Spring Framework 4

Hello!

I am Hiten Pratap Singh

You can find me at:

- hiten@nexthoughts.com
- http://github.com/hitenpratap/
- http://hprog99.wordpress.com/

Spring simplifies Java development

1.

What's Spring?

How did it simplify Java Enterprise development easier?

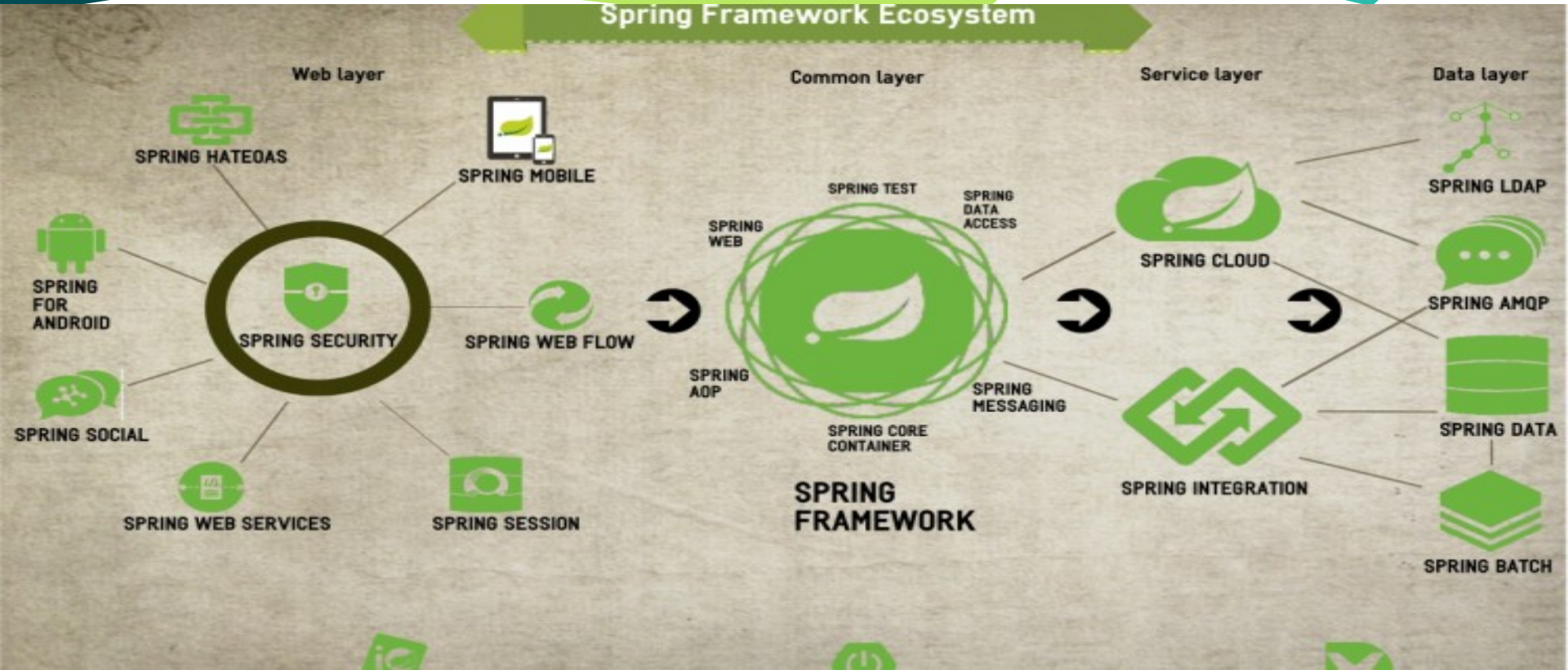
What's Spring?

Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, reusable code.

2.

Spring Framework Ecosystem

Spring Framework Ecosystem



3.
History

History

- First version was written by Rod Johnson.
- First released under the Apache 2.0 license in June 2003.
- Milestone releases in 2004 and 2005.
- Awards:
    1. Jolt Productivity Award
    2. JAX Innovation Award

4.
Goals

Goals

- Make JEE development easier.
- Make the common task easier.
- Promote good software development practice.
- Easily testable, maintainable etc.

