The Spring Framework
“j2ee made easy”

Keith Donald
Software Architect
Computer Science Innovations (CSI)
kdonald@csi-inc.com
http://www.jroller.com/page/kdonald

Agenda

- What Spring Is
- What attracted me to the project
- What sets it apart
- The architecture
- The value-add solutions
- All the technical details you can stand

Spring… It's All in the Name

- *Spring-to-life:* an enabling jumpstart
- *Spring fever:* a philosophy, a passion
- *A new season:* A much needed relief from old-man-winter (heavy J2EE)

Why Spring is the "*Season*" for Me

- A focused mission
- An extremely active, vibrant community
- The developers eat their own dog food
- Emphasis on documentation and testing
- Fosters integration, enables productivity

Spring = Solutions to market faster

Spring Core Competencies

- The team
  - Experts in *the practical application of* J2EE
- The product
  - The best Inversion of Control (IoC) container available
  - Addresses end-to-end requirements, not just one piece
  - Powerful, yet elegantly simple

Spring - Core Architecture

- Manages component lifecycle and dependencies
- Centralizes and externalizes application configuration information
- Decouples business logic from infrastructure
- Decorates components with cross-cutting behavior
- Fully portable across deployment environments

Spring - Solutions

- Solutions address major J2EE problem areas:
  - Web application development (MVC)
  - Enterprise Java Beans (EJB, JNDI)
  - Database access (JDBC, iBatis, ORM)
  - Transaction management (JTA, Hibernate, JDBC)
  - Remote access (Web Services, RMI)
- Each solution builds on the core architecture
- Solutions foster integration, they do not re-invent the wheel

Spring

j2ee Application Framework

Before Spring

- Proprietary, buggy configuration code
- Proliferation of Singletons, Service Locators (glue code)
- Copy-and-paste code, ugly exception handling
- Tied by invasive infrastructure (EJB/JNDI), application server
- Testing? What's that?

Sad

After Spring

- One elegant way of configuring everything
- Dependency Injection – components are handed only what they need
- Cohesive, reusable business logic
- No dependency on infrastructure APIs or container
- Easy, comprehensive testing

Euphoric!

Key Point

With Spring,

your business needs drive the technology,

the technology doesn't drive you!

Key Point

Spring applies OO best practices and integrates best-of-breed technologies to provide a "one-stop-shop" for building high-quality enterprise applications quickly.

Spring Core: The Bean Factory

- **The most complete IoC container**
  - Setter Dependency Injection
    - Configuration via JavaBean properties
  - Constructor Dependency Injection
    - Like PicoContainer: configuration via constructor arguments
  - Dependency Lookup
    - Avalon/EJB-style call-backs

Why all the IoC hype?

- The Hollywood Principle
  - "Don't call me, I'll call you"
- The Law of Demeter
  - "Don't talk to strangers"

IoC: What's the point again?

- **Testing** becomes easy
  - You can test the component in isolation
- **Maintenance** becomes easy
  - Loose coupling facilitates local changes
  - Clear separation of concerns improves modularity
- **Configuration** becomes easy
  - Decoupled from application logic
  - Easy to vary based on context
- **Reuse** becomes easy
  - Loosely coupled components can be reused

Spring: Dependency Injection

- Setter Injection
  - Configuration XML

```
<bean id="hibernateClinic" class="samples.HibernatePetClinic">
   <property name="sessionFactory">
        <ref bean="sessionFactory"/>
    </property>
</bean>
```

  - Java Code

```
public class HibernatePetClinic implements PetClinic {
    public void setSessionFactory(SessionFactory factory) {
        this.sessionFactory = factory;
    }
}
```

Spring: Dependency Injection

- Constructor Injection
  - Configuration XML

```
<bean id="hibernateClinic" class="samples.HibernatePetClinic">
   <constructor-arg index="0">
        <ref bean="sessionFactory"/>
    </constructor-arg>
</bean>
```

