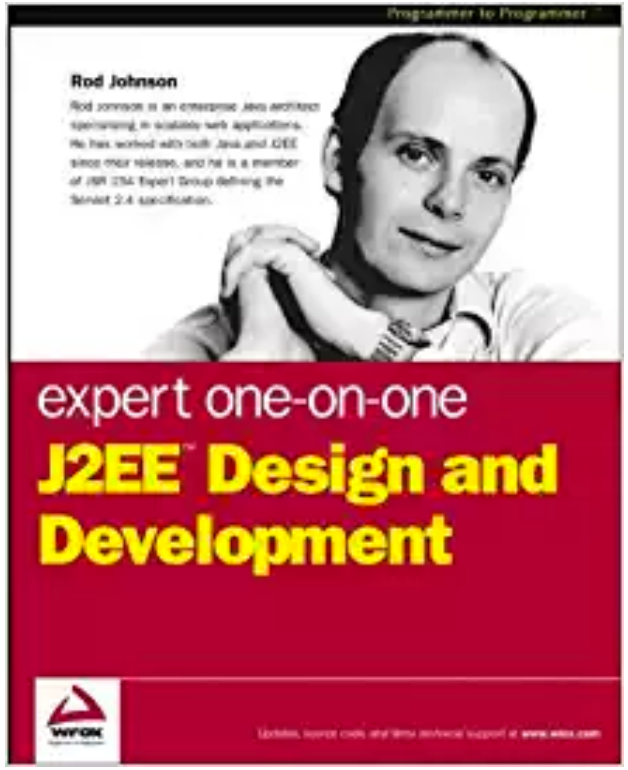
Walmart

Mike Barlotta

Spring Framework Part 2

What is Spring?

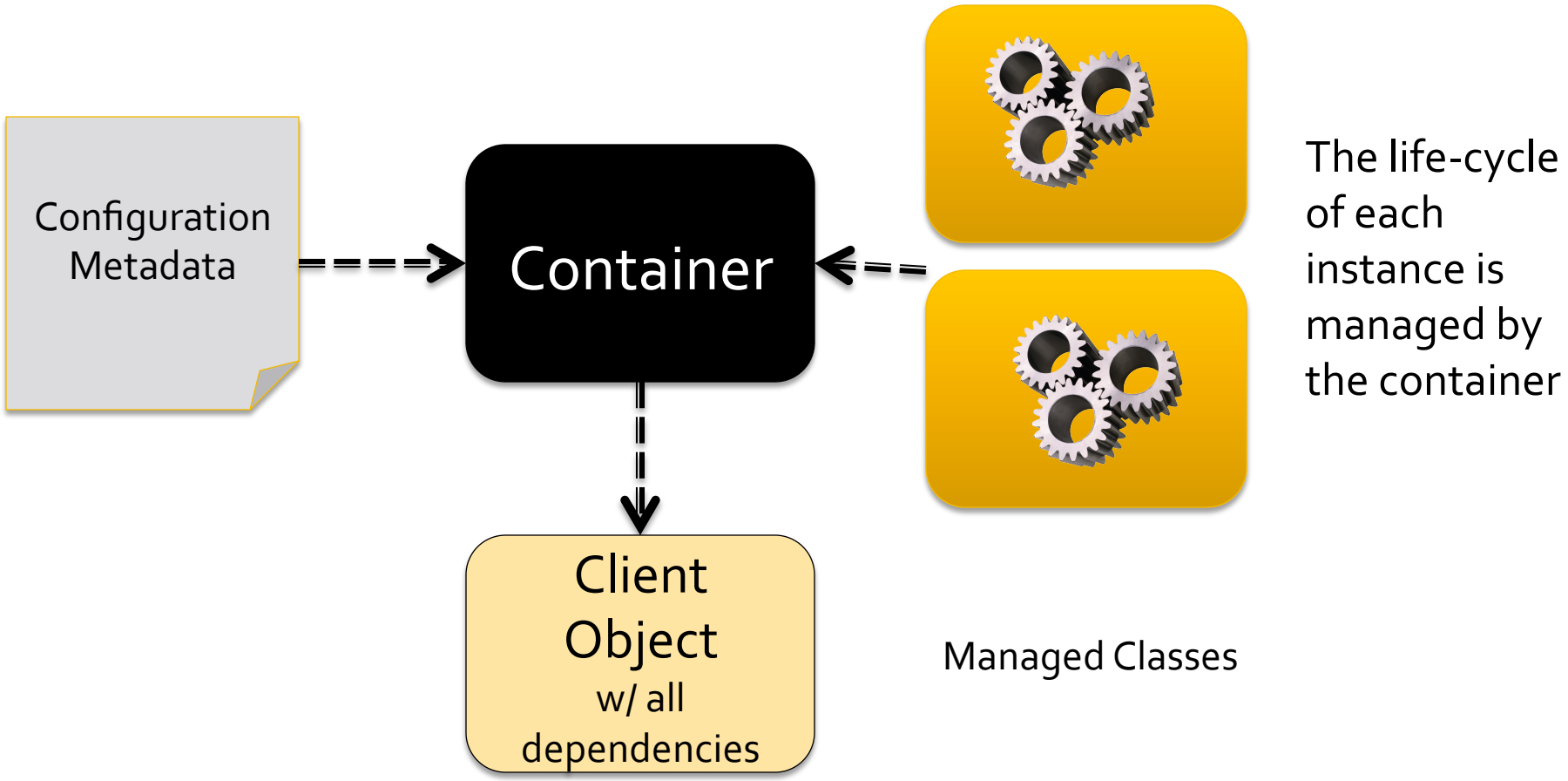
- Dependency Injection framework
  - introduced in 2002 in Rod Johnson's book
  - open sourced in 2003
  - Spring 1.0 available in 2004
- Ecosystem of Capabilities
  - Spring MVC
  - Spring Data
  - Spring Batch
  - Spring Integration
  - Spring Cloud