5.
Key Strategies

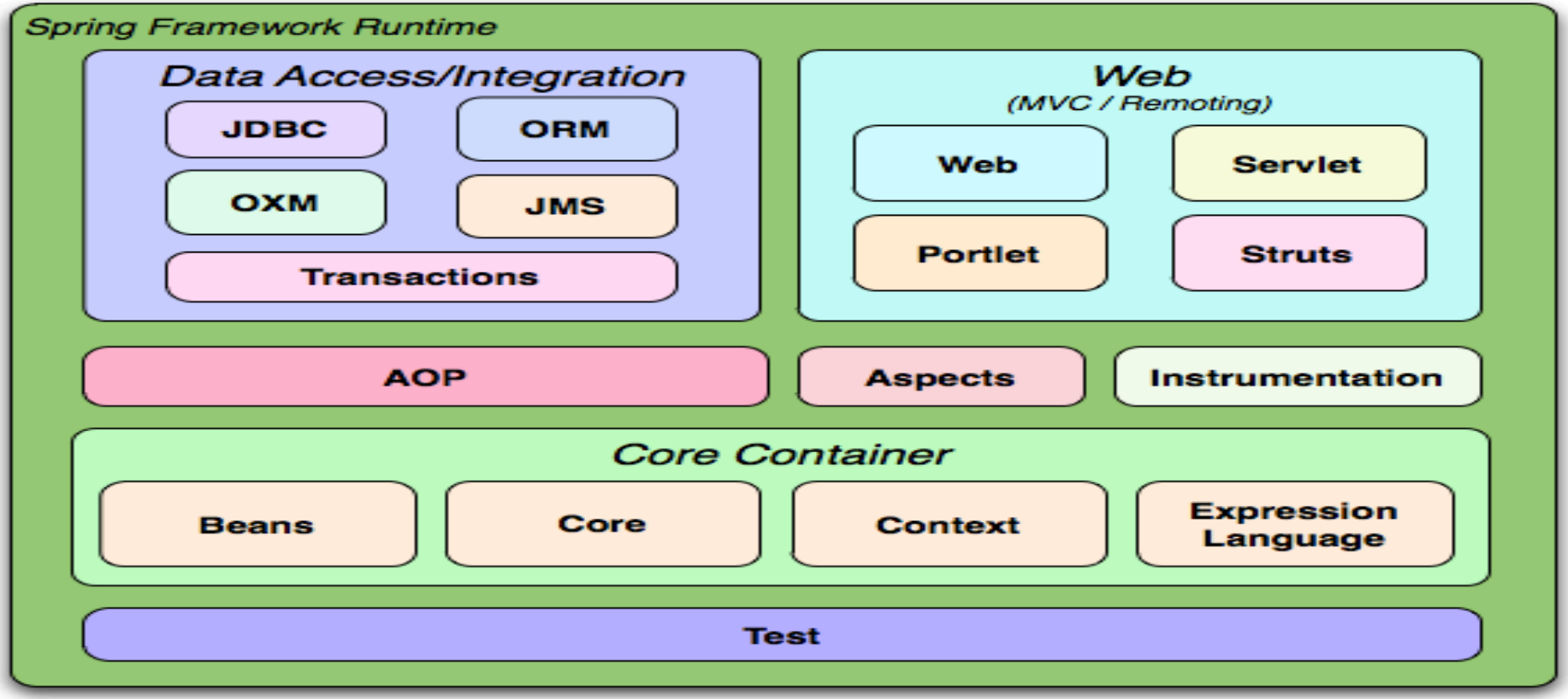
Key Strategies

- Lightweight and minimally invasive development with POJOs
- Loose coupling through DI and interface orientation
- Declarative programming through aspects and common conventions
- Eliminating boilerplate code with aspects and templates

6.

Spring Framework Architecture

Spring Framework Architecture



Core Container

- The **Core** and **Bean** provides the fundamental part of the framework including IoC and DI.
- The **Context** is a means to access any object defined and configured.
- The **SpEL** module provides a powerful expression language for querying and manipulation of an object.

