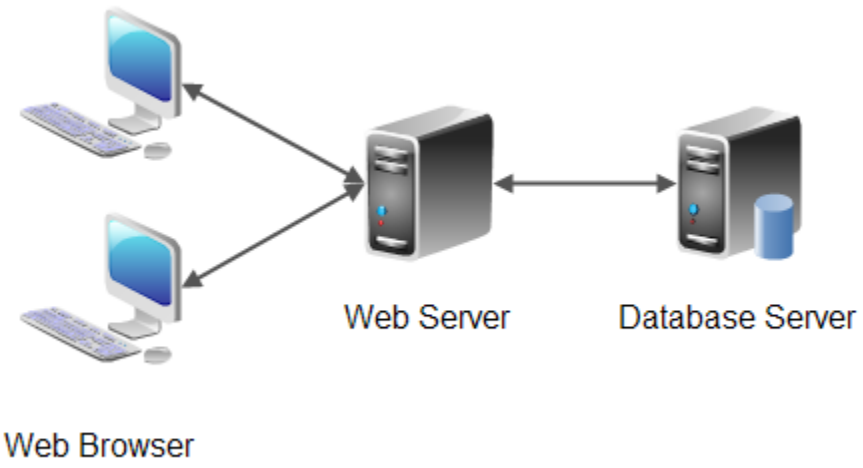
Common Web Application Architecture

Advanced Web Development (SCS3112 /IS 3015/CS3112)

Assignment 1

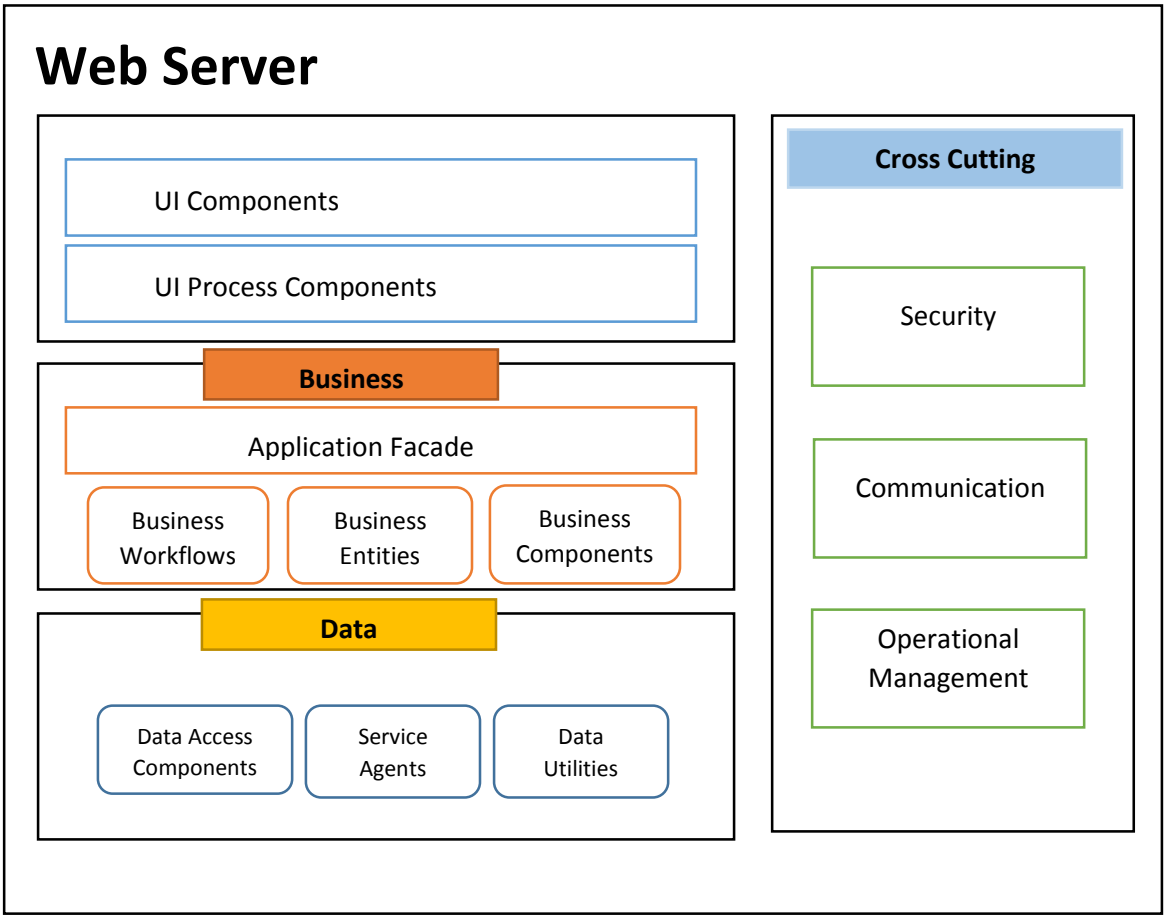


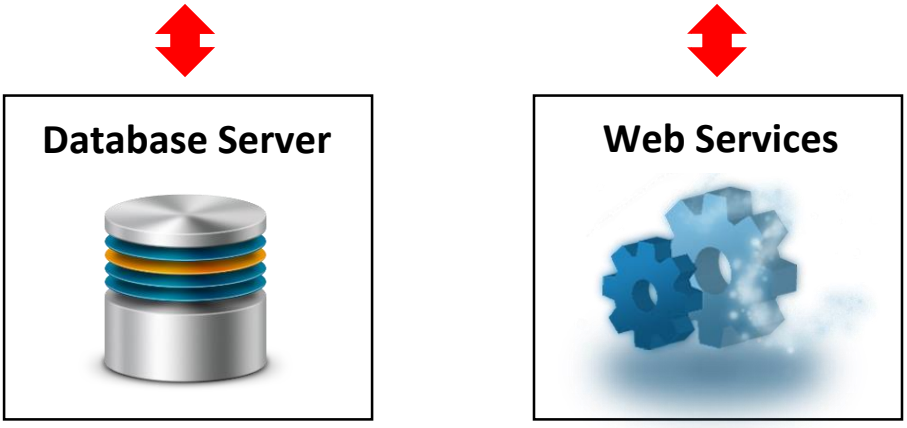
W.D. Hettige

2013/CS/044

13000446

Key Components of Common Web Application Architecture





Web Architecture is a flow diagram. When talking about architecture, we should think of the technologies, methods, and how everything is arranged to form a complete product. In this report I'll be discussing about pros and cons about the common stuff today which we used to build web architecture.