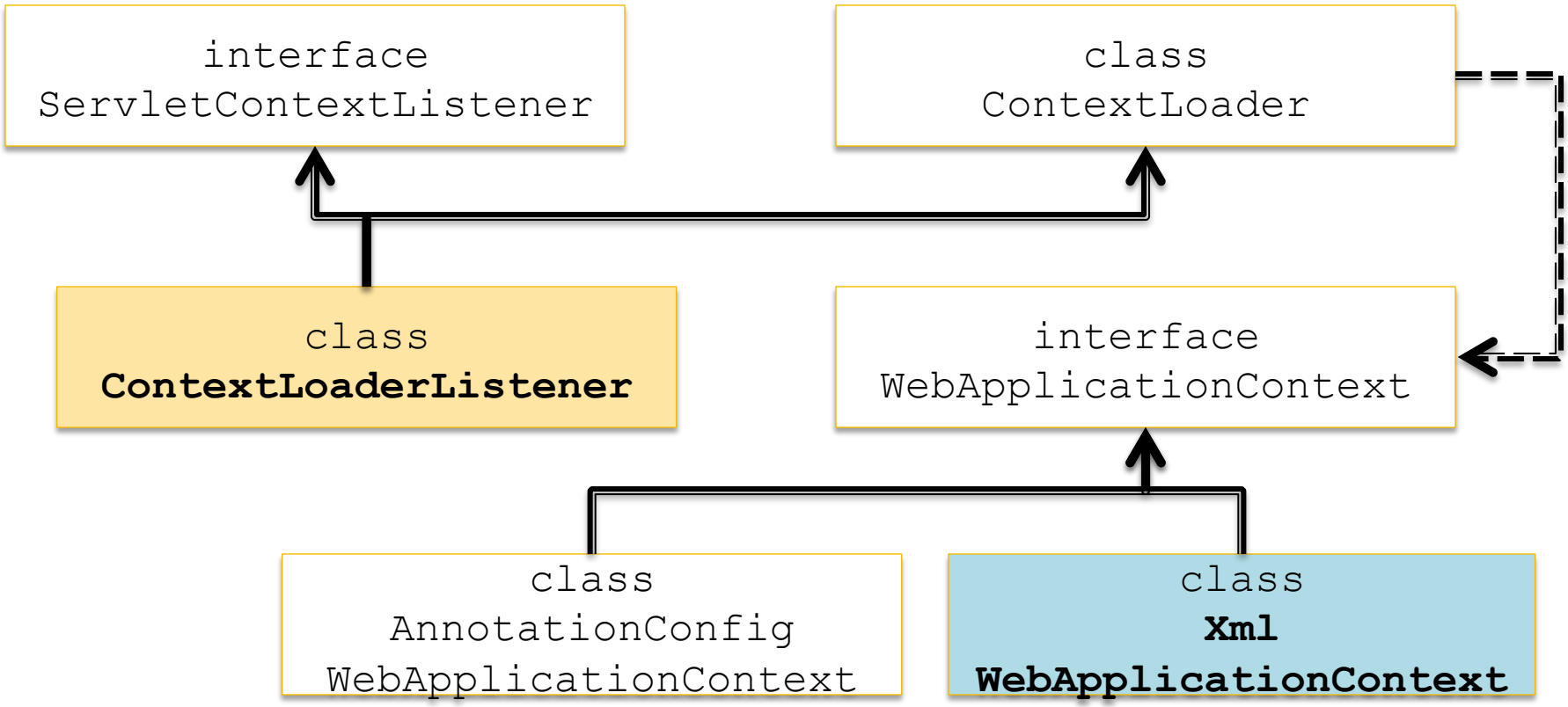
Dependency Injection Framework

- Decides how and when to **create** the client & dependent objects
  - creates objects
  - handles the dependencies of dependent objects
- Decides which type of dependency the client object will **use**
  - which Engine is injected into our Delorean
- Decides the **lifecycle** of the client and dependent objects
  - How many instances of the client object are allowed
  - will the client object get a singleton or a new instance of the dependency

