Importance of Moudlarity
- How to?
- Challenges
- Tools & best practices
- OSGI

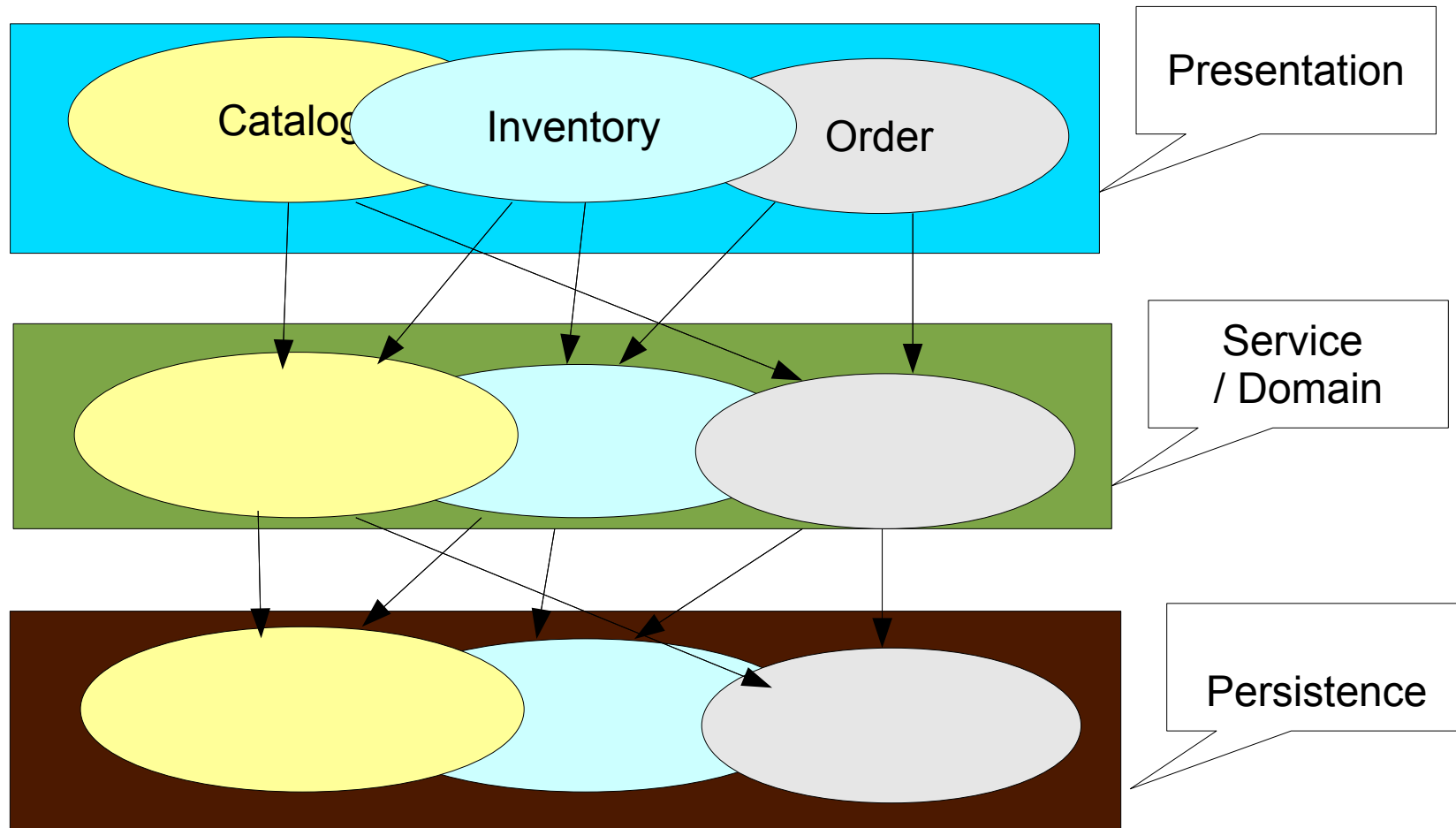


Modularity

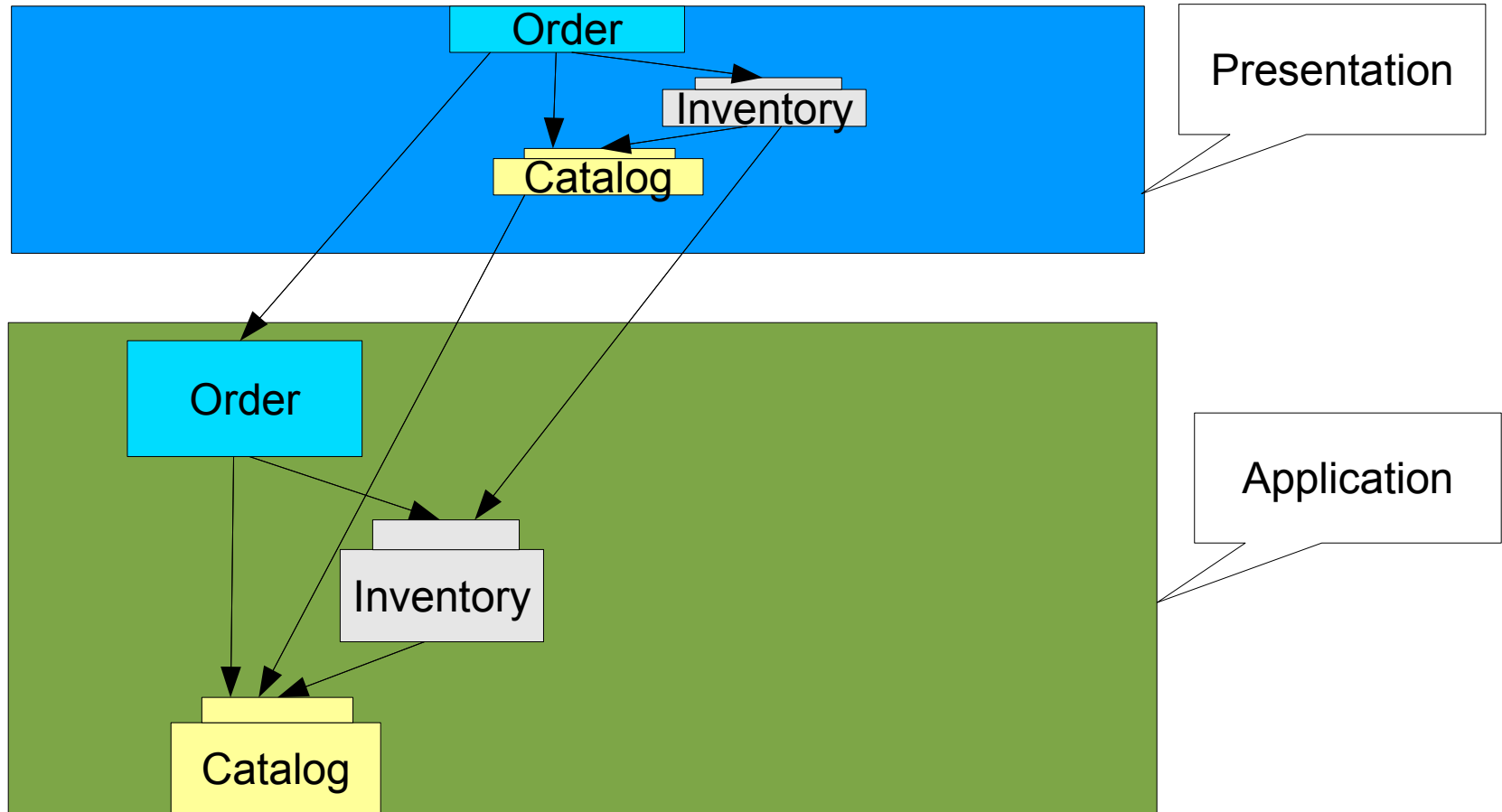
- Break application into smaller modules
- Interaction between module via interfaces
- Helps in dealing with complexity
- Break development into smaller teams
- Reuse
- Easier to maintain



Horizontal / Technology



Vertical / Functional



Vertical is better

- Scales better
 - Team size
 - Feature
- Separate concerns along the lines of business functionality
- Easier to maintain



Physical Vs Logical

- Logical separation provides most of the benefits
- Physical separation issues
 - Performance overheads
 - Reduces reliability
 - Increased Complexity



Package guidelines

▼  tenxperts.ezshop.catalog

Visible to other modules

▶  CatalogService.java

▼  tenxperts.ezshop.catalog.domain

Visible to other modules

▶  CatalogItem.java

▶  Category.java

▼  tenxperts.ezshop.catalog.internal

Private classes /
Not visible outside
this module

▶  CatalogRepository.java

▶  CatalogServiceImpl.java

▶  HibernateCatalogRepository.java



Challenges

- Builds become more complex
- Changes across module boundaries are harder
- Integration issue
- Wiring the system together

