Spring - A Manager's Overview

Prepared by: Eberhard Wolff, Principal Consultant, SpringSource

Summer 2008

SpringSource 411 Borel Ave, Suite 101, San Mateo, California 94402, USA **T** +1 650 425 3515 **F** +1 650 212 3515 **E** insidesales@springsource.com

Executive Summary

This white paper describes the core technologies and values that form the foundation of the Spring Framework and discusses the business value that results from using Spring to build enterprise applications.

The first section briefly describes the history behind Spring and the core values that led to its creation. This is followed by an overview of the architecture of the Spring Framework, explaining the basics concepts of Dependency Injection, Aspect-Oriented Programming and Enterprise Service Abstraction. As Spring is often used in conjunction with Java EE, the next section explains the relation to this technology as well as the differences and advantages of Spring over EJB 1.x, 2.x and 3.x. Spring's exceptionally high quality standard is covered next and as well as Spring's market share, which shows Spring to be the de facto standard. The concepts behind Spring have created much more than just the core framework, so the next section reviews the complete Spring Portfolio and the platforms that Spring can run on. SpringSource, the company behind Spring, is discussed next, including the products and services that provide compelling business value. The penultimate section is a list of case studies that successfully used Spring in large scale projects and the last section provides a simple conclusion.

The Spring Framework

History

After its initial launch in 1997, Java quickly became the predominant platform for mission-critical enterprise applications. An important reason for this success was the standardization of several common services for enterprise application as J2EE (Java 2 Enterprise Edition) later called Java EE. This platform managed to get backing by several large companies such as BEA (to become part of Oracle), Oracle, IBM, Sun and SAP. Despite this success, many Java EE projects failed and the platform seemed overly complex, missing some features while other were not usable in practice:

- According to Java EE business logic should be written in EJBs. Therefore, even for the simplest pieces of logic, three artifacts (Remote Interface, Home Interface, Implementation) and a configuration in a rather complex XML Deployment Descriptor had to be implemented. This was such a tedious task that as a work-around generators like XDoclet were invented.

- EJB 1.x also promoted a distributed component model, i.e. the EJB server should be on a separate machine from the web server. Because of the high overhead of remote calls this led to severe performance problems.

- Testing by deploying the application on the Application Server was slow because of the time needed for the deployment. Also, it was difficult to automate. Running the code outside the Application Server was rather difficult because of the dependencies in the written code to the Application Server. This made the software essentially untestable – ironically, at about the same time the agile development movement stressed the importance of elaborate and automated tests.

- JSP allowed including Java code in web pages. So, design and logic was intermingled in one artifact that had to be modified by designers and developers alike. This approach also proved to be very hard to maintain.

- The approach towards persistence using EJB CMP focused on the very unlikely case of concurrent access to the same persistence entity. This had a huge impact on the performance for standard cases. At the same time loading larger sets of data led to one database query for each item in the set. Loading 10,000 customers actually meant executing 10,000 database queries – of course, in practice this led to performance disasters. The alternative was to use JDBC, the low level API for database access in Java. But it is hard to avoid resource leaks and to implement error handling correctly when using this API. Most of the JDBC tutorials either don't talk about this topic or even show erroneous code. Doing it correctly still means about 10 lines of code for even the simplest database query.

In October 2002, Java-industry expert Rod Johnson published "Expert 1-on-1 J2EE Design and Development" (Wrox, 2002). This book contained the condensed experience of his groundbreaking consulting work. It offered valuable insight on how to deal with the shortcomings mentioned above and how to actually succeed in J2EE projects. Additionally, a complementary download a framework was made available that contained several of the best practices in code. Rod Johnson saw tremendous potential in this approach and after Jürgen Höller joined the development in February 2003 the Spring Framework was released.