A web application architecture can vary greatly depending on the application at hand, its needs, its behaviors. There are lot of layers that we should talk when building up a web application.

Client-Side

This is also called the front end. It's the top most visible layer. This is usually written in HTML, CSS, for styling and JavaScript for "interactivity" and functions. JavaScript has become widely adopted in recent years largely because new frameworks and best practices have increased. The emergence of JavaScript toolkits, such as Dojo, make client-side development easier while hiding many of the problems surrounding cross-browser compatibility. And another front end technology that I want talk is Adobe's Flash. Flash technology has become a popular method for adding animation and interactivity to web pages; several software products, systems, and devices are able to create or display Flash. Flash is commonly used to create animation, various web-page components, to integrate video into web pages, and more recently, to develop rich internet applications.

Server-Side

This is where you use a programming language such as PHP, Python, Ruby, and Java etc.

PHP

Advantages:

- Fairly easy to learn, especially for developers with C/C++, Java, or Perl experience.
- Active user community willing to lend you a hand getting started.
- Everything's built right into the language.
- It's free!
- Cross-platform.

Drawbacks:

- This language was designed to be programmer-friendly, which unfortunately makes it a little less friendly to non-programmers.

Java

Advantages:

- Extremely powerful and scalable.
- Cross-platform.
- Most Java server plugins are free for personal and development purposes.

Drawbacks:

- Java takes a lot of work to learn. Don't even start if you're not serious about learning object oriented programming.
- Most Java server plugins must be paid for if they are to be used to host a commercial Web site.

Frameworks

Assume that we want to write our own solution to something, frameworks makes it easier and simpler for us. "Ruby on Rails" is a Ruby framework, "Django" is a Python framework, "ZendFramework" is PHP framework and "Spring" is a Java framework just to name a few.

I recommend "Django" to anyone. Why I say this because,

- Django is time-tested
- Django has been crowd-tested
- Tons of Django packages available
- Django has excellent documentation
- The Django community is amazing and supportive

Database

In order for most web application to function, they need to be able to store data somewhere. That somewhere is usually a database of some sort. Some big name brands and household names are MySQL, PostgreSQL, Oracle,CouchDB,Redis,HBase and the list is pretty endless.

I prefer MySQL. This DBMS gets the number one nod mainly because the community version is free and is a great platform to begin learning on. There are, of course, commercial versions of MySQL for sale once you get to the point where you are developing large scale commercial applications, but getting started will cost you a big fat nothing.

Oracle Express Edition is also a good database server. Oracle Express also has tools and a separate server application. It also has more operating system options than Microsoft SQL Server Express.

The Server – Software

Now you got everything work, but something needs to be "answering calls" and serving up what you have to offer. The server's role is to accept the "request" of some action , and run it. For this, like everything, various vendors exist: Apache, Nginx, Tomcat.. the list is endless.

Is it good idea to use Apache Webserver in front of GF or Tomcat? Does it improve the performance/security?

- **Clustering**. By using Apache HTTP as a front end you can let Apache HTTP act as a front door to your content to multiple Apache Tomcat instances. If one of your Apache Tomcats fails, Apache HTTP ignores it and your Sysadmin can sleep through the night. This point could be ignored if you use a hardware loadbalancer and Apache Tomcat's clustering capabilities.

- **Socket handling/system stability**. Apache HTTP has better socket handling with respect to error conditions than Apache Tomcat. The main reason is Apache Tomcat must perform all its socket handling via the JVM which needs to be cross platform. The problem is socket optimization is a platform specific ordeal. Most of the time the java code is fine, but when you are also bombarded with dropped connections, invalid packets, invalid requests from invalid IP's, Apache HTTP does a better job at dropping these error conditions than JVM based program. (YMMV).

- **Speed**. Apache HTTP is faster at serving static content than Apache Tomcat. But unless you have a high traffic site, this point is useless. But in some scenarios, Apache Tomcat can be faster than Apache httpd. So benchmark YOUR site. Apache Tomcat can perform at httpd speeds when using the proper connector (APR with sendFile enabled). Speed should not be considered a factor when choosing between Apache httpd and Tomcat.

- **Security**. This topic can sway one either way. Java has the security manager while Apache has a larger mindshare and more tricks with respect to security. I won't go into this in more detail, but let Google be your friend. Depending on your scenario, one might be better than the other. But also keep in mind, if you run Apache with Tomcat - you have two systems to defend, not one.

The most popular of the alternative Web servers, with almost 24 million Web sites and 7% of the global market, is **Nginx** (pronounced “engine X”). Like Apache, Nginx is an open-source Hypertext Transfer Protocol (HTTP) Web server. Nginx also includes an Internet Message Access Protocol (IMAP) and Post Office Protocol (POP) server.

References

http://blog.teamtreehouse.com/choose-django

https://www.sitepoint.com/server-side-language-right/

http://wiki.apache.org/tomcat/FAQ/Connectors#Q3

http://www.computerweekly.com/feature/Choosing-the-right-database-management-system