  - Java Code

```
public class HibernatePetClinic implements PetClinic {
    public HibernatePetClinic(SessionFactory factory) {
        this.sessionFactory = factory;
    }
}
```

Spring Core: The Bean Factory

- Lifecycle support – initialize, destroy
- Singleton or prototype service models
- Lazy or eager instantiation
- Dependency checking
- Dependency auto-wiring
- Automatic property type conversion
- Support for nested properties, collections

Spring Core: The Bean Factory

- Hooks, Hooks, Hooks!
  - BeanFactoryPostProcessor
    - Apply custom processing to bean definitions
      - For example, resolve properties by `${name}` from another source (PropertyPlaceholderConfigurer)
  - BeanPostProcessor
    - Apply custom processing after a bean is instantiated.
      - For example, wrap a bean in a proxy providing additional *cross-cutting* behavior

Spring Core: The Bean Factory

- Factory Bean, special kind of bean
  - Encapsulates object creation or lookup logic
  - Provides a layer of indirection
  - Examples
    - JNDIFactoryBean
      - Abstracts away JNDI lookup from clients
    - TransacationProxyFactoryBean
      - Makes a target object transactional
    - StatelessSessionProxyFactoryBean
      - Facilitates codeless EJB clients

Spring Core: Factory Bean

- Example – StatelessSessionProxyFactoryBean
  - Proxy declaration

```
<bean id="ejbServiceProxy" class="spring.LocalStatelessSessionProxyFactoryBean">
   <property name="jndiName">
       <value>myEjb</value>
   </property>
   <property name="businessInterface">
       <value>com.mycompany.MyBusinessInterface</value>
   </property>
</bean>
```

  - Inject "Business Interface" to Client

```
<bean id="myAction" class = "samples.myAction">
    <property name="myService">
        <ref bean="ejbServiceProxy"/>
    </property>
</bean>
```

Client has no idea "myService" is backed by a EJB!

Spring Core: The Enablers

- IoC - Yea we know already…
- AOP – Aspect Oriented Programming
  - Compliments OO
  - Empowers modularization of cross-cutting concerns

IoC + AOP = a match made in heaven!

Spring Core: IoC+AOP Synergy

- Powerful AOP framework
- Built on Spring IoC for easy configuration
- Out-of-the-box aspects address common concerns

Spring Core: AOP

- Example - StatelessSessionProxyFactoryBean
- How does it work?
  - Creates a dynamic AOP proxy
    - Implements your POJO business interface
    - A single interceptor intercepts each method invocation
      - Creates and invokes the target EJB transparently
      - Re-throws any unrecoverable exceptions as unchecked
  - Chains of interceptors can be easily configured
    - Example:
      - Performance monitor interceptor, then invoke EJB.

Spring Core: AOP

- Example: PerformanceMonitor + EJBInvoker

```
<bean id="monitoredEjb" class="spring.aop.ProxyFactoryBean"/>
    <property name="proxyInterfaces">
          <value>com.mycompany.MyBusinessInterface</value>
    </property>
     <property name="interceptorNames">
        <list>
            <value>performanceMonitor</value>
            <value>ejbInvoker</value>
        </list>
     </property>
</bean>

<bean id="performanceMonitor" class="samples.PerformanceMonitorInterceptor"/>
<bean id="ejbInvoker" class="spring.ejb.RemoteStatelessSessionBeanInvoker">
    <property name="jndiName">myEjb</property>
</bean>
```

Now I can see how slow remote method calls on EJBs really are…
with no change required in application code! ☺

Spring Core: AOP

- Example - TransacationProxyFactoryBean
  - Wraps a POJO in a transactional proxy
  - Methods to advise are fully configurable
  - How does it work?
    - Interceptor intercepts each method invocation
    - If the method is marked transactional:
      - Create or reuse a transaction appropriately
      - Commit or rollback based on configuration policy

