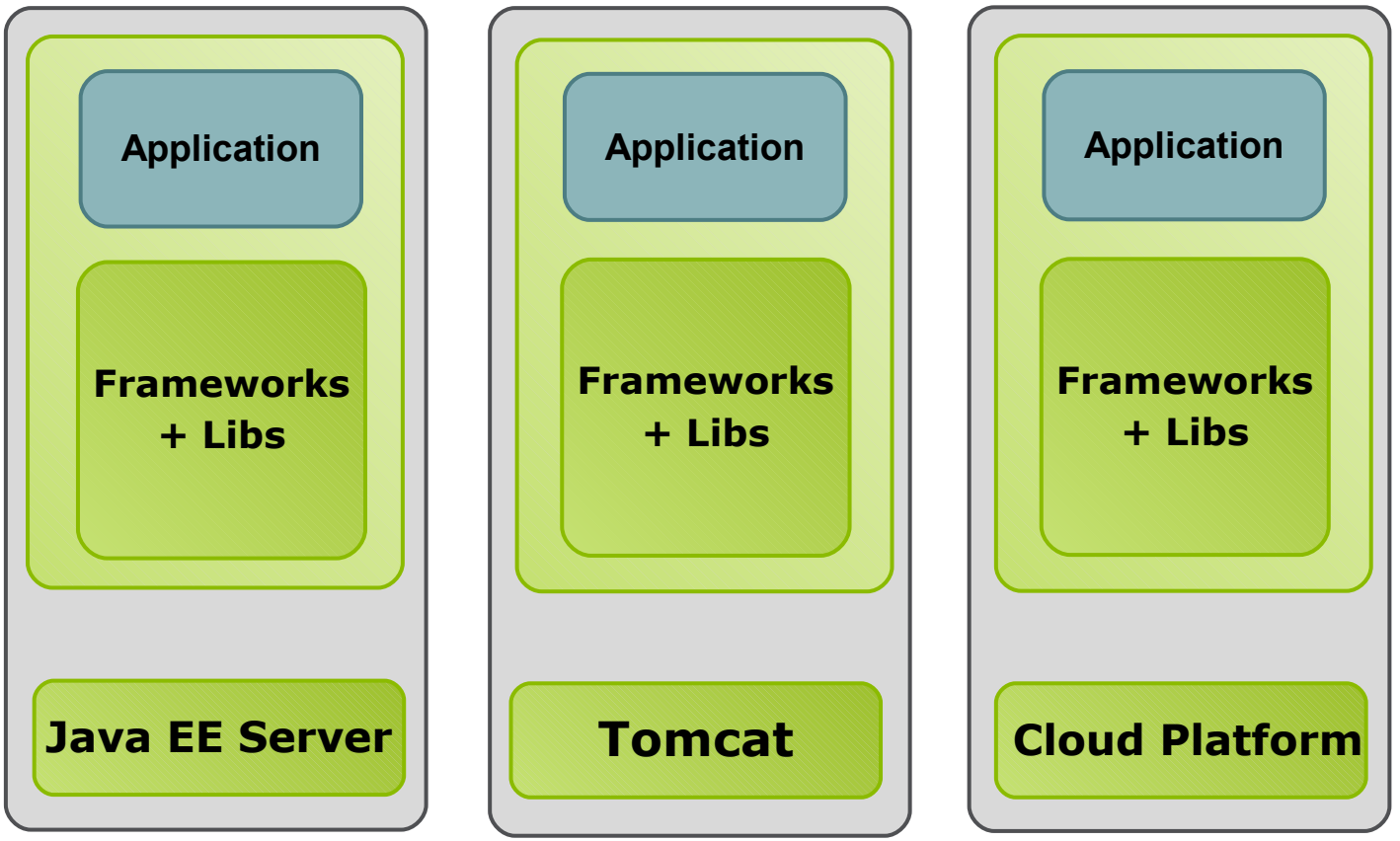
Enterprise Java in 2012 and Beyond
From Java EE 6 To Cloud Computing

