Desarrollo de Microservices Cloud Native

Spring MVC

- @Controller
- @RequestMapping
  - RequestMethod
  - @PathVariable
  - @PathVariable @DateTimeFormat(iso=ISO.DATE) Date day
  - @RequestBody
  - @Valid AppointmentForm appointment, BindingResult result
  - @RequestHeader
  - @RequestParam
  - WebRequest
- @ResponseBody
- @RestController
  - HttpEntity
  - ResponseEntity<T>

TECH IS NOW

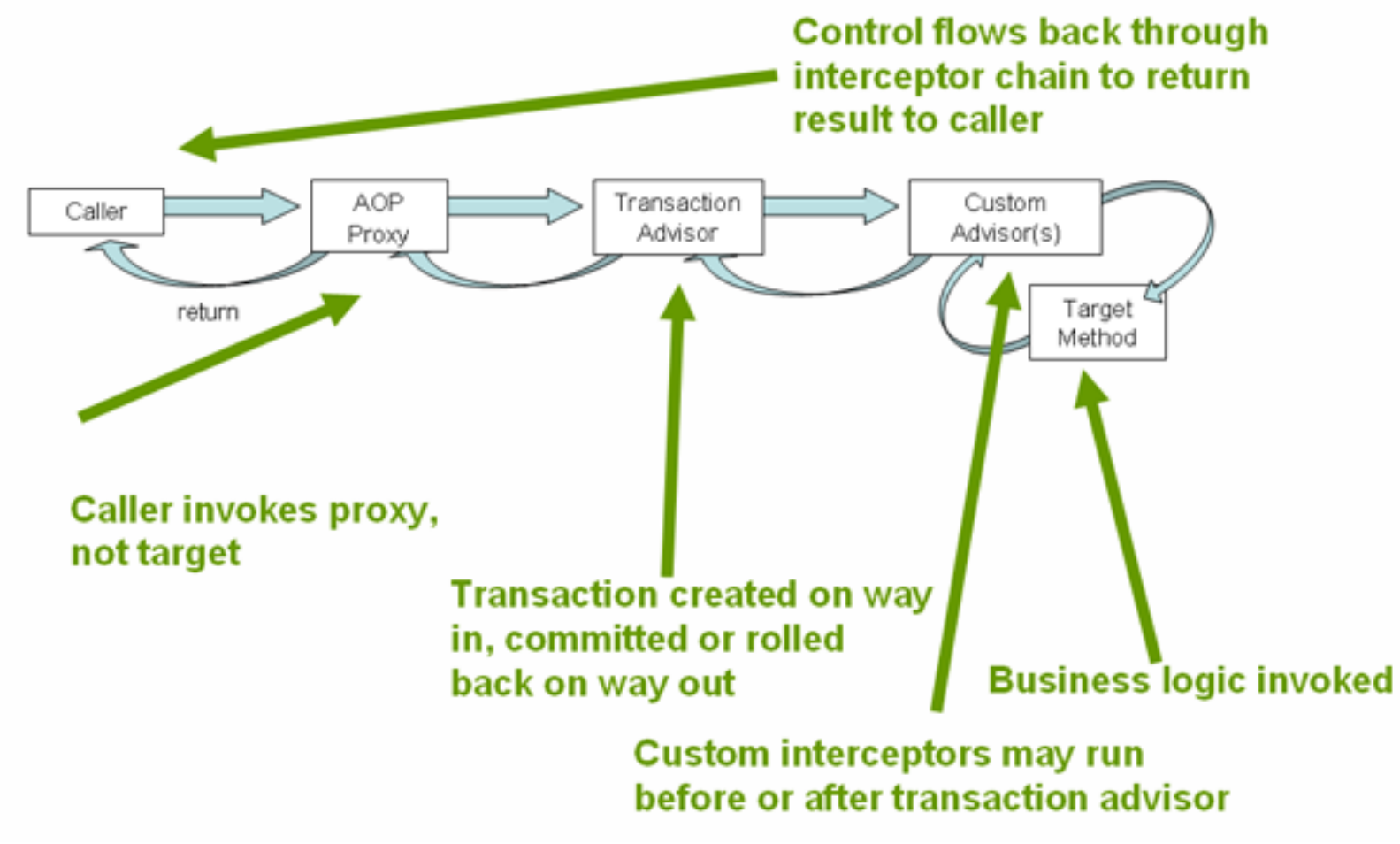
Spring MVC
Anotaciones compuestas

- Me permiten “simplificar” la definiciones de mi configuración
- Solo disponibles a partir de Spring 4.3 y Spring Boot 1.4.0
  - @GetMapping
  - @PostMapping
  - @PutMapping
  - @DeleteMapping
  - @PatchMapping

TECH IS NOW

Spring MVC
Laboratorios

Spring MVC
AOP



Spring MVC
AOP en Web

- @ControllerAdvice
  - @ExceptionHandler
- @RestControllerAdvice

TECH IS NOW

Spring HATEOAS

- Lo esencial que hace es ayudar con la creación de Links y ensamblado de las representaciones.
- Links
- Resources
  - ResourceSupport
- HttpEntity<Resource>
- ControllerLinkBuilder

Joel Test

- Escrito por Joel Spolsky en 2000
- Doce puntos muy sencillos para evaluar la madurez de una compañía.
- Solo se debe responder sí o no a una pregunta.
- Si se obtienen 10 o menos respuestas positivas (sí), la compañía tiene problemas que atender.

TECH IS NOW

Test

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

TECH IS NOW

Joel Test y Microservices

- La complejidad del software crece.
- Desarrollar sistemas distribuidos implica interfaces que pueden romperse.
- Tener una base sólida de prácticas permite adoptar nuevos paradigmas de desarrollo de software.
- Aunque no haga Microservices es muy buena idea mejorar mis prácticas de desarrollo.

TECH IS NOW

Integración continua

- Continuous Integration (CI) es una práctica de desarrollo de software que requiere que los desarrolladores integren el código en un repositorio compartido a lo largo del día.

- Cada commit o check-in de código es verificado por un proceso de construcción automático, lo que permite a los equipos detectar problemas al momento que se presenten.

- Integrar los cambios regularmente (o con cada cambio) se pueden detectar errores rápidamente para corregirlos lo más pronto posible.

Mantra

- Libera a menudo, libera regularmente.
- CI no va a eliminar los bugs, te va a ayudar a encontrarlos más fácilmente.

TECH IS NOW

Beneficios de CI

- Ayuda a reducir los problemas de integración del código, ya que se realiza constantemente.
- Los desarrolladores no deben esperar mucho para ver si el nuevo código va a funcionar en conjunto.
- Ayuda a reducir el tiempo depurando el código para aprovecharlo en agregar nueva funcionalidad
- Incrementa la visibilidad, lo que ayuda a mejorar la comunicación.
- CI es la base para entrega continua
  - Continuous Delivery CD.

Práctica

- Setup de un server (DigitalOcean)
- Setup de Jenkins
- Crear repositorio de código (GitLab)
- Crear un trabajo en Jenkins
- Añadir código productivo
- Añadir pruebas (JUnit)
  - Medir cobertura (JaCoCo)
- Añadir plugins de análisis estático
  - Checkstyle
  - FindBugs

TECH IS NOW

Instalación de Jenkins
Ubuntu Linux

- wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
- sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
- sudo apt-get update
- sudo apt-get install jenkins

TECH IS NOW

Instalación de Jenkins

TECH IS NOW

Plugins de Gradle

- apply plugin: 'checkstyle'
- apply plugin: 'findbugs'
- apply plugin: ‘jacoco’

TECH IS NOW

Herramientas de CI

- Jenkins
- TravisCI
  - OpenSource (http://travis-ci.org/)
  - Comercial (https://travis-ci.com)
- CircleCI
  - https://circleci.com

TECH IS NOW

Spring Framework

Trasfondo

- Java EE es ‘complicado’
- Modelo ágil
- Proceso de desarrollo integrado
- Diseño OO basado en negocio
- Ingeniería de software
- Patrones de diseño
  - Factory
  - Builder
  - Decorator
  - Service Locator

¿Qué es Spring?

