

# Application Architectures - Where We Have Been, Where We Are Going



Srini Penchikala

Detroit Java User Group

December 17, 2008

# Speaker Bio

Srini Penchikala

Enterprise Architect, Flagstar Bank

- Writer: InfoQ, TSS, ONJava, DevX Java, java.net
- Using Java/JEE since 2000
- SOA/Web Services since 2006
- PowerPoint since September 2008

# Goals for this Presentation

- Give an:
  - Overview of application architectures of the past, present and future
  - Update on the most important trends affecting Enterprise Java development

# Contents

- Introduction
- Current Architecture
- Future Architecture
  - DI, AOP, Annotations
  - Spring
  - Java as a platform
  - OSGi
  - Cloud Computing
  - Database Layer
- What's next for J2EE/JEE?

# Presentation Format

- Interactive
- Demos
- Duration: ~60 minutes

# Current Architecture - J2EE Way

- JSP/Servlet/Struts
- EJB 2.x
  - Session Beans with Business Logic
- JMS
  - Unit testing constraints
- POJO's as a last resort
- EAR
- J2EE application servers

# Current Architecture Constraints

- **Business:**
  - Not enough communication between Business Units and IT teams.
  - Architects/Developers are thinking only about infrastructure.
  - We have lost real OOP
- **Technology:**
  - Unit testing constraints
  - EJB 2.x (deployment descriptors)
  - Transaction Management
  - J2EE (EAR/WAR)

# New Architecture - Back to Basics

- POJO's as first-class citizen components
  - Controller components
  - Facade classes (POJO's w/ Transactions managed via Annotations)
  - Domain Classes (with State and Behavior)
  - Message Driven POJO's (JMS)
- Deployed as WAR files (no need for EAR)
- Light-weight JEE containers

<b>Arch Layer</b>	<b>Current</b>	<b>New</b>
UI/Application	JSP/Servlet/Struts	POJO (Spring MVC)
Facade	EJB 2.x <ul style="list-style-type: none"><li>• Session Beans</li></ul>	POJO (EJB3, Spring)
Domain Model	Anemic	Rich (POJO based)
Deploy	EAR	WAR, JAR
Server	J2EE App Servers	OSGi container

# Design Recipe

- Object Oriented Programming (OOP)
- Dependency Injection (DI)
- Aspect-Oriented Programming (AOP)
- Annotations

# OOP Principles

<b>Domain Element</b>	<b>State/Behavior</b>
Entity Value Object Aggregate	State and Behavior
Data Transfer Object	State only
Service Repository	Behavior only

# Dependency Injection

- Decouple and manage the dependencies of the components in the application
- DI Frameworks:
  - Spring
  - Google Guice
  - Pico Container

# Demo

- Data Access Object (DAO) example

# Aspect-Oriented Programming

- Allows developers to add behavior to objects in a non-obtrusive manner through use of static and dynamic crosscutting.
- Main goal is to code cross-cutting concerns in separate modules and apply them in a declarative way.

# Demo

- Profiling example
- Architecture Rules

# Annotations

- Added in Java SE 5.0 as Java Metadata facility (JSR 175).
- Provide a way to add metadata to program elements.
- Defined by nearly every recent JSR standard.
- Also include a mechanism for adding custom annotations to the Java code\*.

# Annotations Examples

<b>Layer</b>	<b>Domain Element</b>	<b>Annotation</b>
Domain	Entity Value Object	@Entity (JPA), @Configurable
Domain	Repository	@Repository
Domain	Service	@Service
Application	Controller	@Controller
All	All	@Component

# Custom Annotations

- Implementation Options
  - Reflection
  - Annotation Processing Tool (APT)
  - Byte-code Instrumentation (Javassist)
  - Aspects/AOP (AspectJ and Spring AOP)

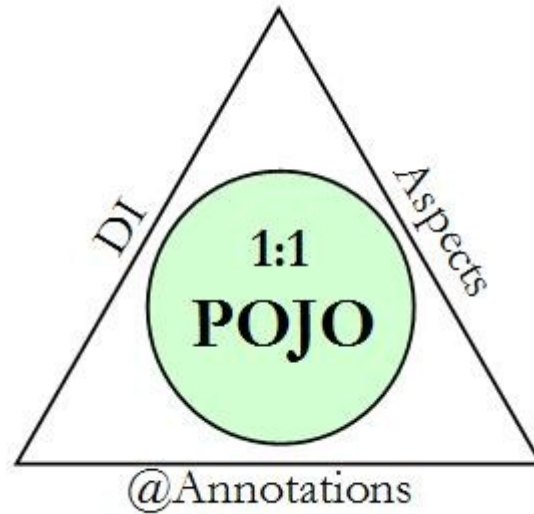
# Demo

- Caching example
- Requirement: To cache specific data (objects) using a custom Annotation
- Annotation: `@CacheEntity`

# Spring Framework

- Separation of concerns (business v. infrastructure)
- POJO Based Design
- Business domain first, only then infrastructure concerns (persistence & txn management)
- Agile Development, Testing, Refactoring & CI

# Spring Philosophy



# Spring Portfolio

- Spring Core
- Spring AOP
- Spring Security
- Spring MVC/Spring WebFlow

# Persistence & Txn Management

- JDBC
- JPA w/ Hibernate, OpenJPA, EclipseLink (TopLink)
- Spring JPA Utils & Data Source DI
- Spring JTA Support
  - Transactions are managed in Service classes using “@Transactional” annotation