Jürgen Höller, Principal Engineer, SpringSource

Deployment Platforms: Becoming More Diverse



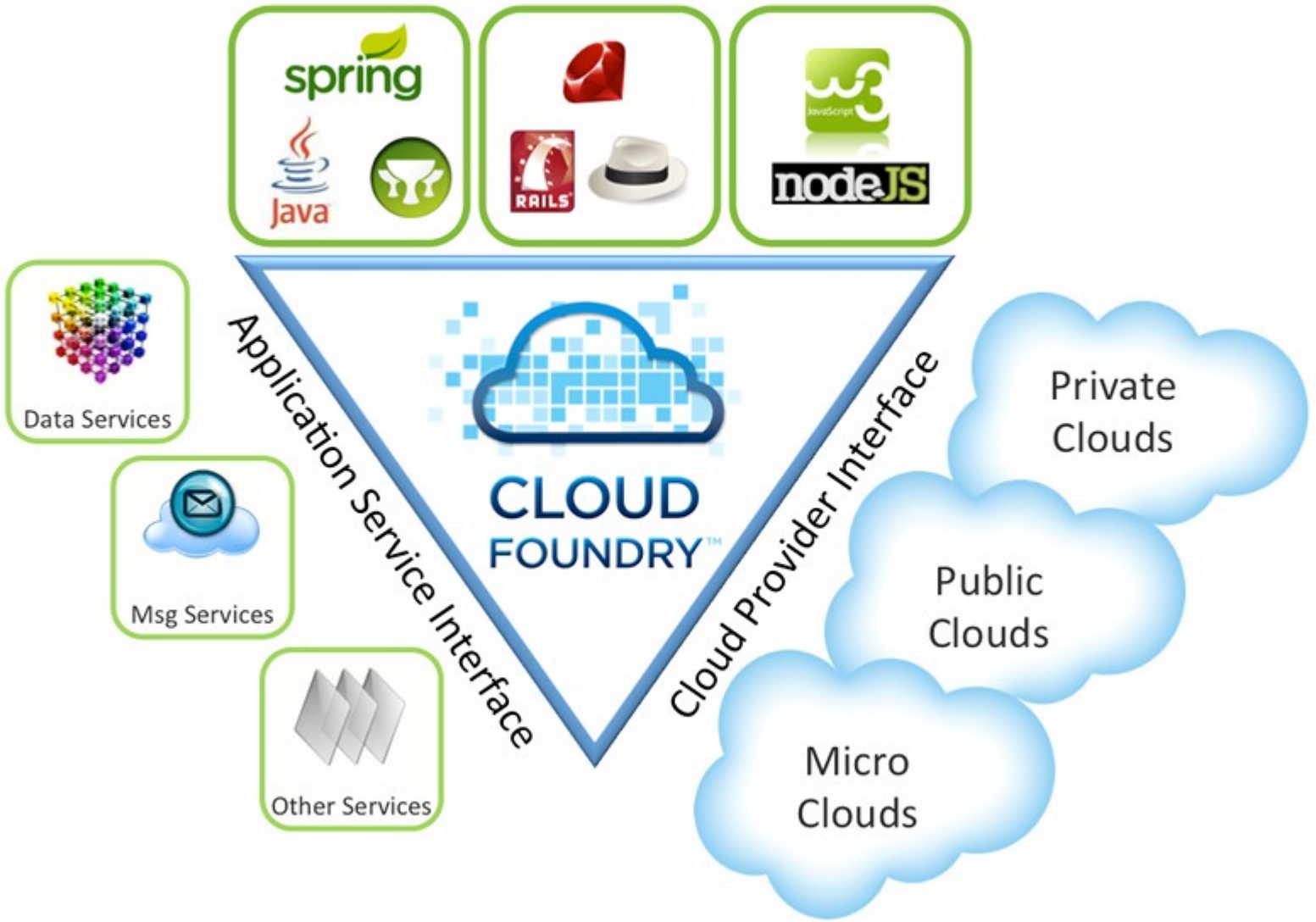
The State of Deployment Platforms in 2011/2012