- Spring es un framework open-source
  - presentado y desarrollado en 2002-2004.
  - Las ideas principales fueron concebidas por el experimentado arquitecto JEE, Rod Johnson.

- Es una alternativa para el desarrollo JEE
  - pero no solo para JEE.

TECH IS NOW

Continuación…

- Spring a menudo es descrito como un framework “ligero” para construir aplicaciones Java.
- No está enfocado a una parte específica de una aplicación. (i.e. Struts, JSF, Hibernate).
- Spring es “ligero” porque para usarlo en una aplicación no hay que hacer muchos cambios, en ocasiones ninguno. Esto al menos para usar el “Core” de Spring.
- Impacto mínimo. Principio de la filosofía de Spring.

TECH IS NOW

¿Por qué Spring?

- Porque reduce la complejidad de desarrollo JEE
  - Simplificar sin sacrificar poder.
  - Facilitar mejores prácticas, que de otra manera son difíciles de seguir.

- Porque nace de la experiencia práctica de muchos desarrolladores en todo el mundo.

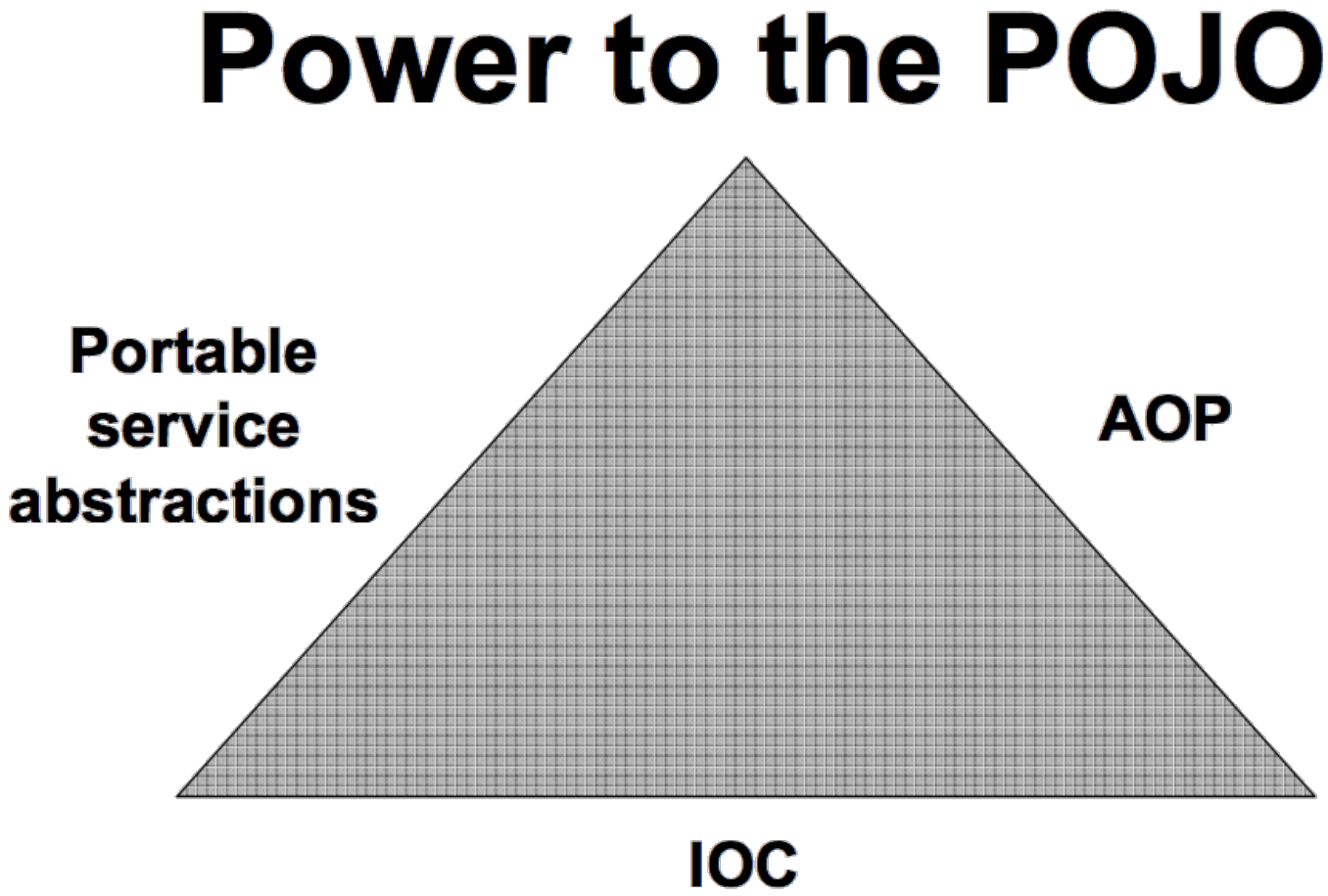
TECH IS NOW

Continuación…

- Desarrollar aplicaciones usando POJOs.
  - Spring ofrece capacidades avanzadas de configuración que permiten escalar a una complejidad del mundo real.
  - Permite aplicar servicios empresariales a los POJOs, de forma declarativa y no invasiva.

TECH IS NOW

Continuación…



TECH IS NOW

¿Qué es Spring?

- Framework de contenedor ligero
  - Framework: base para construir aplicaciones java; incluso empresariales
  - Contenedor: administra el ciclo de vida de los componentes de las aplicaciones
  - Ligero: mínimamente invasivo

TECH IS NOW

¿Qué no es Spring?

- Spring NO es un Servidor de Aplicaciones
  - Aunque soporta e integra casi todas las APIs de Java EE.
  - SpringSource tiene runtimes para aplicaciones
    - tcServer: Tomcat con características avanzadas de monitoreo.
    - SpringSource ERS: Enterprise Ready Server, Apache WebServer + Tomcat.