Dependency Injection Framework

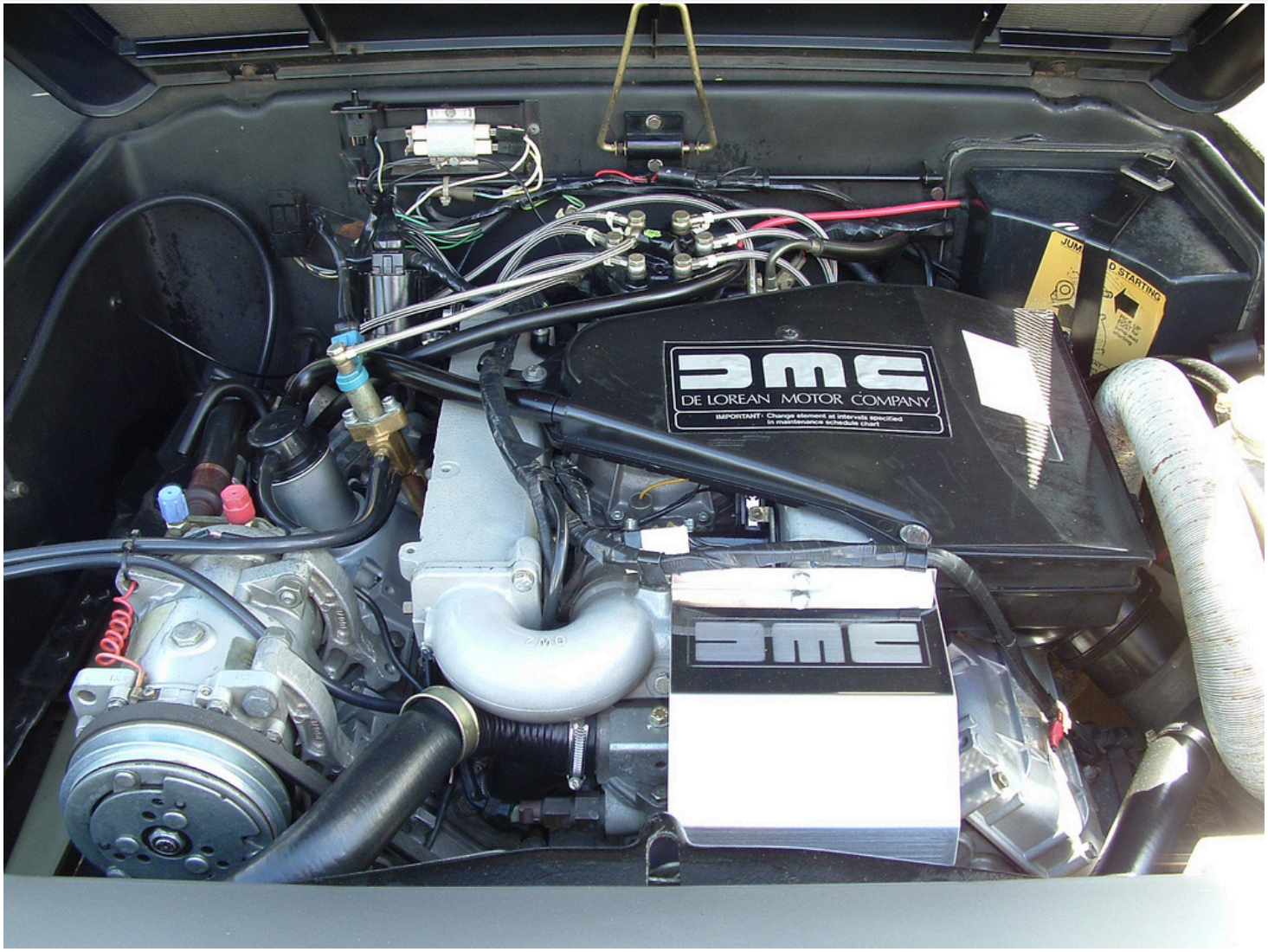


Typical Use Case: Servlet Container

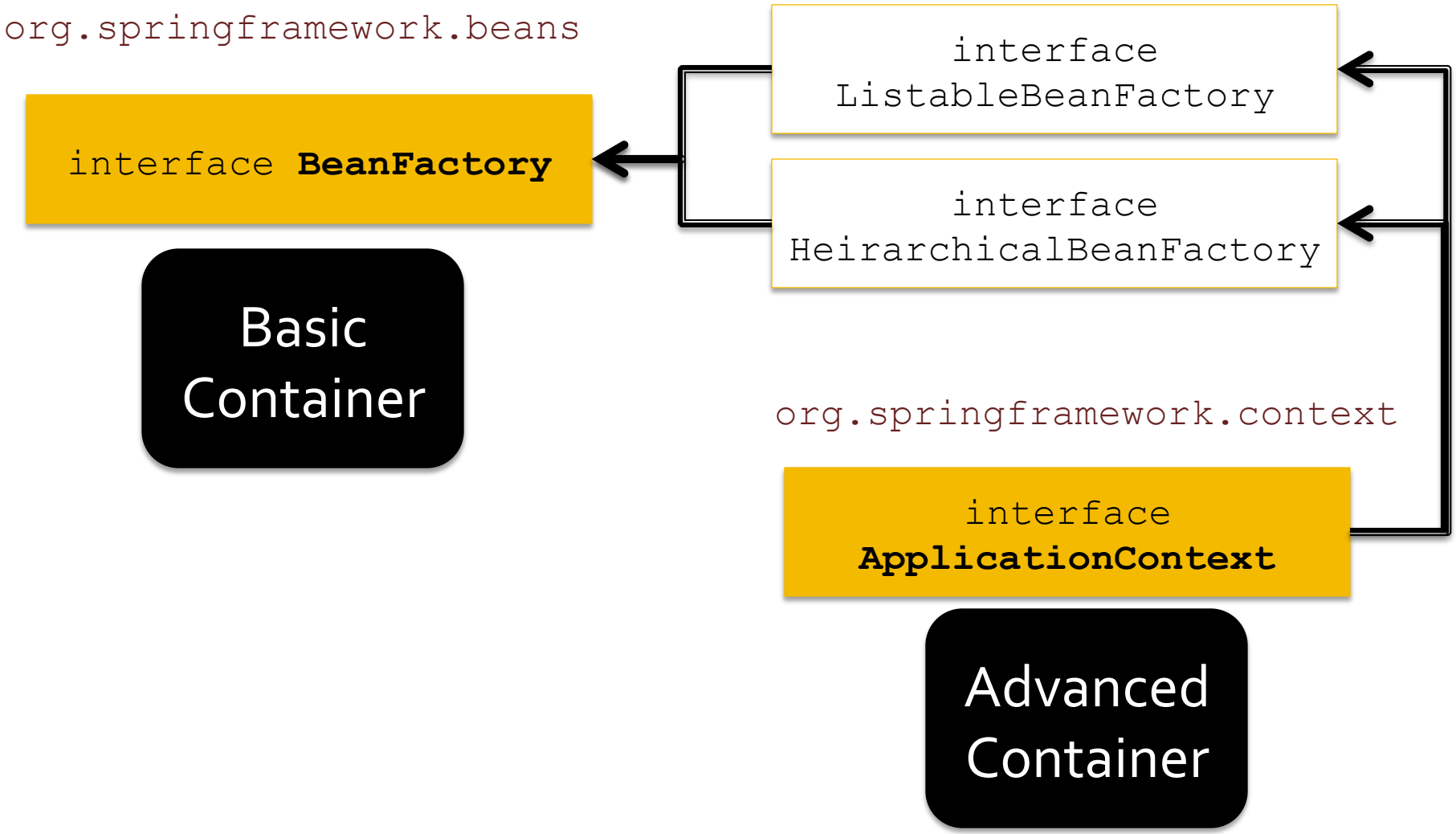


When configuring the ContextLoaderListener the default is XML configuration