Data Access/Integration

- **JDBC** provides a JDBC-abstraction layer.
- **ORM** provides integration layers for popular ORM APIs including JPS, JDO and Hibernate etc.
- **OXM** provides an abstraction layer that support object/XML mapping implementation for JAXB, Castor, XMLBeans and XStream.
- **JMS** provides messaging service.
- **Transaction** module supports programmatic and declarative transaction management.

Web

- **Web** – provides basic web oriented integration features.
- **Servlet** – Spring's MVC implementation.
- **Struts** – for integration with classic Struts web tier within a Spring application.
- **Portlet** – provides MVC implementation to be used in a portlet environment.

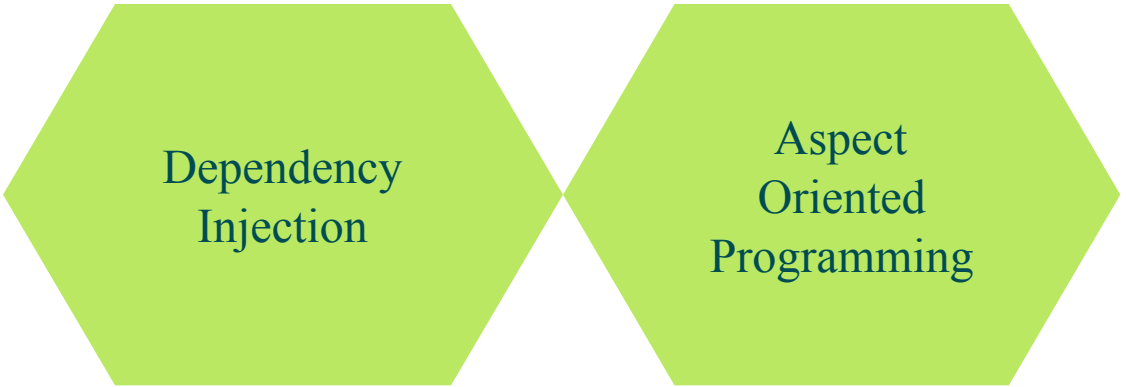
Miscellaneous

- **AOP** – provides aspect-oriented programming impl.
- **Aspects** – provides integration with AspectJ.
- **Instrumentation** – provides class instrumentation support and class loader implementations to be used in certain application servers.
- **Test** – supports the testing of Spring components with JUnit or TestNG frameworks.

7.

Key Components of Spring Framework

Key Components of Spring Framework



8.

Spring IoC Container

What’s IoC?

- Concept of application development.
- “Don't call us, we'll call you.”
- Dependency Injection is form of IoC.

