Adrian D. Finlay
@thewiprogrammer. Writer @hackernoon. Code, LOTS of it. Mangos, LOVE THEM! Barbering. Health. Travel. Business. & more! Network w/ me @ adriandavid.me/network
Sep 25, 2017 · 49 min read

How To Become a Java Web Developer (2017): The Common Core Skills

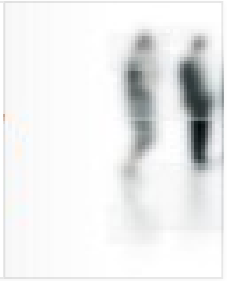
Java Banner [10]

Due to it's length, this publication is alternatively available as a three-part series. You can find the first part of the series below.

How To Become a Java Web Developer (2017) — Part I: Java Web Development Rudiments

Somehow you've stumbled into programming and gained interest in the Java Programming languag...

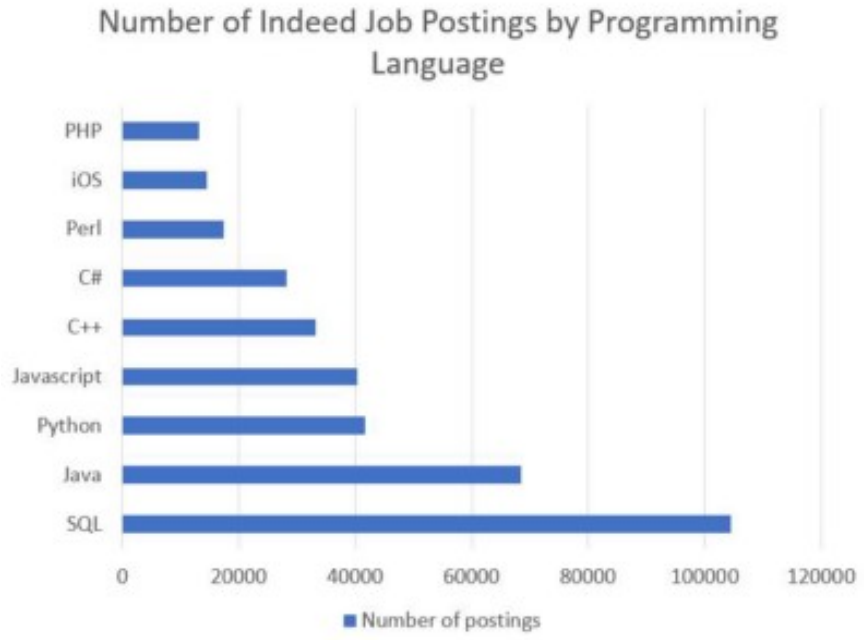
medium.com



Somehow you've stumbled into programming and gained interest in the Java Programming language. Maybe it was because of school. Or maybe you saw the Java logo on your desktop. Perhaps you figured out that **Java is almost everywhere**—from Blu Ray software to Android devices via the openJDK. You may have noticed that Java routinely ranks as the #1 (or among the top) most popular programming language according to the TIOBE Index and that 6 billion devices use Java [1][2]. You may have also noticed that Java is the second most popular programming language to JavaScript (excluding SQL, which is a query language) according to Stack Overflow[35].

According to the BLS: Software Developers raked in an average of $102,280/yr and $49.17/hr in 2016 and demand is growing at a rate of 17% from 2014 into 2024; By 2020, there will be 1.4 million CS related jobs and only 400,000 CS graduates with the necessary skills. **Java is commonly one of said skills.**

Peradventure you've done some research and discovered that ~**70,000 job postings** (a +30,000 increase from 2016) on the popular job site Indeed.com included the Java programming language as a skill in 2017[2] .



[3]—codingdojo.com

Curious as you are, you may have noticed that 3/4ths (depending on the source, 65%–85%) of Java developers work on **web or enterprise applications** and that 90% of the Fortune 500 use Java on the server-side[4]—a claim stated by Oracle. Indeed, it's an excellent time to be an Enterprise Java developer.

Browsing through the job boards you may have noticed a theme—keywords like "J2EE", "Spring", "Servlets", "Hibernate", and the like. You may have asked yourself—"How do I get from here to there?". Perhaps you're enticed by reports of above average compensation packages for Java Enterprise developers or dreams of designing the "next big thing", or both.

Maybe you've done a Google search and have only found John Thompson's article on the topic [5]. Maybe you haven't asked any of these questions: all you want to know is **how to become a Java web or enterprise developer.** Either way, **the answer is—there is no single definitive answer**. The skills you will need will ultimately depend on the specific role you are seeking. Furthermore, the skills you may generally find Java roles demanding may vary based on a bevy of various factors, such as region and industry, among others. The truth is (and I might be vilified for this) that **you don't need to have every skill listed on a job advertisement** (even among those marked required!). Many employers hire individuals based on aptitude, ability to learn quickly, and, to put it bluntly, because they like them[6]. Despite this, **skills still matter—the work doesn't get done without them.**

Having said all this, perhaps the question to start with is: **"Which are the common core skills of Java Enterprise & Web Developers?"**. The skills discussed will describe the skills of engineers who work on the full software stack. However, as engineers work on various parts of the stack—front end, back end, EJBs, etc.—one may read the portions relevant to them.

It is important to note that this article will not discuss the interview and pre-interview aspects involved in acquiring a job in java development and will instead focus on the core skills of Java Enterprise Developers. To dig into that material, I overwhelmingly recommend **Cracking The Coding Interview by Gayle Laakman McDowell**. You will not be disappointed!

It is also important to note also that this article will focus on technical skills as opposed to soft skills. While soft skills are **invaluable,** they fall out of the scope of this article. However, it is worth noting that good organization, communication, and interpersonal skills are key. **Check out Soft Skills by John Somnez** and **The Clean Coder: A Code of Conduct for Professional Programmers by Robert C. Martin**.

Without further adieu, let's take a peek.

Impatient? Here's a condensed sneak peek: http://bit.ly/2gBUBRJ If the bit.ly link expires or throws a 404, comment on the article, and I will provide a new hyperlink.

Got Suggestions? Hate the article? Love the article? Anything in between? **Let me know in the comments.**

Java—This one is fairly obvious.

> *"The Java Programming Language is a is a general-purpose computer programming language that is concurrent, class-based, object-oriented,and specifically designed to have as few implementation dependencies as possible."[7]*

Java is designed to *Write Once & Run Everywhere*, a notion otherwise known as **WORE**. This is accomplished through the use of executing *Java bytecode* on the *Java Virtual Machine*, which is itself implemented on several platforms. We won't spend much time here, as it should be obvious to the reader that one needs to know the Java language to be a Java developer (duh). **The point to grasp** is that a serious Java developer should **thoroughly understand and aim to master all the elements of the Java programming language itself** and core features as well as important packages (including but not limited to): java.lang, java.util, java.concurrent, java.io.

Why I Personally Like Java: You can develop robust, scalable, trusted, tested, reliable **Desktop** (JavaFX, Swing), **Mobile** (Android), **Web & Enterprise** (JEE, Spring, Struts) solutions.

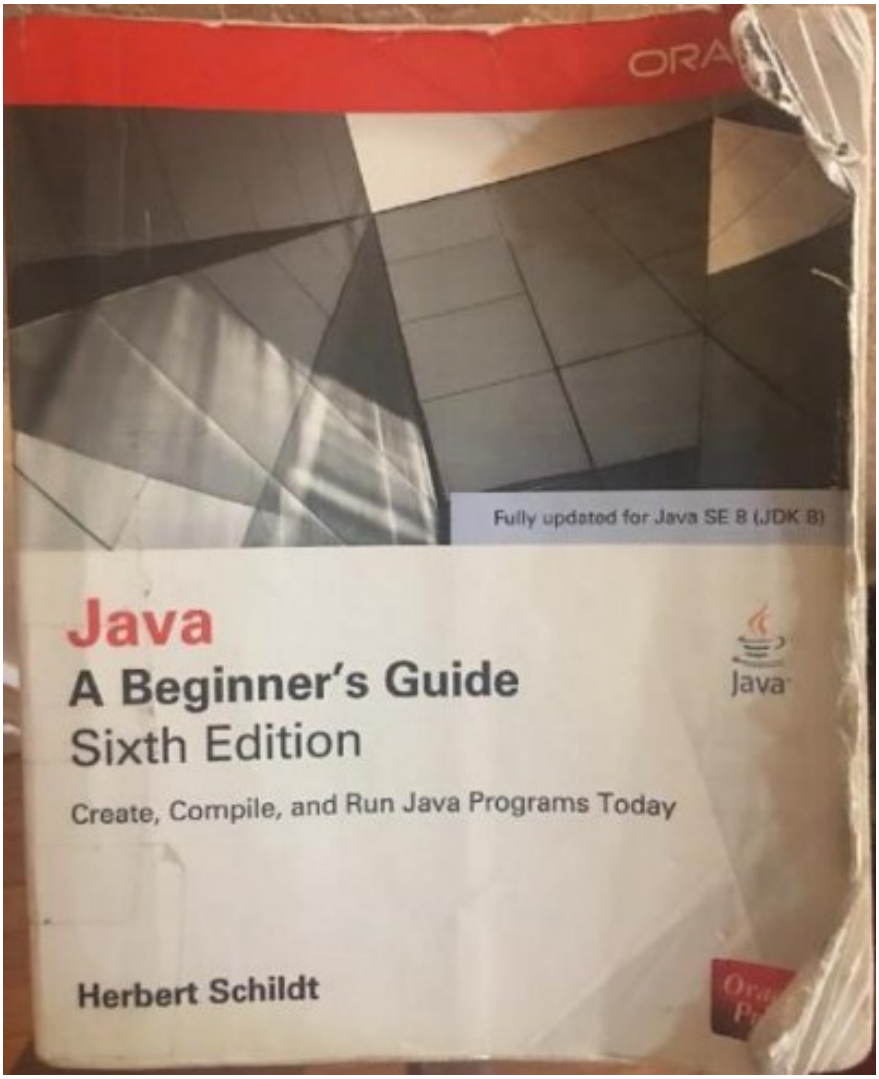
Keep Track of Java Evolution: Java 9 is (as of publication) slated to be released in 17 days on September 21. Will it actually meet it's release date? You have to keep up to date to find out. Don't get left behind the curve—Java 9's modules will fundamentally change how we approach Java programming. Consider attending (or at least follow online) the Java One Conference hosted by Oracle every year during early October. You should also follow a magazine or online publication such as DZone / Java Zone, which features articles talking about the latest and greatest in Java Development.

UPDATE: Java SE 9 / Java EE 8 met it's release date! Release notes for Java SE are here. Release notes for Java EE are here.

Where to learn, recommended resources:

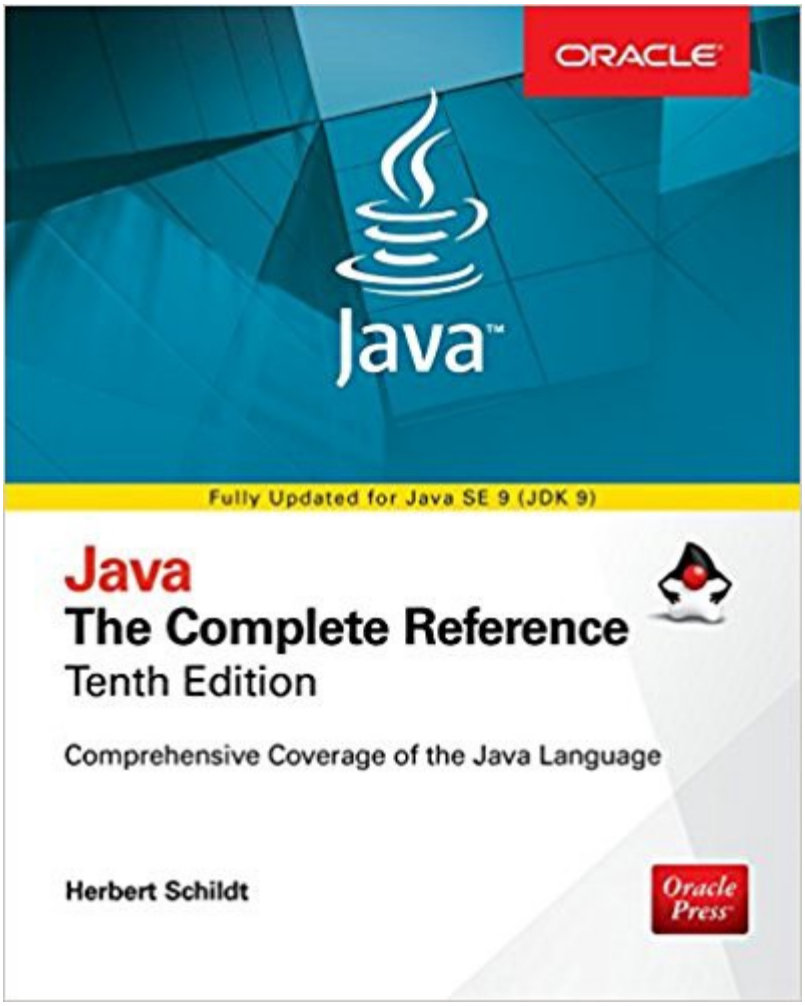
Programming is something best learned by both understanding the tools you are working with as well as plain old writing code, to be blunt. My personal recommendations for learning Java are:

Java: A Beginner's Guide, 7th Ed. (Oct. 13. 2017 release date)
- The 6th Edition to this book was my Introduction to Java. I came to Java after having learned C++ in school. The details seem all to fuzzy. I remember the Fall 2014: I would sit in my IST 302 class wishing the material were about Java instead of Project Management, and I loved Project Management very much & still do. This book is my number one resource for beginner's. I have personally spoke to the Author, Herbert Schildt, thrice, and have zero question in my mind that the 7th edition will be just as excellent. Published by the Oracle Press. Covers Java 9.



My tattered, worn personal 6th Ed. Copy. Affectionately known as my second love.

Java: The Complete Reference, 10th Ed. (Oct. 13. 2017 release date)
The next logical step after reading the Beginner's Guide. Jam packed with critical information. **Compulsory reading.** Also written by Herbert Schildt and published by the Oracle Press. Covers Java 9. **I personally believe that all java programmers should own this text, with no exception.** I own the 9th Edition of this text.



Java, The Complete Reference, 10th Ed.—Herbert Schildt

Core Java: Volume I, Fundamentals, 10th Ed.
Let it be known: After reading this text, there is no way you could tell Cay S. Horstmann that he doesn't know his stuff. This text reeks of the Author's very deep and thorough knowledge of the Java Programming Language. Goes beyond the language basics and leaves the user satisfied. Covers Java 8.

Core Java: Volume II, Advanced Features, 10th Ed.
The next logical step after reading the First Volume. Every bit as satisfying. Covers Java 8.