Spring's Values

Spring is based on several values:

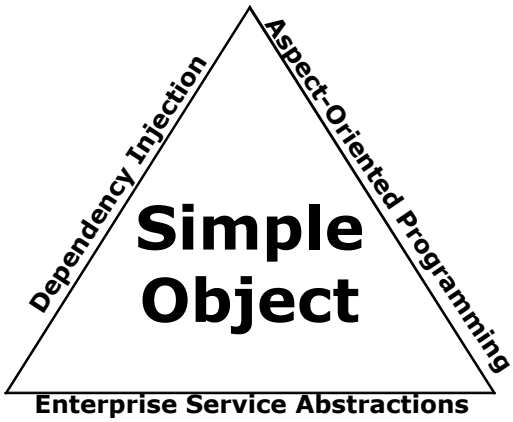
- **Simple, yet powerful**: While the approach offered by the framework should be easy to use it must not sacrifice any of the power. A good example is Spring's JDBC abstraction: While it is possible to execute database queries without caring about the details of the JDBC API at all, it is also possible to expose any part of the underlying API without losing the advantage of the automatic error handling. It is even possible to access the proprietary extensions of specific databases – a feature hard to implement without Spring.

- **Flexibility**: Instead of using a monolithic approach Spring contains several modules that are independent from each other. For that reason it is possible to use Spring in those areas where it adds the most value and it can be adapted step-by-step instead of all-at-once. Also Spring offers a lot of hooks to extend the given behavior in specific way for the context at hand.

- **Choice**: This is actually closely related to Flexibility. Choice acknowledges that one size does not fit all. Spring does not force the developer to use a specific approach or technology. For example, Spring supports a great variety of persistence frameworks (JDBC, JPA, JDBC, Hibernate, JDO, Apache OJB, …). Also, in areas like web frameworks that are addressed by Spring itself, several other approaches are supported (JSF, Struts, Tapestry, WebWork). Spring has several different ways of configuring applications that range from XML to several Java-annotation-based approaches.

These core values lead to a framework that is not limited to a single area of applications or technologies, and can easily be adapted and changed for different purposes and remains easy to use.

Spring's Architecture



The architecture of the Spring Framework is designed to support Simple Objects. Sometimes in the Java world they are also called POJOs (Plain Old Java Objects). The basic characteristic of such an object is that it does not depend unnecessarily on any technology. Independence from technology is a well-known best practice in software development. This has several benefits:

- *Investment in business code is retained even if a new technology is used*. So, if you decide to migrate the application to a new environment such as a new version of an Application Server or a lightweight alternative such as a Web Server, the business logic itself is not changed.

- *The developer can focus on creating business value instead of dealing with the technology*. This makes him/her more productive in terms of business value generated over time.

- *The objects can easily be tested in isolation*. Instead of using a production-like environment, all collaborating objects can be exchanged against a mock object. These provide a behavior similar to the production environment but are much simpler. This allows faster tests and simulation of exceptional situations otherwise hard to provide in a production-like environment. The main benefit is that errors can be tracked down to the single object that was not mocked and is therefore the only possible source of the error.

- *Tests of single objects in isolation are not enough*. By using the Simple Object approach it is also possible to use a lightweight alternative environment to test a collaboration of several objects. So, instead of deploying the application to a server infrastructure, which is a time consuming task, such a test can be started from the developer's Integrated Development Environment (IDE), which greatly speeds up development and enables easier debugging of the applications.

Obviously, a system still needs an environment to run on. Spring provides three important features that allow building an application from Simple Objects:

- **Dependency Injection** (DI) means that collaborating objects are injected. So objects do not depend on any technology to find other objects. Instead it is the responsibility of Spring to provide the collaborators. This way they can also easily be exchanged for tests as mentioned above.

- **Aspect Oriented Programming** (AOP) is a technique proven for 10 years to add aspects such as security, transactions etc to the code. Without AOP every method in the code has to take care of these aspect and therefore depends on the security or transaction technology. Using AOP all code that deals with such an aspect can be placed in one specific location. The business code doesn't have to care about it. So, the business concerns are separated from technical concerns such as transactions or security – a goal named "Separation of Concerns" by Computer Science pioneer Edsger W. Dijkstra in 1974.