- **Java EE servers having moved on to Java EE 6**
  - GlassFish 3
  - JBoss 7
  - WebSphere 8
- **Apache Tomcat having moved on to Tomcat 7**
  - Servlet 3.0 based (Java EE 6 level)
- **Cloud platforms becoming a serious option for regular Java web application deployment**
  - Google App Engine: Jetty++
  - Amazon Elastic Beanstalk: Tomcat++
  - VMware's CloudFoundry: Tomcat++

Cloud Platforms

- **a.k.a. „Platform as a Service“ (PaaS)**
  - „public cloud“: available through a shared, public host
  - „private cloud“: virtualization platform inside a corporate data center
- **Typically: a pre-installed web container with additional services**
  - datastores (not necessarily a relational database!)
  - messaging, clustered sessions, clustered cache, etc
- **The aforementioned Google App Engine and Amazon Elastic Beanstalk are good reference examples**
  - common ground: WAR deployment, Servlet API, JPA – ignoring Java EE?
  - several further offerings to show their promise in the course of 2012

CloudFoundry: VMware's Open Cloud Platform



Wide Variety of Data and Datastores

- **Not all data resides in relational databases**
  - cloud environments often suggest alternatives for scalability reasons
  - HBase, Redis, Mongo, Neo4j, etc
- **Distributed caches add challenges as well**
  - not least of it all in terms of application-level access patterns
  - GemFire, Coherence, Infinispan, etc
- **Hardly any standardization available**
  - JSR-107 – for caching – did not make progress for a long, long time
  - finally getting picked up in Java EE 7, but again only for caching
  - alternative datastore space is too diverse for standardization

Wide Variety of Web Clients

- **More and more client-side web technologies**
  - HTML 5 as a next-generation standard for desktop and mobile browsers
  - native Android and iOS clients on smartphones and tablets
- **Server-side state to be minimized or even removed completely**
  - in particular: no server-side user interface state
  - strictly controlled user session state
- **JSF's state-centric approach not a great fit anymore**
  - except for special kinds of applications (which it remains very useful for)
  - web application backends and web services based on JAX-RS / MVC style
  - nevertheless: JSF keeps evolving – JSF 2.2 coming up in 2012

HTML 5 on Mobile Devices

See mobilehtml5.org

Java SE 7: Concurrent Programming

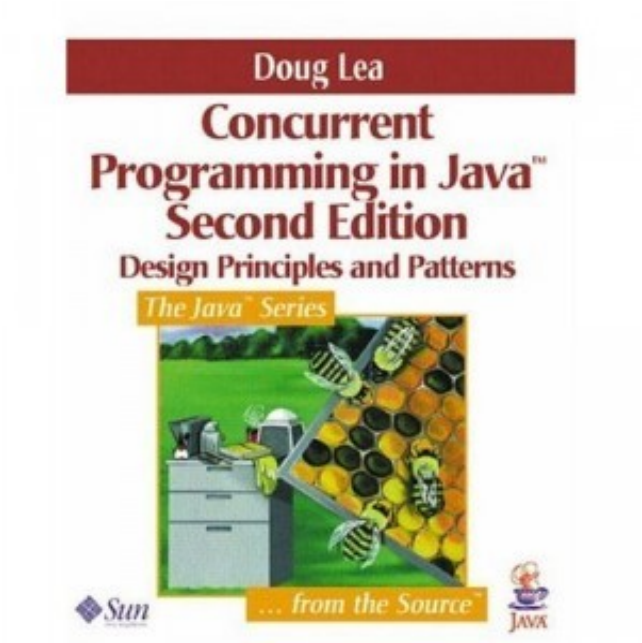
- **A challenge: concurrent programming in a multi-core world**
  - user-level APIs and recommended programming styles?
- **Servers with more cores than concurrent requests**
  - how to actually use your processor power in such a scenario?
- **Java SE 7: java.util.concurrent.ForkJoinPool**
  - specialized ForkJoinPools to be locally embedded within the application
  - different kind of pool, separate from regular Runnable-oriented Executors
- **Oracle's OpenJDK 7 released in summer 2011**
  - IBM JDK 7 followed surprisingly soon after

