International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 6 Issue 9|| September 2017 || PP. 65-74

Analysis of PHP and Java Languages for Enterprise Applications

Muhammad Bello Aliyu[1] Sunday Kissinger[2]
[1](*Computer Science Unit, Department of Mathematics, Usmanu Danfodiyo University, Sokoto*
aliyu.mbello@udusok.edu.ng)
[2](*Computer Science Unit, Department of Mathematics, Usmanu Danfodiyo University, Sokoto*
sundaykissy@gmail.com)
Corresponding Author: Muhammad Bello Aliyu

Abstract: *Enterprise applications are now becoming pervasive in our today's world simply because of the high demand for it by industries, government agencies, companies, and individuals for the purpose of enhancing their mode of operations. Several enterprise applications have been developed by business enterprises through which their products would not only be made available on the internet, but also enable their prospective consumers to be able to follow some procedures to make their purchases online. Bespoke applications are increasingly becoming common this days; this ensures that organizational needs are met readily. This is normally achieved with the help of some technologies such as Java, PHP and a host of others. This paper presents an analysis of these technologies with a view to finding out the most appropriate technology for adoption in bespoke applications development. It was found that Java has more security edge over PHP and has better language structure although PHP is simpler to learn.*

Keywords: *Bespoke, choice of programming language, Enterprise, Java, PHP.*

Date of Submission:24-08-2017 Date of acceptance: 09-09-2017

I. Introduction

Enterprise applications are software systems that help organizations to run their businesses. They add a degree of automation to the implementation of business processes as well as supporting tasks such as planning, data analysis, and data management. A key feature of an enterprise application is its ability to integrate and process data from different business areas, providing a holistic, real-time view of the entire enterprise [1].

The Java programming language is a general-purpose, concurrent, class-based, object-oriented language. It is designed to be simple enough that many programmers can achieve fluency in the language. The Java programming language is related to C and C++ but is organized rather differently, with a number of aspects of C and C++ omitted and a few ideas from other languages included. It is intended to be a production language, not a research language [2]. Java is one of the best-suited languages for building distributed components for the following reasons:

Firstly as a result of its rich Interface/implementation separation. This entails the separation between the interface and implementation mainly to keep the component upgrades and maintenance to a minimum. Java supports this separation at a syntactic level through the interface and class keywords.

Secondly due to its safety and security features. In this regard the java architecture is much safer than traditional programming languages. In Java, if a thread dies, the application stays up. Pointers are not an issue since the language never exposes them to the programmer. Memory leaks occur much less often. Java also has a rich library set, so that Java is not just the syntax of a language but a whole set of prewritten, debugged libraries that enable developers to avoid reinventing the wheel in a buggy way. This safety is extremely important for mission-critical applications.

Thirdly it is Cross-platform. Java runs on any platform. There is a Java Virtual Machine (JVM) for all platforms. Vendors provide support for their application servers across all the platforms most of the time. This means that EJB applications could be deployed on all these platforms. This is valuable for customers who have invested in a variety of hardware platforms, such as Intel, AMD X32-X64, SPARC, and mainframes, as well as operating platforms, including various flavors of UNIX, Windows, and so on, in their data centers.

On the other hand, PHP which is known as Hypertext preprocessor is a scripting language initially designed for the web. PHP was just starting its emergence as more than a niche scripting language for hobbyists. That was the time of PHP 4 and the first Zend Engine had made PHP faster and more stable. PHP deployment was also increasing exponentially, but it was still a hard sell to use PHP for large commercial web sites. This difficulty originated mainly from two sources:

Firstly, Perl/ColdFusion/other-scripting-language developers who refused to update their understanding of PHP's capabilities from when it was still a nascent language. Secondly, Java developers who wanted large and complete frameworks, robust object oriented support, static typing, and other "enterprise" features.