- Finally, the **Enterprise Service Abstraction** allows the use of common APIs such as JDBC or JMS in a much simpler and less error prone way. Usually, even simple tasks like doing a database query with JDBC or sending a message with JMS take >10 lines of code. Also, it is quite hard to get the error handling right – some tutorials don't even mention this topic. Spring's Enterprise Service Abstraction takes away this unnecessary complexity and allows the developer to do a database query or the sending of a message in just one line of code. It also takes care of the error handling and unifies the APIs where needed, i.e. all persistence exceptions from all supported frameworks are translated into a common hierarchy.

Because of the choice and flexibility each of these features can be used on its own without any of the other. This allows easier migration and a pick-and-choose-approach.

Spring, Java EE and EJB

Spring is not comparable to Java EE. While Java EE defines a platform implemented by an Application Server, Spring defines a programming model that can run on the Java EE platform or alternative platforms. Of course, Java EE has its own programming model but this cannot offer abstractions like Spring's Enterprise Service Abstraction that is, by definition, a layer atop Java EE.

EJB originally started as the preferred way of implementing business logic in Java EE applications. It is commonly agreed that EJB 1.x and 2.x are better to be avoided – a view that has been advocated by Spring's founder Rod Johnson in his book "J2EE without EJB." This title also says that Java EE itself is still a valid model. The Spring Framework even includes support to expose Simple Objects as EJBs and to use EJBs in a way that makes them look like Simple Objects. So, even for the implementation or usage of EJBs, Spring should be used for EJB 1.x and 2.x.

EJB 3.x has taken some of the approaches introduced by Spring and therefore simplified the EJB model. However, it falls short of providing the same level of features Spring does in each of the three areas of the Spring Architecture:

- **Dependency Injection using EJB 3**: Dependency injection is only possible for fields and setters using Java Annotations (except for primitive types) and it is only possible on EJB 3 artifacts. Spring offers support for constructor, setter, method and field injection using Java Annotations in the class itself, an external configuration with Java Annotations or XML. Spring is perfectly capable to integrate code that has not been designed with Spring or Dependency Injection in mind or uses alternative approaches like Factories. Spring also supports all Java Annotations for Dependency Injection defined by EJB 3.

- **EJB 3's AOP model:** This model does not allow the selection of specific targets for an aspect. Every other AOP implementation contains support for this concept known as pointcuts. So, an aspect can either be applied to all EJBs or the EJB has to be changed. Thus, it is not possible to use an aspect on specific EJBs or methods without changing it. Interestingly enough, EJB 3 itself has implementation of aspect such as security and transactions that cannot efficiently be implemented using the EJB 3 AOP model. Spring uses the AspectJ pointcut model to choose the targets of aspect. It has seen 10 years of development and is recognized as the most powerful and mature pointcut model available. Of course, this model is also used to implement all other aspects in the Spring Framework or other technologies of the Spring Portfolio.

- There is **no match for the Enterprise Service Abstraction in EJB 3**. This is not in the scope of EJB 3.

- Additionally, transactions are supported using Java Annotations in EJB 3. These can also be used with Spring. Spring also offers its own Java Annotation that is considerably more powerful. It also allows the definition of transactions using XML in a much simpler and more powerful way than EJB does.

- Furthermore, the security model used by EJB 3 is role based. It does not directly support advanced security requirements. For example it is not possible to define that certain business object may only be access by certain users i.e. a clerk may only use data related to his customers. This made security the least used feature of EJB. Spring Security allows role-based security as well as security based on the individual business objects and users. Apart from Java Annotations and XML the security definitions can be read from arbitrary sources such as databases. Spring Security can also easily be integrated in existing security infrastructures.

So, EJB is still unable to match the power of Spring in the areas mentioned above and it also introduces many drawbacks. It ties the code to EJB 3 and therefore a specific API and a specific environment, i.e. an application server. Behind EJB 3, the complexity of EJB 1.x and 2.x that remain part of the specification still lurk and sometimes – such as in the case of XML based configuration – clearly show. As Spring supports a considerable part of the EJB 3 programming model the question is why EJB 3 should even be considered as Spring offers many more features – without the drawbacks.

However, if EJB 3 must be used for some non-technical reason, Spring-based code can still be integrated with EJB 3 and used as a platform.