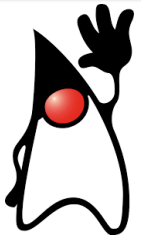
What is going on under the hood



Spring DI Container = Bean Factory



Basic Container with XML



```
DefaultListableBeanFactory factory = new
DefaultListableBeanFactory ();

XmlBeanDefinitionReader reader = new
XmlBeanDefinitionReader(factory);

reader.loadBeanDefinitions(
     new ClassPathResource("beans.xml"));

Car car = factory.getBean("dmc12”,Car.class);
car.go();
```

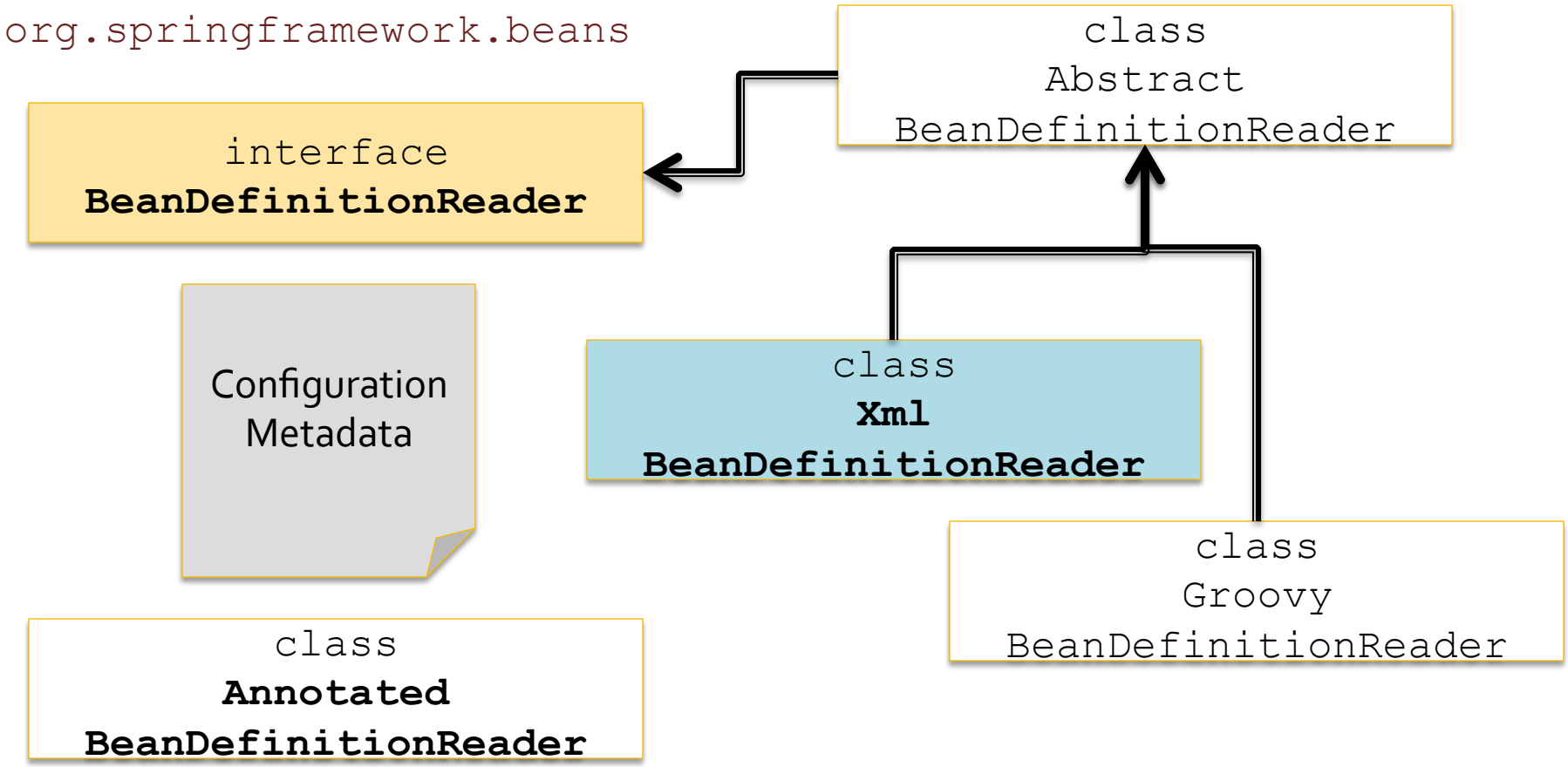
What is the difference?