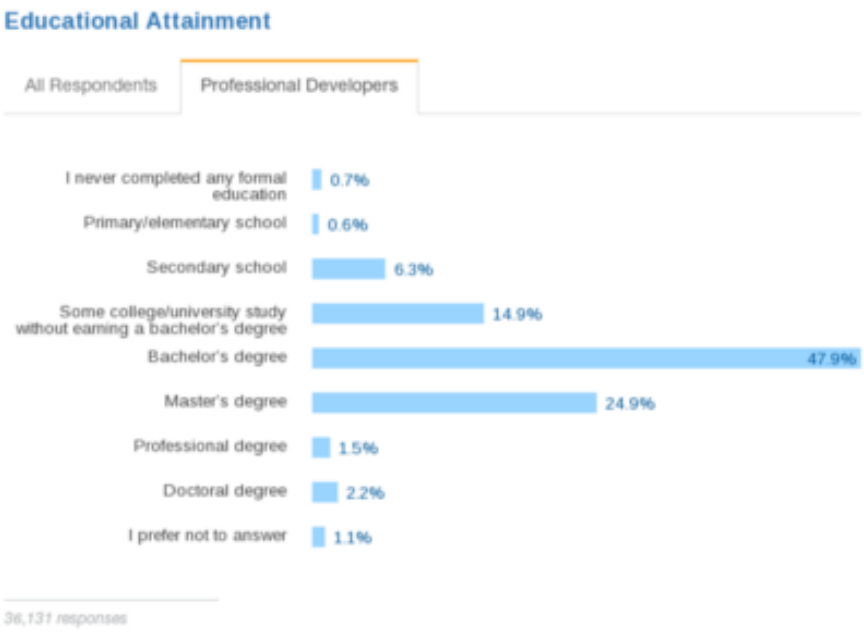
Qualifications—How to say you know what you know

This one is a bit tricky. The most common qualifications of software developers are: **college degrees, high-school diplomas, technical certifications,** and **technical bootcamp completion certificates.** Qualifications are important, in my opinion, because they are a standardized, mostly-reliable way of estimating a person's level of competence and knowledge of a particular domain.

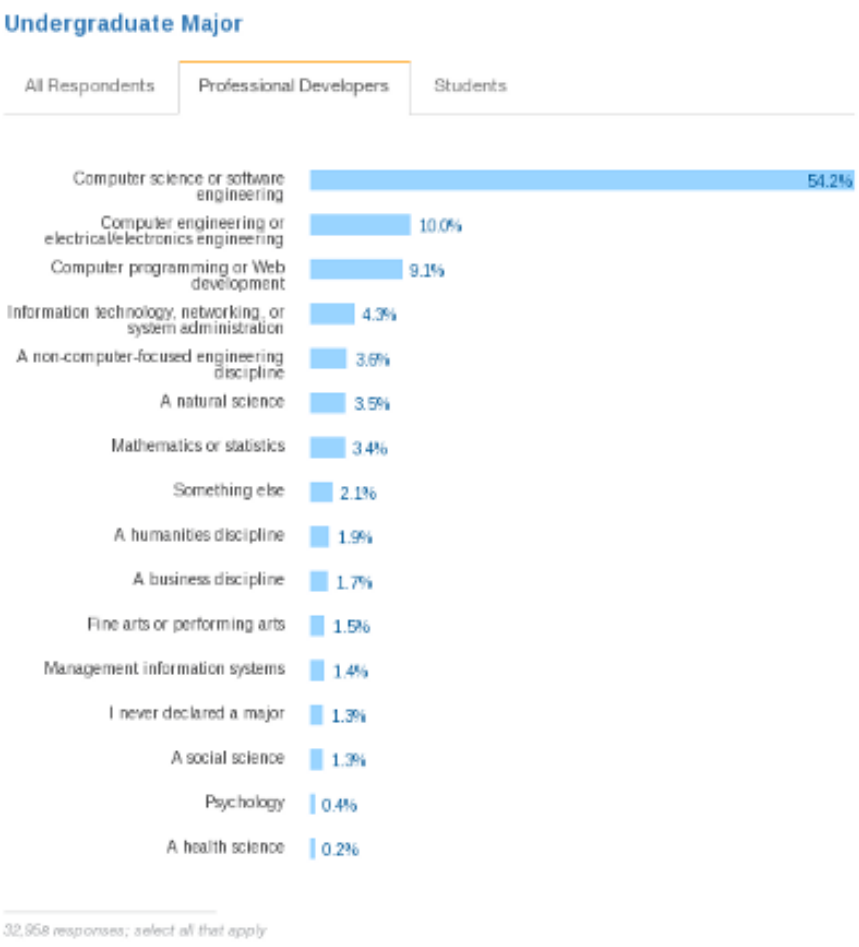
> ***Formal Education:*** *Degrees & Diplomas*

According to Stack Overflow (the most popular programming community on the web and the most popular programming informational site on the internet [36]), **76.5% of professional developers hold a bachelor's degree or higher, 14.9% had some college experience, 7.6% had below college education,** and 1.1% preferred not to answer. Slightly over half (~56%) of people who took this survey provided a complete response to the following:

Stack Overflow Developer Survey 2017—*Educational Attainment* [35]



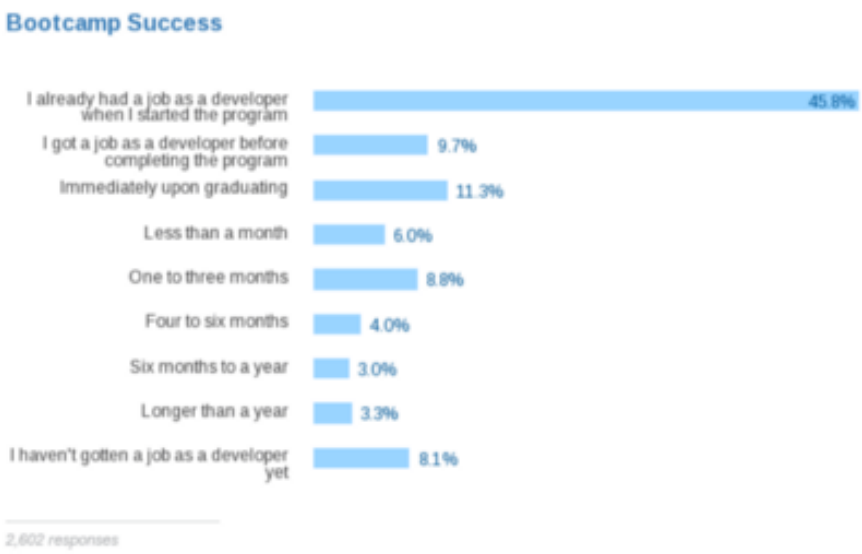
Of those who had studied at a college or university, **54.2% studied Computer Science or Software Engineering**, 24.9% majored in a closely related discipline, and 20.9% focused in non related fields[35].



Miscellaneous Highlights of the Responses by Professional Developers on the Education portion of the Stack Overflow Survey to various questions:

- 67.9% said formal education was at least somewhat important.
- 91.1% of developers reported being at least partially self-taught.
- 9.4% of developers used bootcamps. 22.8% recommend coding bootcamps.
- **16.4% sought industry certifications**.
- Several others used on-the-job training, online courses, open source contributions, hackathons, coding competitions, & part-time courses as a means of education.

> *Bootcamp Success among Software Developers [35]*



The Stack Overflow Developer Survey 2017 data is very interesting and you can find it here. You may regard it as a SOTU of sorts, for the developer community. I strongly recommend reading their annual reports.

> *What does this all mean?*

There are a number of conclusions (and indeed a number of data sources from which to draw said conclusions) one can make from the available data to try to determine the importance of qualifications to the software engineering profession. I won't make them for you, but a few things stand out to me. These are my opinions.

- While you do not need a degree to be employed as a Software Developer, **the majority of developers seem to hold degrees**. Try browsing the indeed software developer listings for your local area—I'm willing to bet that you will find as I found that it all depends. Some organizations state a degree as a requirement while others state is at a preference. Your mileage truly does vary, but I willing to guesstimate that the majority will at least prefer a degree. **Even if they do state that they require a degree, apply anyway**. I will again be vilified for this, but at the end of the day a company is looking to do business with you—if you can make them money, they may be happy to hire you. **It's a dirty yet oft-repeated secret that companies may hire people who don't have all the requirements**. You've got nothing to loose. The worst they could do is not hire you.

- **There are several qualifications that employers recognize when hiring Software Developers**. On average, an immediately relevant degree seems to me to be the most prominent qualification above bootcamps and closely related degrees. While this may change, this trend seems to be holding fast, for now at least.

- Perhaps the most astonishing findings—among professional developers who did seek a degree, **slightly over a fifth didn't study a relevant topic**, **slightly over a fourth studied a closely related field**, and the most astonishing**—only 54 studied either CS or Software Engineering!** What this makes clear to me is that **your college major** is probably **not that important when being considered for a developer job**. I wish Stack Overflow would separate Computer Science from Software Engineering for the 2018 Survey, however, as the two curricula are definitely not the same.

- That 91.1% of developers are self-taught is indicative of the fact that **real world work is simply beyond the topics taught in the classroom**. The classroom incontrovertibly has it's use. However, if you want to be a developer, you **most certainly will never be able to rely on the classroom alone to give you all the tools you need**. You will also **not be able to afford not continuing to learn**. Period.

- **Advanced degrees are not as important as in many other fields**. ~48% of professional developers have just a bachelor's degree. While the data alone may not fully support the claim that advanced degrees are not as important, I think the fact that the data shows that the vast majority of professional developers (69.2–70.3%) lack a degree past a bachelor's is telling. Take a browse through the job listings—how often do you see a Master's Degree (or higher) as a requirement? I rest my case.

> ***This is all fine and dandy but… which qualification should I get?***

At the end of the day, I can't tell you which option is best for you.

Some **Colleges & Universities** can be devastatingly and exorbitantly expensive, to the ire of many people. At a top ranked CS institution (I won't say where) between Tuition, Fees, Room, Board, Books, &

Supplies,etc. could cost as much as $70K USD per year. **That's equivalent to someone's annual salary and higher than the median household income in the United States.** Should you finish school in 4 years that would run you about $280K. Should you take out loans for school, you would pay more than that, obviously, due to interest. The decision is a personal one—one is tasked to determine for their own how much they are willing to pay for a degree as much as they'd be willing to pay for a car, or anything else. My thinking is to look on the **return on investment—how much money are you in a position to earn because of the particular degree you achieved**.

On the other hand, financial aid is available at many schools and low cost education exists. **Degrees are the longest standing and most marketable qualification, by far.** A college degree provides you the flexibility to more easily leave software development should you later decide to. Also, a college degree is critical to obtaining a higher-ranking or managerial job. If thought about carefully, a college education can be a very rewarding investment. After all, bachelor's degree graduates are reported by some to earn well over a million more than their non bachelor degree holding peers [37]. I think there are many explanations for this that make sense. However, I do not believe that the correlation of incomes and bachelor's degrees is evidence alone of cause (degree) and effect (money). Notwithstanding, a college degree is valuable and is a solid option. I do not, however, recommend a for-profit or non-accredited college or university.

In my opinion **Certifications** are useful and I really like some of them, but they will not give you as much employ-ability as would a university degree. Nevertheless, they can potentially be a good (In my opinion) qualification to make you stand out of the pack for a targeted area, like Java. **Check out the Oracle Certified Associate & Programmer in Java Programming**, here. Oracle makes several strong claims about the industrial value of their Java certification and I think very positively of it. I have met software engineers for which this certification has benefited them and I have seen the certification asked for in a few job postings.

I was previously not a fan of **Coding Boot-camps** and had largely dismissed them as a scams but my mind has changed. After-all—Programming is not a gimmicky tricky that can be learned in a mere 16 weeks? That was my former thinking. While I still don't believe that one

can become a full fledged developer in 16 weeks, I think a lot of progress can be made. I think Boot-camps can be useful if the boot camp has strong industry connections, if the individual is self-motivated and learns outside of the boot-camp, and if the individual posts their code on the web for organizations to see. Check out Gayle Laakman McDowell's (author of *Cracking The Coding Interview*, among other books) article on the matter: So that whole coding bootcamp thing is a scam, right?

Self Taught. Usually these individuals hold a high-school diploma or GED. In my opinion, many of these individuals wind up more knowledgeable about their area of study **because they had the freedom to focus on said area**. All things considered, I personally believe that all serious developers have to be self-taught to a very significant degree, and that seriously self taught people are very commonly passionate. After all—willingly applying the discipline needed to become a Software Developer without school holding your hand is hard work. If you take it upon yourself to do all the dirty work of discipline and discovering what it is you need to know to become a developer, then immediately I know that you take software development **very seriously. Aside:** CNN Money reported than in 2010 about 38% of web developers had less than a four-year college degree, according to the US Census.

You should check out Cory Althoff's Self-Taught Programmer's group ;)

Fundamental Computer Science—Core Foundations

> *"Computer Science is the study of computers and computational systems. Unlike electrical and computer engineers, computer scientists deal mostly with software and software systems; this includes their theory, design, development, and application." [38]*

Whether you are self-taught or you graduated from University, you should have a handle on core fundamental computer science. There are several reasons why you should understand the fundamentals of Computer Science. Instead of listing them, I will try to summarize them all with one statement—You need to understand: **what exactly it is that you are doing in coding, what exactly it is that you are coding on, and how exactly it works**. Can you skate by without knowing CS

fundamentals well? Maybe, maybe not. Should you try? **Definitely not**. The situation is loosely akin to playing basketball while blindfolded. You may have some success, you may score a basket, but you **can't really see so as to properly navigate what it is you are doing.** You should understand what it is that you are actually doing when you compile, link, and then execute a program. All the way down to ones and zeros, and beyond.

Perhaps Basketball isn't your taste or that analogy didn't quite work the way I wanted it to. Take for example, driving: It's useful to know when you are driving how the car works—but you can get away with not knowing many of the details, often, but not always, when all you need to do is to drive. The situation with programming is a little more subtle than that. When all you need to do is build an application, you might be able to get away with not understanding how the compiler does what it does. However, the subtlety is that **programming requires a little more understanding of the underlying phenomena than driving does to perform the task well**. Understanding and refining how you use and design data structures and algorithms is key to just about everything—from security to performance and everything else.

You should understand why the first method of integer addition is less efficient than the second method of addition in the following:

Another thing: Did I need to use the **varags**? Nope, absolutely not. In actual reality the JVM doesn't notice the difference—it still sees an array—but you should understand the difference.

And so on.

Side Note: In the process of crafting this example I learned that the Integer constructors are deprecated in Java 9 in favor of static methods! You **never** stop learning in programming. For those curious, you can find out more, here.

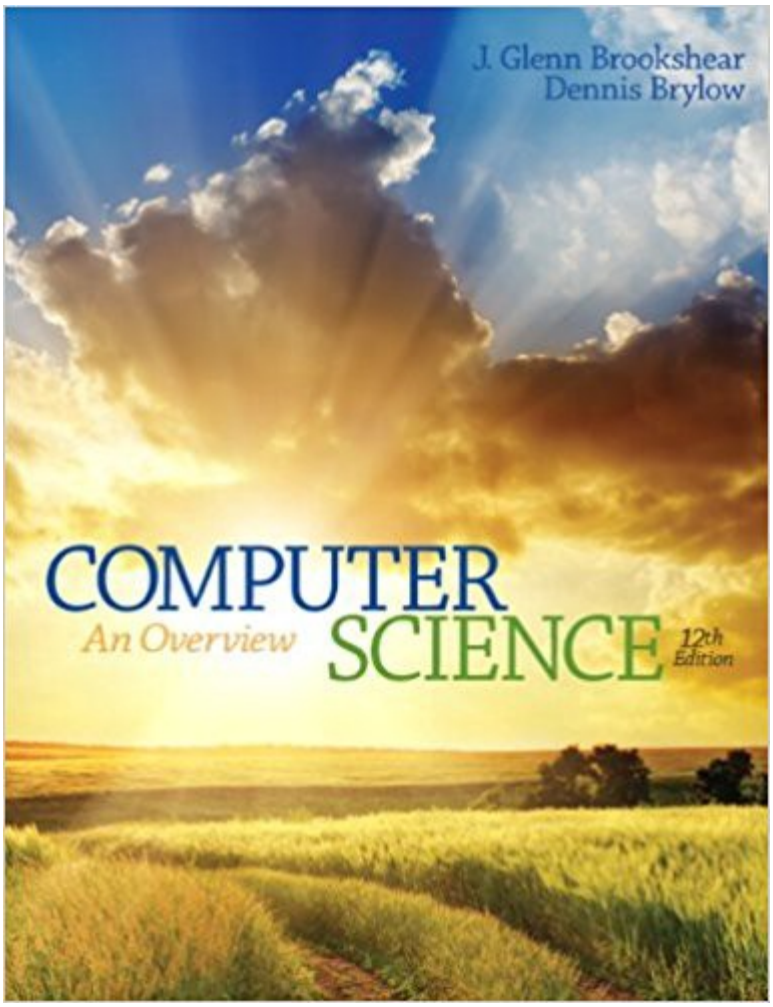
According to the **Computing Sciences Accreditation Board**, a board that includes folks from the Association for Computing Machinery (ACM) and the IEEE Computer Society, there are for foundational areas to Computer Science [38]:

- The Theory of Computation
- Algorithms & Data Structures
- Programming Methodologies and Languages
- Computer Elements & Architecture

It is beyond the scope of this article to discuss in depth the foundations of Computer Science. However, I will point you in the right direction. See below.