Quality

Spring constantly receives highest marks for its quality:

- Chris Chedgey noted in his blog [1]: "Serious credit to the Spring guys for building possibly the best structured code-base in the world!". He is CEO of Headaway Software, a company that builds products to measure the quality of software.

- Ivica Aracic (PhD student at the Darmstadt University of Technology) compared some Open Source applications and came to the conclusion that "except of Spring, all of these projects had some more or less bad dependency cycles at the architectural level, making the comprehension difficult. However, after analyzing Spring I was surprised that there was not a single cycle at all in the dependency graph at the architectural level." [2] Dependency cycle link

two part of the software in a way that changes influence both parts making them hard to maintain and finally inseparable.

Jürgen Höller talks about how this quality was achieved in several presentations [3]:

"This level of quality can only be achieved using an Open Source approach that allows feedback on the code and the documentation. Spring comes with an extensive documentation on the API as well as a Reference Documentation. Due to the constant stream of feedback from the community the level of quality can be kept pretty high. This is basically an ever ongoing review of code and documentation that ensures that errors are fixed, the documentation is updated and that the Spring Framework works on the production systems on the users' sites."

Market Share

Spring has passed the five million downloads mark and now has a bigger market share than any Java Application Server. It is the predominant approach for Java Enterprise Applications. A survey by BEA in 2006 showed that 64% of all enterprises were already using Spring. So, Spring has become the de facto standard for Enterprise Java. It is also interesting to note that in the job market Spring has passed other technologies like EJB. The job market is interesting because it shows what companies are actually willing to pay money for and what skills they want in their staff. These numbers are a clear indication that organizations are using Spring extensively.

The Spring Portfolio

Spring's values can also be applied to other areas. For this reason the Spring Framework is now complemented by a set of technologies that solve other common problems in Enterprise Computing while sharing the same values of the Spring Framework:

- **Spring Security** (formerly Acegi Security) is the only security solution that leverages the power of AOP. It allows the implementation of a powerful, fine-grained security system without influencing the business logic. It is not limited to role-based security but can also use Access Control Lists and can easily be extended to use other approaches. It offers an integration of security technologies such as x.509 Certificates, JAAS, LDAP, CA Siteminder, HTTP Basic, HTTP Digest and can be adjusted to specific requirements, as needed.

- **Spring Dynamic Modules for the OSGi Platform**: OSGi allows the creation of services and dynamic updates of them. This means that parts of the application can be updated without restarting the whole application. However, the application has to deal with situations in which certain services are not available at runtime. The Spring Dynamic Modules framework simplifies the creation and discovery of services to become just a simple configuration. Also, dynamic updates of such services are automatically dealt with.

- **Spring Integration**: Enterprise Integration is becoming more and more important for enterprise. A set of Patterns of Enterprise Integration has been established. Spring Integration provides a simple yet powerful programming model based on these patterns. It allows the implementation of routing, splitting and other message handling with a minimum of coding. It can be connected to several different data sources and output channels like JMS or file import/export.

- **Spring Batch**: Batch applications remain mission critical for many enterprises. Spring Batch is the first Open Source framework that supports batch processing applications. It addresses common challenges in this area like retry, failure handling and efficient handling of batch updates to database.

- **Spring .NET** is a port of the Spring Framework to the Microsoft .NET platform. It offers features like Dependency Injection, Aspect Oriented Programming and Enterprise Service Abstraction for a set of .NET APIs like ASP.NET, ADO.NET and NHibernate.

- **Spring Web Flow** allows developers to model user actions as high-level modules called flows that are runable in any environment. The framework delivers improved productivity and testability while providing a strong solution to enforcing navigation rules and managing application state.

- **Spring Web Services** focuses on creating document-driven Web services. Spring Web Services aims to facilitate contract-first SOAP service development. This is done without writing WSDL manually which is known to be a very tedious task. Spring Web Services also allows for the creation of flexible web services using one of the many ways to manipulate XML payloads.

- **AspectJ** extends the AOP approach used by Spring to be even more powerful. It allows more efficient implementation of aspects and a bigger set of pointcuts, among others.