TECH IS NOW

Historia

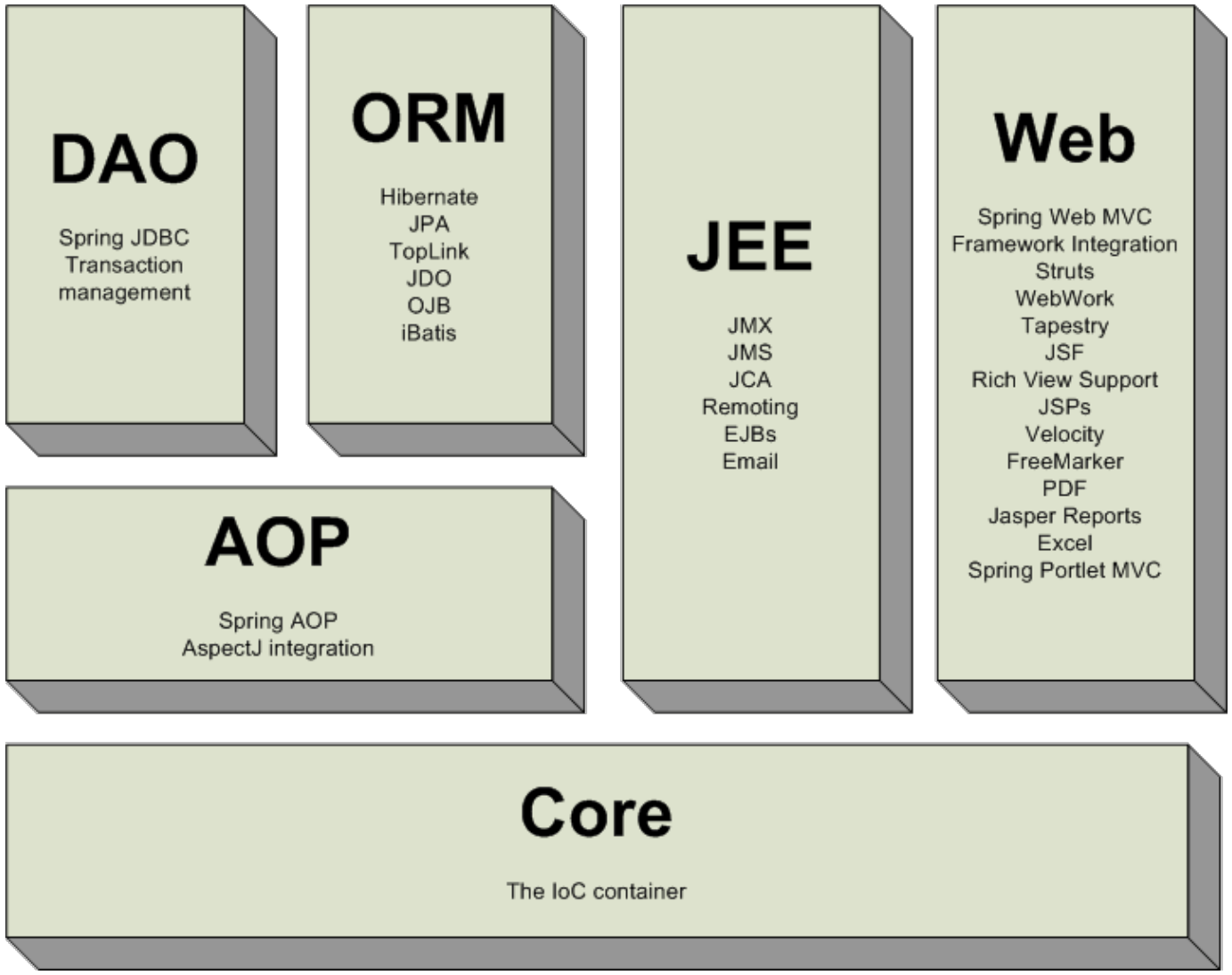
- Spring 1.0
  - DI, AOP, DataAccess, web framework, JNDI, EJB 2.1
- Spring 2.0
  - Configuración en XML extensible, nueva librería de tags, soporte a lenguajes dinámicos, nuevos alcances de los beans
- Spring 2.5
  - Configuración con anotaciones, integración con jUnit 4, nueva versión de Spring MVC, descubrimiento automática de beans
- Spring 3.0
  - REST, SpEL, validación declarativa, configuración basada en Java
- Spring 4.0
  - Mejora la configuración basada en Java
  - Empieza el soporte de Java 8
  - Soporta Groovy Bean Definition DSL (Domain Specific Language)
- Spring 5.0
  - Java 8
  - Compatibilidad Java 9
  - Java EE 7
  - Compatibilidad Java EE 8

MetaFramework

- Spring Security
- Spring WebServices
- Spring Roo
- Spring LDAP
- Spring WebFlow
- BlazeDS Integration
- dmServer
- Spring Batch
- SpringIntegration
- SpringSource Tool Suite
- Spring Extensions
- Spring JavaConfig
- Spring RichClient
- Spring .NET
- Spring BeanDoc
- Grails
- Spring Data
- Spring Cloud

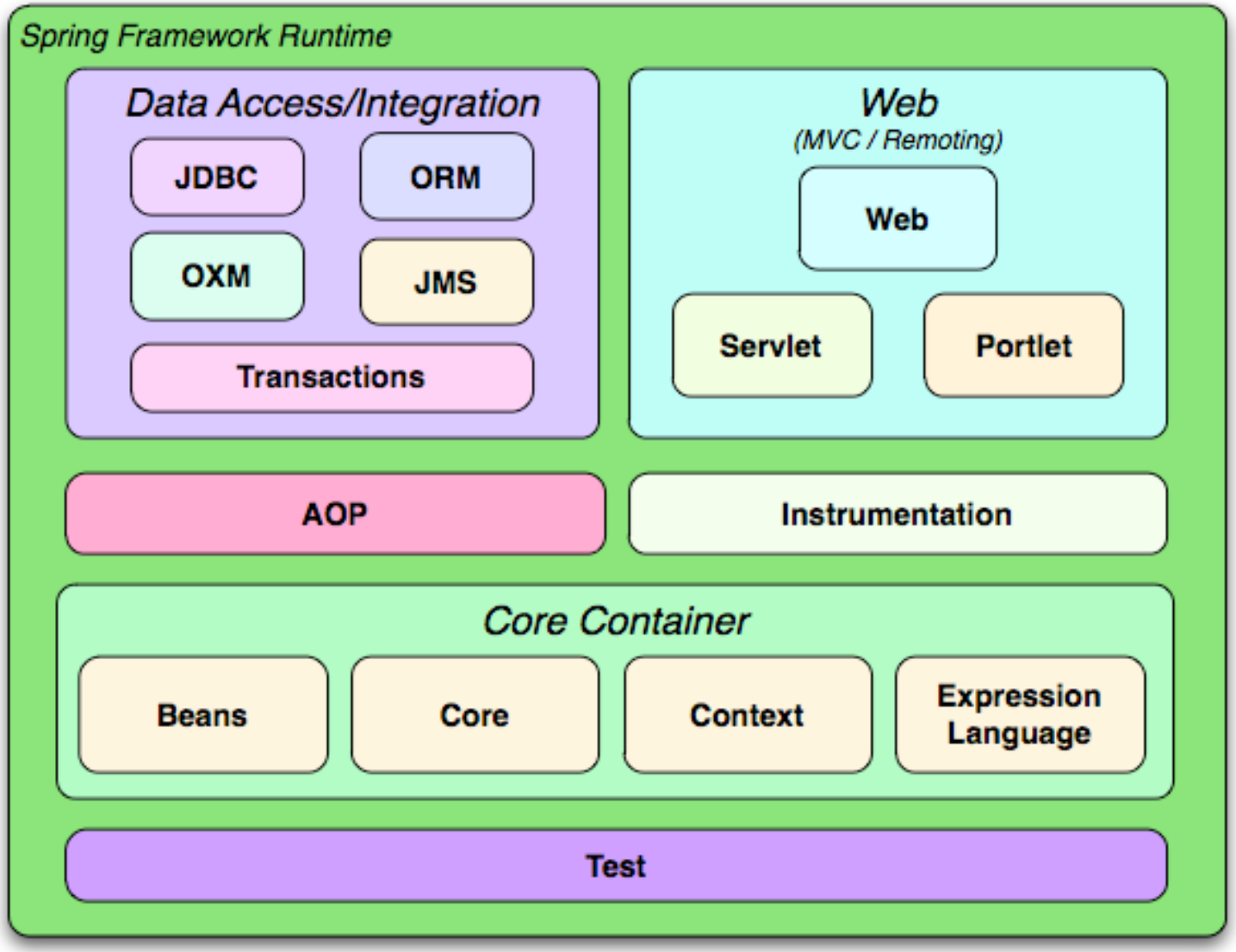
TECH IS NOW

Componentes de Spring



TECH IS NOW

Componentes de Spring



Características (cont).