Where to learn, recommended resources:

Computer Science: An Overview, 12th Ed.



While there may be many texts on several different topics concerning Computer Science, this text was among the only ones I could find that discussed as broad an over view of the fundamental ideas of computer science. While this book may not provide you with everything you need to know, it will equip you with starting knowledge to work with when thinking about Computer Science.

The Theory of Computation—Chapter 12
Algorithms & Data Structures—Chapters 5 , 8
Programming Methodologies and Languages—Chapter 6
Computer Elements & Architecture—Chapters 1 , 2

SQL—The world is all about data

> *"Structured Query Language—SQL is a domain specific language used in programming to manage data held in a relational database management*

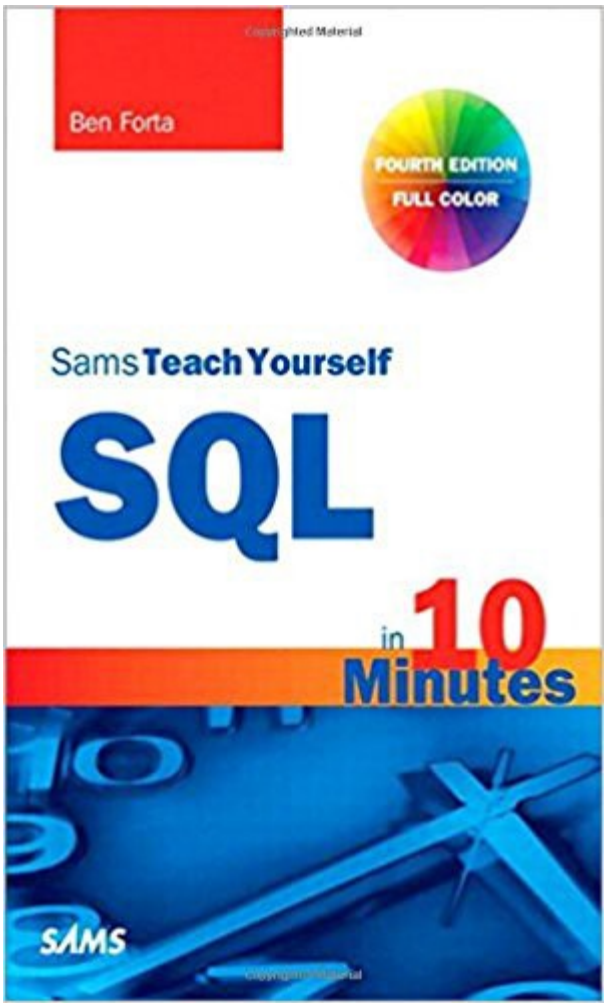
> *system (RDBMS), or for stream process a relational data stream management stream (RDBMS)."[8]*

If you're going to be doing back-end Java development, you'll likely be dealing with relational databases in some fashion; you will need to understand SQL. In addition to learning SQL, it's safe to say that **one is pretty much required to learn a RDBMS platform such as MySQL or PostgreSQL**, as ANSI SQL can only get you so far. The current edition of ANSI SQL is SQL:2016.

According to Rebel Lab's Java Tools & Technologies Landscape 2016 Survey [27], the following RDBMSs have the leading mind-share among Java Developers: Oracle DB with 39%, MySQL entrenched with 38%, & PostgreSQL with a strong 29%. It is important to note that respondents could choose more than one RDBMS.

Where to learn, recommended resources:

Sam's Teach Yourself SQL in 10 Minutes a Day ~ Ben Forta
This text should be considered compulsory reading for application developers. At 288 pages, it is reasonably digestible for a week's reading. You will not be able to grab a DBA job after this book, but it doesn't matter, because you're not trying to be a DBA, you're trying to be a Java Enterprise Developer. It helps that this book is one of the most widely (if not #1) most printed text on SQL a testament to it's quality and consequently Forta is widely revered for his work. At present, Forta's text is in it's 4th Edition which covers SQL:2011 and SQL:2016 is upon us. Despite being one version behind, SQL:2011 is the most popularly implemented version as of present and the updates to the SQL standard (SQL:2016) are available freely online[11]; Most of them concern JSON. Because it is compact, concise, and technically sound, I still recommend reading this book.



Sam's Teach Yourself SQL in 10 Minutes — Ben Forta

Tutorial's Point MySQL Tutorial
MySQL is a very popular and very powerful common open source RDBMS. MySQL was made very popular in part by the popularity of LAMP (Linux/Apache/MySQL/ Perl, Python, PHP) stacks of the late 1990's and early 2000's. You can read more about the history, here.

OOP— Grouping together the "do" & the "stuff"

> *"It is dangerous to make predictions, especially in a discipline that changes so rapidly, but one thing I can say with confidence is that I have seen the future, and it is object-oriented"—Grady Booch, Father of UML, 1996 [9]*

Object Oriented Programming (OOP) is a programming paradigm by which an application is structured in terms of object behavior and interaction. In OOP, an object is an abstraction that binds attributes and

subroutines. Commonly, contemporary OOP parlance refers to attributes as fields or data members and subroutines as methods. Class based object design is the most common contemporary approach to OOP.

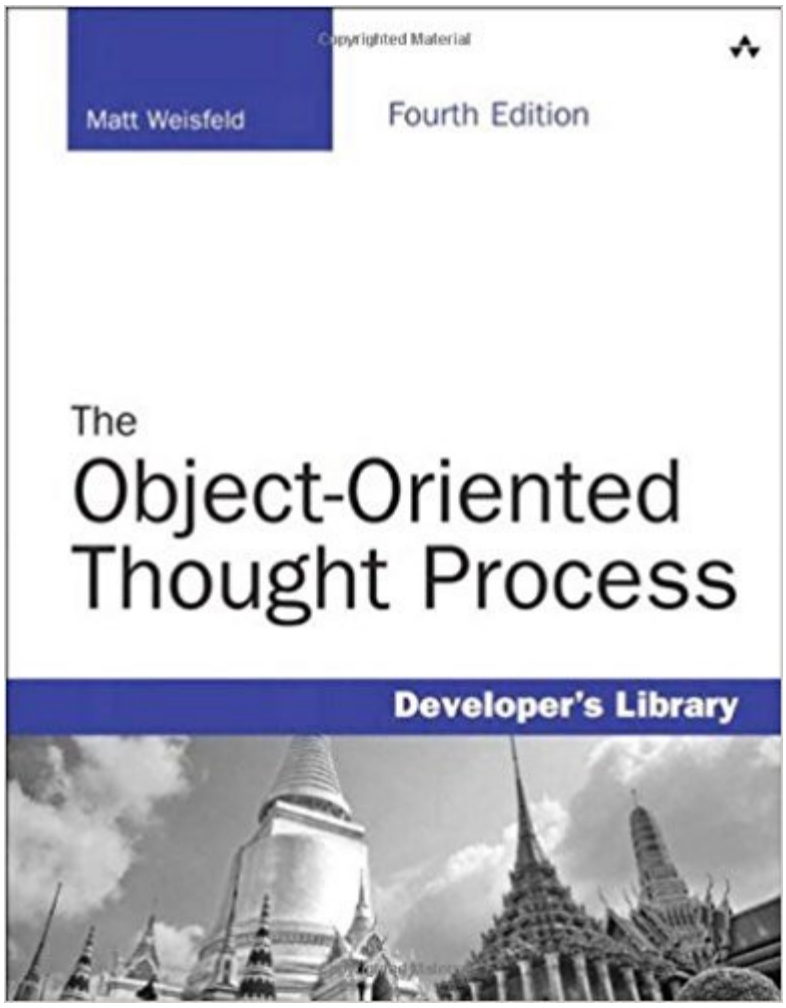
Most of the popular and modern languages: Python, Java, C#, C++, Ruby, PHP, Scala, etc. all include support for Object Oriented Programming as a core component of their programming model. The fact is—a contemporary developer will probably not be able to circumvent the Object Oriented Paradigm (unless you're into weird stuff like the programming language "Brainf***", which already has developers extending it to be object oriented!). Get the point? OOP is ubiquitous.

Key to the architectural design of Object Oriented Software is the **Unified Modeling Language (UML)**. UML is a common means by which Software Developers describe and design the way objects are to behave in an Application. "The Unified Modeling Language (UML) is a graphical language for OOAD that gives a standard way to write a software system's blueprint. It helps to visualize, specify, construct, and document the artifacts of an object-oriented system. It is used to depict the structures and the relationships in a complex system."[42] UML is not restricted to Object Oriented Design (OOD), however.

The despicable truth: Most of us will probably not go out of our way to grab a generic Object Oriented Programming book (phew!, I said it). Instead, we will probably learn Object Oriented Programming while learning a programming language or through a university course. However you do it, **you must do it**, as this skill is borderline quintessential to the field. You will be a better developer for it. You will thank yourself for it. Your boss will thank you for it, and so will the future maintainers of your code ;) .

Where to learn, recommended resources:

The Object-Oriented Thought Process, 4th Ed., Developers Library
My personal recommendation. This text is written in a language agnostic way (though primarily C#), with examples in several languages available online from the publisher. I really like the way this book is written and I have not seen something (as modern at least) written quite like it.



Head First Object-Oriented Analysis and Design
Although I am not a personal fan of the Head First series (just give it to me raw, forget the kiddy stuff), many people are. This text is very popular and I have confidence that you won't be steered wrong. To be technically accurate: This book is more concerned with Object Oriented Analysis than the last text which is not exactly the same as Object Oriented Programming. Likewise, Object Oriented Design does not necessarily mean programming, and as such *Your Mileage May Vary* when considering this book for OOP specifically. That being said, although the philosophical approach to the topic differs, they concern the same and similar material.

Some Notes: While book reading is a great approach, you may also find benefit from taking a course, like I did. This is the approach by which (in my estimation) the majority have learned OOP.

Tutorials Point Unified Modeling Language(UML) Tutorial

Tutorials Point Object Oriented Analysis & Design Tutorial

While OOP is the dominant paradigm, **Functional Programming** is experiencing a resurgence in popularity and you should definitely be aware of it. This influence has been felt community wide and example such as 1) Clojure, Scala, Jaskell, 2)Emergence of functional interfaces, lambda expressions, and method references in Java 8, 3)Emergence of Functional Support in Spring 5[12], among others, testify to this.

HTML, CSS, JavaScript— Visual Representation of web pages & their behavior

> *"HTML is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web.* ***HTML specifies the content of web pages, CSS specifies the presentation of web pages, and JavaScript specifies the behaviour of web pages.****"[13]*

This trio represents the cornerstone core technologies involved in what is called *front end programming.* In addition to learning these languages you should also considering learning a View/MVC/MMVM framework such as AngularJS, Angular 2/4, or ReactJS + Redux, Vue, etc. The inclusion of one of these frameworks greatly simplifies development for many projects, relieving you from having to become bogged down in every detail of HTML. Today's Java Developer should be familiar with at least one of these groundbreaking frameworks.

Where to learn, recommended resources:

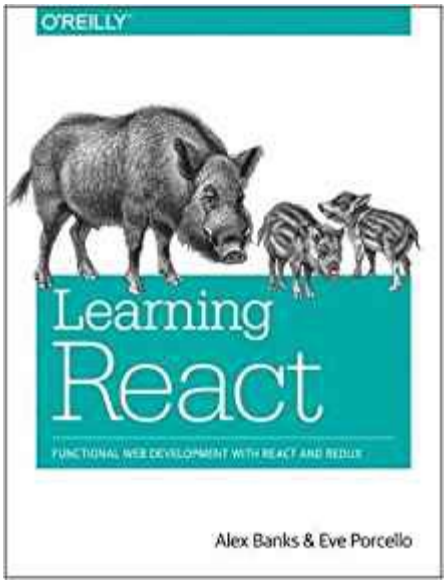
HTML5
https://www.tutorialspoint.com/html/

CSS3
https://www.tutorialspoint.com/css/

JavaScript
https://www.tutorialspoint.com/javascript/
JavaScript: The Good Parts

JavaScript Framework—ReactJS
https://www.tutorialspoint.com/reactjs/
https://facebook.github.io/react/tutorial/tutorial.html
Learning React

Learning React: Functional Web Development with React and Redux



JavaScript Framework—Angular 4(Angular Platform 2+)
https://angular.io/docs
ng-book: The Complete Guide to Angular 4

A special note on Angular:
TypeScript—An ECMAScript superset that compiles to JavaScript (consequently all JavaScript programs are valid TypeScript programs). TypeScript notably adds **classes** among several different language features.

AngularJS/Angular 1.X—The original Angular Framework. Written in JavaScript.

Angular/Angular 2.X—A complete rewrite of AngularJS. Written in TypeScript.

Angular 4—The next version in development of Angular2, Backward Compatible with Angular 2. Written in TypeScript.

All Three: HTML, CSS, JavaScript
https://www.amazon.com/Web-Design-HTML-JavaScript-jQuery/dp/1118907442

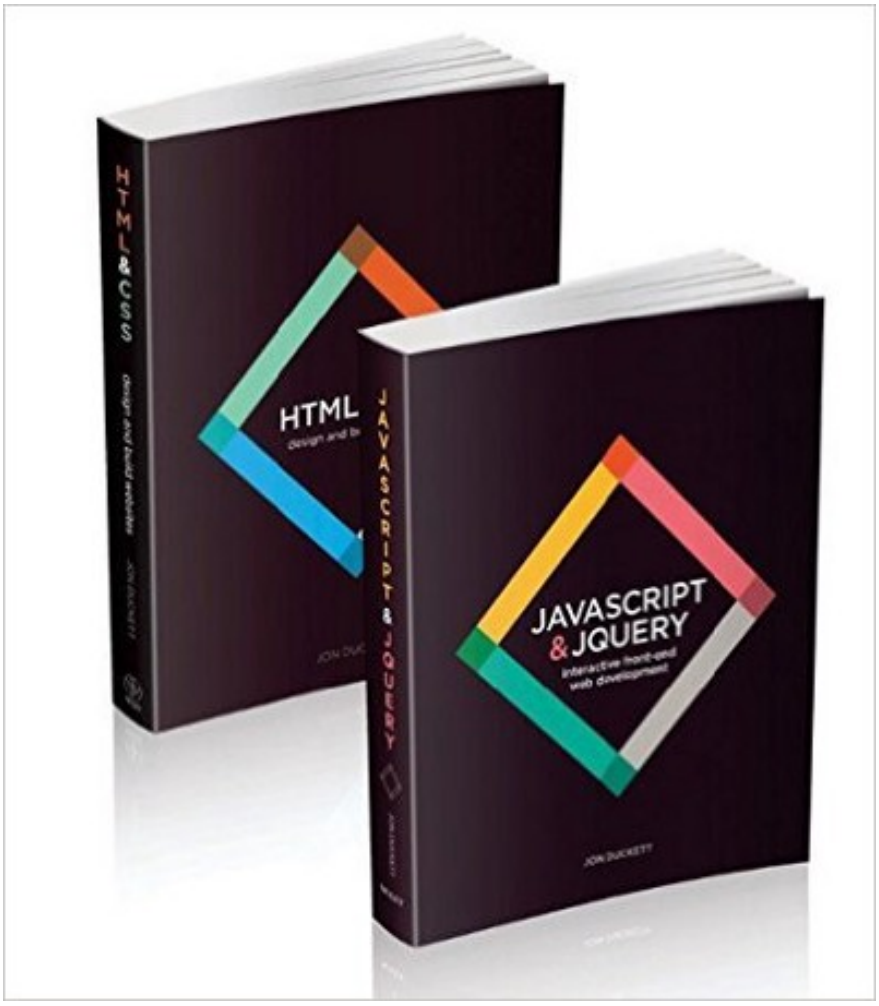
While Duckett's text was published in July of 2014, and evolution, particularly in the CSS and JavaScript communities has led to newer editions of both languages, this text is still relevant. The reason for that is largely due to version adoption. According to the W3Schools[21], **ECMAScript 5 (2011) is the only fully supported version by all major web browsers**, while ECMAScript 6 (2015) is partially supported and ECMAScript 7 (2016) is poorly supported. Similarly, although CSS4 is upon us, many browsers have still not fully supported CSS3.