1.1 Enterprise Application

As seen in the introductory part, an enterprise application is designed to be used in a corporate environment such as business, government, non-governmental (NGO) etc. By nature, such applications need to be scalable, secured, mission critical, interoperable e.t.c. However, these features, in addition to the programmer skills and experience, can only be implemented in a language that was developed to provide such support. By the nature of the enterprise application, they tend to be used by users from a disparate locations/regions over a network hence, most of the enterprise applications are web based. PHP and Java have dominated the web application domain, providing features and libraries for the various aspect of the web paradigm. They (PHP and Java) are different in design, platform support and applicability.

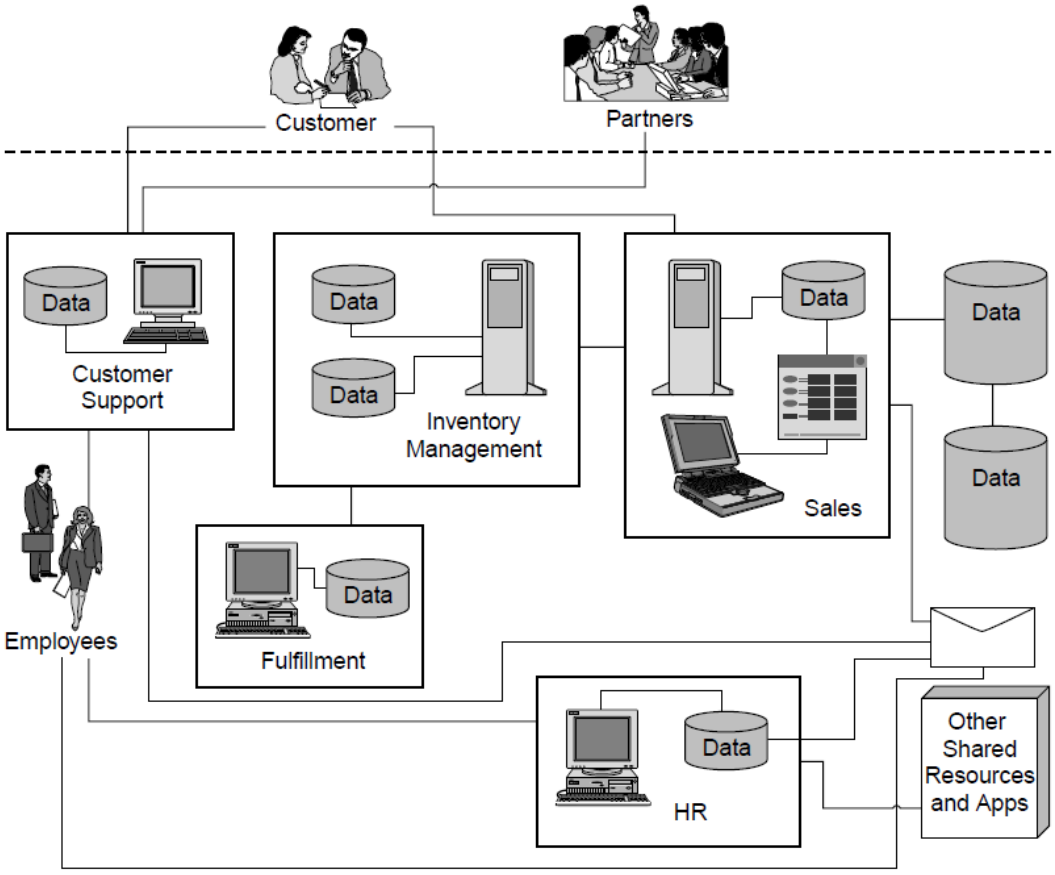

Figure 1.1 Enterprise environment

1.2 Challenges of Enterprise Application Development

Enterprise application development is constrained by time. The applications need to go live as soon as possible i.e. be put in operation. Timing has always been a critical factor when organizations adopt new technologies, and the accelerated pace of the information-driven business model puts greater emphasis on response times [3]. In this environment, timeliness, productivity, security, and predictability are all absolutely critical to building and maintaining momentum. A number of factors can enhance or impede an organization's ability to deliver bespoke enterprise applications quickly and to maximize their value over their lifetime.