Spring Core: AOP

- Example: TransacationProxyFactoryBean

```
<bean id="transactionalService" class="spring.TransactionProxyFactoryBean"/>
    <property name="target">
        <ref bean="myBankingService"/>
    </property>
    <property name="transacationManager">
        <ref bean="localTransacationManager"/>
    <property name="transactionAttributes">
        <props>
            <prop key="transfer*">PROPAGATION_REQUIRED</prop>
            <prop key="withdraw">PROPAGATION_REQUIRED</prop>
            <prop key="deposit">PROPAGATION_REQUIRED</prop>
            <prop key="*">PROPAGATION_REQUIRED,readOnly</prop>
        </props>
    </property>
</bean>
```

Now I can have pluggable, declarative transactions without the overhead of an EJB container… ***again, with no change required in application code!*** ☺

Spring Core AOP: Gettin' Fancy

- Auto proxy facility
  - Automatically proxy beans matching a criteria
- Source-level metadata
  - Markup configuration as .NET-style attributes
  - Concise alternative to XML
  - Keeps configuration metadata close to where it applies

Spring Solutions: Web MVC

- Need to build a web-app?
  - Spring offers a powerful, easy-to-use MVC framework
- Wish to integrate your existing web-app with a Spring middle tier?
  - Struts
    - http://struts.sourceforge.net/struts-spring/
  - Web Work
    - http://wiki.opensymphony.com/space/Spring+Framework+Integration
  - Tapestry
    - http://www.springframework.org/docs/integration/tapestry.html

Spring Solutions: DAO JDBC

- Offers much simpler programming model than raw JDBC
- No more try/catch/finally blocks
- No more leaked connections
- *Meaningful* exception hierarchy
  - Spring auto-detects database and knows what ORA-917 means
  - No more vendor code lookups
  - More portable code
- Stored procedure support
- Ideal for the places Hibernate doesn't go

Spring Solutions: DAO ORM

- Manages Hibernate sessions
  - No more ThreadLocal sessions
  - Sessions are managed by Spring declarative transaction management
- HibernateTemplate makes common operations easy
- Consistent exception hierarchy
  - Runtime exceptions (Gavin King now believes Hibernate should have opted for unchecked exceptions)
- Mixed use of Hibernate and JDBC *within the same transaction*
- JDO support

Spring Solutions - Remoting

- Synchronous binary RPC protocols
  - Hessian
  - RMI
- XML-based protocols, web-services
  - Apache Axis (SOAP)
  - Burlap
- Viable option for lightweight distributed solutions

There's more!

- Full Internationalization support
- Elegant resource abstraction layer
- Job Scheduling support
  - Quartz Integration
- Velocity / FreeMarker support
- Java mail abstraction layer
- Samples, Samples, and more Samples

Spring Roadmap – What's Next?

- ***Spring 1.0 final out this week! It's here to stay!***
- Spring 1.1
  - JMX, JMS support
  - Enhanced lifecycle support
  - Declarative, rules-based validation
  - More aspects!
- Sub-projects
  - Rich client platform (RCP)
  - Eclipse plug-in
  - Declarative security framework
- **Spring 1.1 RC1 planned for late April**

The Spring Community

- 14 developers (I'm lucky number 13!)
- Rod Johnson/Juergen Hoeller
- Test-first development
- *Very* active mailing lists and forums on SourceForge
- JIRA Issue tracker at Atlassian, Confluence Wiki coming soon
- DocBook reference manual
- www.springframework.org
- At least three books coming out in 2004:
  - *Spring Live* (May): Matt Raible
  - *J2EE Without EJB* (May): Johnson/Hoeller
  - *Professional Spring Development* (Q4): Johnson/Risberg/Hoeller/Arendsen
- Over 15K downloads before 1.0 RC1

Got Spring? They Do!

- Banking
  - Global investment bank: two projects live with Spring MVC, IoC, JDBC; 10,000 users
  - German domestic bank (rolling Spring out in new projects)
- Healthcare
  - Two large US healthcare organizations
- Computer Science Innovations – software consultancy
- Numerous websites
  - Canadian web consultancy
  - Austrian web consultancy
  - Dutch consultancy: major European delivery tracking service
  - FA Premier League

Questions?