Platform Independence

As mentioned above, the Spring Framework offers a platform independent approach. For this reason, Spring has broad support on many different Java EE platforms:

- Spring is certified on IBM WebSphere. It has been extensively tested on all platforms supported by IBM WebSphere. [4]

- Spring is used at the core of BEA WebLogic since version 10. The EJB 3 implementation of BEA is based on Spring. Therefore, it is not only possible to use Spring in BEA WebLogic out-of-the-box but BEA WebLogic EJBs are actually just a different flavor of Spring components. [5]

- Oracle has submitted code for the integration of their TopLink product to Spring and has also helped with the integration of the JPA persistence approach into the Spring Framework. Spring is also supported by Oracles OC4J server and by JDeveloper. [6]

Forrester Research also noted that the majority of Enterprise Java users use Spring. Spring integration is used as an evaluation metric for Application Servers.

However, recent analyst reports show that Java EE as a platform is facing significant challenges. Gartner recently published the study, "Trends in Platform Middleware: Disruption Is in Sight" [7]. This analysis reports that Java EE is increasingly inadequate to cover needs for extensive scalability and performance, event-based programming styles, advanced service-oriented architecture (SOA) and dynamic application developments. A sensible set of standard

technologies, such as Java application programming interfaces for JSLEE (a telco standard), Spring and OSGi is being adopted by many vendors in this space. By 2009, at least 75% of Java-based platforms and integration middleware products will provide embedded support for the Spring Framework, which is an increase from fewer than 20% in 2007. So, a growing number of users are turning to alternatives, (particularly Spring) to work around the growing complexity of Java EE. A new generation of Java-based platforms addressing XTP (extreme transaction processing) requirements has selected Spring as its programming model. Spring can potentially support any container (Java EE or not), thus enabling a good degree of application portability across different platforms.

In another report, [9] Gartner notes that the simplicity of Spring Framework programming initially made it popular among Java developers for simple transactional applications; however, it is now increasingly used in the context of large, business-critical projects. Leading-edge organizations looking for an easy-to-use, lightweight Java programming model alternative to Java EE to support even business-critical development should look at Spring Framework as a primary option.

Currently there are several promising approaches for the future of Enterprise Java:

- A lightweight container based on OSGi. Spring is well prepared because of the Spring Dynamic Modules for OSGi project. Submitters from Oracle and BEA also back this project.

- Alternative approaches such as JavaSpaces-based high-throughput computing are usually based on Spring as it allows technology neutral code, has a high adoption rate and is therefore the perfect vehicle to make the transition to such infrastructure easy for the developers.

- Many enterprises are turning toward lightweight, open source application servers, namely Apache Tomcat. A recent survey by BZ research found usage of Tomcat at 64.3%, ahead of WebSphere at 36.9%, JBoss at 32% and WebLogic at 23.7%.

SpringSource

SpringSource employs all Spring Framework committers and almost all of the committers to the Spring Portfolio. Therefore, the Spring platform does not carry the usual risk associated with Open Source where no clear responsible entity is present.

SpringSource acquired Covalent Technologies in January 2008 and now provides commercial support of Tomcat, the leading Java application server, as well as the world's leading web server Apache HTTP Server. Principal committers for both technologies and several other Apache projects are now part of SpringSource. Covalent provides support for over 50% of Fortune 500 and 20% of Global 2000. SpringSource also employees a committer to Equinox, the most successful OSGi implementation. So, besides the Spring Framework and the Spring Portfolio, SpringSource can also offer support for the strongest Enterprise Java platform in the market today i.e. Spring running on Tomcat.

SpringSource is led by the original inventors of the Spring technologies. Benchmark Capital invested $10M in venture capital and has also invested in several other Enterprise Open Source companies such as MySQL, Red Hat, Hyperic,

Zimbra and JBoss. As experts in this field Benchmark invested more than just money: They also add their experience and contacts for the benefit of SpringSource.