II. Statement Of Problem

Enterprise application is having a revolutionary effect on business. It is changing the way businesses interact with consumers, as well as the way they interact with each other. Electronic interactions are increasing the efficiency of purchasing, and are allowing increased reach across a global market. Over the last few years, PHP adoption within the enterprise community has improved significantly making the language go mainstream within the IT industry because of it is fast and cheap cost [4]. However, sceptic views about the language are still surrounding its choice for implementing enterprise applications. The sceptic views about PHP for

enterprise applications are limitations due to its design. Problems such as scalability, performance, modules, and integration, seem to be the major problem with this technology. Java has enjoyed better security reputation for applications, in addition more robust language design and libraries and hence make Java to be the language of choice for enterprise applications. Therefore, the need to analyse the languages' features and practices for enterprise adoption with a view to justifying or refuting their adoption in the enterprise domain is of great demand.

III. Literature Review

The Internet and World Wide Web represent a foundation on which enterprises are working to build an information economy. In this economy, information takes on as much value as goods and services, and becomes a vital part of the market. The information economy challenges today's enterprises to radically re-think the way they do business [5].

Distributed bespoke applications are the packages in which an organization delivers information as a commodity. Custom applications add value to and extract value from the information assets of an organization. They allow IT organizations to target specific functionality to specific user needs. By making information available within an organization, they add strategic value to the management and planning processes.

Computer programs are designed to solve human problems. A process called dynamic programming is a technique for breaking down larger problems into smaller ones. The plan is to solve each smaller problem and then put everything back together into a single, larger solution.

The Java programming language was originally called Oak, and was designed for use in embedded consumer-electronic applications by James Gosling. After several years of experience with the language, and significant contributions it was retargeted to the Internet, renamed, and substantially revised to be the language for enterprise applications [6].

According to [2], a programming language needs to meet six criteria to be usable in an enterprise application. These are fast prototyping and implementation, Support for modern programming paradigms, Scalability, Performance, Interoperability and Extensibility. According to him, Object Oriented Programming is an essential feature for the development of any enterprise application; in this regard he posited that since the Java language was designed with OOP in mind, it will be natural for Java to perform better than any other language. PHP on the other hand has OOP features retrofitted in its design although the firs Zend Engine had made PHP faster and more stable. PHP deployment was also increasing exponentially, but it was still a hard sell to use PHP for large enterprise web sites. Scripting languages such as PHP in general are great for agile products because they allow you to quickly develop and test new ideas without having to go through the whole compile, link, test, debug cycle. PHP is particularly good for this because it has such a low learning curve that it is easy to bring new developers on with minimal previous experience. He also emphasized that PHP 5 has fully embraced the rest of these ideas as well. As you will see in this book, PHP's new object model provides robust and standard object-oriented support. PHP is fast and scalable, both through programming strategies you can apply in PHP and because it is simple to re-implement critical portions of business logic in low-level languages.

According to [7] VP product development at Addepar, Java, at least in Silicon Valley, is one of the *leading* languages for building web applications. When you're talking about web applications at large scale, you have to consider things like security, performance, concurrency, developer productivity, scalability as a language, interoperability, and how easy it is to find skilled persons who are familiar with the technology. According to him, other languages might also be secure or scalable, but when looking at the industry landscape, Java is certainly on top of that list. Google (Android also), Amazon, Yammer, Oracle, eBay, LinkedIn, and *tons* of start-ups all use Java as one of their primary languages of choice. On the other hand, PHP enjoys wider acceptability especially by beginners mostly due to its simplicity.

IV. Methodology

First of all, the requirements of the enterprise applications were identified. From the identified requirements both languages were analysed in view of the requirements. Then we looked at both languages' potentials in the light of such requirements. The requirements and sub requirements were taken one at a time and appropriately matched against the potentials of the PHP and Java; and then analysed. The demonstration involving implementation of the features was done and the results was analysed.