For example, the results of css3test.com running on my Win10 x86–64 box,

Opera Browser v. 46—**58% CSS 3 Support**
Google Chrome Browser v.60— **58% CSS 3 Support**
Google Canary Browser v.63—**59% CSS 3 Support**
Google Chromium Browser v.60— **58% CSS 3 Support**
Firefox Browser—**67% CSS 3 Support**

As of: 09/16/2017.

Quality work takes time. Consequently, this book is still very much relevant.



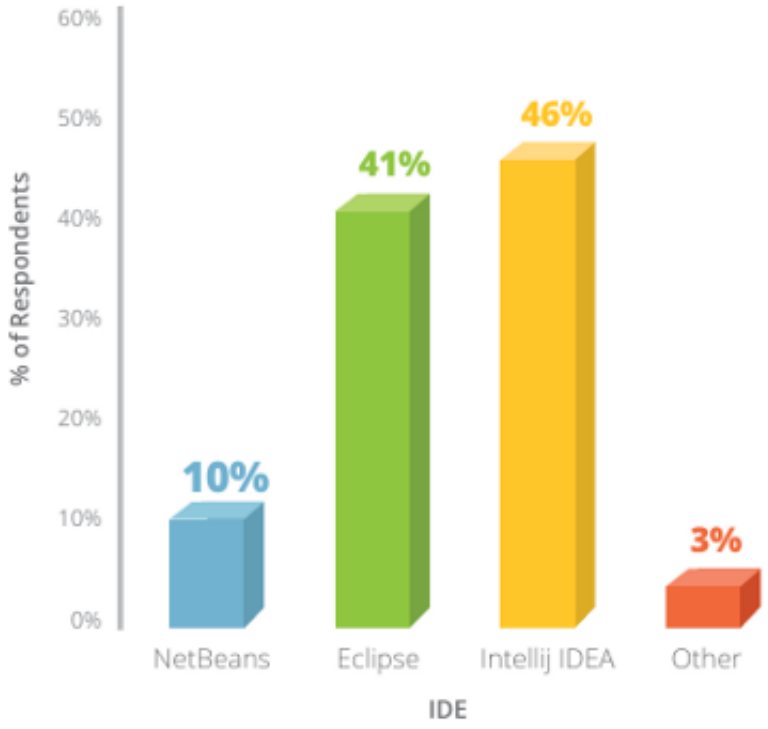
Web Design: HTML, CSS, JavaScript—Jon Duckett

Integrated Development Environment — The Developer's Workdesk

The Java Environment is rich with tools. Among them, **Integrated Development Environments(IDE)** are standard tools that all Java engineers need understand. While it is not necessary to use an IDE to build software, IDEs dramatically increase productivity by integrating the various common tools used in building software into one environment. Among a whole host of productivity tools include: a debugger, utilities for managing dependencies, facilities for plugins, auto-complete, automation tools, templates, compilers, and many more. Practically speaking, real world development demands that you understand how to use and IDE.

IDE Mindshare among Java Developers [27].

Figure 1.11 Battle of the IDEs



IDE Mindshare [27]

There are several IDEs available and they provide varying support for various languages and technologies. My personal favourites happen to be **IntellIJ IDE** and **NetBeans IDE**. The company who develops IntelliJ, JetBrains, has in particular had unique influence in the Java community. In addition to producing a very popular IDE, the Android Community has adopted IntelliJ as the basis of the Android Studio IDE. Additionally, Google now officially supports Kotlin, a JVM language that is source-to-source compatible with Java, for Android Development.

Java EE—Building an Enterprise System

> *"Since that first release, the platform has become the foundation of server-side development for much of the corporate world, being adopted by and improved by industry heavyweights such as IBM and Oracle."*—Danny Coward, Principal Architect, Oracle Co. [14].

The Java Enterprise Edition is a platform for developing and deploying enterprise software—that is web and networking software; **It is defined by specification**. This is unlike the other editions of Java,

namely, ME and SE. By providing a specification for EE components, this leaves the software vendor with more control over how said components are managed and implemented by the application sever.

My opinion: Knowledge of the Java Enterprise Edition is **compulsory knowledge** to Java Web & Enterprise developers, even if they wind up developing in Web/Enterprise Frameworks such as Struts II, Spring MVC, Grails, Play, etc.

There are several core components of the platform that you should be aware of. Everyone will not agree as to what should or should or not be known. In the end, one should probably invest the majority of one's time into the things that one will actually use. Nevertheless these are the **core ideas.** I will break them down into categories:

Web Components: Servlets, JSP/JSTL/EL, JSF

Servlets are the foundation of the Web Components of the Java EE Platform. Java Servlets provide the functionality to do next to anything with a client-server protocol such as HTTP and a markup language like HTML. A Java servlet is an Java object that processes the server side of HTTP interactions[14].

JavaServer Pages (JSPs) are inside-out servlets. This model came from the observation that wrapping static HTML inside the output stream of a Java servlet was counterintuitive for web designers and led to large, messy, code. Instead of putting static HTML into Java Code, JSPs dynamically put Java code into HTML. This, among many improvements led to a more intuitive web view approach and provided a way to dynamically inject Java components such as beans into static HTML. The **JavaServer Pages Standard Tag Library (JSTL)** is a Java EE Web Component that adds to the JSP specification by providing a standard tag library of JSP tags for various tasks.

Unified Expression Language (EL) is a small language that was design to read and process the values of Enterprise Java Bean but is now used to generally process data in a JSP. EL can use Java Beans, JSP EL Literals, and JSP implicit objects.

JavaServer Faces (JSFs) take JavaServer Pages to the next level by providing a MVC Framework based on reusable components.

Web Services: REST via JAX-RS + JSON-P, Jersey, Familiarity with SOAP

JAX-RS is a Java EE specification for the creating of web services according Representational State Transfer (REST) architecture. **Jersey** is a framework for creating RESTful Web Services. Unlike JAX-RS, it is not a Java EE specification but rather the JAX-RS reference implementation. These are not the same—**without an implementation, Java EE specifications can't do anything, they are merely a definition provided so as to ensure common core functionality across Java EE compliant Application servers**. As mentioned earlier, designing the EE platform by specification has allowed Java EE compliant vendors more freedom. Consequently, if you limit yourself to the JAX-RS API, it should run the same (though it may be implemented differently) on all Java EE Compliant Application Servers. I mention this to say that Jersey extends the core functionality described by the JAX-RS specification. Hence, all JAX-RS is valid Jersey but not all Jersey is valid JAX-RS. The reason for listing Jersey here is that it has a very large market share among Java developers for REST development and it further simplifies the development of RESTful web services. **JSON-P** is Java's API for **JSON (JavaScript Object Notation)** processing. The reason for mentioning JSON-P is that RESTful Web Resources are commonly exposed as JSON.

SOAP stands for Simple Object Access Protocol. SOAP is neither a specification implementation nor is it a part of the Java EE Platform. SOAP is an architectural specification. Like REST, SOAP is a means by which people approach the development of Web Services. Like REST, Java provides support for SOAP via **JAX-WS**. Modern Software Development has largely abandoned SOAP oriented approaches for the development of web services in favour of REST oriented approaches. This is controversial, however, and **it is indeed true that SOAP is still used for new work**. However, the trends show that RESTful approaches are overwhelmingly used for new web services work above SOAP oriented approaches. Nevertheless, **because developers are often working on older systems and SOAP is still used to some degree for new work, I have listed Java's SOAP support here for**

completeness. Web Resources exposed via the SOAP protocol are commonly exposed as XML.

Web Sockets: Java WebSockets API

Java API for WebSockets is a specification that provides APIs to handle WebSocket connections. WebSockets are unique among the Web Components in the Java EE API in that they can they push data to the client without the client having to request it (i.e. an HTTP GET request).

In deeper detail: "Java WebSockets are a departure from the HTTP-based interaction model, providing a way for Java EE applications to update browser and non-browser clients asynchronously. For many kinds of web applications, having the user always in the driver's seat is not desirable.
From financial applications with live market data, to auction applications where people around the world bid on items, to the lowly chat and presence applications, web developers have long sought means by which the server side of the web application can push data out to the client. The WebSocket protocol is a TCP-based protocol that provides a full duplex communication channel over a single connection. In simple terms, this means that it uses the same underlying network protocol as does HTTP and that over a single WebSocket connection both parties can send messages to the other at the same time."—Java EE 7, The Big Picture ~ Danny Coward[14]

Security: Web Component Security, Bean Security

Web Component & Enterprise Java Bean Security mainly involve two notions: the **declarative security model** and the **programmatic model.** To be brief, the declarative security model describes (in metadata) the protection model that you want the web container to apply to the application. This metadata can be defined in the deployment descriptor and can also be specified via certain annotations in source code. The fundamental concepts of the declarative security model involve authentication, authorization, and establishing data privacy based on user roles. The programmatic model involves building against the Java Security API's to define you own security framework.

Security skills are critical to all Software Developers. The recent Equifax [31] and Yahoo! [32] breaches further cement this very serious fact. Globally it is estimated that 4 billion data records were stolen in 2016 [33].

While Enterprise Developers should not be expected to possess the skills of a a security engineer, they should **always write code with security in mind**. A detailed conversation regarding security could fill many books and is beyond the scope of this article. However, you cannot afford to code in ignorance of security. The evidence is daunting. The risks are very real.

Middle Tier: Enterprise Java Beans, Accessing EJBs

Enterprise Java Beans (EJBs) are Java's groundbreaking middleware (between the client and the database) components. EJBs are a Java EE specification concerned with representing the business logic of an Enterprise Application. Enterprise Java Beans run in the Enterprise Bean Container. Once hailed as the golden child of server-side development in the corporate world, The Enterprise Java Bean API provides stock functionality for a number of tasks, simplifying web development. EJBs are trusted, reliable, scalable, proven technology for sever-side development. The EJB is one of Java's most significant technologies.

> ***How Enterprise Beans are accessed:***

Remote Method Invocation (RMI) over the **Internet Inter-Orb Protocol** is the standard, TCP based way for clients to communicate with an EJB in *a different Java Virtual Machine*.

Enterprise Java Beans can be accessed over **HTTP** if said EJBs are exposed as **RESTful** Web Services. Needless to say, **EJBs** can be accessed via local method calls as well.

Last but not least, a special kind of Enterprise Java Bean called a **Message Driven Bean**, can be accessed by making a call to the **Java Message Service**.

Persistence: JDBC (SQL based), JPA/JPQL (ORM)

The **Java Database Connectivity API (JDBC)** is a Java SE API for interacting with a database. It is primarily concerned with relational databases, however an Open Database Connectivity API bridge exists to allow interacting with an ODBC compliant data source. In practice, the most common use of the JDBC is to query or create relational databases via SQL queries processed via the Statement.executeUpdate() method. The JDBC is the *de facto* way to use *SQL* to deal with relational databases in Java programming.

The **Java Persistence API (JPA)** is a Java EE API specification that provides the means to persist Java objects in a relational database. You might ask: "Why do I need the JPA when the JDBC is available?". The Answer: The JPA simplifies the common process of storing application data in a relational database. Instead of the ad-hoc method of creating SQL query statements to map a Java object's properties to a relational database and restructuring said persisted objects back to Java objects, the JPA eases the pain of these common tasks. In addition, the JPA API includes it's own query language called the **Java Persistence Query Language (JPQL)**. The JPA API also includes APIs such as the EntityManager API for dealing with the transition of objects between the Application & Persistence layers. The JPA API provides support for *Object Relational Mapping*.

CDI: Contexts & Dependency Injection (CDI) API

The **Contexts & Dependency Injection (CDI) API** is a Java EE Specification for modularizing applications. The need for the CDI API grew from the need to decouple Java EE Application components in a type-safe manner. The motivation for the CDI API sprung from the desire to integrate bind web and enterprise bean components by way of injection, leaving the responsibility of determining which instance you need to the injection service.

Contexts & Dependency Injection is a pivotal tool in contemporary Enterprise Development across the board. You should understand how to use this well. **Every JEE project you will encounter will use it in some way.**

Structure: Packaging & Deployment,WAR, JAR,EAR, JNDI, etc.

Important to Java EE development is understanding the containers in which Java EE Components operate and how they are grouped together and structured. Several concepts are important in understanding this. Among them: **WAR, JAR, EAR, Deployment Descriptors, RAR, Application Manifest, Modules etc.** A Java EE Application is a single executable containing any of the following:

> ***Brief Descriptions:***

Web Application Archives (WARs)—"A web application [archive] is a collection of static content, markup pages, and images, together with dynamic web components such as Java servlets, JSPs, JavaServer Faces, JAX-RS resources, and Java WebSocket endpoints, with any resources that these web components need such as managed beans or tag libraries and their collective deployment information." [14] A **WAR** is a special type of file format build on top of the JAR archive.

Java ARchive (JARs)— An archive file format for grouping Java class files and application resources for deployment in a single executable unit. JAR files build on the ZIP archive format and include a Java specific Application Manifest**.** The **Application Manifest—**"a metadata file contained within a JAR.It defines extension and package-related data. It contains name-value pairs organized in sections. If a JAR file is intended to be used as an executable file, the manifest file specifies the main class of the application. The manifest file is named MANIFEST.MF. The manifest directory has to be the first entry of the compressed archive."[15]

Enterprise Application aRchive (EARs)—An archive file format for packaging Java EE modules for deployment on an Application Server. In Java EE the **Deployment Descriptors** are XML files used to specify deployment configuration information of Java EE modules by an Application Server or Enterprise Container. ***web.xml*** is the deployment descriptor for Web Applications while ***application.xml*** is the deployment descriptor for Enterprise Applications.

Resource Adapter Archives (RARs)—An archive file format that implements the Java EE Connector Architecture for to foreign

enterprise or non-java systems.[14]

In Java EE, all of the aforementioned archives are known as **modules.**

Awareness of various JEE Components:

Lastly, you should be aware of the overall environment in which you are working in. While you may or may not need to use many of these tools (*Your Mileage May Vary)*, you should at least be aware that they exist and have (at least) a basic, superficial understanding of what they are used for. There are several JSRs and several Java EE APIs and I will not name them all.