Offerings

SpringSource's offerings include:

- **SpringSource Enterprise** which includes the following support as well as certain Shared Source Components:
  - **Support for the Spring Portfolio and Apache products** such as Apache HTTP, Tomcat, Axis, ActiveMQ, Geronimo and Roller directly from the source i.e. from the developers. The support includes development support that gives access to subject matter experts who can increase developer productivity and effectiveness. Production support is also offered directly from the key committers when required and allows fast turnaround for bug-fixes and patches. Patches, bug fixes and product enhancements provided by SpringSource will be committed and stay in the code base – a service only committers are able to provide.
  - The **SpringSource Application Management Suite** is shared source software, made available only to subscribers. It monitors production deployments of Spring applications on platforms like Tomcat, WebLogic, WebSphere and JBoss. It can track application performance, utilization and resource control, monitor servers and Spring's core components. You can define critical application alerts with automated corrective control and control and record configuration changes. This is compatible with enterprise wide management tools like HP OpenView and IBM Tivoli
  - The **SpringSource Tool Suite** is a complete Integrated Development Environment (IDE) based on a custom Eclipse distribution. Eclipse is by far the dominant IDE in the Java area. The SpringSource Tool Suite adds samples, cheat sheets and automated code reviews as well as access to the SpringSource Knowledge Base to this established foundation. This way the collective community wisdom accessible in the tool and it can suggest error fixes, test unit tracing and best practices. It also offers task based enterprise development based on Eclipse Mylyn. This allows the developer to focus on the task at hand. The IDE store information about which windows are open while working at a specific task. For each task the complete layout of the IDE is adjusted so that the developer sees the information needed to complete the task.
  - The **SpringSource Advanced Pack for Oracle Database** provides Spring support for Oracle RAC Fast Connection Failover. All database operations that were done during the transaction are automatically retried on the new RAC node. The Oracle Streams Advanced Queuing makes the Spring code use a single local transaction for messaging and database access which makes application that use JMS and JDBC on Oracle very efficient. XML and other Advanced Data Types are supported and specific Oracle DataSource wrappers and pooling for Spring are included.
  - **Certified releases** from SpringSource.com are available for the subscribers. Those are production ready versions only and they are certified to be virus-free, worm-free. At Gold and Platinum support levels legal indemnification is offered. Certified patches and binary upgrades are immediately available to all subscribers.

- **SpringSource dm Server** is a completely module-based Java application server that is designed to run enterprise Java applications and Spring-powered applications with a new degree of flexibility and reliability. The SpringSource dm Server is based on the new SpringSource Dynamic Module Kernel™ (dm Kernel). The dm Kernel provides a module-based backbone for the server, which also harnesses the power of Spring, Apache Tomcat and OSGi-based technologies.

- **SpringSource Training** offers education for the Spring Framework as well as other products of the Spring portfolio, AOP, Hibernate and a general introduction to Enterprise Java. So far SpringSource has successfully trained more than 4000 developers. The courses are offered as public training and in-house. Based on the training a certification is available that allows the participants to document their knowledge of Spring.

- **SpringSource Consulting** offers high impact, short duration engagements delivered by Spring experts. Usually consulting engagements are client driven and can be used to address specific issues. This includes reviews, jump starts, proof of concept or performance tuning.

Return on Investment

SpringSource is the only place to go for superior support and services surrounding the Spring Framework. SpringSource has developed a brand known for technical quality and excellence, commitment to customers, and a pragmatic view on software development and application creation.

- Support:

    - Traditional support organizations are so far removed from the real world and the pain of escalating cases to the right person requires alignment of the moon and stars. Since every SpringSource employee has been on the other side of this experience, there is a strong culture of service in our support offerings.

    - Support in the Open Source world has been amateurish at best to date. The stories abound of customers being called "stupid" or worse, support tickets simply being ignored, support people unfamiliar with the products that they are responsible for, and support organizations who lack the power to effect change in the underlying code base has resulted in distrust and often lack of adoption of many open source projects. SpringSource has taken a commercial approach to offering support with professional support personnel.

    - The importance of support for SpringSource is seen as the company's leaders lead by example. Some of our most senior Spring committers answer support questions. SpringSource believes strongly that its employees should also see the effects of the decisions they make on the front line and requires that all employees spend time in the support queues.