4.1 Language features Analysis

The features of the language were those identified by the literature that have to do enterprise applications/systems.

4.2 Information handling

PHP is weak typed. This means that the variables declaration do not have explicit data type declarations, hence re-assignment is easily done. This does not however mean variables do not have data types but variables are not

restricted a particular datatype. This is not a safe programming especially where users have to enter data in an unsupervised way. Take a look at the codes below:

```php
<?php
$amount = $_POST['amount'];
$amount = $amount + 100;
echo ($amount)

?>
```

Figure 4.21 posting data from HTML form

```php
<?php
$name="Ali";
$name = $name + 100;
echo $name

?>
```

Figure 4.22 variable declaration

In the figure 4.21 above example, $amount is a value received from a form (which is obviously a number type). Depending on what is entered, the amount is increase by 100 and stored in a variable called $amount. The problem however, is that if the user enters a string value in place of number, an error is supposed to be generated at line 3 but instead, the code runs successfully generating an output of *100*. This is an unsafe practice especially in the enterprise domain. One way to possibly fix the problem in figure 1 is to validate the form at the front end, while the validation could be bye-passed by users. Well, we could have another level of validation using the PHP codes. It could be done but at a cost of having an extra cost for this functionality. Also, when type is not attached to variables (as this is the case with PHP), the interpreter must determine the type at run time thus, incurring another overhead.

Java is strongly typed. This is means that variables are only bound to a particular data type. This ensures the strict adherence to a particular data type to spot errors earlier hence, speeds development. It also generates an optimised code from the compiler as it does not involve the determination of the type at the run time [8].

For most modern enterprise applications, they integrate one or more information sources as integral component of the application. Strongly typed languages are key to providing a strongly typed access to these resources to ensure a low impedance mismatch to information access [9]. The figure below shows the various applications with open API timeline that generate data which is used by other applications. This data has a format that must be preserved as it is being passed to other applications.