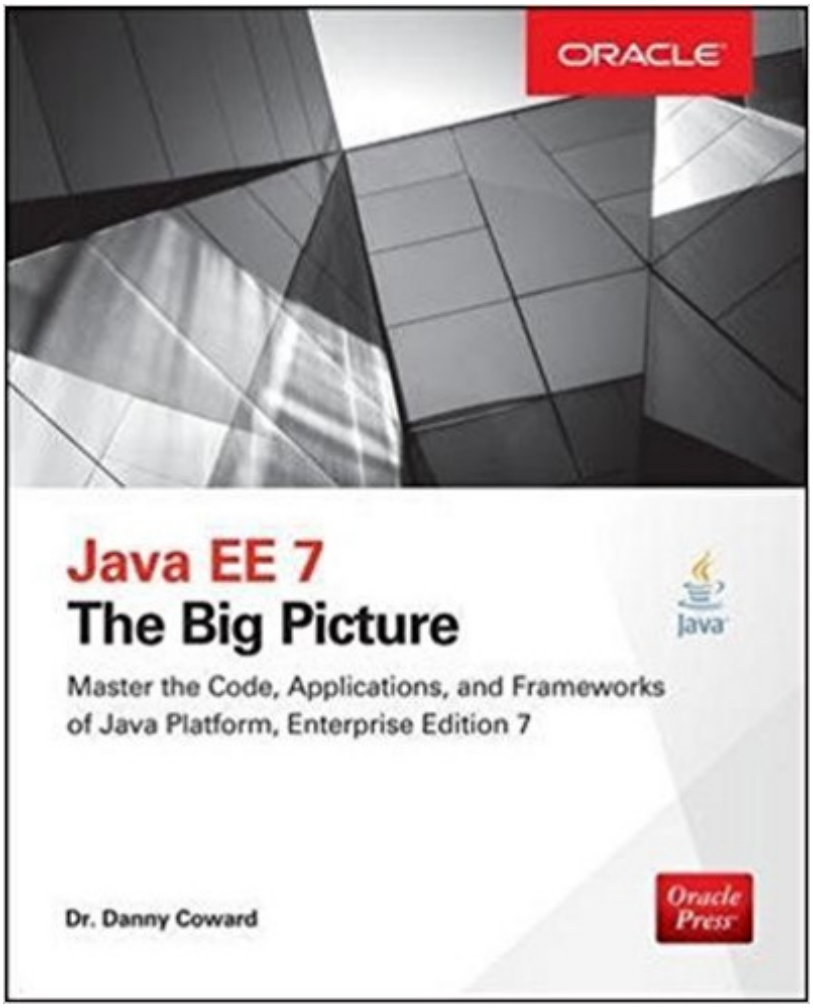
> *Some that you should be aware of:*

How **Concurrency** works in Java EE Applications.
The **Java Message Service** for loosely coupled distributed communication.
The **JAX-WS API** for XML Web Services (SOAP).
The **Java Transaction API(JTA)** for distributed transaction.
The **Java Connector Architecture** for connecting to non-Java EISs.

And so on.

Where to learn, Recommended Resources to learn Java EE:

Java EE 7: The Big Picture
Danny Coward's text is compulsory reading on Java EE. No question about it. I have learned all of the aforementioned Java EE concepts from reading and applying the content of Dr. Coward's book. I strongly recommend it for Java EE beginners and those looking to grasp the core ideas of the Java EE platform specification.



Java Platform, Enterprise Edition: The Java EE Tutorial
This is the Oracle Official Java EE Tutorial.

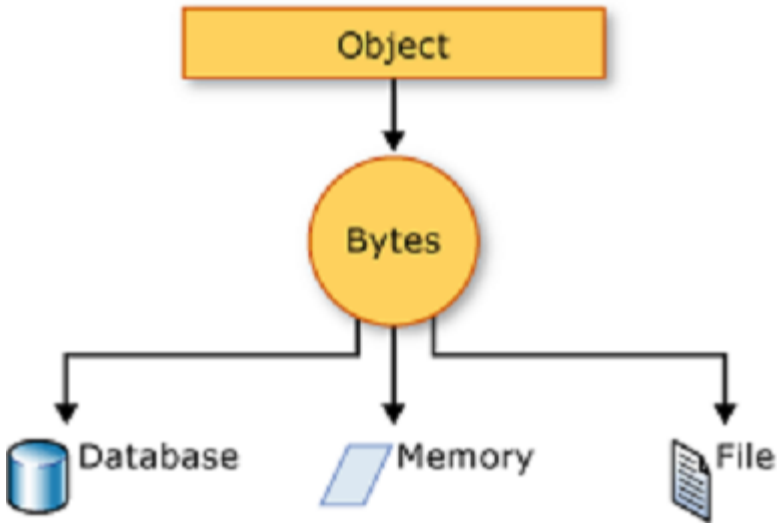
Java Platform, Enterprise Edition: The Java EE Tutorial Release 7
This a more expansive, Oracle Official reference document on the Java EE platform specification.

Serialization Formats— Translating Object State

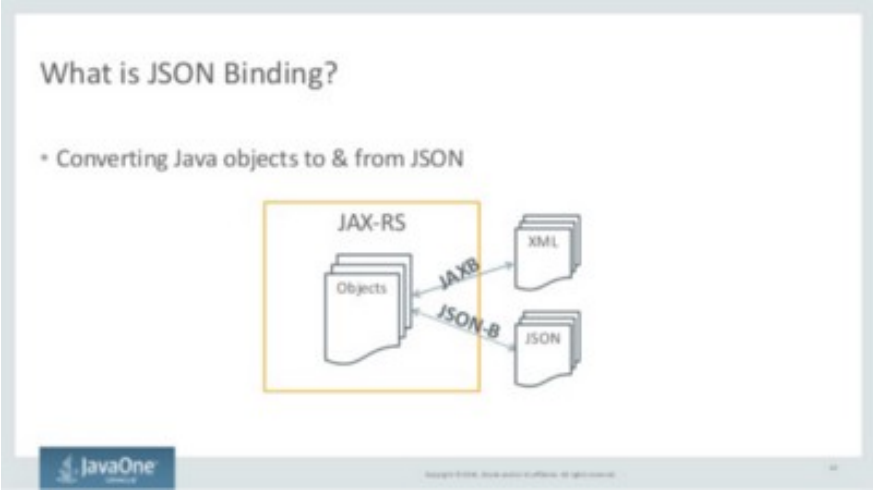
> *Serialization is the process of translation data structures or objects into a format that can be stored or transmitted and then later reconstructed (possibly in a different computing environment).[17] Deserialization is the reverse of this process.*

As an Enterprise Developer you should understand how to perform serialization and deserialization. However, I will leave the topic of how

to perform Serialization to another Author and instead briefly touch on the formats used in the process. Commonly, the formats that application objects will be translated to are: **Binary, XML, & JSON.** The two that we will briefly discuss here are XML & JSON. There are several advantages to serialization, of which three may be considered principal: communication, portability, and storage. In the real world, web service resources are commonly transmitted as JSON or XML. Consequently, and for several other reasons, serialization is a critical concept to understand in enterprise development. **Commonly, RESTful web resources are transmitted as JSON, and SOAP oriented web resources are transmitted as XML.**



Binary Serialization[16]



Java to JSON, XML Slide—JavaOne Talk [18]

Extensible Markup Language (**XML**) "is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable through use of tags that can be created and defined by users." [19]

One of the key features of XML that make it useful for serialization is the ability to define document elements. The ***XML Information Set*** is used with the SOAP protocol.

JavaScript Object Notation (JSON) "is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser/server communication, including as a replacement for XML in some AJAX-style systems.

JSON is a language-independent data format. It was derived from JavaScript, but as of 2017 many programming languages include code to generate and parseJSON-format data."[20]

Where to learn, Recommended Resources:

To learn about XML:
https://www.tutorialspoint.com/xml/

To learn about JSON:
http://www.tutorialspoint.com/json/

Java EE Persistence Implementations—Tucking Objects Away For Later Use

> *"Persistence refers to object and process characteristics that continue to exist even after the process that created it ceases or the machine it is running on is powered off." [22]*

As an Enterprise Developer, strong understanding of Persistence is key. While the JPA provides a very solid core set of features for persistence, contemporary application development has demanded features that exceed the JPA. For this reason, there are several JPA implementations that add to the JPA. Among JPA implementations there are many

options: EclipseLink(Reference Implementation), Hibernate, Toplink, Apache OpenJPA, DataNucleus, & ObjectDB.

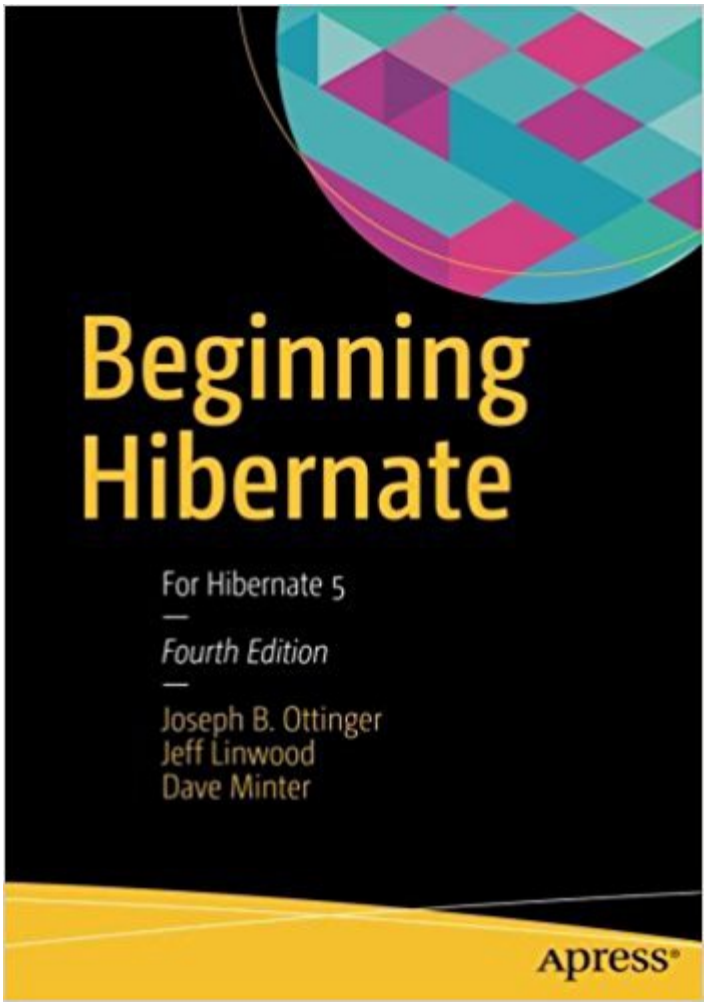
Returning to the topic of persistence, we will touch on the most popular API for persisting application objects—Red Hat's **Hibernate ORM**.

Hibernate ORM—an Object Relational Mapping tool (ORM) for persisting Java Application Objects. So popular is Hibernate that a few polls on the net suggest that it has 70% +/- market share among Java developers as an ORM solution. In 2014, Rebel Labs estimated this market share to be 67.5% [23].

Where to learn, Recommended Resources:

Hibernate 5.2 Official Tutorial & Documentation

Beginning Hibernate: For Hibernate 5, 4th ed. Edition



Spring & Hibernate for Beginners, Chad Darby—Updated 08/2017
The most popular course on uDemy on the topic of Spring/Hibernate, Chad's course is relied upon by thousands. I spent $10 on this course, a small investment that should pay back many times more.

Protocols—How we talk to each other on the net

> *A protocol is a set of rules and guidelines for communicating data. [24]*

HyperText Transfer Protocol (HTTP) is the foundational communication protocol for the World Wide Web (WWW). It is a request-response model between a client and a sever implemented atop a transport protocol, such as TCP/IP.

Basic knowledge of how HTTP works may be considered bar none in importance to web development. The web speaks HTTP. In order to understand the web, you must understand HTTP.

In Java, we deal with HTTP primarily with the **java.net.*** and **javax.servlet.*** packages.

HyperText Transfer Protocol Secure (HTTPS) is an HTTP communications protocol encrypted with ***Transport Layer Security*** (TLS) or ***Secure Socket Layer*** (SSL).

In Java, we deal with HTTPS primarily with the package **java.net.ssl.*** and **Java EE platform**.

Transmission Control Protocol/Information Protocol (TCP/IP) is the foundation of the Internet, commonly referred to as the **Internet Protocol Suite.** While it was named after the two primary protocols, it is more than just TCP and IP.

"TCP/IP is a [four-layered] suite of protocols designed to establish a network of networks to provide a host with access to the Internet. TCP/IP is responsible for full-fledged data connectivity and transmitting the data end-to-end.."[25]

In Java, we interact with TCP/IP in none other than the **java.net.*** packages.

You can learn more, here. TCP is the foundation of the **WebSocket protocol**. The truth is, however, TCP/IP is the engine by which we most often conduct HTTP communication—TCP/IP is everywhere, and is by far the most dominant protocol of internet communication. It is important to understand, that HTTP functions on a layer underneath TCP/IP.

TCP/IP are in the transport and internet layers, while HTTP is much higher, at the Application layer. The **four layers,** in hierarchical order are: the link layer, the internet layer, the transport layer, and the application layer. Consequently, **we most commonly use HTTP atop TCP/IP.**

User Datagram Protocol (UDP) is a unreliable, stateless transport layer protocol that communicates by way of messages called datagrams. Datagrams are not guaranteed to be delivered and there are no error checking mechanisms in place. It finds common application in domains such as video conferencing.

In Java, we interact with UDP via the **java.net.*** packages, particularly **java.net.DatagramSocket**.