- **`BeanFactory`** provides basic DI container
- **`ApplicationContext`** provides
  - Preconfigured `BeanDefinitionReader`
  - Automatic support for `BeanPostProcessor`
    - often used to create proxy to add capabilities
  - Automatic support for AOP
  - Automatic support for Transactions

7.16.1 BeanFactory or ApplicationContext?

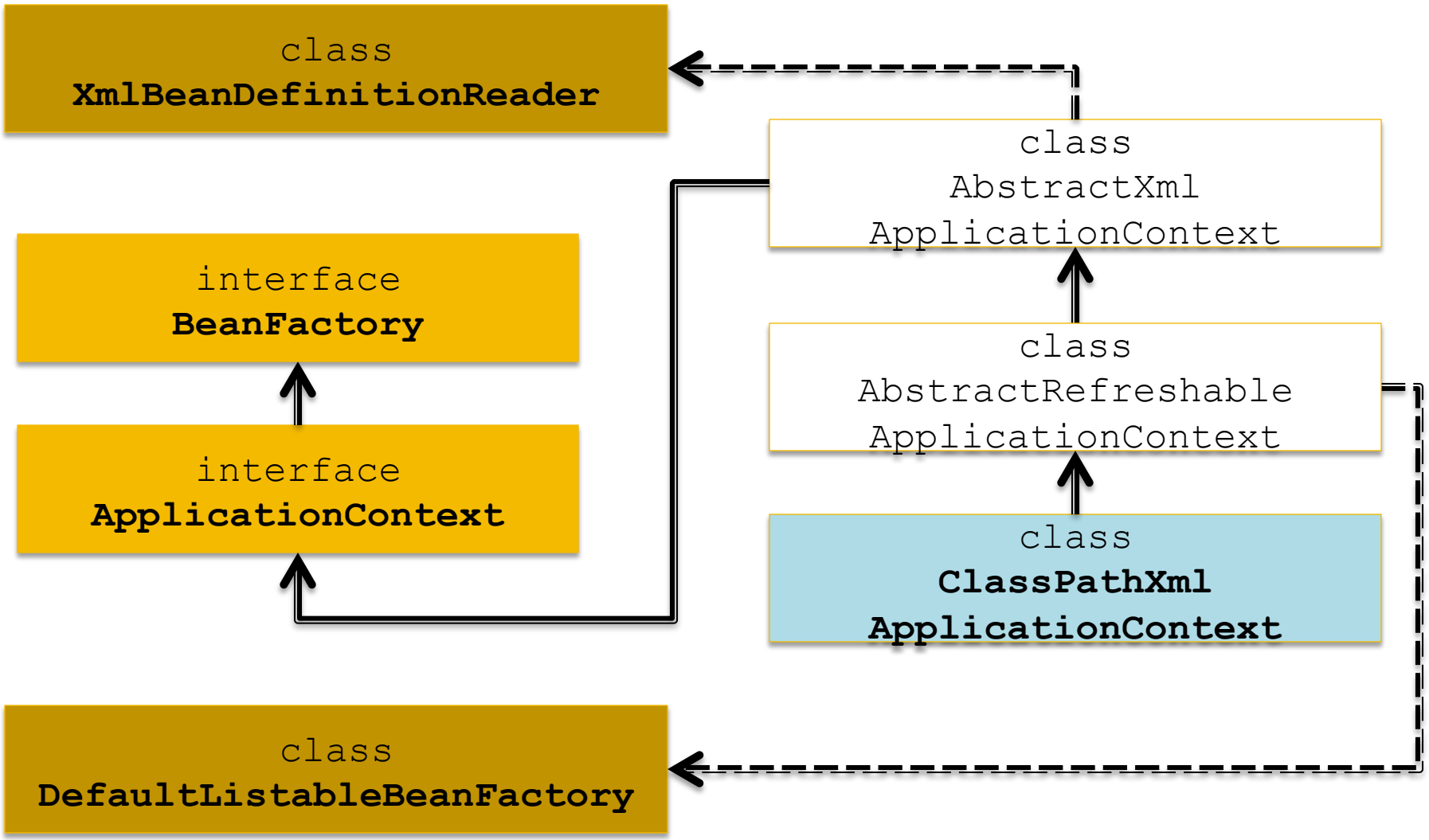
Use an `ApplicationContext` unless you have a good reason for not doing so.