- AOP
  - Facilita la implementación de soluciones a problemas recurrentes, incluso en contextos diferentes.
- Acceso a datos.
  - Integración con Hibernate, Jdbc, iBatis, JPA.
- Administración de transacciones.
  - Capa de abstracción para manejo de transacciones.
- Integración y simplificación con JEE.
  - EJB
  - Session Bean
    - SLSB
    - SFSB
  - JMS
- JNDI
- JMX
- Java Mail
- Muy importante. Pruebas
  - Unitarias
  - Integración

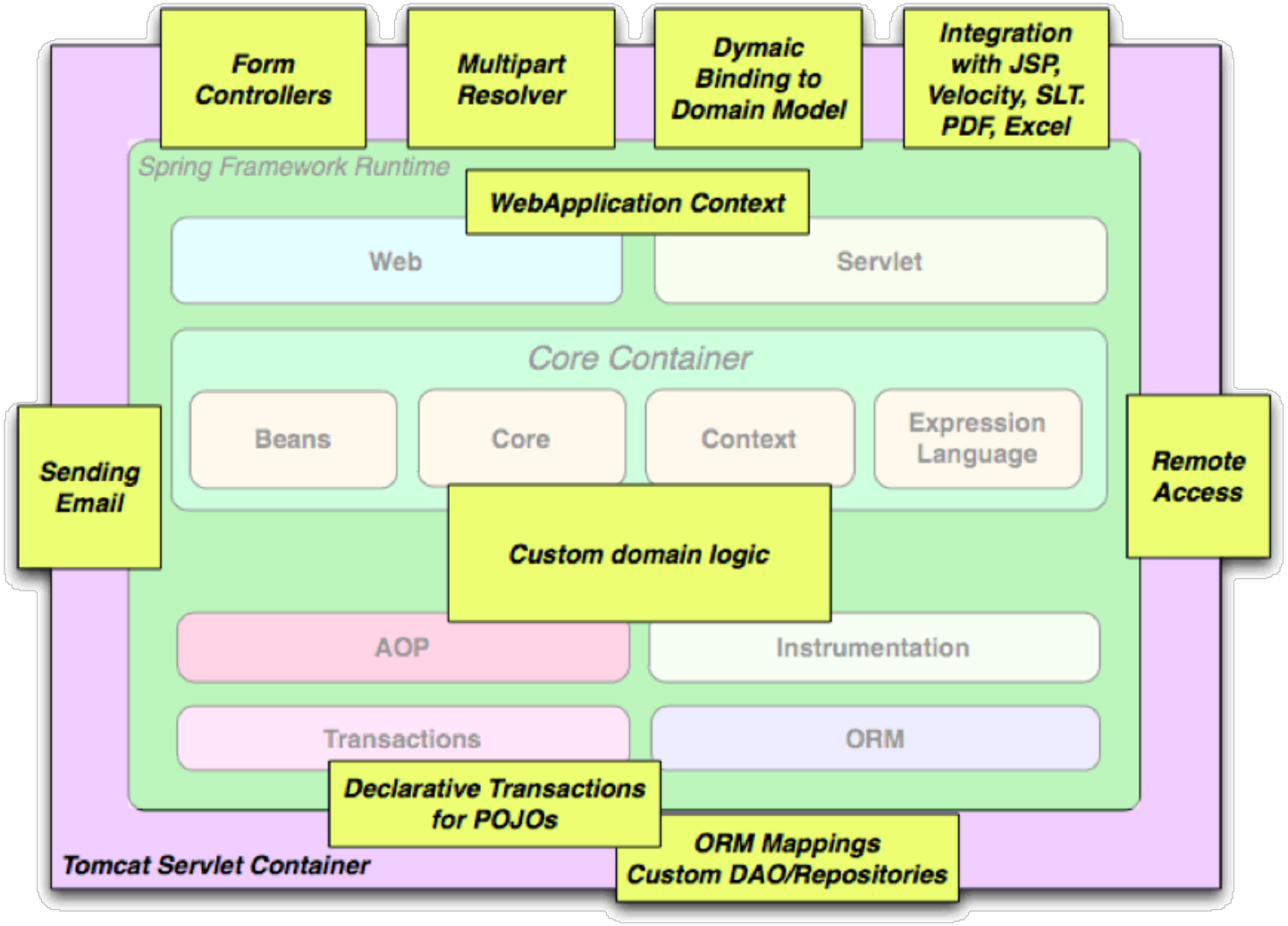
TECH IS NOW

Características (cont).

- Spring en Web.
  - SpringMVC
    - JSP
    - Velocity
  - Struts
  - JSF
- Calendarización de Procesos
  - Quartz
- Manejo de excepciones simplificado
  - Evita el tedioso manejo de excepciones.
  - Cátedra de manejo de excepciones.