Application Server—It's all talk until it's deployed

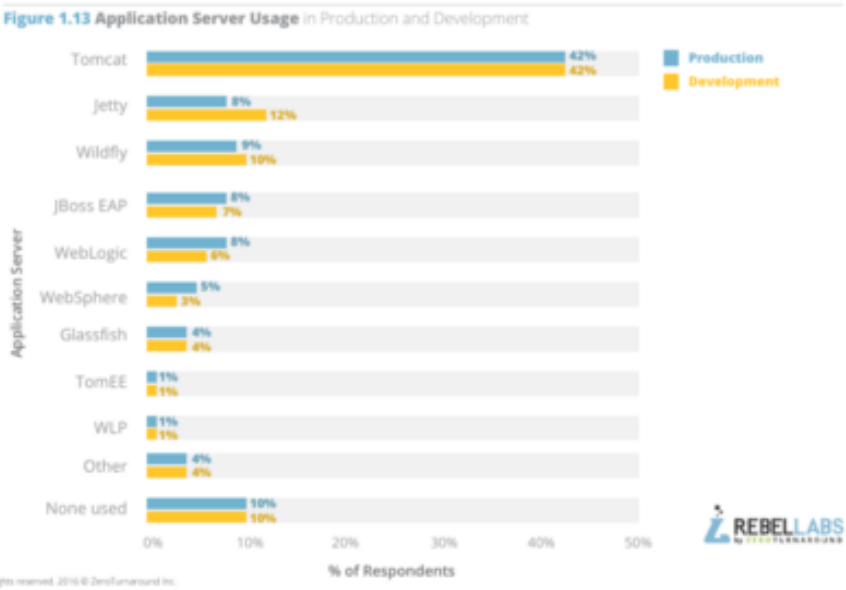
> *An Application Server provides the the tools to create Web Applications as well as a sever to run them [26].*

Remember how we said that Java EE was merely a specification? Java EE Application Servers provide the implementation for Java EE specifications as well as a whole host of resources needed to build and deploy web applications. The reference implementation of the Java EE Specification is Oracle's **GlassFish Application Server.** Additionally, it is important to understand that not all Java EE Application Servers implement the full breadth of the platform. Apache Tomcat (arguably the most popular one), for example, is mostly a Servlet Container implementation. Also, key in understanding Java EE is understanding that each implementation of the Java EE Specification may add to or build on the API specification, and this is commonly the case. One such example is found in Jersey, the JAX-RS reference implementation that we touched on earlier. While Jersey provides a reference

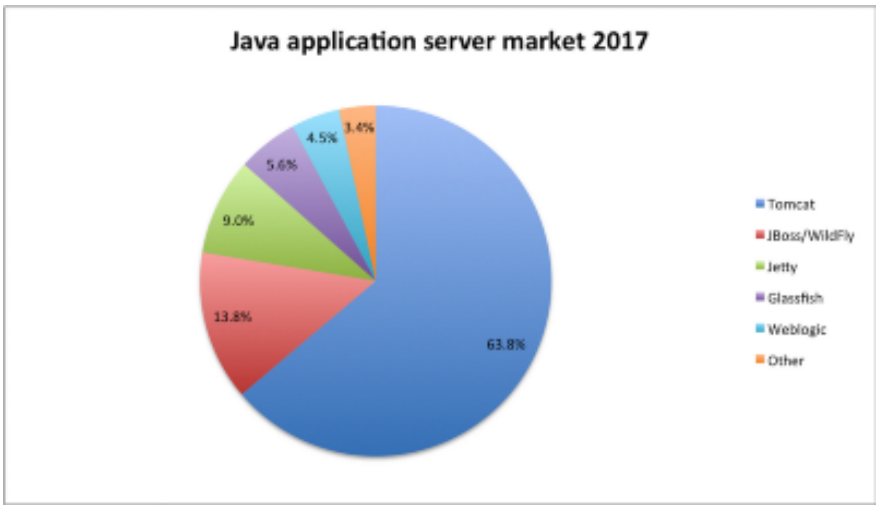
implementation for the JAX-RS API specification, it additionally provides APIs for other things, such as GUICE integration.

The take away from this is that the situation as it pertains to portability is a little more subtle as compared to Java SE development. **Portable code is accomplished by sticking to the Java EE API libraries.** The various implementations of the Java EE API specification almost always include more than just the Java EE APIs. **This is true even for various Reference Implementations. Consequently, in practice this leads to code that is often not portable across Java EE Application Servers.** Unless you examine the contents of a Enterprise or Web archive, you cannot be sure that it will run on your application server. This is further complicated by the fact that there are partial as well as full Java EE Application Servers.

According to Rebel Labs [27], **Apache Tomcat** is a leading Application Server choice with 42% market share followed by a distant **Eclipse Jetty** and **RedHat's Wildfily** (formerly popularly known as JBoss).



According to ***Plumbr*** [28], an application performance monitoring company, **Apache Tomcat** has a 63.8% market share, followed by 13.8% for **Redhat's WildFly** and 9.0% for **Eclipse's Jetty.**



The following data is again from ***Plumbr***, showing trends over time.

	2013	2014	2015	2016	2017
Tomcat	45.2%	40.5%	58.7%	58.2%	63.8%
JBoss/WildFly	17.4%	18.0%	15.7%	20.2%	13.8%
Weblogic	3.3%	5.5%	9.9%	2.9%	4.5%
Jetty	24.7%	31.0%	8.8%	10.7%	9.0%
GlassFish	7.4%	4.0%	5.1%	5.6%	5.6%
Other	2.0%	1.0%	1.8%	2.4%	3.4%

There is only one major conclusion that can be drawn from the trends: Tomcat remains the clear #1 preference, extending its footprint slowly but steadily year over year.

Other trends or changes should be interpreted carefully. For example the reason why Jetty dropped to just third of its former glory on 2015 is likely caused by Plumbr transformation from a development tool to a monitoring solution. Instead of the developer-friendly Jetty the production deployments with other Java application servers took its share of the deployments.

It is a safe bet to say that a contemporary Java Enterprise Developer should understand the way **Apache Tomcat** works. Minimally, a Java Enterprise Developer should be familiar with the environment of at least one Application Server.

Where to learn, Recommended Resources:

Apache Tomcat 9 Official Documentation

Redhat's WildFly 10 Official Documentation

Enterprise Framework—The Engine of Contemporary Development

> *"While Java EE does a great job of standardizing the enterprise infrastructure, providing an application model, and providing components adequate to develop web applications, two major problems are associated with it."*
>
> *- Interacting directly with the Java EE Components often results in massive boilerplate code and even code redundancy.*
>
> *- Creating an Enterprise Application using the Java EE infrastructure is a nontrivial task that requires a great deal of expertise.*
>
> *Frameworks address these two major problems (among others)." [40]*

The Spring Framework, much like Hibernate, is unique in this area not only due to it's commanding mindshare among Java developers as an Enterprise Solution, but also because much like Java EE it is a suite of various different tools for building Enterprise Solutions. It is not merely an MVC Framework. Spring MVC is only one aspect of the behemoth that is the Spring Framework.

An Enterprise Framework is a collection of APIs and tools used to develop Enterprise Software. While the term "Enterprise" is a finicky one in that it is not very well defined, when contrasted to Web Development it usually is used to indicate that it is software beyond merely a web framework. Enterprise Frameworks include solutions for networking, persistence, web services, and much, much more.

Spring Framework is unique in that it is one of the only frameworks that are as broad and comprehensive in the amount of APIs and tools available. There are other Frameworks, however, such as the ZK Enterprise Framework, JBoss Seam (now deprecated), and a combination of various Apache Enterprise tools, but in my estimation, **Spring & JEE are the most cumulative Enterprise Solutions**.

OK, I lied. The Spring Framework is actually an application framework and inversion of control container for the Java platform. It can be used in several different applications. It merely contains components for building Enterprise Applications atop the JEE (mostly the Servlet API).

Those components “happen” to be extremely popular for building Enterprise Applications.

It is beyond the scope of this article to discuss the Spring Framework in it’s totality. You may never even use the Spring Framework, but I’m willing to bet that you will certainly encounter it throughout the course of a Java Enterprise Career. Because it’s influence and widespread use in Java Enterprise Development, it must be discussed. The specific components that you may need to create Java Enterprise Applications will of course depend on need. **Much like the JEE, there are components for several different things**. Often enough, you will find JEE development done in tandem with the Spring Framework. The Current Stable version of the Spring Framework is Spring 4.3.10 released in mid July 2017. Spring 5 is expected to be released any day now, in September 2017; It is Java 9 compliant. You can see what’s new in Spring 5, here.

You will want to take learning Spring Framework very seriously, as this is the predominant Enterprise Framework of choice for many Java developers.

There are various other pieces of Java Enterprise Software. For more, see here.

Where to learn, Recommended Resources:

Tutorial’s Point Spring Framework 4.1.6 Tutorial

Pivotal’s Official Spring Documentation, Tutorials, & Guides
You will visit this web-page. Trust me. Feel free to bookmark it ahead of time.

Free Spring Framework Tutorials—Spring Framework Guru (John Thompson)

Spring & Hibernate for Beginners, Chad Darby—Updated 08/2017
The most popular course on uDemy on the topic of Spring/Hibernate, Chad’s course is relied upon by thousands. This is how I started.

Learning Spring 5.0

Mastering Spring 5.0

Pro Spring 5.0 : An In-Depth Guide to the Spring Framework and Its Tools, 5th Ed.

Building Web Apps with Spring 5 and Angular

What's New in Spring Framework 5, John Thompson (DZone/Java Zone)

Web Frameworks—Building Web Applications

Because we have already discussed the Java EE Web Tier (Servlets, JSP/JSTL/EL, JSF) as well as Enterprise Frameworks (Spring, Apache), this portion will be relatively brief.

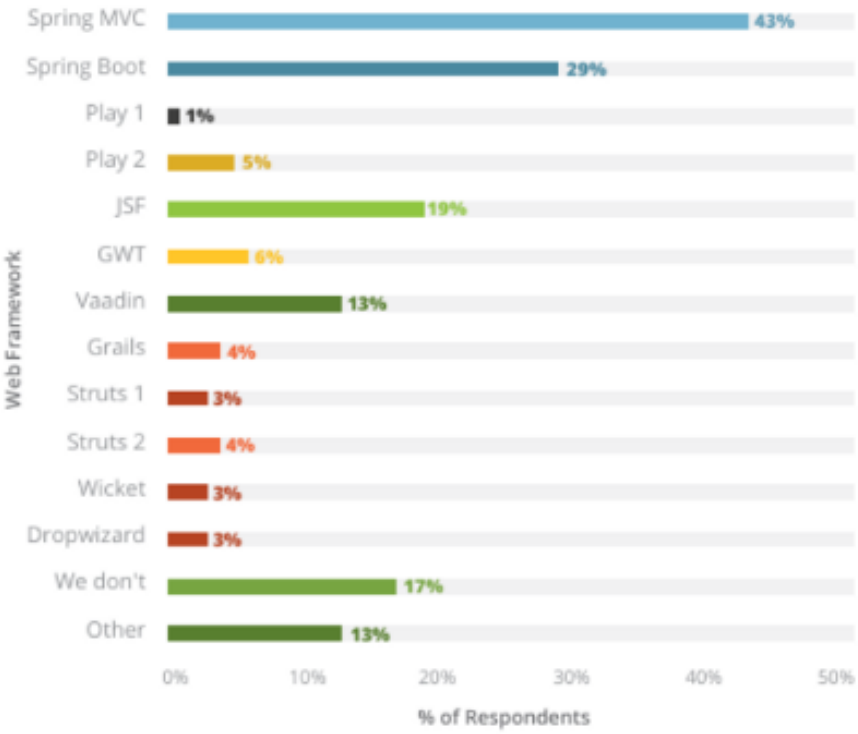
A **Web Framework** is an Application Framework that is designed to ease in the development of Web Applications. They are a common staple of Enterprise Development.

According to Rebel Lab's latest data (Sep 19 2017), Spring-Technology based Web Framework (Spring MVC, Spring Boot, Grails) has 47.81% of the popularity index among developers with JSF a distant second at 15.19% [41].

Rank	Framework	Popularity
1	Spring mvc	29.39
2	JSF	15.19
3	Spring Boot	11.69
4	GWT	7.6
5	Grails	6.73
6	Struts	7.47
7	Play framework	4.16
8	Seam	1.88
9	jax-rs	3.1
10	Vaadin	2.45
11	Wicket	1.92
12	Tapestry	1.83
13	JHipster	0.73
14	Dropwizard	1.05
15	Sparkjava	0.76
16	Lagom	0.71
17	Vert.x	0.72
18	Ratpack	0.15
19	Rapidoid	0

According to Rebel Lab's 2016 Java Tools & Technologies Landscape Report [27], 76% of developers used at least one Spring-Technology based Web Framework (Spring MVC, Spring Boot, Grails), with stock JSF again holding a distant second with 19%. It is important to note

that participants of this survey could choose more than one web framework. This makes sense as the use of multiple frameworks is common.



John Thompson claims that some estimates report that Spring is used in over 60% of Java Web Based Applications [5]. In my opinion, the unalienable truth is that Spring technology has made an indelible mark on contemporary Java Enterprise Development.

> *What about the Jobs?*

At the time of posting this article,

An Indeed US search for “Java Spring” yields 11,133 results
An Indeed US search for “Java Play” yields 4,067 results
An Indeed US search for “Java Struts” yields 1,847 results
An Indeed US search for “JSF” yields 1,035 results

This should not be taken as proof, but on the surface, it does appear that Spring is in much more demand than any other web framework.

> *Which Framework is the best?*

Are apples better than oranges? I will leave it to the user to decided which Framework **best meets their needs.** Many feel that the Spring MVC Web Framework is the golden child of Java Enterprise Development and following that guise, is the *de facto* best web framework available. The mere asking of this question presumes a defined notion of what qualifies something for being "the best". I am generally dismissive of these kinds of conversations as I generally do not believe there is a best language, tool, or framework for all tasks in a specific domain. I generally believe that some tools do some things better than others and that the quality of a tool is relative to need as well as both the community and engineering team behind the development of the tool. Try a framework or three. Let me know which you like best, and why, in the comments.

Where to learn, Recommended Resources:

For Spring, refer to the links in the Enterprise Frameworks segment above.

Play 2 Official Documentation & Tutorial

Tutorials Point Struts 2 Tutorial

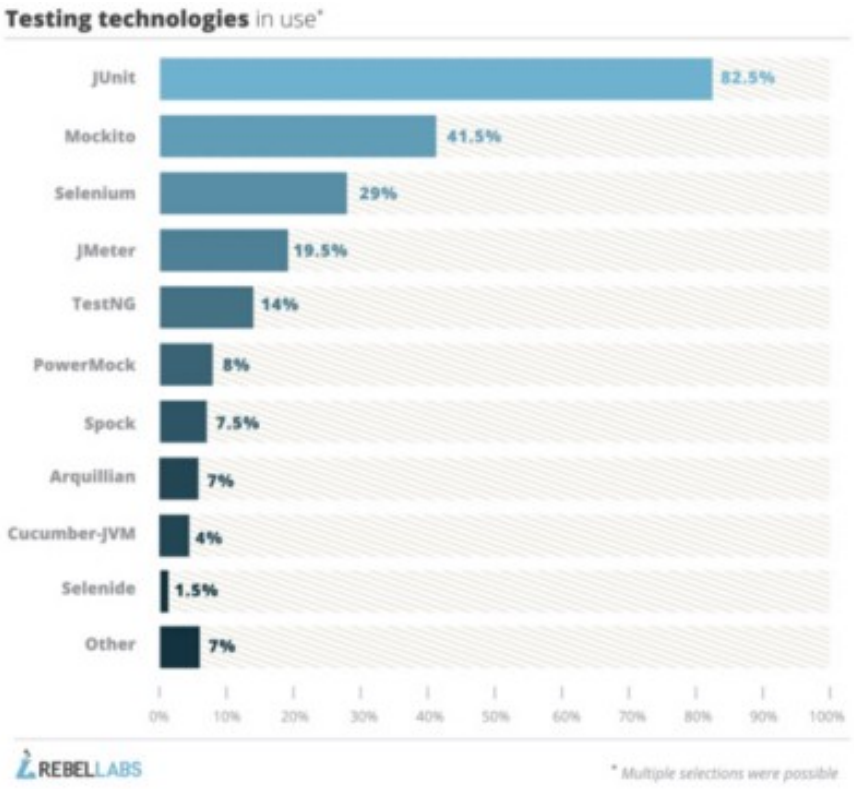
Unit Testing— Does it work the way we say it should?

While there is more to testing than unit testing, I will leave the task of discussing Testing to another author. Techopedia puts it best:

"A unit test is a quality measurement and evaluation procedure applied in most enterprise software development activities. Generally, a unit test evaluates how software code complies with the overall objective of the software/application/program and how its fitness affects other smaller units."[29]

Unit Testing is important because it provides us away to examine if individual application components are meeting end-user requirements. Unit Testing is useful in building components for large systems where other components rely on the confidence that the components it depends on work the way they are supposed to. Some community discussion on the topic is available here.

Of the Unit Testing Frameworks, JUnit holds a commanding market share, with 82.5% of Java developers making use of the technology, according to Rebel Labs [23]. While JUnit may not be all for your Applications needs, it is a very valuable tool and as such it is a very familiar among Java Enterprise developers.