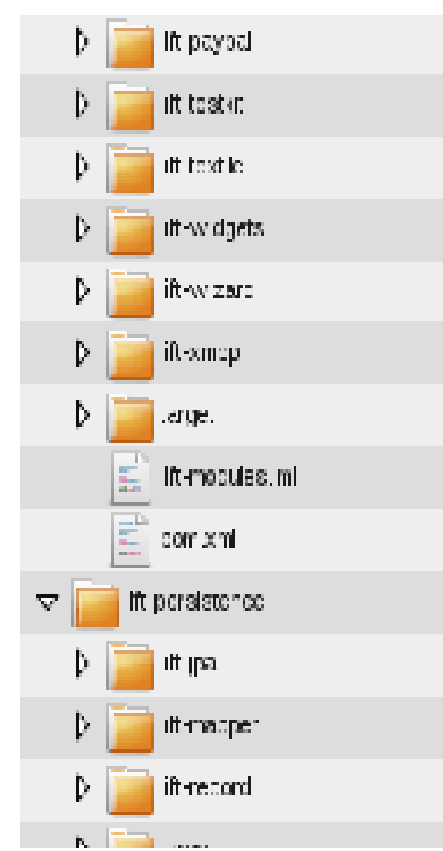
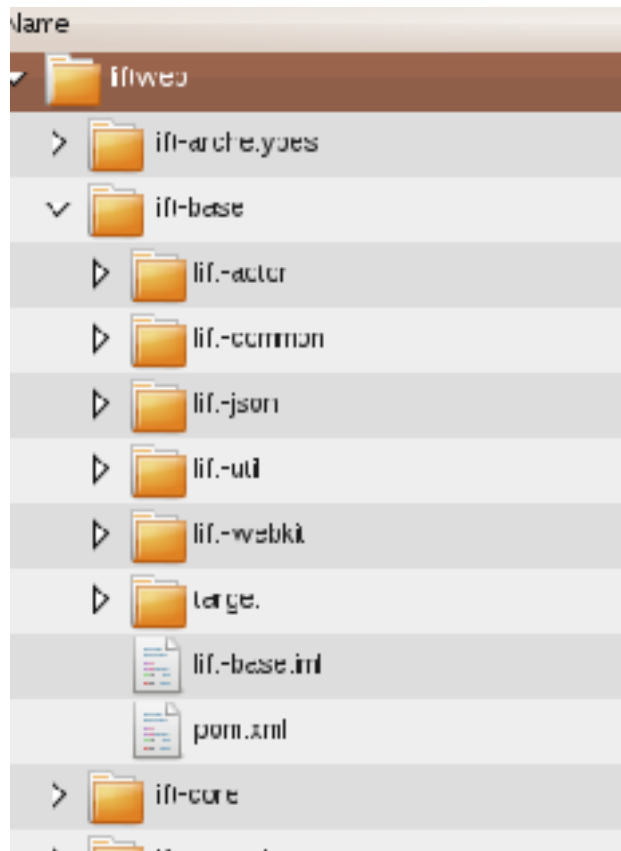


Tools & Frameworks

- Maven
 - Simplifies multi module builds
 - Declarative dependency
 - Transitive dependency handling
 - Versioned



Maven in action



Dependency Injection

- Takes care of wiring together components from different modules
- The modules are loosely coupled depending only on the published interfaces
- Spring Framework / Google Guice / EJB 3.0 ..



Java module limitation

- Jar is the basic construct for defining modules
- Only a compile time construct
 - Modules are not preserved at run time
- No way to enforce strict separation
- Difficult to have multiple versions of same module in a vm
- Not easy to share modules across multiple applications



Runtime modularity

- OSGI
 - A mature specification & platform
 - Eclipse is built on OSGI
 - So are many of the application servers
 - Multiple implementations (Equinox, Felxi, Knopflerfish)
- JSR 294 – Simple module system for Java



OSGI

- A module is a jar with metadata (MANIFEST file)
 - exported packages
 - Imported packages with versions
 - Other module dependencies with versions
- OSGI container
 - Manages module lifecycle
 - Resolve dependencies
 - Enforces strict separation

OSGI != modular

- OSGI can make a existing modular application work better
- Breaking up a system into well defined modules and maintaining it that way is the harder part
- Tools can help – but ...



Questions ?



Thank You.