# Web Services, Async Messaging & ESB's

- Web Services
  - REST
- Message Driven POJO's (MDP)
- AMQP standard
- Enterprise Service Bus (ESB)
  - Apache ServiceMix
  - Mule, JBossESB
  - Spring Integration Framework

# Java as a platform (not a language)

- Dynamic Languages (Groovy, JRuby, Scala)
- Domain Specific Languages (DSL's)
  - Internal
  - External

# Deployment

- OSGi v. Java Module System (JSR-277)
- Implementations:
  - Eclipse Equinox, Apache Felix, KnopflerFish
  - Spring DM Framework
- Light-weight & OSGi compatible containers
  - WebLogic 10.3
  - WebSphere 7
  - Tomcat 6
  - SpringSource dm Server

# Distributed Computing

- Parallel / Concurrent Programming
  - Concurrency package in Java SE 5
  - Fork/Join in the upcoming Java SE 7
- Cloud Computing
  - Virtualization
  - Amazon EC2, Gigaspaces
  - Terracotta
  - GridGain, JPPF

# Database Layer

- New Distributed Data Storage Frameworks:
  - Amazon S3
  - Big Table
  - Hypertable (High Performance, Scalable DB)
  - MapReduce
  - Hadoop
  - AtomServer (Publishing for Data Distribution)
  - Neo4j (Graph database)
  - CouchDB

# Other Trends

- Web 2.0 turning Internet into application platform
- AJAX and Rich Clients
- Rich Internet Applications (RIA)
  - Flex, Silverlight and JavaFX
- Rich Client Platform (RCP)
  - Eclipse RCP, Spring RCP
- Conversational Web Frameworks
  - Seam, Spring WebFlow
- Batch Frameworks
  - Spring Batch

# What's next for J2EE/JEE?

- Java EE 6 (JSR-316)
  - Profiles, Web Beans
- JPA 2.0 (JSR-317)
  - Criteria Expression Support
- EJB 3.1 (JSR-318)
  - Deploy EJB's in a WAR (no need for EAR's any more)
- Spring 3.0
  - REST/EJB 3.1 Support

# Conclusion

- Java EE 6 may keep Java EE relevant, but Java EE no longer shapes the future
- Light-weight frameworks will be used more widely.
- One of the key technologies that will shape the future is OSGi

# Resources

- Patterns of Enterprise Application Architecture, Martin Fowler
- Enterprise Integration Patterns, Gregor Hohpe
- Spring Framework
- Domain Driven Design [Website](#)
- [Domain Driven Design](#) by Eric Evans
- Domain Driven Design Quickly, InfoQ [mini-book](#)
- [Domain Driven Design & Development In Practice](#)
- [Can DDD be Adequately Implemented Without DI and AOP](#)

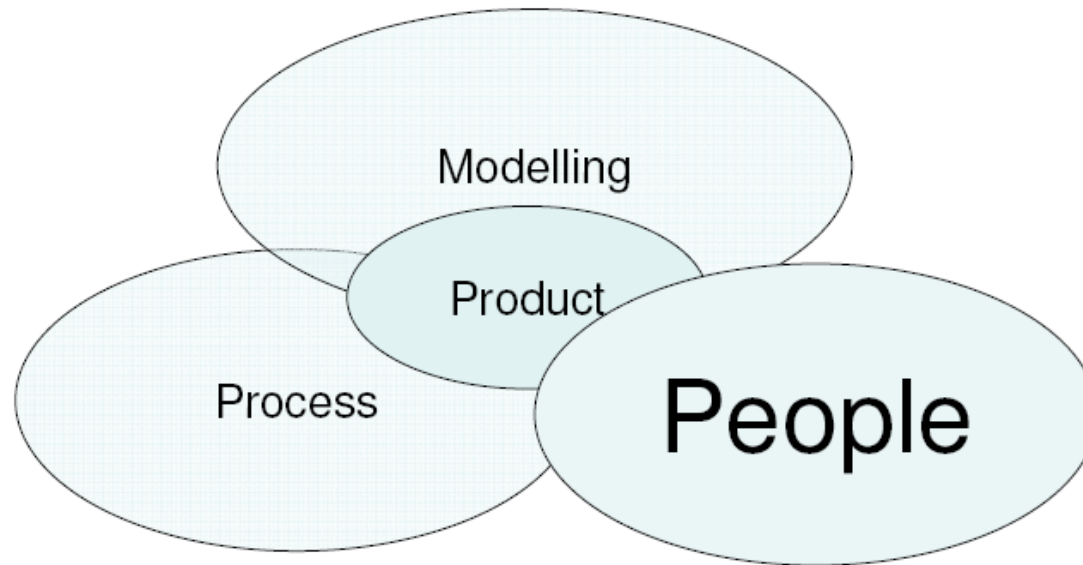
# Resources - 2

- AspectJ
- AspectJ Development Tools (AJDT), Eclipse Plugin

# Contact Information

- Domain-Driven Design and Enterprise Architecture articles on InfoQ.
- InfoQ website (<http://www.infoq.com>)
- E-Mail: [srinipenchikala@gmail.com](mailto:srinipenchikala@gmail.com)
- Blog: <http://srinip2007.blogspot.com>

# Questions



*Non-Linear First-Order Components*

*Alistair Cockburn*