Scala & Akka: Concurrent Programming Revisited

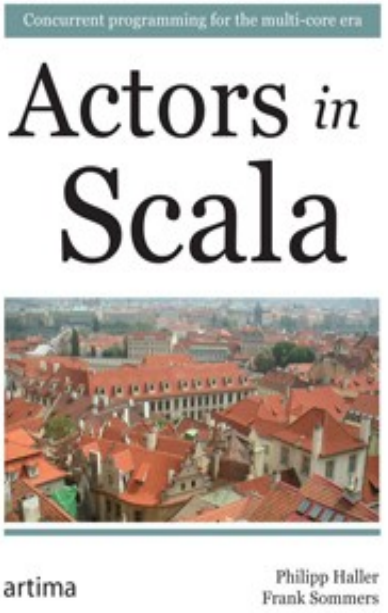
- **Scala as a next-generation language on the JVM**
  - combines object orientation with functional programming
  - particularly well suited for concurrent programming
  - integrates reasonably well with existing Java APIs

- **Akka as an actor-based framework for Scala and Java**
  - event-driven architectures
    - strong architectural impact (if you fully go for it)
  - different approach towards concurrent programming
    - raises the concurrency abstraction level (but not too much)
  - provides Scala and Java APIs
    - however, being most convenient with Scala message passing

Concurrent Programming: Then and Now



The classic one



Higher-level patterns for 2012 and beyond

Java EE 6 Revisited

- **How relevant is Java EE 6 in the context of recent trends?**
  - as usual, Java EE 6 tends to solve yesterday's problems...
  - the fate of specifications with a multi-year expert group process
  - even worse, EE server vendors take years to implement a full platform release