Where to learn, Recommended Resources:

JUnit Tutorial for Beginners—Learn Java Unit Testing (Ranga Karanam, in28Minutes on uDemy)—This short video tutorial will equip you with the basics to start using unit testing in your Java Applications.

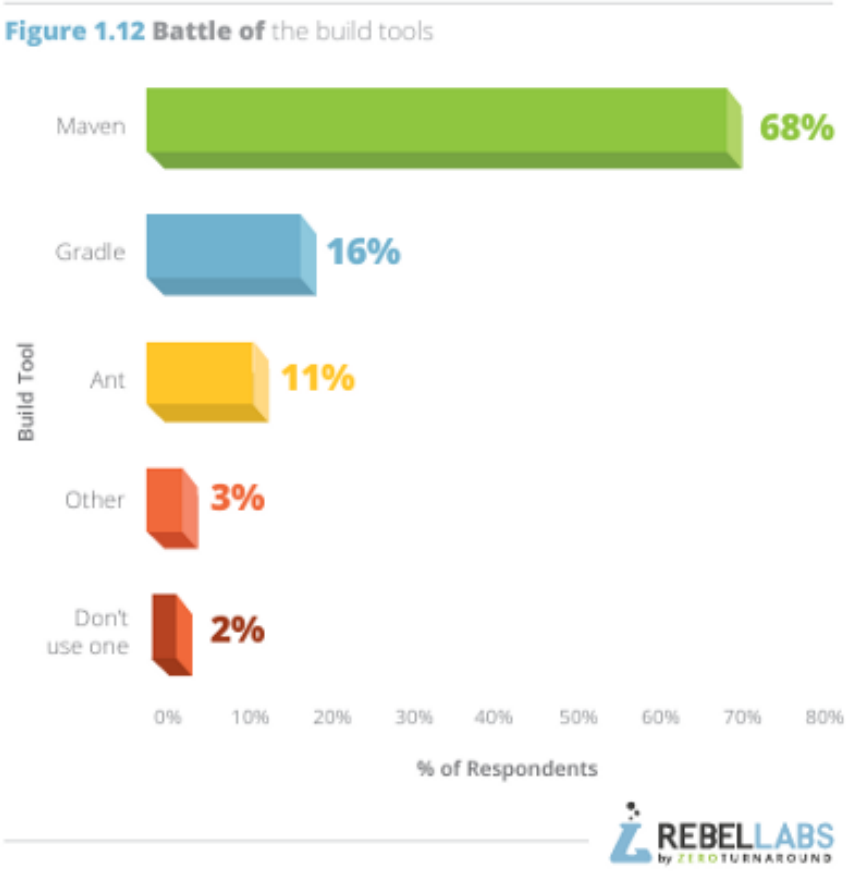
JUnit 5 User Guide—Official Documentation & Tutorial

It is also important to understand how to perform **assertions** using the **assert** keyword in the Java programming language.

Build Tools—Automating The Build Process

Build tools are automation tools used to manage the dependencies of software, describe how software is to be built, and automate the process of compiling,linking, & producing an executable. They are a standard staple of enterprise projects, which will almost always contain numerous dependencies that will need to be managed over the evolution of the software.

According to Rebel Labs [27], Maven is still holding a pseudo-monopoloy in Java mind-share with ~7/10 developers opting for the tool. Gradle notched a distant second, with ~ 1/6th of mind-share.



Where to learn, Recommended Resources:

Tutorial's Point on Apache Maven

Apache Maven Official Documentation

Spring: Building Java Projects with Gradle

Grade Build Tool Official Documentation

Tutorial's Point Gradle Tutorial

Version Control— Collaborative Development

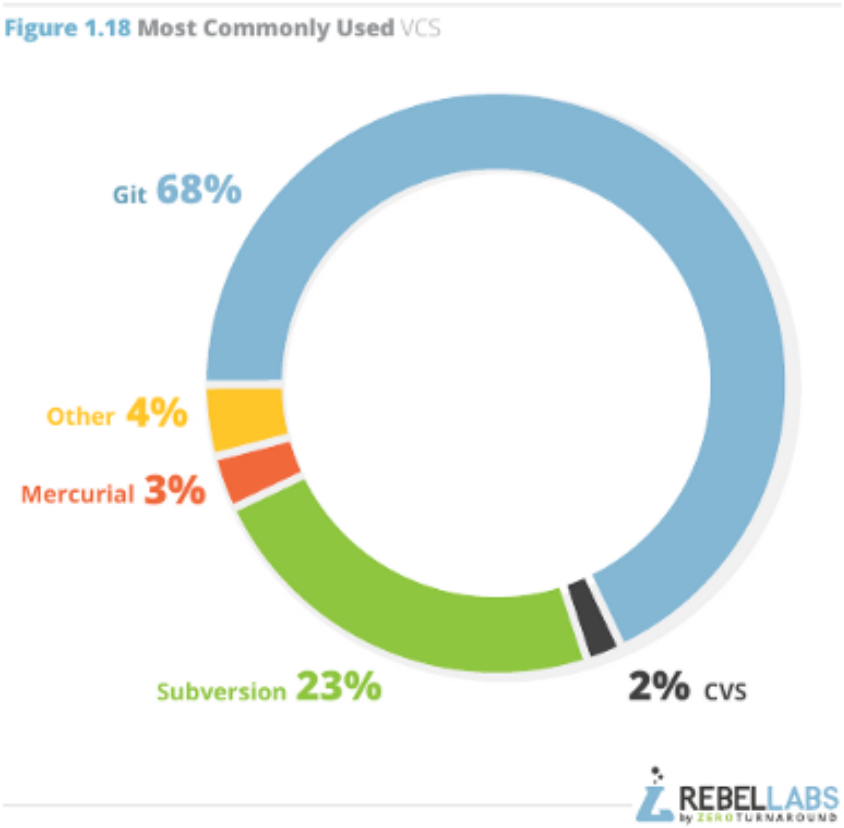
> *"Alone we can do so little; together we can do so much" — Helen Keller*

Version Control is important because it is the means by which managing versions of software in the development process takes place. This is relevant to real-world Enterprise Development where **several people work on various components of a single application**, often in teams. Whether you like it or not, real-world development will require you to work in teams, or at least, to collaborate with other people.

There are many benefits of version control:

- A Complete History of the Application and it's changes
- Ability to collaborate
- Ability to merge changes
- Ability to maintain different versions of the same application
- Branching
- Much more….

Git seems to have a commanding lead with 68% market share, while Subversion seems to hold a distant yet sizable second place spot with 23% [27].



According to the Stack OverFlow Developer Survey 2017 [35], Git had a 69.2% & Subversion had a 9% mind share among developers for choice in version control.

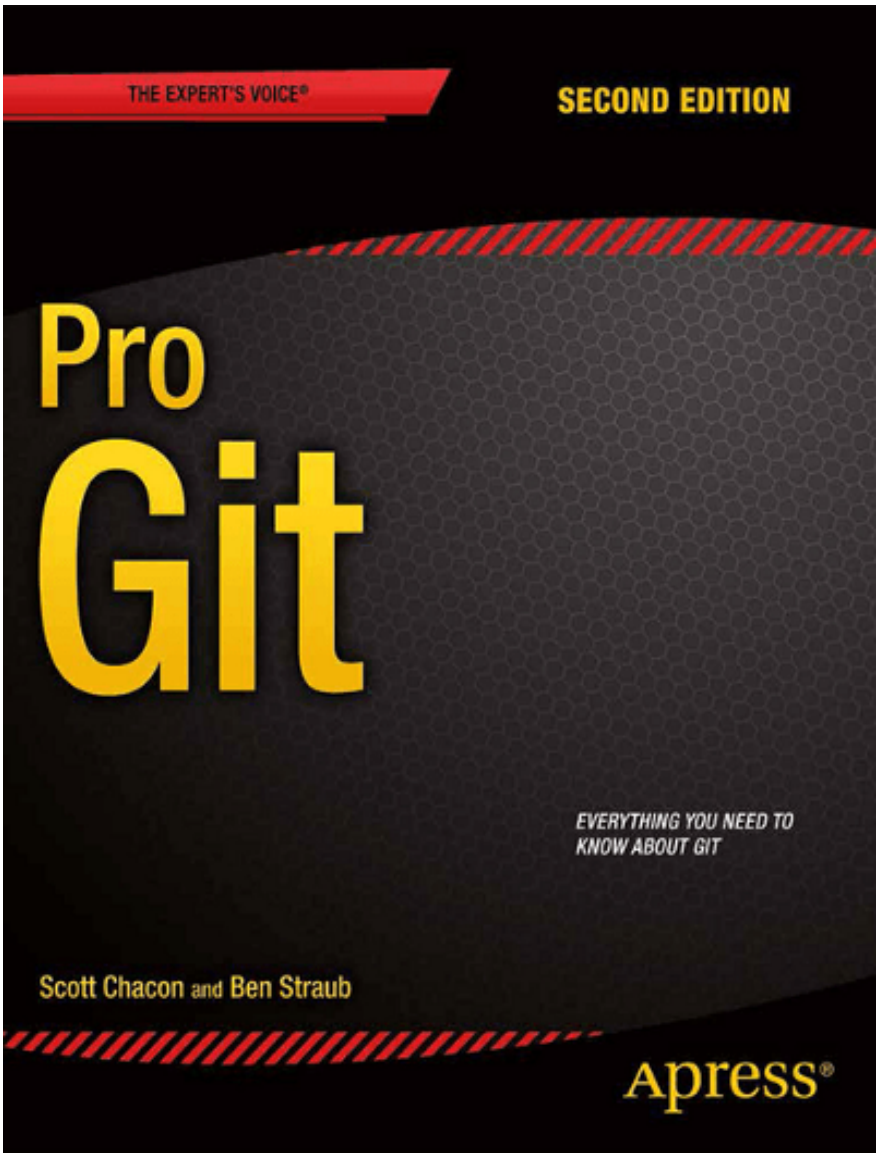
A Final Note: You should also be aware of version control repositories such as **GitHub** or **BitBucket**. These two repositories are among the most common used in development, with the former having similar levels of market share as it's companion, **Git**.

Where to learn, Recommended Resources:

TutorialsPoint—Git Tutorial

Git Official Documentation

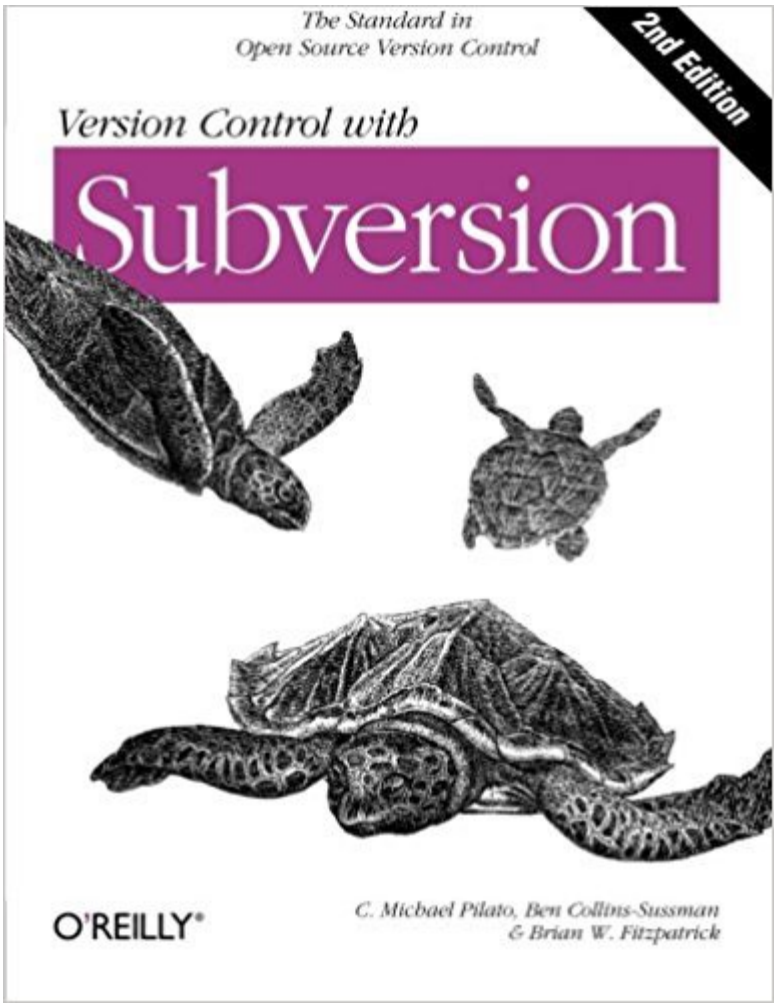
Pro Git (Free Book)



GitHub's Official List of Tutorials

TutorialsPoint—Subversion Tutorial

Version Control with Subversion (Free Book)



Debugging— If the bug's big enough, it's a feature

> *" The wages of sin is debugging."—Ron Jeffries*

> *"If debugging is the process of removing software bugs, then programming must be the process of putting them in."—Edsger Dijkstra*
>
> *"The most effective debugging tool is still careful thought, coupled with judiciously placed print statements."—Brian Kernighan*

Debugging is the process of removing bugs, otherwise known as errors in software, from an application. There are many tools and techniques involved in debugging, enough to fill many books. The objective of this

portion is to point you in the right direction regarding debugging in Java.

For more, see here.

Enterprise Developers should be aware of how to use the **JDB** / JPDA, best practices, as well as how to use **breakpoints** and **IDE tools** to perform debugging.

Where to learn, Recommended Resources:

Java Platform Debugger Architecture (JPDA) SE 8

Tutorials Point—Java Debugger Tool

Java Platform Debugger

IntelliJ IDEA Java Debugging Tutorial

Code Quality— Practice Makes Perfect

> *" Quality is not an act, it is a habit."—Aristotle*

Poorly written code can be devastating to an organization in terms of the cost (both in time and money) of managing the consequences of said code. Obscurely written code can lead to developer inefficiency which leads to more time spent trying to understand and/or restructure the code which costs the organization money and which ultimately hampers progress.