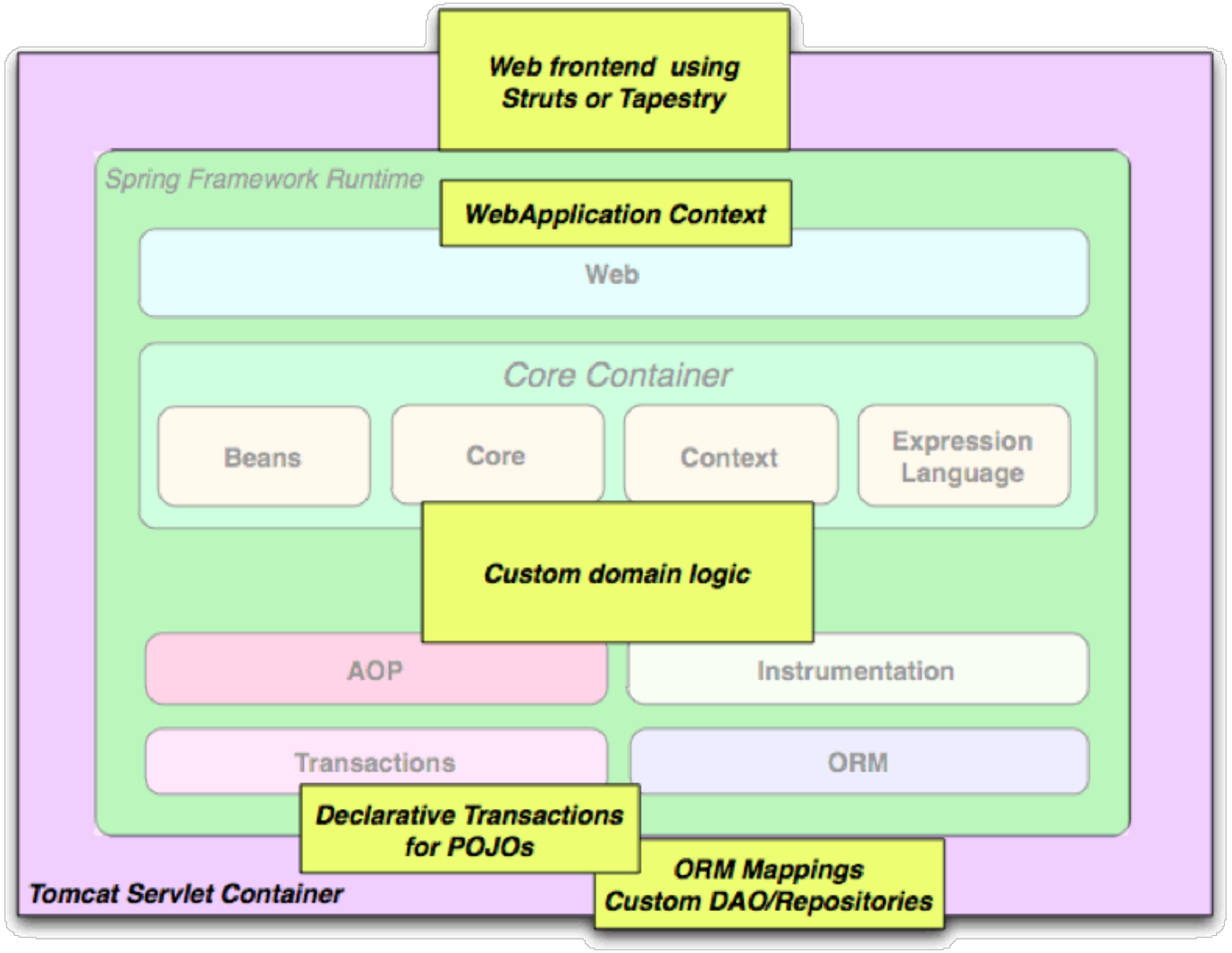
TECH IS NOW

Escenario de uso:
Spring en todas las capas



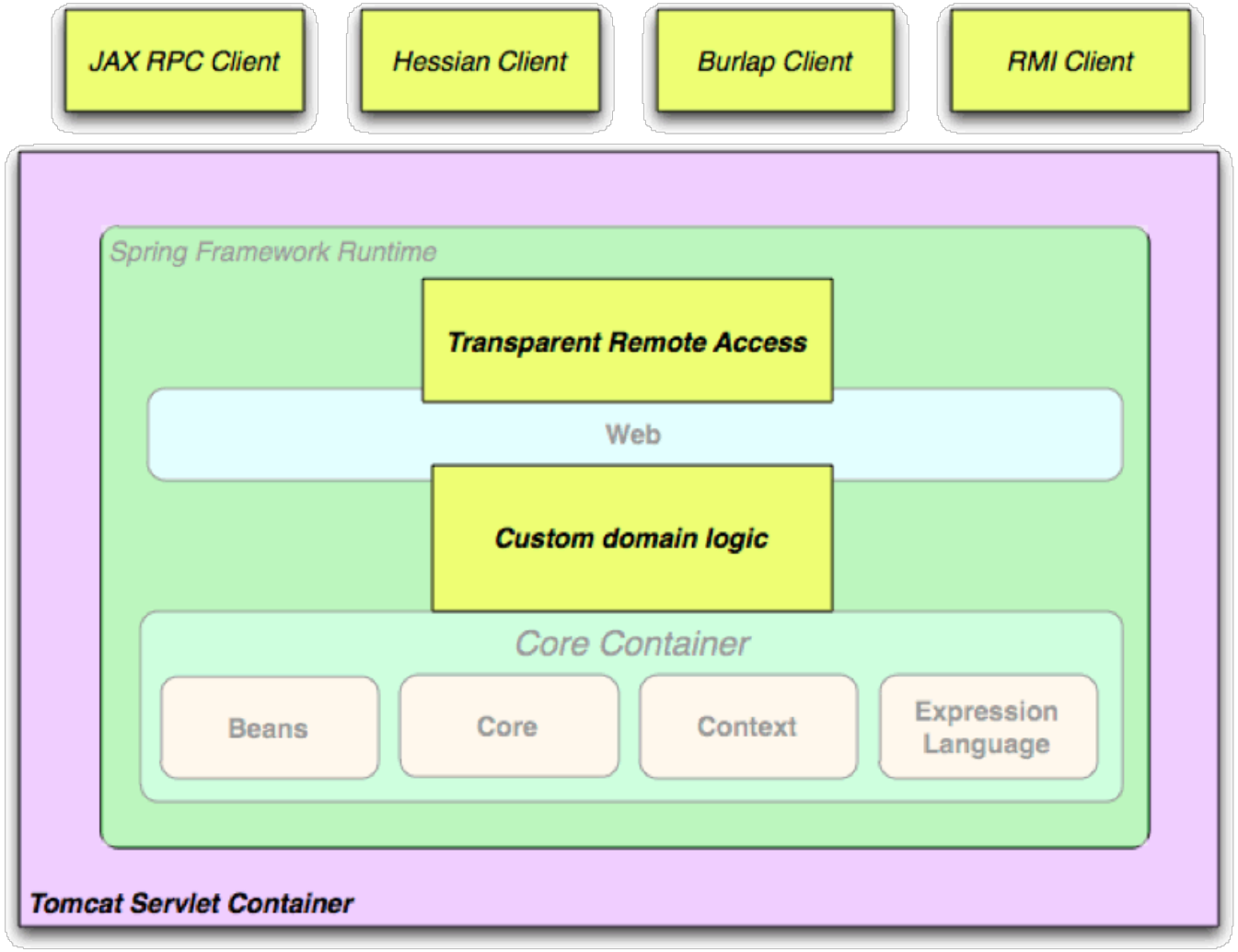
TECH IS NOW

Escenario de uso:
Framework web de terceros



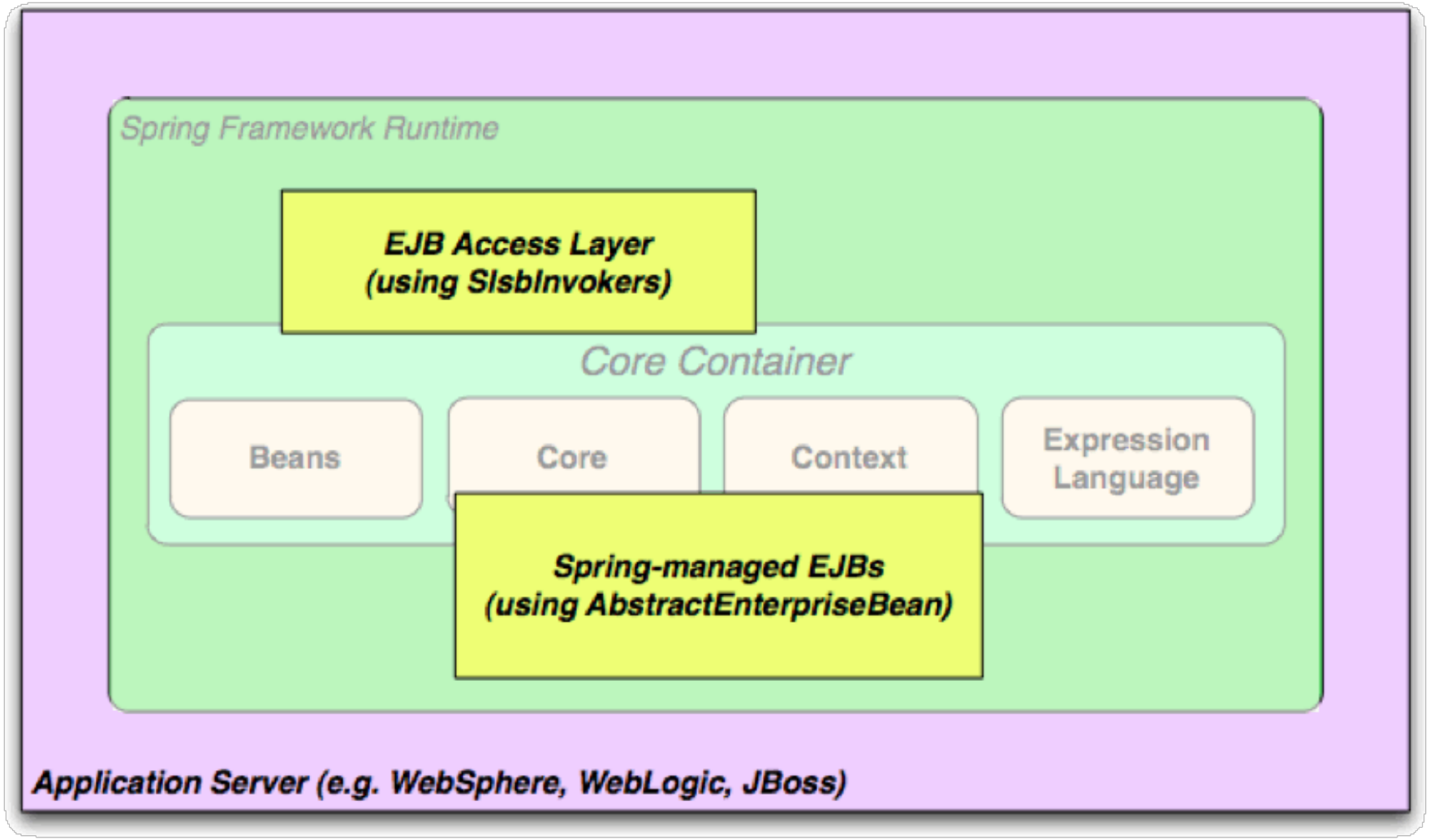
TECH IS NOW

Escenario de uso: Haciendo remoting



TECH IS NOW

Escenario de uso: EJBs



TECH IS NOW

Spring Framework
Elementos Básicos
Contenedor de IoC
(Inversion of Control y Dependency Injection)

TECH IS NOW

Inversion of Control

- Técnica que externaliza la creación y manejo de las dependencias de componentes.
- IoC también es conocido como Dependency Injection (DI).
- La implementación de DI de Spring está basado en dos conceptos clave de Java: JavaBeans e Interfases.

TECH IS NOW

Continuación…

- En el contexto de DI, Spring actúa más como un contenedor que como un framework.
  - Nos provee instancias de clases de nuestra aplicación con todas las dependencias que ellas necesitan.
  - Usando JavaBeans se facilita esta labor del contenedor.

TECH IS NOW

Ventajas de DI

- Reduce el código de plomería.
  - El contenedor se encarga de muchas cosas por nosotros.

- Externalizan las dependencias.
  - No es necesario recompilar la aplicación para alterar las dependencias.

- Administración de las dependencias en un solo sitio.