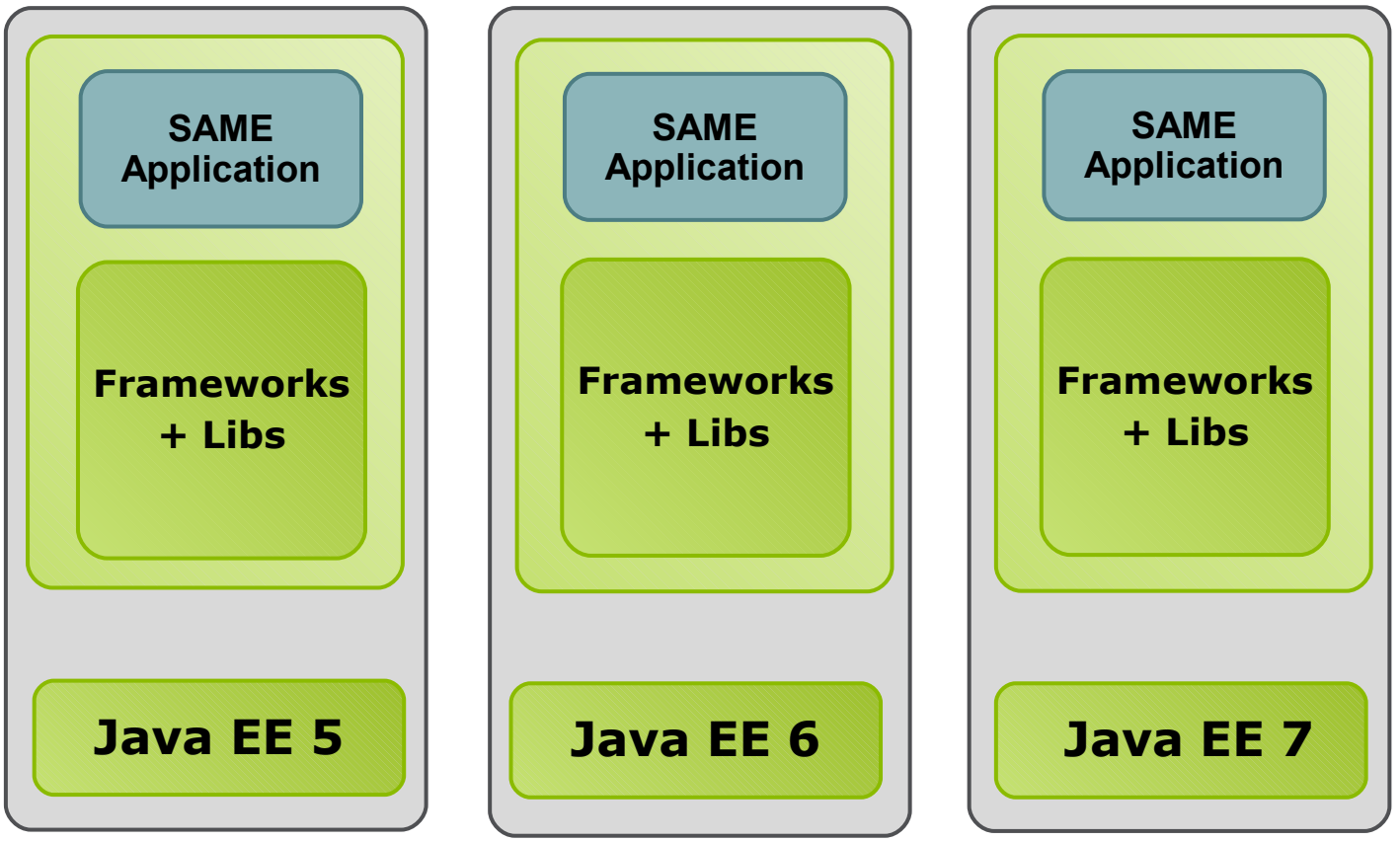
- **Some recent trends change this industry quite rapidly and radically**
  - cloud platforms challenge the notion of dedicated servers
  - alternative datastores challenge relational databases and their access APIs
  - concurrent programming trends do not match traditional EE assumptions

- **Java EE 7 to the rescue in 2012?**
  - Let's see...

A Quick Preview: Java EE 7

- **A broad set of expected updates**
  - JCache (JSR-107)
  - JAX-RS 2.0
  - JMS 2.0
  - JPA 2.1
  - EJB 3.2
  - CDI 1.1
  - Bean Validation 1.1
  - Servlet 3.1
  - JSF 2.2
- **Key theme: multi-tenancy for cloud environments**

Application Portability: Platform Generations?



Java EE 7 Timeline

- **Java EE 7 is certainly going to deliver key foundational updates**
  - the key question is: when, and how well adopted by major vendors?

- **Umbrella specification scheduled to go final in Q4 2012**
  - more realistically, most individual specifications going final in late 2012
  - with the full EE 7 umbrella specification not being ready before 2013