Spring DI Configuration = BeanDefinitionReader



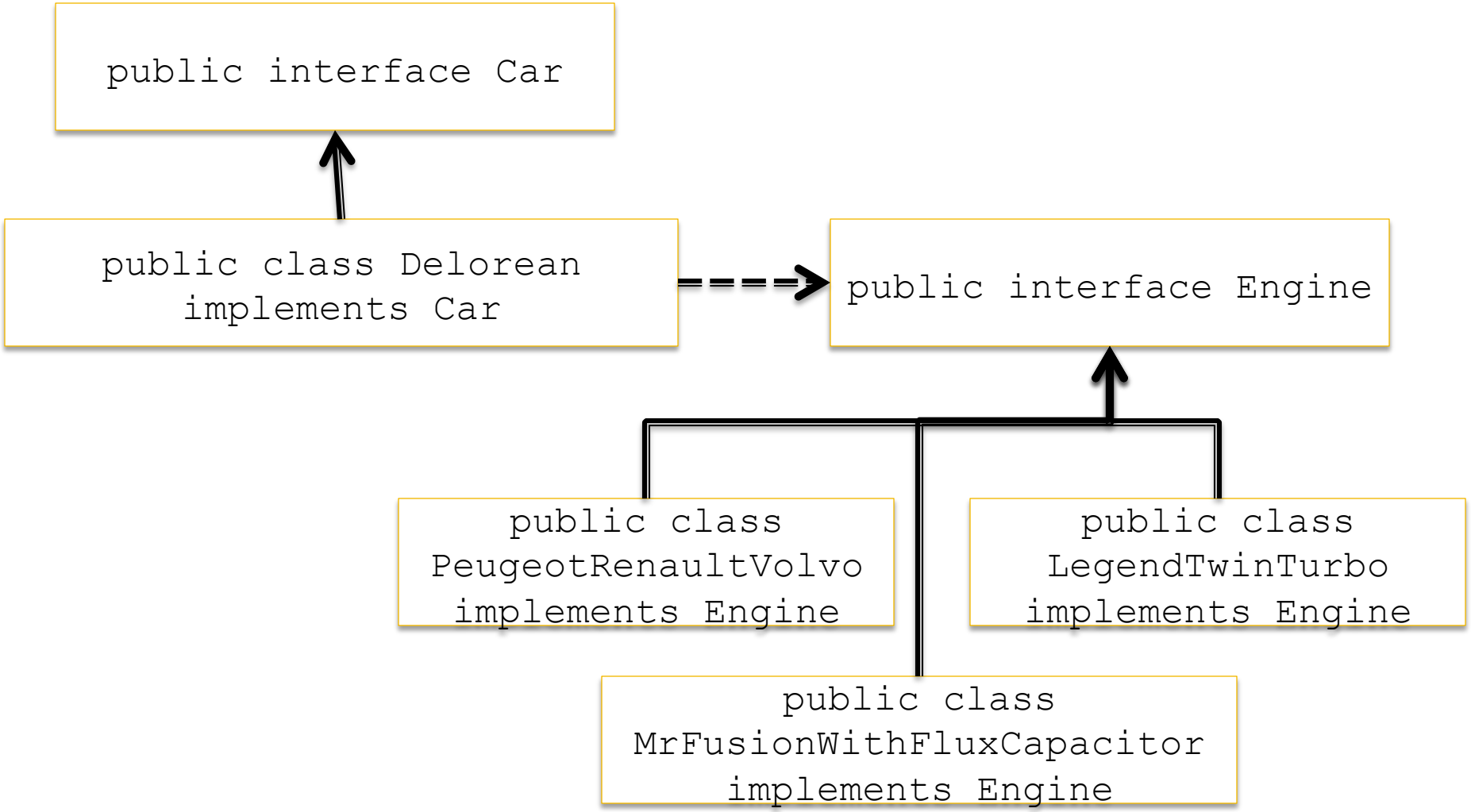
Does NOT implement the interface

ApplicationContext uses BeanDefinitonReader & BeanFactory

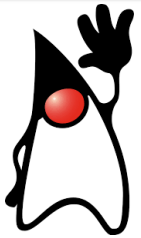


Using Delorean and Engine Design

DI + DIP


public interface Car
public class Delorean implements Car
public interface Engine
public class PeugeotRenaultVolvo implements Engine
public class LegendTwinTurbo implements Engine
public class MrFusionWithFluxCapacitor implements Engine

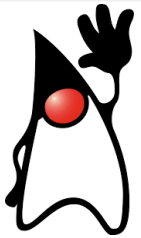
Configuration Options



- **XML**
  - original way to configure Spring

```
<bean id="dmc12" class=
    "codesmell.back2future.Delorean">
      <property name="engine" ref="prv"/>
 </bean>

<bean id="prv" class=
"codesmell.back2future.PeugeotRenaultVolvo"/>
```

Configuration Options

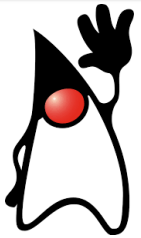


- **XML & Annotations**
  - Introduced in Spring 2.5
  - constructors & getter/setters can be removed

```
<context:annotation-config />

<bean id="dmc12"
class="codesmell.back2future.AutowiredDelorean" />

<bean id="engine"
class="codesmell.back2future.PeugeotRenaultVolvo" />
```

```
public class AutowiredDelorean implements Car {
    @Autowired
    @Qualifier("twin")
    private Engine engine;
```

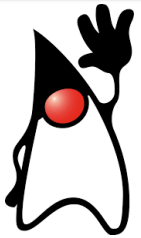
Configuration Options



- **XML & Annotations & Component Scan**
  - Introduced in Spring 2.5
  - constructors & getter/setters can be removed
  - don't need to configure any beans in XML