    - Once an application is in production, SpringSource will be there with best of breed production support giving you peace of mind that should something go wrong, the best and brightest will be there as a partner to get you back up and running as quick as possible.

- Training:
  - The attendees evaluate each training at SpringSource. The feedback usually rates the course material and instructor "excellent". As SpringSource's employees do more than just trainings they know what the most important aspects in practice are and what needs to be emphasized. For that reason they also know the technologies in detail and can answer in-depth questions. Some of the trainers have shown their technical and didactic expertise in several conferences and books.
- Consulting:
  - All of SpringSource's consulting services are designed to help customers become extremely successful with their implementations. SpringSource's Design Consulting and Architecture Review offerings are specifically designed to help clients' get their projects started correctly and to ensure that these projects are given the best opportunity to succeed. Coupled with SpringSource's support offerings and Project Health Checks, SpringSource is there through the course of your project to make sure that it stays on track and to make sure that developers continue to remain productive and efficient in meeting their management's targets and goals.
- Other value drivers of SpringSource's offerings include:
  - Customers can leverage the expertise and experience of SpringSource to prevent instances of production downtime and to minimize the cost of production downtime.
  - SpringSource customers can focus their limited resources on core competencies, not infrastructure coding, especially in current environments where finding and retaining strong technical talent is extremely challenging.
  - Strong quality assurance – SpringSource provides higher reliability through rigorous commercial testing of Spring and by also leveraging the vast Spring community to provide the breath of testing only available to ubiquitous Open Source technologies like the Spring Framework.
  - Ongoing cost savings in maintenance (traditionally, companies underestimate maintenance of internally developed frameworks).
  - Greater level of integration with third-party technologies. Again, Spring's ubiquity is very powerful in creating a platform for many technologies from which to launch. The success of Spring's simplified programming model is available for all to leverage.

By coupling the simplified Spring programming model with enterprise-class professional services and support, SpringSource gives you the best value in the software industry:

- The pricing and quality benefits of Open Source.
- Low risk because of the de facto industry standard that is Spring, the technology independence of your code and the superior quality of the Spring software.

- An organization that influences Spring and the Spring Portfolio whose revenue streams come from our ability to deliver ongoing value.

Case Studies

Many organizations experience tremendous financial benefit from deploying with the Spring Framework and leveraging SpringSource for technical expertise through professional services, training and product support. The following success stories showcase specific examples:

Voca

Voca is Europe's leading processor of direct debit and direct credit transactions handling over 5 billion transactions worth $5 trillion each year, including 70% of UK salaries and 90% of UK utility payments. The core systems of Voca run on Spring.

Prior to switching to Java, Voca used mainframes with applications containing 10 million lines of Cobol code that cost $50M per year to maintain. In addition to the maintenance costs, many of the engineers involved in the project retired or left the company and it was increasingly difficult to replace them with experienced Cobol programmers. Voca hired one of the world's largest system integrators to create a replacement system in Java. After a cost of $20M, this system integrator created a system capable of 700 transactions per second. This was far below the 4000 transactions per second required by Voca. A second major system integrator was hired but concluded that the performance and "no loss" tolerance requirements could not be achieved with Enterprise Java and proposed a C/C++ solution. Finally, Voca hired SpringSource. SpringSource helped Voca implement a Spring-based solution on the same hardware and infrastructure that was used by the initial system integrator but with far better results.

With Spring and SpringSource's support, Voca ran tests that showed they could processes 12,000 transactions per second, 6x the throughput of mainframe system. In Voca's tests, the 12,000 transactions per second do not appear to be a system maximum but rather the limit that Voca could achieve with their current set of tests and hardware. The application itself appears to scale linearly. Spring's lightweight container exceeded performance requirements where other well financed solutions failed.

These results are confirmed by Voca[8].

Bank of America

Nine out of ten of the world's most profitable banks are SpringSource customers. Bank of America serves more than 59 million consumer and small business relationships with more than 6,100 retail banking offices, nearly 19,000 ATMs and online banking with nearly 24 million active users.