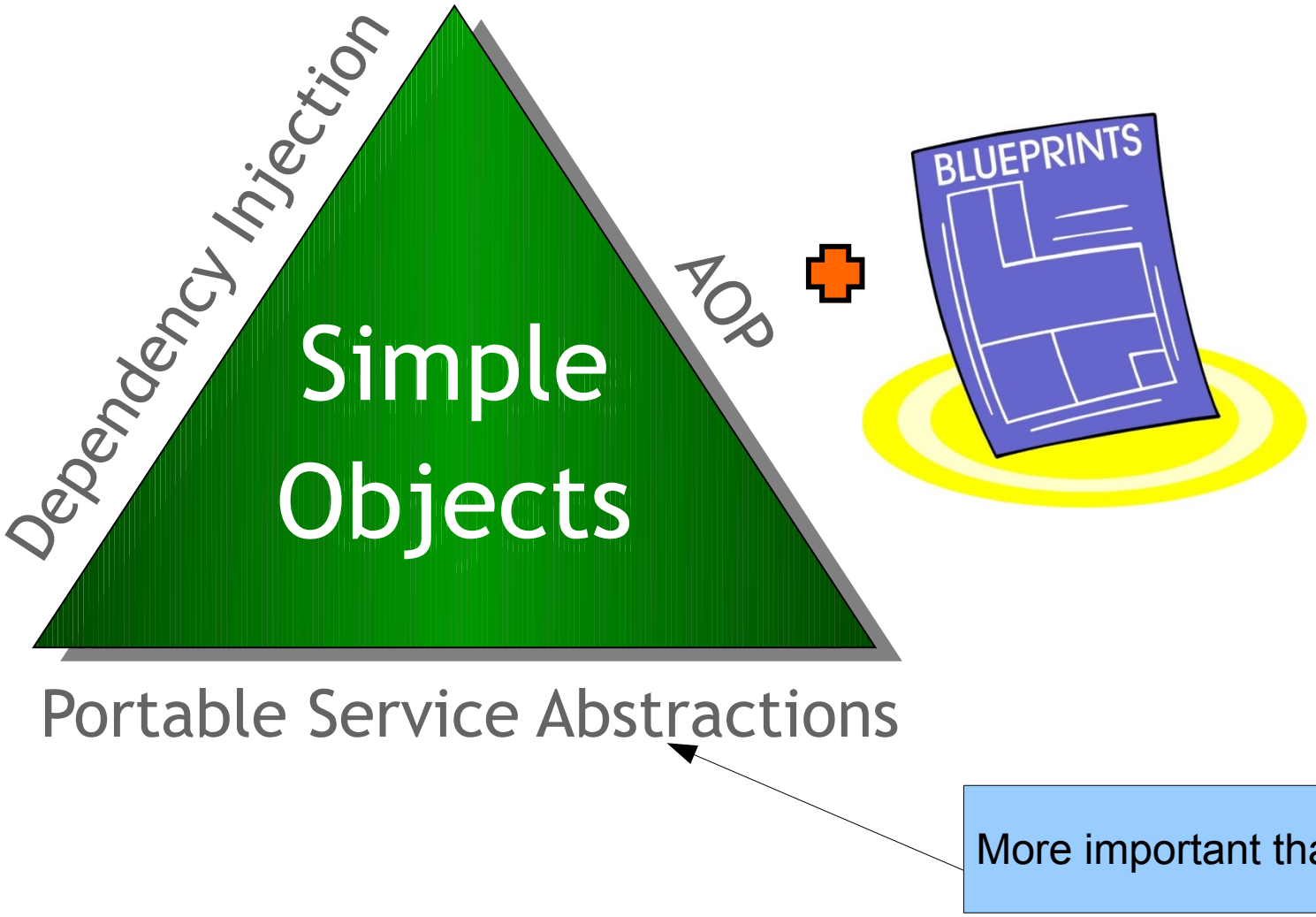
- **Individual implementations for JCache, JMS 2.0, JPA 2.1 etc**
  - to be available in late 2012, for immediate use on Tomcat etc
  - embedded into the application's deployment unit
  - hard to use on top of existing EE servers due to API version conflicts

Java EE in Cloud Environments

- **Java EE 6 in mainstream cloud environments?**
  - Red Hat's OpenShift is based on JBoss AS 7 now
    - nearly two years after the Java EE 6 specification release
  - however, Oracle's cloud offering is just about to become fully Java EE 6 compatible in early 2012
    - more than two years after the Java EE 6 specification release