  - Toda la información de las dependencias es contenida en un solo repositorio.

TECH IS NOW

Continuación…

- Mejora la prueba del software (TDD).
  - Facilita intercambiar implementaciones.
  - Uso de mocks para probar.
  - Ejecución de pruebas muy rápidas.

- Permite mejor diseño de aplicaciones.
  - Se diseña orientado a interfases.
  - Te concentras en la lógica de tu aplicación, no en la implementación del framework.

TECH IS NOW

Valores de Spring

- Solidez en sus abstracciones demuestran un gran valor en un amplio rango de entornos.

- Adopción estratégica en muchas empresas, evitando algunos de los enfoques ineficientes existentes en Java EE.

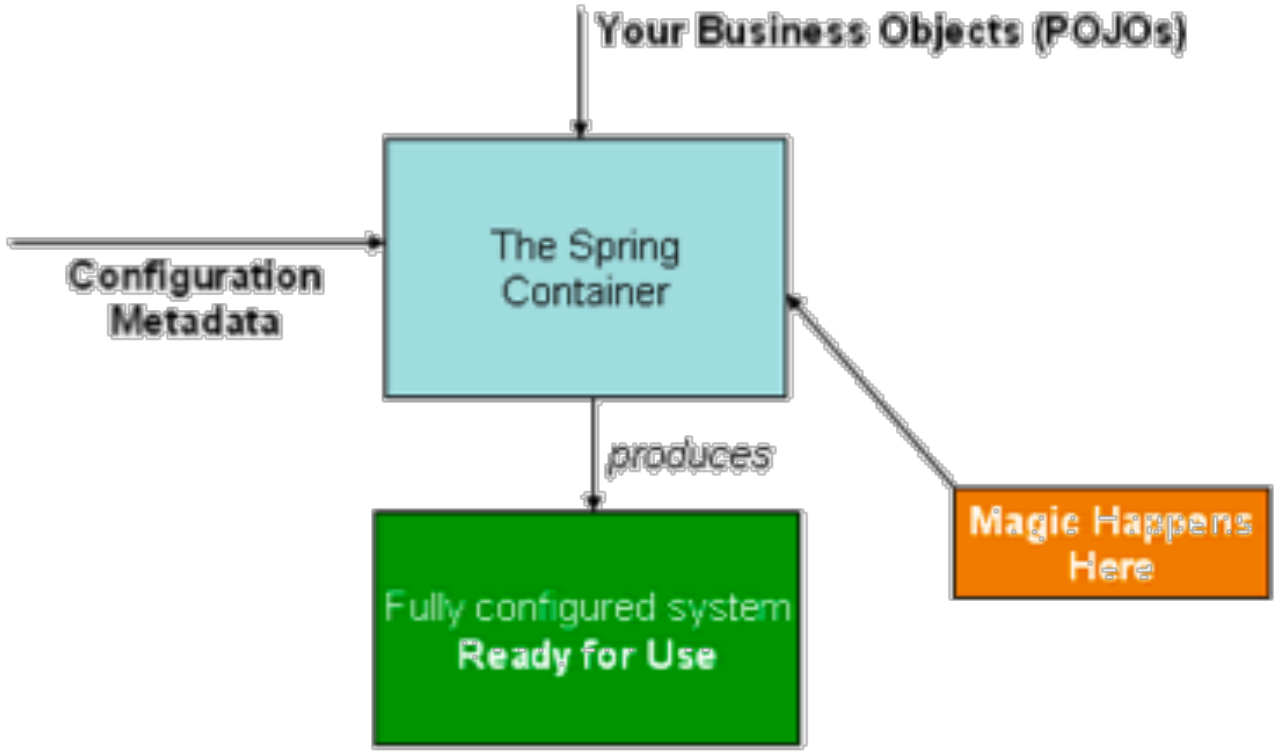
TECH IS NOW

Spring Framework
Screencast recomendado
https://vimeo.com/8660559

El contenedor de beans de Spring (IoC Container)

- En Spring, los objetos que forman la columna vertebral de una aplicación son administrados por el contenedor IoC.
- En Spring, a dichos objetos se les conoce como Beans.
- Un bean es cualquier objeto simple, el cual es instanciado y administrado por el Contenedor IoC.
- La definición de los Beans y sus relaciones es expresada a través de diversos mecanismos de configuración.
- El contenedor es representado por implementaciones de org.springframework.beans.factory.BeanFactory

El contenedor de beans de Spring (Cont..)