Dependency Injection

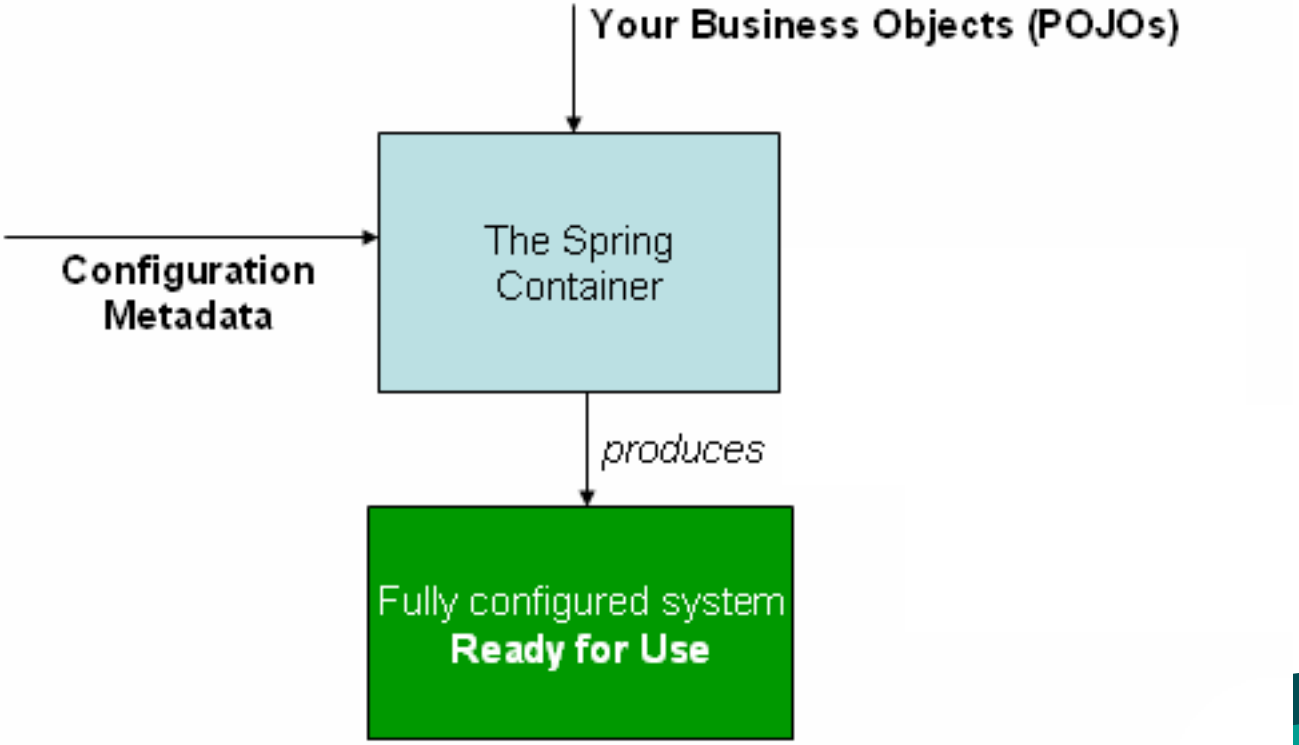
- When applying DI, the objects are given their dependencies at creation time by some external entity that coordinates each object in the system. In other words, dependencies are injected into objects.
- “Don't call around for your dependencies, we'll give them to you when we need you”
- Exist in two form:
    1. Constructor Injection
    2. Setter Injection

IoC vs DI vs Factory

- DI is one form of IoC.

- The Factory pattern's main concern is creating.
- The DI's main concern is how things are **connected together**.