- **Key problem: major cloud vendors not adopting Java EE at all**
  - but rather a more minimal selection of Java platform services
    - typically Java SE + Servlet + JPA + custom cloud service APIs
  - reconsider Google App Engine and Amazon Elastic BeansTalk
    - also Cloud Foundry, Heroku, etc

Key Elements of Spring: Ready for 2012 & Beyond



Environment Abstraction

- **Grouping bean definitions for activation in specific environments**
  - e.g. different stages: development, testing, production
  - e.g. different deployment environments: Tomcat, EE server, Cloud Foundry
  - resolution of placeholders from environment-specific property sources
- **Environment association of specific bean definitions**
  - XML 'profile' attribute on <beans> element
  - @Profile annotation on configuration classes
  - @Profile annotation on individual component classes
- **Ideally: no need to touch the deployment unit across different stages/environments**

Cache Abstraction

- **CacheManager and Cache abstraction**
  - in org.springframework.cache
    - which up until 3.0 just contained EhCache support
  - particularly important with the rise of distributed caching
    - not least of it all: in cloud environments

- **Backend adapters for EhCache, GemFire, Coherence, etc**
  - EhCache adapter shipping with Spring core
  - plugging in custom adapters if necessary

- **Specific cache setup per environment profile?**
  - potentially adapting to a runtime-provided cloud caching service

Spring Web Applications on Servlet 3.0

```
/**
 * Automatically detected and invoked on startup by Spring's
   ServletContainerInitializer. May register listeners, filters,
   servlets etc against the given Servlet 3.0 ServletContext.
 */
public class MyWebAppInitializer implements WebApplicationInitializer {


    public void onStartup(ServletContext sc) throws ServletException {
        // Create the 'root' Spring application context
        AnnotationConfigWebApplicationContext root =
                new AnnotationConfigWebApplicationContext();
        root.scan("com.mycompany.myapp");
        root.register(FurtherConfig.class);


        // Manages the lifecycle of the root application context
        sc.addListener(new ContextLoaderListener(root));
        ...
    }
}
```

Support for Java SE 7 & Java EE 7

- **Java SE 7 is an important driver for Spring 3.x**
  - making best possible use of JRE 7 at runtime
  - support for JDBC 4.1 in Spring 3.1
  - as of Spring 3.2, building the framework against Java 7
    - while preserving runtime compatibility with Java 5 and 6

- **Early support for Java EE 7 related specifications**
  - coming in this year's Spring 3.2 generation as well
  - JCache
  - JMS 2.0
  - JPA 2.1
  - JSF 2.2
  - Bean Validation 1.1

Forward Compatibility with Java SE 8

- **Java 8's language enhancements in mind already**
  - preparing Spring APIs for Java 8 lambda expressions
    - a.k.a. Java 8 closures
- **“Single Abstract Method“ (SAM) types**
  - interfaces with one method
  - common in Spring already
    - ResultSetExtractor
    - RowMapper
    - MessageCreator
    - TransactionCallback
- **Java 8 enhancements will work with existing versions of Spring**
  - once JDK 8 is released

Beyond Spring Framework: Recent Key Projects

- **Spring Data**
  - support for many alternative datastores
  - GemFire, Hadoop, Redis, Mongo, Neo4j

- **Spring AMQP**
  - messaging beyond JMS, e.g. for RabbitMQ

- **Spring Mobile & Spring Android**
  - conveniences for mobile app development

- **Spring Social**
  - programmatic access to social networks

Summary

- **Disruptive forces approaching the Enterprise Java space**
  - deployment to cloud platforms
  - access to alternative datastores
  - HTML 5 based web architectures
  - concurrent programming challenges

- **Common ground in deployment platforms is once again unclear**
  - selected specifications (Java SE, Servlet, JPA) as one key ingredient
  - 'proprietary' APIs and embedded frameworks as another key ingredient

- **Either way, there are exciting times ahead of us. Let's embrace them!**