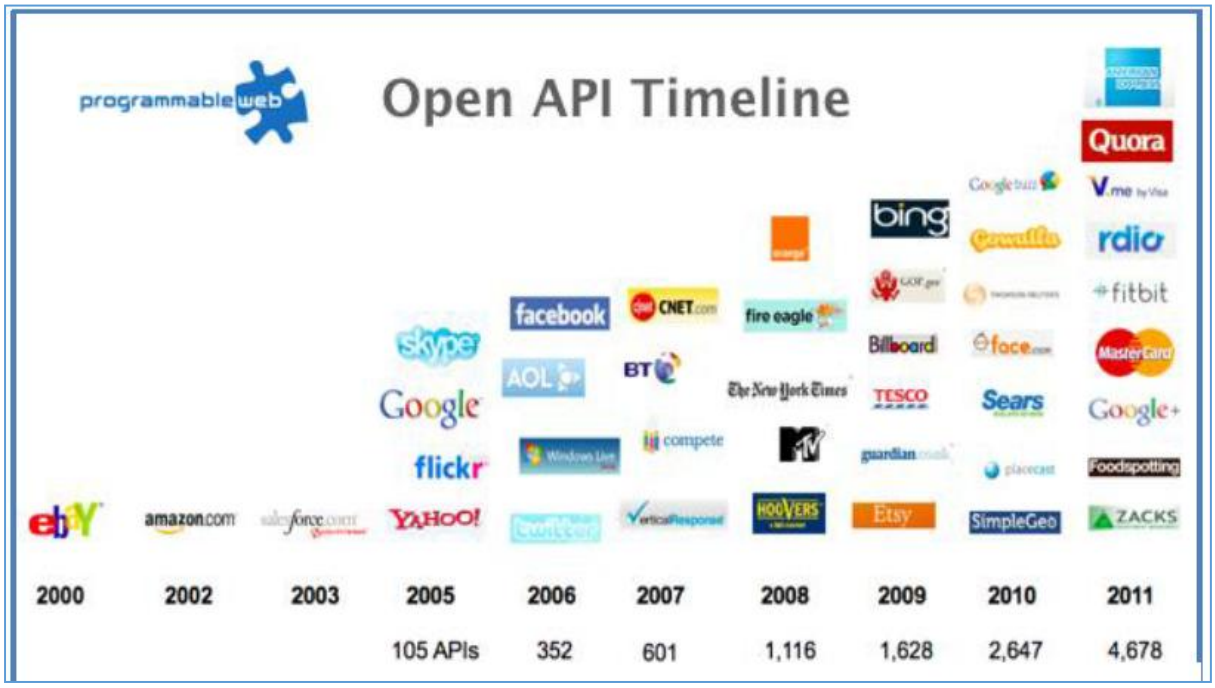

Figure 4.23 API timeline for enterprise applications

Java, a strongly typed language, is able to seamlessly integrate external information sources as if they were strongly typed components from the programmers' perspective. This is because, the enterprise applications now tend to be more information-focused as opposed to code-focused hence, strongly typed programming languages tend to have more ground for adoption in the enterprise world. This is because the information format needs to be preserved against the external interference.

4.3 Exception Handling

A key aspect of an efficient system is to withstand all situations. In programming terminologies, it must have a good exception handling structure. Many PHP Built-ins and libraries do not have exception handling built within them. This makes bigger the tendency for error to propagate. This does not mean that PHP as a language does not have exception handling feature, but the libraries and the frameworks lack the in-built exception handling feature. This makes it easier to break into the applications' code. The PHP libraries instead report errors by the user in some ways (the default exception handling in PHP). This adds an additional overhead. The PHP's frameworks such as DOM, PDO, MySQL etc. Usually throws their own exceptions without catching them. To demonstrate the above claim, consider the following piece of code.

```
$db_link = mysqli_connect('localhost', 'username', 'Password', 'dbname');
function user_access($current_user) {
    global $db_link;
    $username = mysqli_real_escape_string($db_link, $current_user->username);
    $res = mysqli_query($db_link, "SELECT COUNT(id) FROM blacklisted_users WHERE
username = '$username';");
    $row = mysqli_fetch_array($res);
    if ((int)$row[0] > 0) {
        return false;
    } else {
        return true;
    } }
if (!user_access($current_user)) {
    exit();
}
```

Figure 4.31 *mysql*i database logging

The code above could face many runtime setbacks. For example, the database connection could fail, due to wrong password or the server non-availability because it could be closed by the server after it was opened at the client side. In these cases, by default the *mysqli* functions will issue warnings or notices, but will not throw exceptions or fatal errors. This means that the code simply carries on! The variable $row becomes *NULL*, and PHP will evaluate $row[0] also as *NULL*, and (int) $row[0] as 0, due to weak typing. Eventually the *can_access* function returns *true*, giving access to all users, whether they are on the login list or not. If these native database APIs are used, error checking should be added at every point. However, since this requires additional work, and is easily missed, this is insecure by default. It also requires a lot of boilerplate.

It's often best to turn up error reporting as highly as possible using the *error_reporting* function, and never attempt to suppress error messages — always follow the warnings and write code that is more robust.

Libraries and projects written in PHP are often insecure due to the problems highlighted above, especially when proper web frameworks are not used. Do not trust PHP code that you find on the web, as many security vulnerabilities can hide in seemingly innocent code. Java on the other hand, catches library specific exceptions which are raised when the libraries are used. For example, the database connectivity exception which are raised if any goes wrong during database connectivity and operations. The figure shows the exception is raised and caught in the codes.

```
public void getData(){
    try {
         String query = "select * from login";
         rs = st.executeQuery(query);
         while(rs.next()){
         String user = rs.getString("username");
         String pass = rs.getString("password");
         System.out.print("Username " + user + "Password "+ pass);
         }
    } catch (SQLException ex) {
          System.out.print(ex);
    }
}
```