One Spring application at Bank of America is a middle office data intensive Trading Analysis Application where the development team replaced the old Excel/Reporting Databases system with a Spring-based Java application. Spring allows Bank of America to focus on their application as opposed to the framework while providing patterns for simplifying data access. Spring supplies a container for hosting services with Dependency Injection along with AOP Services for remoting. Finally, Spring ensures no vendor lock in for the project.

HSBC

HSBC is one of the largest banking and financial services organizations in the world. HSBC's international network comprises over 10,000 offices in 83 countries and territories in Europe, the Asia-Pacific region, the Americas, the Middle East and Africa. HSBC based its next-generation architecture on Spring. This led to 40% reduction in framework code and 25% reduction in reference application code. HSBC has a long relationship with SpringSource and purchased Enterprise-wide Support from SpringSource.

Symantec

One of Symantec's Spring-based applications, Titan, ensures responsive and timely critical support. The Spring-derived benefits in Titan include:

- Loose coupling between tiers including physical tiers and tiers within the applications themselves
- The abstraction of remoting technology supporting HTTP remoting, EJB, SOAP, and JMS
- Over 75% of test code coverage in the server tier
- Their server runs on JBoss and Tomcat with little modification providing vendor independence

French Online Taxation System

The French online taxation system, serving over 34 million French taxpayers is deployed with a Spring runtime. The French Minister of Finance has described this system as a "triumph of IT". Accenture implemented it. Thomas van de Velde, Lead Java Architect of Accenture Delivery Architectures, who designed the system, commented that, "Spring has had a significant impact on the productivity of our Java EE technology developments. Thanks to its simple yet powerful programming model we were able to significantly improve time to market and build better quality solutions."

European Patent Office

Another high-profile, high-volume Spring user is the European Patent Office, which uses Spring in the core of its solution to manage Intellectual Property across the European Union. EPO Architect Roland Nelson comments that, "ever since the introduction of Spring, I have been able to focus on what really matters: the business focus of an enterprise application. Low-level plumbing is a thing of the past and as an architect I can tie modules and functionality together – injecting concerns I think matter, where they matter. What had previously taken numerous man-years to compose/ understand and implement has been achieved in a few months using Spring".

Conclusion

- Spring is based on the values of simple, yet powerful, flexibility and choice. It allows the creation of Enterprise-grade software using Simple Objects by the techniques of Dependency Injection, Aspect Oriented Programming and the Enterprise Service Abstraction.
- The Spring Framework is a proven, stable, high-quality platform ubiquitous in today's Enterprise Java world. It can be integrated into a Java EE environment and is considerably more powerful and complete than EJB 3.0.

- An entire Spring Portfolio has been created that uses the Spring values in other important areas. This includes security, batch operations, the .NET platform, web flows and web services.
- Spring is supported on the leading Java EE platforms IBM WebSphere, BEA WebLogic and Oracle OC4J. Spring enables the adoption of new platforms, Apache Tomcat and OSGi being the main ones. New approaches choose Spring as their programming model because Spring allows technology independent code and, therefore, easy migration to new platforms.
- SpringSource offers commercial backing for the product and is a reliable partner for Spring users. Services include training, support and consulting.
- Numerous large companies have deployed Spring in mission-critical applications with very good results.

Links & Literature

[1] http://chris.headwaysoftware.com/2006/07/springs_structu.html

[2] http://sourceforge.net/mailarchive/message.php?msg_id=44B56DD1.40307%40gmx.de

[3] http://se-radio.net/podcast/2008-01/episode-82-organization-large-code-bases-juergen-hoeller

[4] http://blog.SpringSource.com/main/2007/06/21/spring-framework-certified-on-websphere/

[5] http://www.springone.com/display/SpringOne06/Spring+in+the+Core+of+WebLogic

[6] http://www.oracle.com/technology/tech/java/spring/index.html

[7] http://www.gartner.com/DisplayDocument?id=525420

[8] http://www.theserverside.com/news/thread.tss?thread_id=42460#219407

[9] http://www.gartner.com/DisplayDocument?id=577512