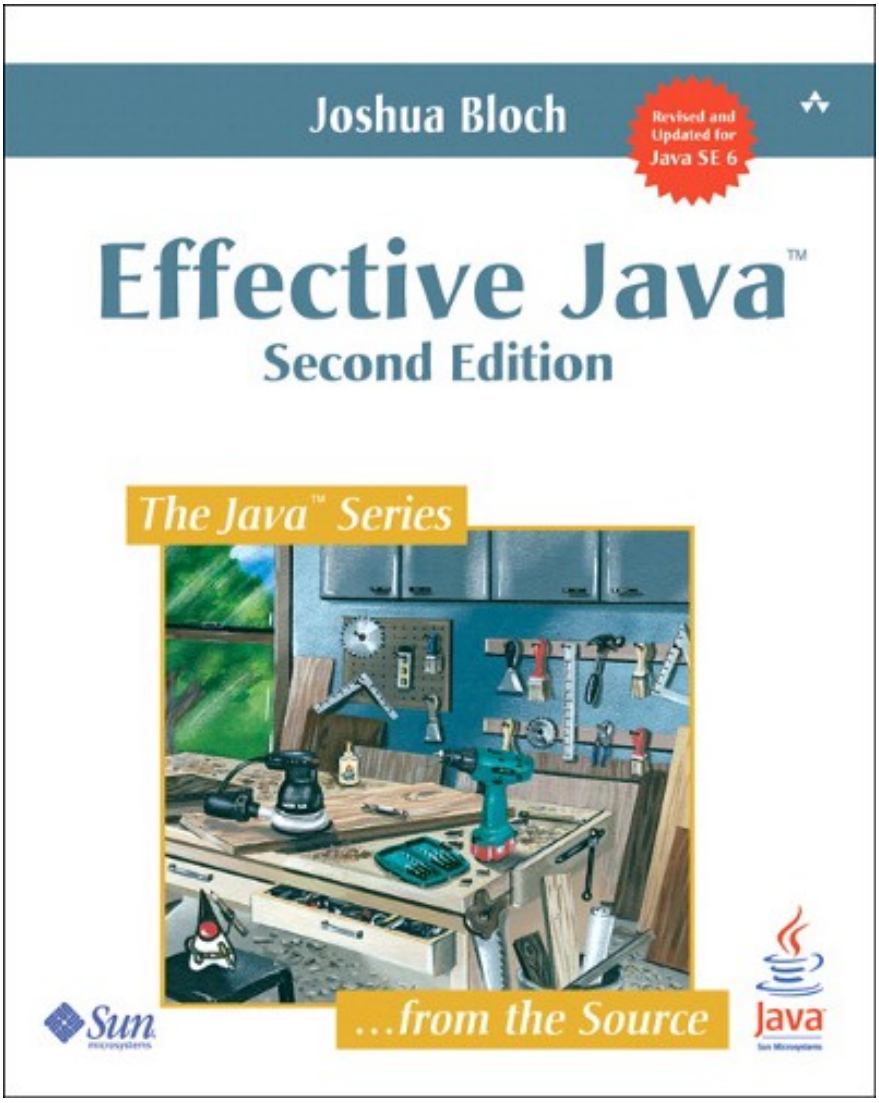
Badly written code can lead to performance penalties which may (depending on the nature of the organization's aims) lead to less adoption of their product/service which could lead to the organization loosing business or ceasing to operate.

Badly written code can lead to security vulnerabilities which may lead to the breach of sensitive information which will assuredly lead to demise and utter destruction.

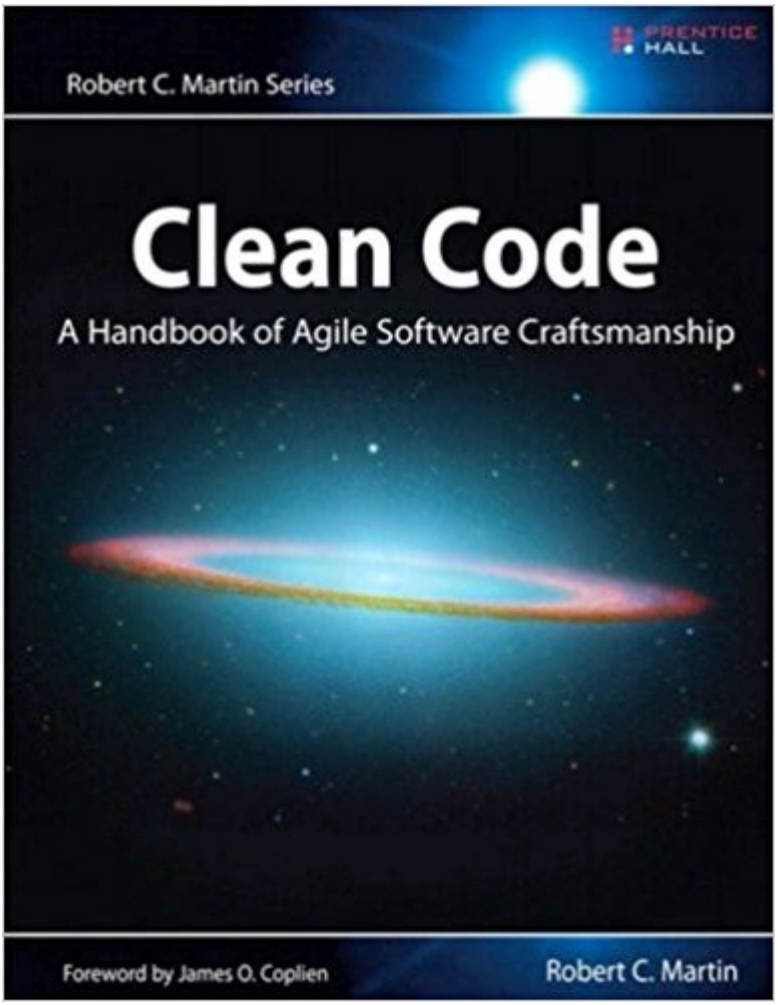
Get the point? It's not enough to know the elements of a programming language—one needs to know how to use the tools the language provides in a way that is safe, effective, & efficient.

With that in the mind, I recommend the following books for writing quality code:

Effective Java, 3rd Ed.—Joshua Bloch (**Release:** December 29, 2017)



Clean Code



Another good book that is often recommended is Code Complete by **Steve McConnell**. The reader be cautioned, however, that it is a lengthy read. At 960 pages it is much longer than Clean Code, about an additional ~500 pages by comparison.

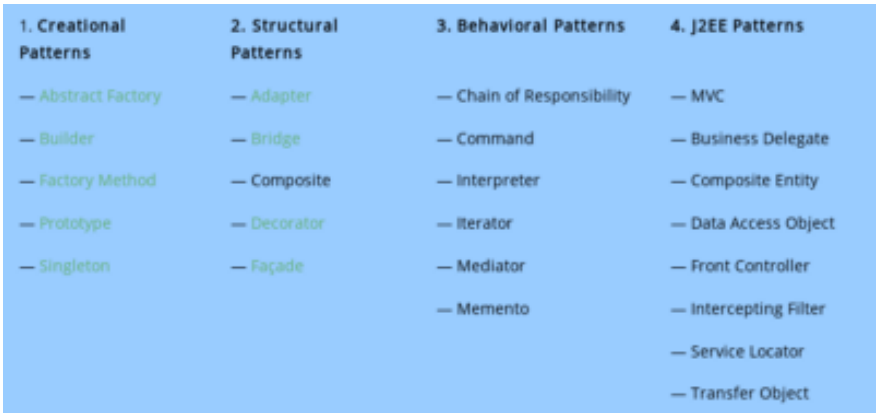
Design Patterns and Best Practices in Java 9,Adrian Ianculescu (**Dec 6, 2018**)

Design Patterns—Templates for Solving Common Problems

Design Patterns are useful because they save time and (arguably) effort. They are the result of observing a general, repeatable means by which to solve commonly occurring problems. **Enterprise Design Patterns** save time by providing a template for common Enterprise Tasks.

In 1994, Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm, cordially referred to as the ***Gang Of Four***, wrote the groundbreaking book **Design Patterns: Elements of Reusable Object-Oriented Software**. Though the text was written with C++ in mind, the concepts may be applied to many Object Oriented languages.

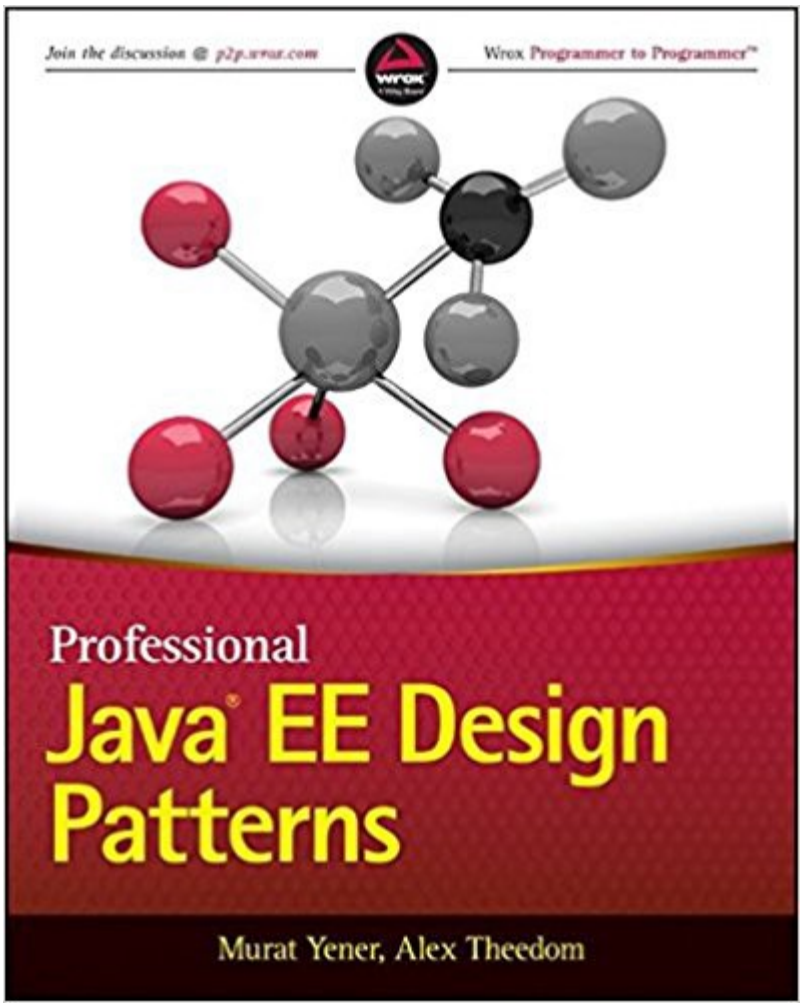
It is valuable to be familiar with **The 23 Design Patterns** of the Gang Of Four Design Patterns (Elements of Reusable Object-Oriented Software)[30].



Though many of the concepts of the *GOF Design Patterns* are timeless, significant technical evolution has occurred since then which merit (also) reading a more modern book. In particular, I recommend reading: **one of the first four resources** listed below, **Clean Code,** and if developing with the Spring Framework, **Spring 5 Design Patterns**. A great person to follow is Martin Fowler and his work. The sequel to the original Design Patterns book can be found here.

Where to learn, Recommended Resources:

Professional Java EE Design Patterns



uDemy Course: Java EE Made Easy—Patterns, Architecture and Frameworks

Martin Fowler's Patterns in Enterprise Development

Modern Java EE Design Patterns: Building Scalable Architecture for Sustainable Enterprise Development, RedHat(Markus Eisele)

Patterns of Enterprise Application Architecture

Clean Code: Design Patterns Videos by Robert Martin

Spring 5 Design Patterns, Dinesh Rajput (**December 11, 2017)**

The Last Step— BUILD THINGS. Build in tandem with learning.

> *“Knowledge without practice is useless. Practice without knowledge is dangerous.”—Confucius*

> *“Tell me and I forget . Teach me and I remember. Involve me and I learn.” —Benjamin Franklin*

This portion is purposefully brief. I believe the aforementioned quotes by Confucius & Benjamin Franklin could not be more fitting.

When one attempts to build something, say an Android App, without fully understanding the Java programming language or the Android APIs, it is, again, akin to playing basketball while blindfolded. Sure, you may make a basket here and there, but you may be ultimately clueless as to what it is you are truly doing. Worse still, you may make some dangerous code.

Conversely, if one has knowledge but does not apply their knowledge to practice, it is almost useless. Anyone who has built even the most primitive systems can attest to the fact that programming is best learned in practice. Learning without practice can in many cases be akin to learning French without engaging in verbal or written dialogue.

Hopefully I have successfully described what I believe to be fallacies on camps that ignore the importance of either theory or practice. The truth is, a unique blend of both is essential. **Reading & building go hand in hand**. There is a problem with reading without enough application and building without sufficient understanding.

To market yourself to organizations with which you wish to work with, it is useful to build applications based on what you have learned and post them to a public repository like GitHub to demonstrate your competence and passion. A blog or vlog is useful as well ;).

Beyond The Core Skills—Things to explore when need necessitates it

> *“Once you stop learning, you start dying.”—Hans Albert Einstein*

At this point I believe that we have touched on the core skills of Enterprise & Web Developers. **The truth is, a career as a Software**

Developer is a never-ending journey in learning. These are merely the core skills. This is very likely to be an incomplete list of all the skills that you may need for your specific job. This is also likely to be an incomplete or outdated list of skills as evolution in technology barrels forward. While these are the core skills, it is useful to pay attention to **technology trends.** In addition to trends, there are many valuable tools and concepts that are useful for various application domains. Among them:

Continuous Integration (i.e. Jenkins)
Virtualization (i.e. Docker)
Agile Practices
Microservices Architectures (This is one is especially popular.)
Visual Profiling (i.e. Visual VM)
Advanced User Experience & User Interface Tools
Advanced Performance & Security Techniques
NoSQL

And so on. These are things to check out when the need for them arises.

[34]—KEEP CALM and CODE JAVA

Works Cited

[1]—http://www.oracle.com/technetwork/articles/javame/bluray-142687.html

[2]—https://www.tiobe.com/tiobe-index/java/

[3]—http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2017/

[4]—https://zeroturnaround.com/rebellabs/reports/

[5]—https://springframework.guru/how-do-i-become-a-java-web-developer/

[6]—http://www.asanet.org/journals/ASR/Dec12ASRFeature.pdf

[7]—https://en.wikipedia.org/wiki/Java_%28programming_language%29#cite_note-21

[8]—https://en.wikipedia.org/wiki/SQL

[9]—Grady Booch (1996) *Object Solutions: Managing the Object-Oriented Project*. p. 277

[10]—http://www.decimustechub.com/expertise/java-expertise/

[11]—https://en.wikipedia.org/wiki/SQL:2016

[12]—https://howtodoinjava.com/spring5/spring5-features-and-enhancements/

[13]—Flanagan, David. *JavaScript—The definitive guide* (6 ed.). p. 1.

[14]—Java EE 7, The Big Picture ~ Danny Coward

[15]—https://en.wikipedia.org/wiki/JAR_(file_format)#Manifest

[16]—https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/

[17]—https://en.wikipedia.org/wiki/Serialization

[18]—https://www.slideshare.net/mgrebac1/java-api-for-json-binding

[19]—https://en.wikipedia.org/wiki/XML

[20]—https://en.wikipedia.org/wiki/JSON

[21]—https://www.w3schools.com/js/js_versions.asp

[22]—https://www.techopedia.com/definition/8842/persistence-computing

[23]—https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-for-2014/10/

[24]—https://www.techopedia.com/definition/4528/protocol

[25]—https://www.techopedia.com/definition/2460/transmission-control-protocolinternet-protocol-tcpip

[26]—http://www.theserverside.com/news/1363671/What-is-an-App-Server

[27]—Java Tools And Technologies Landscape 2016 ~ Rebel Labs by Zero Turn Around

[28]—https://plumbr.eu/blog/java/most-popular-java-application-servers-2017-edition

[29]—https://www.techopedia.com/definition/9847/unit-test

[30]—https://www.dineshonjava.com/design-patterns_25/

[31]—https://www.consumer.ftc.gov/blog/2017/09/equifax-data-breach-what-do

[32]—https://www.nbcnews.com/tech/tech-news/yahoo-must-face-litigation-data-breach-victims-judge-rules-n797871

[33]—https://www.nbcnews.com/storyline/hacking-in-america/more-4-billion-data-records-were-stolen-globally-2016-n714066

[34]—https://www.keepcalm-o-matic.co.uk/p/keep-calm-and-code-java/

[35]—https://insights.stackoverflow.com/survey/2017

[36]—https://www.alexa.com/siteinfo/stackoverflow.com

[37]—https://www.usnews.com/education/best-colleges/articles/2011/08/05/how-higher-education-affects-lifetime-salary

[38]—https://undergrad.cs.umd.edu/what-computer-science

[39]—https://en.wikipedia.org/wiki/Computer_science

[40]—Learn Java for Web Development ~ Vishal Layka

[41]—https://zeroturnaround.com/webframeworksindex/

[42]—
https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_uml_analysis_model.htm