TECH IS NOW

El contenedor de beans de Spring (Cont..)

```
<?xml version="1.0" encoding="UTF-8"?>
 <beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
  <bean id="..." class="...">
        <!-- collaborators and configuration for this bean go    here -->
  </bean>
  <bean id="..." class="...">
     <!-- collaborators and configuration for this bean go    here -->
  </bean>
     <!-- more bean definitions go here... -->
</beans>
```

TECH IS NOW

Instanciando el IoC

```
Resource resource = new FileSystemResource("beans.xml");
BeanFactory factory = new XmlBeanDefinitionReader(resource);

ClassPathResource resource1 = new ClassPathResource("beans.xml");
BeanFactory factory1 = new XmlBeanDefinitionReader(resource);

ApplicationContext context = new ClassPathXmlApplicationContext( new
String[] {"applicationContext.xml","applicationContext-part2.xml"});
//ApplicationContext es un BeanFactory
BeanFactory factory = (BeanFactory) context;
```

TECH IS NOW

BeanFactory

Principales Métodos

boolean	**containsBean**(String name)
Object	**getBean**(String name)
Object	**getBean**(String name, Class requiredType)
Class	**getType**(**String name**)
boolean	**isSingleton**(String name)

API de Spring:
http://www.springframework.org/docs/api/

ApplicationContext

- Un ApplicationContext extiende la funcionalidad de BeanFactory.
- La idea principal es integrar en forma “declarativa” la obtención y manipulación de los beans administrados por el IoC sin tener que instanciar la fábrica en forma programática.
- Se apoya en el uso de algunos frameworks que emplee dicha aplicación para cargar e instanciar la fábrica, por ejemplo:
  - Uso de ContextListener en una aplicación web para incluir la referencia de la fábrica dentro del contexto de la aplicación web: ServletContext.
- Apropiado especialmente en aplicaciones empresariales.

El elemento <bean>

- Principales atributos

`class`	***Nombre calificado de la clase que representa al bean.***
`name`	***Nombre del bean.***
`id`	***Identificador único del bean.***
`scope`	***alcance o ámbito del bean.***
`property`	***Propiedad o atributo de un bean***
`constructor-arg`	***Argumentos del constructor***

Scopes

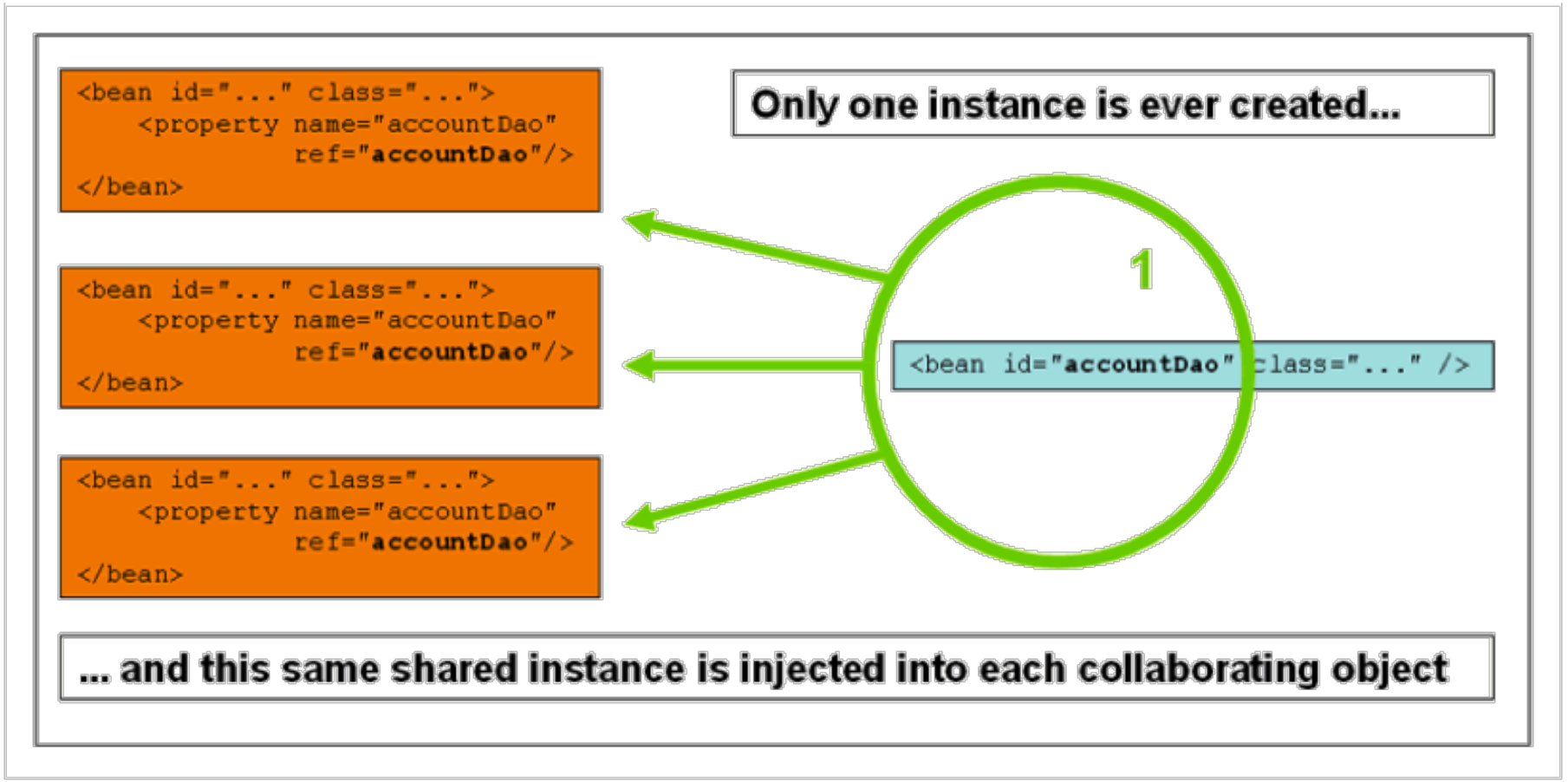
- La definición de un bean (por lo general en un archivo XML) representa el template para crear instancias de la clase indicada.
- Pueden existir 1 o más instancias a partir de la definición de un bean.
- Lo anterior se controla a través del concepto de “alcance” de un bean:

Scopes

`singleton`	***Existe una sola instancia por cada IoC***
`prototype`	***Se crea una nueva instancia cada vez que se haga referencia***
`request`	***Asociado al ciclo de vida de una petición HTTP.***
`session`	***Asociado al ciclo de vida de una sesión (HttpSession)***
`global session`	***Asociado al ciclo de vida de una sesión global***