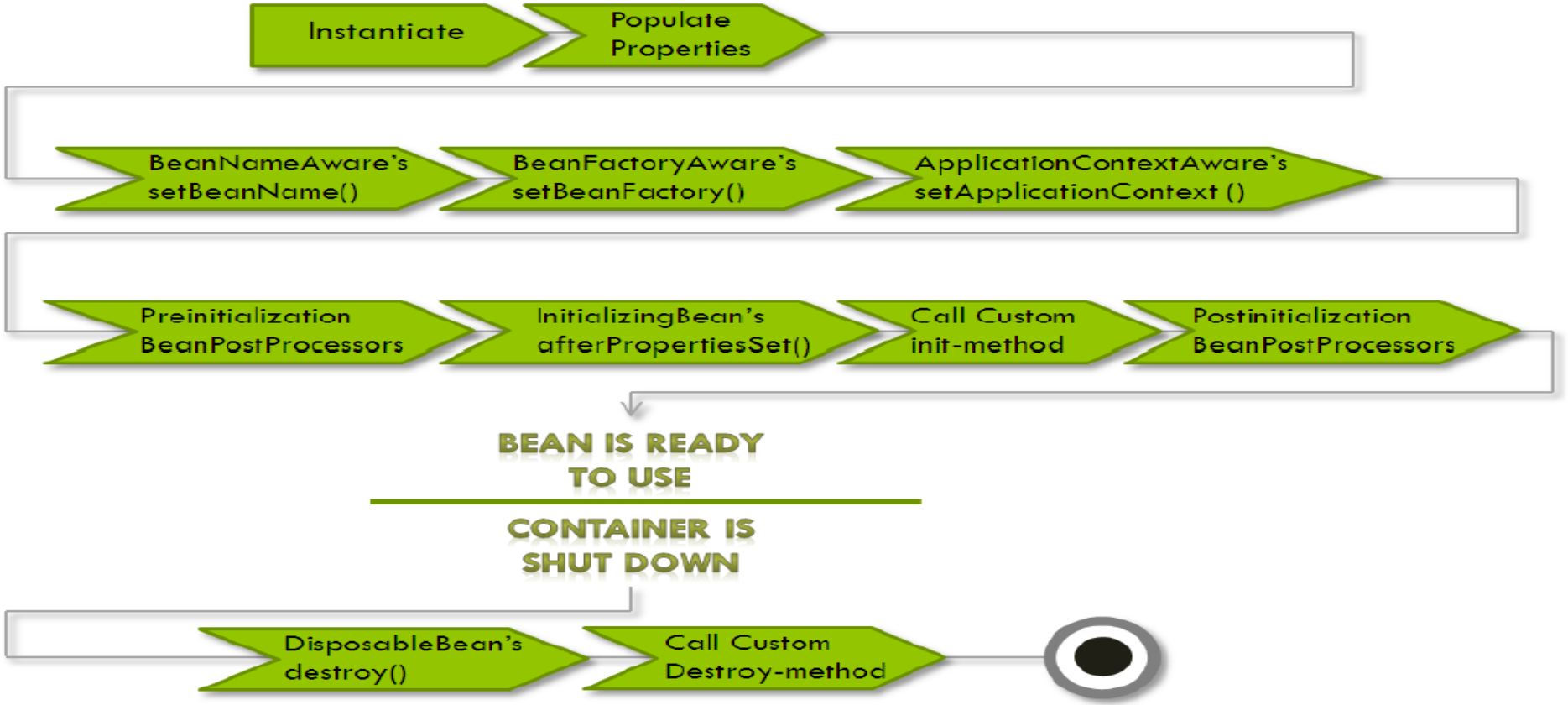
Essence of IoC Container



Working With An Application Context

- AnnotationConfigApplicationContext
- AnnotationConfigWebApplicationContext
- ClassPathXmlApplicationContext
- FileSystemXmlApplicationContext
- XmlWebApplicationContext

Bean Lifecycle



9.

Exploring Spring’s configuration options

Spring’s configuration options

- Explicit configuration in XML
- Explicit configuration in Java
- Implicit bean discovery and automatic wiring

9.1
Automatically wiring beans

Automatically wiring beans

Spring attacks automatic wiring from two angles:

- **Component scanning** — Spring automatically discovers beans to be created in the application context.
- **Autowiring** — Spring automatically satisfies bean dependencies.

Automatically wiring beans

Spring attacks automatic wiring from two angles:

- @Component-Scan
- @Configuration
- @Component
- @Named
- @Autowired
- @Inject

9.2

Wiring beans with Java

Wiring beans with Java

- @Configuration
- @Bean

9.3
Wiring beans with XML

Wiring beans with XML

- Creating an XML configuration specification
- Declaring a simple <bean>
- Initializing a bean with constructor injection
- Setting properties

9.4

Importing and mixing configurations

Importing and mixing configurations

- Referencing XML configuration in JavaConfig
  1. @Import
  2. @ImportResource
- Referencing JavaConfig in XML configuration
  1. <import>
  2. <bean>

10

Spring Profiles and Environments

Environments and profiles

- Configuring profile beans: @Profile
- Activating Profiles: spring.profiles.active and spring.profiles.default
  1. As initialization parameters on DispatcherServlet
  2. As context parameters of a web application
  3. As JNDI entries
  4. As environment variables
  5. As JVM system properties
  6. @ActiveProfiles

11

Conditional Beans

Conditional Beans

- @Conditional

```
public interface Condition {
        boolean matches(ConditionContext ctxt,
                AnnotatedTypeMetadata metadata);
}
```

12

Addressing Ambiguity in Autowiring

Addressing Ambiguity in Autowiring

- Designating a primary bean: @Primary
- Qualifying autowired beans: @Qualifier

13

Scoping Beans

Scoping Beans

Spring defines several scopes: @Scope

- **Singleton** — One instance of the bean is created for the entire application.
- **Prototype** — One instance of the bean is created every time the bean is injected into or retrieved from the Spring application context.
- **Session** — In a web application, one instance of the bean is created for each session.
- **Request** — In a web application, one instance of the bean is created for each request.

Scoping Beans(Contd.)

