Term Paper for Distributed Enterprise Computing
Team : BucKonnect

12/12/2014

Instructor: Dr. Rajiv Ramnath

Priyanka Sadavartia (sadavartia.1@osu.edu)
Jagannath Narasimhan (narasimhan.23@osu.edu)
Nandkumar J. Khobare (khobare.1@osu.edu)

Over the course of the semester there were various interesting and useful technologies that were discussed in the course. The discussion here would be on those technologies encountered and where do we believe these technologies would shape up to in maybe 10 years from now.

EJB - Enterprise JavaBeans

The main use of the Enterprise Java Bean (EJB) architecture is to encapsulate the business logic for enterprise applications. Enterprise JavaBeans had an important milestone release with 3.0 with lot of functions inspired by Spring mainly because of EJB 2.x complexity and Spring 1.x was introduced in 2006. There were functionalities which were clearly inspired from Hibernate as well in EJB 3.0 . Thus if we were to use this technology before the introduction of the upgrade, the inclination would have probably been towards Spring or Hibernate as EJB 2.x is a very heavy and difficult to use technology. But we have EJB 3.2 at our disposal which brings a lot of functionalities(although inspired) to the table. The most useful feature which was included is the use of Metadata Annotations[1] instead of Deployment descriptors which helps in the use of Plain Old Java Object(POJO), inspired by Spring. The use of this comes in deployment when the need was to just deploy a jar file onto the application server and no explicit need of XML descriptors. With that another important feature which is very useful is dependency injection which is now annotated and attached to the bean class avoiding JNDI looking. Apart from this important advancements would be incorporation of Singleton and Asynchronous Session beans , transaction lifecycle callbacks in Stateful Session beans.
[2]EJBs , thus are essential in making the application modular and handling transaction management, security , persistence(taken care of by Java Persistence API) , inter process communication using web services. Thus, this server side component of our three - tiered architecture containing entity , session and message driven beans could be EJB with its new release or Spring with its new release as both have very similar functionalities but EJB being a Java EE standard would be the better choice with all its new properties which have fixed all previous bugs and made life easier for the developer.

Enterprise Service Bus(ESB)

[3]ESBs are the tool which form the connector between mutually interacting software applications in a service-oriented architecture(SOA), which was used to connect our application with our loosely coupled component.
The ultimate result - lightweight, tailor-made integration solutions with guaranteed reliability, that are fully abstracted from the application layer, follow a consistent pattern, and can be designed and configured with minimal additional code with no modification to the systems

that need to be integrated. This mature version of the bus-based EAI model eventually came to be known as the Enterprise Service Bus, or ESB.
The technology used to implement ESB was Apache Camel service mix. As good as this technology is , there is a technology called MuleESB which could serve as a better alternative in many scenarios.[4] If the main criterion for the developers are the non functional requirements like reliability, scalability, security then MuleESB is the preferred choice as it is a complete integration platform compared to Apache Service mix. With a better visual development environment Mule becomes the better choice over ESB again. Thus when both are open source and light weight products the maybe choosing MuleESB over Apache Camel would be better if the need demands so.

JavaServer Faces (JSF)

[5]JSF is a Java EE standard technology that enjoys wide support with Java EE application servers including JBoss AS/WildFly, Oracle WebLogic, and IBM WebSphere. In addition, JSF developers benefit from tooling support built-in to Eclipse, NetBeans, and IntelliJ. JSF component suites like ICEfaces, PrimeFaces, and RichFaces offer a plethora of UI components and advanced features, all built on top of the core JSF standard.
JSF 2.0/2.1 was released with Java EE 6 and was well received by developers thanks to the addition of standards-based Ajax features and the adoption of Facelets as the standard template engine. JSF 2.2 was released with Java EE 7 and added fantastic new features like Faces Flows. Although JSF is an evolving technology , the future for it could include improvement in the JSF AJAX API ,multi-field validations which could come in handy, cross-form Ajax implementation , performance optimization.
When it comes to creating a web framework , JSFs get the job done. But if the job at hand is to implement simple CRUD operations then Grails could be the better choice because its comparatively simpler and code auto reloading mechanism[6].

JBoss

JBoss or Wildfly which we used is the technology to use when JBoss is the best choice for applications where developers need full access to the functionality that the Java Enterprise Edition provides and are happy with the default implementations of that functionality that ship with it. If the full range of JEE features is not needed, then choosing JBoss will add a lot of complexity to deployment and resource overhead that will go unused. For example, the JBoss installation files are around an order of magnitude larger than Tomcat's. Thus ,Tomcat should be chosen ,which is a Java servlet container and web server, and, because it doesn't come with an implementation of the full JEE stack, it is significantly lighter weight out of the box. For developers who don't need the full JEE stack that has two main advantages. Significantly less complexity and resource use and modularity which could be advantageous over Wildfly if the need demands it.

H2

[7]H2 is a terrific database engine, the biggest advantage being that its light weighted. The advantages of using a H2 database would be its speed and the embedded mode is amazing and gives us the flexibility to install the software in a portable device, even share the database on the cloud, let's say via Dropbox. H2 also provides a built-in Java profiler which could come in handy. Although H2 has a lot of advantages the future for it needs to incorporate a few more technologies to make its usage more global and adaptable to all requirements. Some of which could be multi-threaded statement processing, resolve problems with durability(part of ACID), which prevents data loss. It could provide fine grained row level rocking which is provided by mySQL and a couple of others. Also H2 needs to incorporate the concept of Cursors which is not present as of now.

General Idea

Thus for the coming years the key takeaways would be firstly the arrival of Java8 which for one has lambda functions which makes it that more useful. Secondly would distributed computing be switched to be incorporated in Scala and the TypeSafe ecosystem.[8] Its the language of pragmatism and even after maintaining inter-operability with Java it provides modern multicore hardware.

"Scala allows us to write better, more beautiful code, and it just makes us better engineers. It strikes the balance between conciseness, expressiveness and practicality, and you can do amazingly complex things with few lines of code." -Wix
That said Java is the standard for a reason but Scala programming could be the next face of distributed enterprise system , the adoption of it will take a lot of time no doubt,

Another thing to ponder upon about this technology is that will cloud computing/systems replace distributed systems or will a amalgamation of these two will co-exist in the future or will everything go to the cloud. This has various challenges mainly being security and also readiness of developers to migrate but again can't rule it out.

References

1. https://today.java.net/pub/a/today/2005/08/18/ejb3.html
2. http://en.wikipedia.org/wiki/Enterprise_JavaBeans
3. http://en.wikipedia.org/wiki/Enterprise_service_bus
4. http://www.mulesoft.com/resources/esb/mule-vs-camel-comparison
5. http://en.wikipedia.org/wiki/JavaServer_Faces
6. https://grails.org/Auto+Reloading
7. http://www.h2database.com/html/features.html
8. https://typesafe.com/community/core-projects/scala