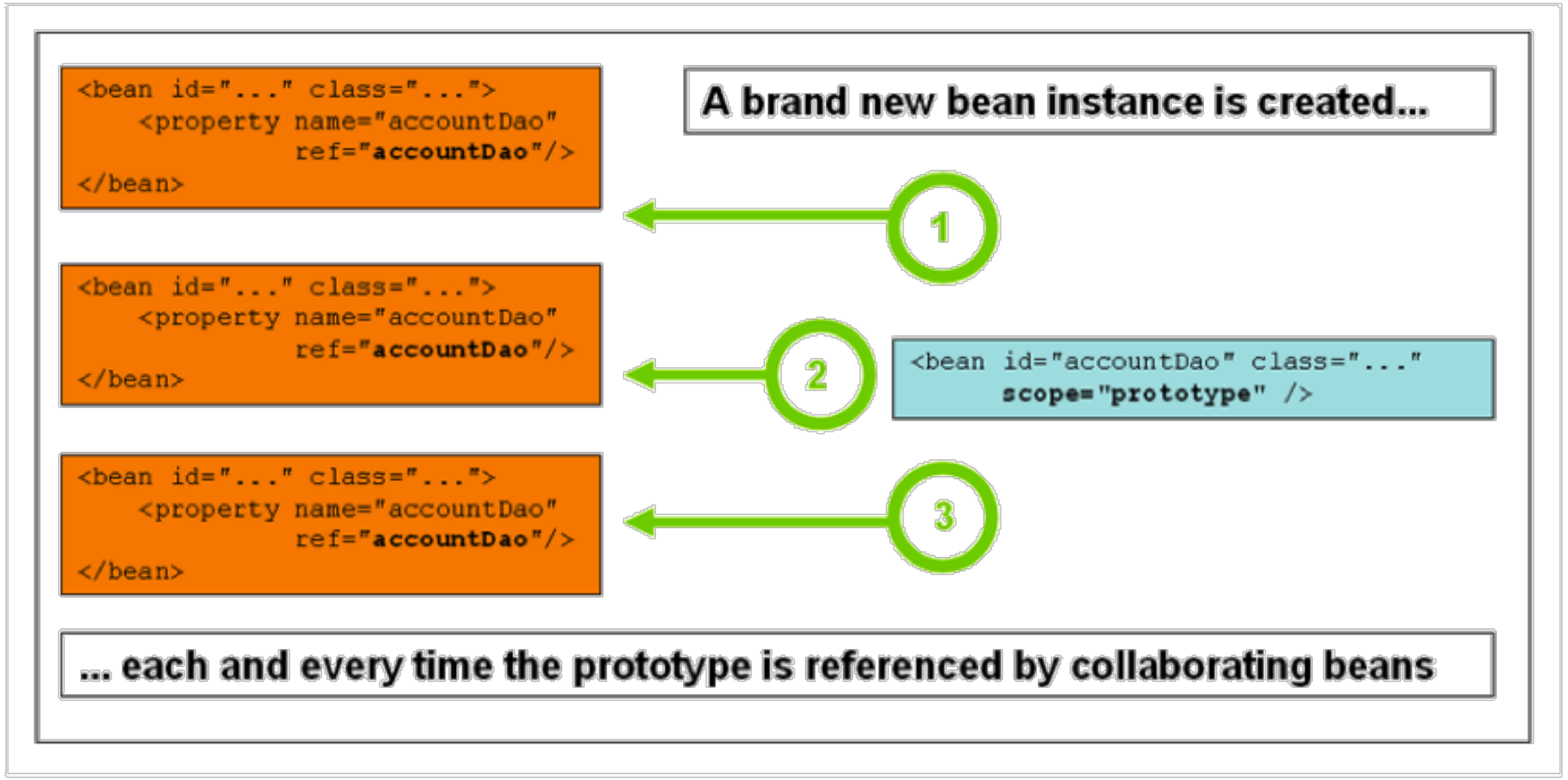
TECH IS NOW

Singleton Scope



TECH IS NOW

Prototype Scope



TECH IS NOW

Manejo de dependencias

- Cualquier aplicación OO por simple que sea, contiene un conjunto de relaciones entre sus objetos.
- Cada clase define algunas de estas relaciones a través de atributos de instancia.
- Generalmente las propias clases de la aplicación se encargan de instanciar y proporcionar dichos atributos empleando patrones (ServiceLocator, Factories,etc), generando el llamado “código de plomería”.

Manejo de dependencias

- Con Spring, el contenedor se encarga de detectar dichas dependencias, para luego realizar una “Inyección” del objeto empleando el concepto de IoC “Inversion of Control”

- A nivel de archivo XML, se emplea el elemento ref para “alambrar” las dependencias de un bean.

TECH IS NOW

Manejo de dependencias – Tipos de Inyección

- Inyección vía métodos “setter”.

```
public class Curso {
   //dependencia de tipo Profesor
   private Profesor profesor;
   // metodo setter empleado para que Spring
         //“inyecte”
   public void setProfesor(Profesor p){
      this.profesor = p;
   }
}
```

TECH IS NOW

Manejo de dependencias – Tipos de Inyección

- Inyección vía Constructor.

```
public class Curso {
   //dependencia de tipo Profesor
   private Profesor profesor;
   // Constructor para que Spring "inyecte"
   public Curso(Profesor p){
      this.profesor = p;
   }
}
```

TECH IS NOW

Manejo de Dependencias – Colecciones de Objetos

- Para manejar dependencias de colecciones de objetos, se emplean los siguientes elementos:
  - `<set>` Empleado para objetos tipo `Set`
  - `<map>` Empleado para objetos tipo `Map`
  - `<list>` Empleado para objetos tipo `List`
  - `<props>` Empleado para objetos tipo `Properties`.
- El valor o key de estas colecciones se puede inicializar empleando alguno de los siguientes elementos:
  - bean
  - ref, idref
  - list, set, map, props
  - value
  - null

TECH IS NOW

Explicando IoC y DI

- DI es un mecanismo para proveer dependencias de componentes (colaboradores).
- Además de manejar esas dependencias en su ciclo de vida.
- Un componente que requiere una dependencia es llamado objeto dependiente (dependent object)

TECH IS NOW

Tipos de IoC

- Dependency Injection
  - El contenedor es encargado de inyectar las dependencias.
- Dependency lookup
  - Dependency Pull
    - Similar a los lookups en JNDI.
    - Ejemplo en la prueba de unidad del Lab2.
  - Contextualized Dependency Lookup (CDL)
    - Cuando el contenedor está listo para pasar la referencia a un componente invoca a un método del objeto dependiente.

TECH IS NOW

Inyección de dependencia por constructor

- La dependencias se inyectan en el constructor del objeto:

```
public class ConstructorInjection {
  private String depen;
  public ConstructorInjection(String depen) {
    this.depen = depen;
  }
}
```

Pruebas con Spring

- Spring ofrece un excelente soporte para frameworks de pruebas
- En el curso usaremos JUnit 4
- Se configura con anotaciones y es muy sencillo de usar
- Existen dos tipos de pruebas que haremos
  - Unitarias
  - Integración
- Es prueba unitaria cuando solamente probemos de manera aislada una sola clase
- Es de integración cuando en la prueba intervienen varias clases bajo prueba

TECH IS NOW

Inyección por setter VS Inyección por constructor

- Es preferible usar por constructir, favorece la inmutabilidad y previene crear setter.
- Considerar cuando se van a inyectar por constructor más de 6 parámetros.
- Elección del desarrollador.

TECH IS NOW

Spring Boot

Introducción

- Spring Boot es una herramienta que facilita la creación de aplicaciones usando como marco de referencia el modelo de programación ligero de Spring Framework.

- Añade al amplio soporte de tecnologías del ecosistema Spring un mecanismo de configuración simple y que permite el rápido desarrollo de proyectos.

TECH IS NOW

Características

- Crea aplicaciones Spring Stand Alone.
- Soporta WAR tradicionales o empotrar Jetty, Tomcat o Undertow.
- Provee configuraciones para herramientas de construcción que permiten minimizar la configuración.
  - Maven
  - Gradle
- Configura Spring automáticamente cuando es posible.
  - Al menos toda la infraestructura del Contexto de Spring
- Provee características de producción tales como métricas, monitoreo de la aplicación y configuración externa.
- No genera código automáticamente y no requiere configuración XML.

TECH IS NOW

Starters

- Un starter es un conjunto de configuración y valores por omisión para la integración de alguna tecnología.

- Spring Boot provee un amplio soporte de starters de caja; solo hay que configurarlos.

TECH IS NOW

Configuración

- Por omisión los parámetros de configuración se establecen en archivos en la raíz del classpath:
  - application.properties
  - application.yml

- Se pueden definir perfiles de configuración.

- Se pueden “sobrescribir” mediante:
  - Variables de ambiente
  - Especificando un nuevo archivo de configuración

TECH IS NOW

Spring Boot Gradle Plugin

- Permite administrar dependencias
- Añade 2 tareas muy importantes
  - bootRun (ejecuta mi aplicación en modo desarrollo). Nunca usar en producción
  - bootRepackage. Prepara un jar “gordo” que incluye todas las dependencias de mi aplicación. Completamente autocontenida.

TECH IS NOW