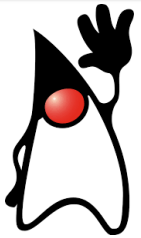
```
<context:component-scan base-package=
      "codesmell.back2future"/>

@Component(value="dmc12")
public class AnnotatedDelorean implements Car {
    @Autowired
    @Qualifier("twin")
    private Engine engine;
```

Configuration Options



- **Java & Annotations**
  - Introduced in Spring 3.0
  - constructors & getter/setters can be removed
  - Similar to XML & Annotations w/o XML
  - JavaConfig class takes place of XML

Configuration Options



- **Java & Annotations & Component Scan**
  - Introduced in Spring 3.0
  - constructors & getter/setters can be removed
  - Similar to XML & Annotations & Component Scan
  - JavaConfig class takes place of XML

Configuration Options

- **Groovy**
  - Groovy Bean Definition DSL (Grails)
  - Can be used instead of XML or Java Config

```
beans {
    groovyEngine(PeugeotRenaultVolvo){
    }

    groovyDelorean(Delorean){
        engine = groovyEngine
    }
}
```

Configuration Options

- **XML**
  - ClasspathXmlApplicationContext
  - FileSystemXmlApplicationContext
  - XmlWebApplicationContext
- **Java Config** *(available since 3.0)*
  - AnnotationConfigApplicationContext
  - AnnotationConfigWebApplicationContext
- **Annotations** *(available since 2.5)*
- **Groovy**
  - GenericGroovyApplicationContext

What's Next?

- the Bean Lifecycle in Spring
  - Singleton vs. Prototype
  - PostConstruct
  - PreDestroy
- What is going on inside the DefaultListableBeanFactory
- What is going on w/ BeanPostProcessors

Q & A