Figure 4.32 Java Database query with inbuilt Exception

4.4 Naming Convention

Java has a consistent naming convention for all its classes, variables, functions and libraries. This consistency manifest in all situations. For example, all class names should begin with an upper case, variables and methods begin with a lower case (except for compound words which with an initial of lower case and subsequent upper case letters) and constant names must consist of capital letters entirely.

```
package lab3;

public class Salary extends Employee {
private double salary;//Annual salary
public Salary(String name,String address,int number,double salary){
super(name, address, number);
setSalary(salary);
}
public void mailCheck(){
System.out.println("Within mailCheck of Salary class ");
System.out.println("Mailing check to "+ getName()
+" with salary "+ salary);}
public double getSalary(){
return salary;}
public void setSalary(double newSalary){
if(newSalary >=0.0){
salary = newSalary;
}}
public double computePay(){
System.out.println("Computing salary pay for "+ getName());
return salary/52;
}}
```

Figure 4.41 Java naming conventions

This paradigm is shared by PHP as well. However, the PHP does not maintain this consistency. Variable names are case sensitive but the function names are not. Consider the piece of code below

```
<?php
FoR( $start= 0 ; $start < 5 ; $start++ ) {
EcHo('Good morning <br />');
}
?>
```

Figure 4.42 PHP naming conventions

The *for* keyword and the *echo*() function are not case sensitive, so the above code would run without errors. But the variables are very case sensitive such that $start and $Start are understood differently by the parser and hence an endless loop result in the codes below.

```
<?php
FoR( $start = 0 ; $start < 5 ; $Start++ ) {
EcHo('Good morning <br />');
}
?>
```

Figure 5.25 PHP naming conventions

For the enterprise application development, the language features are very much significant in choosing the language for implementation.

4.5 Multi-threading

Multi-threading is a means of allowing more than one program to run. This involves systems with multiple core CPU. PHP does not support native multi-threading. The *threads* extension in PHP cannot be used in a web server environment. Therefore, *threading* in PHP remains CLI-based application only. However, PHP gives a solution that allows the retrieval of multiple web-pages for processing instead of sequential retrieval using native PHP codes. Java on the other hand, fully supports multi-threading natively.

4.6 Interoperability

One of the key requirements for the enterprise business is the interoperability. Applications needs to exchange and use data with other applications in a seamless way. This communication can be enabled via web services. PHP very much support interoperability. It provides the interfaces such as REST, XML support, SOAP, Zend Framework for web services. Java however, has broader support.

4.6.1 The Interoperability Architecture

Interoperability is the ability of a system or product to work with other systems or products without special effort on the part of the customer (Rouse 2006). Enterprise systems need to operate with each other in a seamless way; hence interoperability has become a feature of increasing importance. To achieve such interoperability between products and services, standard interfaces are published which must be adhered to by the systems. Examples of these standards include TCP/IP, Hypertext Transfer Protocol (HTTP) and Mark-Up Language (HTML) for web applications. Another way is to make use of 'broker' of services that can convert one product's interface into another product's interface "on the go". Examples of this include: Object Request Broker (ORB) that uses CORBA specification (Common Object Request Broker Architecture). An application is compatible with a standard but interoperable with other systems or products that meets the same standard.

CORBA uses an interface definition language (IDL) to specify the interfaces that objects resent to the outer world. CORBA then specifies a mapping from IDL to a specific implementation language. Standard mappings exist for Java and PHP. Java EE compatibility requires neither interoperability from the provider of identical Java EE products from the same provider nor among heterogeneous products from multiple providers. Instead, IIOP transaction propagation protocol (defined by OMG, described in the OTS specification and implemented by the Java Transaction Service) be used for the transaction interoperability. However, Java products must support resource adapters (Connector) that use *XATransaction* as transaction resource managers. The platform also enables transactional access to the adapter from servlets, JSP pages and enterprise beans [10].

Both PHP and Java EE provide support for both client of web services and web service endpoints.