- Working with request and session scope
- Declaring scoped proxies in XML

14

Runtime Value Injection

Runtime Value Injection

Spring offers two ways of evaluating values at runtime:

- Property placeholders
- The Spring Expression Language (SpEL)

Injecting External Values

- @PropertySource(${ ... })
- @Value

Wiring with the Spring Expression Language

SpEL(#{ ... }) has a lot of tricks up its sleeves:

- The ability to reference beans by their IDs
- Invoking methods and accessing properties on objects
- Mathematical, relational, and logical operations on values
- Regular expression matching
- Collection manipulation

15

Aspect Oriented Programming

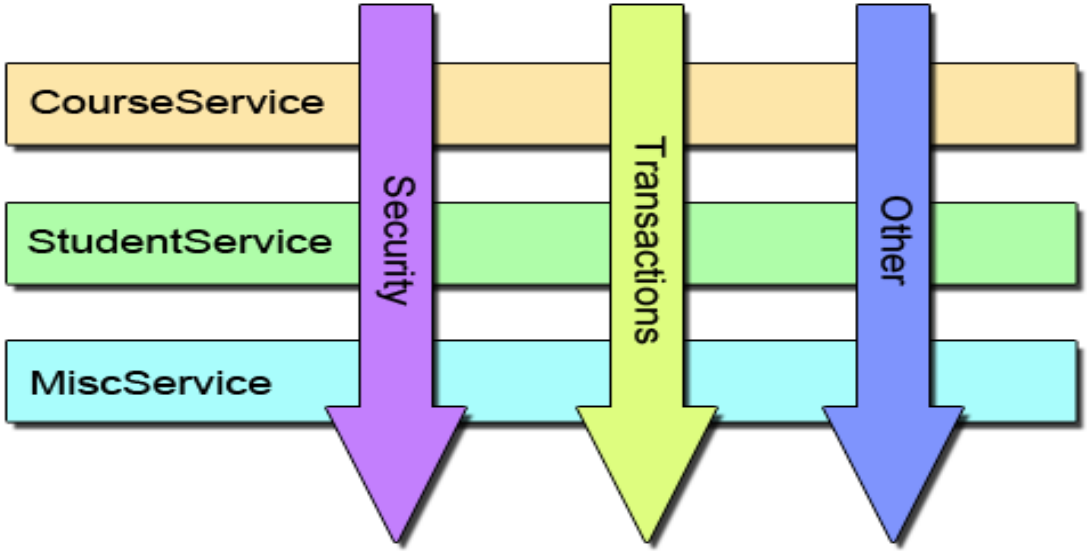
What's AOP?

In software development, functions that span multiple points of an application are called cross-cutting concerns. Typically, these cross-cutting concerns are conceptually separate from (but often embedded directly within) the application's business logic.

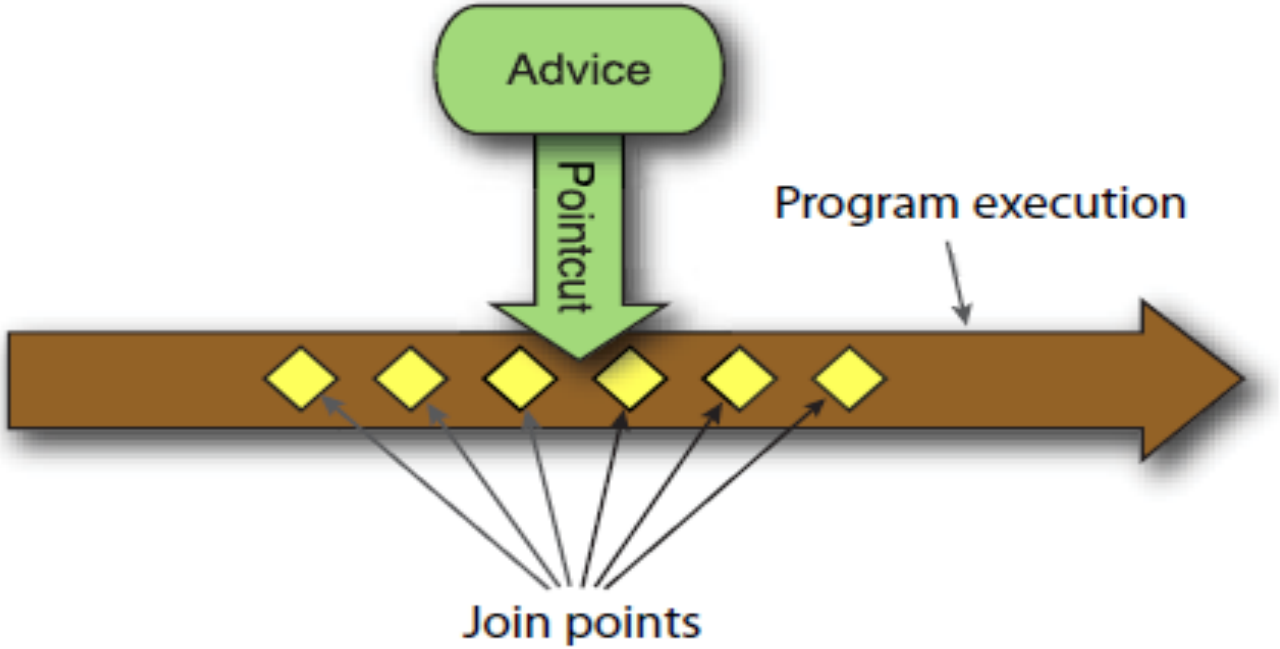
Separating these cross-cutting concerns from the business logic is where aspect oriented programming (AOP) goes to work.

What's AOP?

- Logging
- Validation
- Caching
- Security
- Transactions
- Monitoring
- Error Handling
- Etc...



AOP Terminology



Advice

- In AOP terms, the job of an aspect is called advice.
- Spring aspects can work with five kinds of advice:

1. Before—The advice functionality takes place before the advised method is invoked.
2. After—The advice functionality takes place after the advised method completes, regardless of the outcome.
3. After-returning—The advice functionality takes place after the advised method successfully completes.

Advice(Contd.)

4. After-throwing—The advice functionality takes place after the advised method throws an exception.
5. Around—The advice wraps the advised method, providing some functionality before and after the advised method is invoked.

Join Points

An application may have thousands of opportunities for advice to be applied. These opportunities are known as join points.

A join point is a point in the execution of the application where an aspect can be plugged in. This point could be a method being called, an exception being thrown, or even a field being modified.

Pointcuts

An aspect doesn't necessarily advise all join points in an application. Pointcuts help narrow down the join points advised by an aspect.

If advice defines the what and when of aspects, then pointcuts define the where. A pointcut definition matches one or more join points at which advice should be woven.

Aspects

An aspect is the merger of advice and pointcuts. Taken together, advice and pointcuts define everything there is to know about an aspect—what it does and where and when it does it.

Introductions

An introduction allows you to add new methods or attributes to existing classes.
The new method and instance variable can then be introduced to existing classes without having to change them, giving them new behavior and state.

Weaving

Weaving is the process of applying aspects to a target object to create a new proxied object. The aspects are woven into the target object at the specified join points. weaving can take place at several points in the target object's lifetime:

- **Compile time** — Aspects are woven in when the target class is compiled. This requires a special compiler. AspectJ's weaving compiler weaves aspects this way.

Weaving(Contd.)

- **Class load time** — Aspects are woven in when the target class is loaded into the JVM. This requires a special ClassLoader that enhances the target class's bytecode before the class is introduced into the application. AspectJ 5's load-time weaving (LTW) support weaves aspects this way.
- **Runtime** — Aspects are woven in sometime during the execution of the application. This is how Spring AOP
- aspects are woven.

Thanks!

Any questions?

You can find me at:

hiten@nexthoughts.com
http://github.com/hitenpratap/
http://hprog99.wordpress.com/

References

- http://www.javaworld.com/article/2071914/excellent-explanation-of-dependency-injection--inversion-of-control-.html
- http://www.slideshare.net/analizator/spring-framework-core?qid=017038a4-f5e8-444c-afdb-4e08611bd5c0&v=&b=&from_search=1
- http://www.slideshare.net/iceycake/introduction-to-spring-framework