Another flexible way for inter-operability is the Extensible Mark-up Language (XML). XML is a simple, very flexible text format that plays an important role in the exchange of a wide variety of data on the Web and elsewhere. The Axis2 is an open source web service that provides a web service engine [5]. It was designed to support robustness and support for a variety of WS-* standards. PHP support the Apache Axis2 which was written in C. Axis Object Model (AXIOM) provides the simple interface for SOAP. The XML processing is enabled through AXIOM which is based on the StAX API (streaming API for XML). The SOAP provides a SAX-like interface and the DOM-like approach for pull-parser streaming [11].

For Java, the default XML parser in the application server is the Sun Java System XML Parser (SJSXP). However, the woodstox parser is normally bundled with the application server for improving the

performance. The combination of SJSXP and Woodstox both provides the implementations of the stAX API. The figure below shows the configuration for enabling the Woodstox parser.

```
<config name=server-config>
...
    <system-property name="javax.xml.stream.XMLEventFactory"
   value="con.ctc.wstx.stax.WstxEventFactory "/>
   <system-property name="javax.xml.stream.XMLInputFactory"
   value="con.ctc.wstx.stax.WstxInputFactory "/>
   <system-property name="javax.xml.stream.XMLOutputtFactory"
   value="con.ctc.wstx.stax.WstxOutputFactory "/>
</config>
```

Figure 4.61 Woodstox parser configuration.

A study by (Farwich & Hafner 2010) has revealed that LibXML2 XML parser written in C (which the Apache support) performs better than other XML parsers including woodstox (StAX parser which was written in java). The result showed that the Apache's StAX-like implementation of LibXML2 performs best in all situations with various document size ranges. Specifically, the result showed that LibXML2 implemented in C provides nearly twice the throughput provided by other parsers. Also, [12] reported that the speed of creating AXION or DOM for LibXML2 is much faster than all other implementations.
This means that PHP is a good candidate for interoperability feature and hence, can pretty much be used for enterprise application development.

4.6.2 Integration

One of the key requirements of the enterprise application is how easily it is to integrate with the existing enterprise information systems. This is key to the nature and the value of the enterprise systems and products. Systems need to communicate with each other in a seamless way. J2EE platform provides standard APIs for such integration. Some of the APIs include:

- JDBC is the API for accessing relational data from Java.
- The Java Transaction API (JTA) is the API for managing and coordinating transactions across heterogeneous enterprise information systems.
- The Java Naming and Directory Interface (JNDI) is the API for accessing information in enterprise name and directory services.
- The Java Message Service (JMS) is the API for sending and receiving messages via enterprise messaging systems like IBM MQ Series and TIBCO Rendezvous.
- JavaMail is the API for sending and receiving email.
- Java IDL is the API for calling CORBA services.

4.7 Security

Security is a huge consideration for enterprise applications as it forms a critical aspect of the business. Therefore, it is necessary to prevent the stealing of confidential data, breaking of data integrity and authorised access to the business. This is obvious for enhancing the security enterprise systems such as e-commerce, banking systems and security enhanced online registration prepaid scratch payment [13]. Therefore, any language chosen for implementing enterprise application must be well-featured in this aspect.
PHP as a language has suffered security reputation problems. PHP challenges security were identified by [14] they include:

1. SQL Injection
2. Remote File Inclusion and Remote File execution
3. Cross Site Scripting (XXS)
4. Session and Cookie Hacking
5. Directory Transversal
6. Source Code Revelation

These security vulnerabilities were contained in the earlier versions of the language and have been fixed in the newer version of the language [15]. However, it is down to the skill, productivity and experience of the programmer to avoid such security traps.

4.7.1 SQL Injection Prevention

PHP has a very good language feature for preventing SQL injection. This is available for all the three APIs to access MySQL backend using prepared statements and parameterised queries. These are SQL statements that are sent and parsed by the SQL server separately from any parameters [16]. This way, it becomes impossible for an attacker to inject malicious SQL statements. These are illustrated as follows.

Using PDO

```
$stmt = $dbConnection->prepare('SELECT * FROM employees WHERE name = ?');
$stmt->bind_param('s', $name);
$stmt->execute();
$result = $stmt->get_result();
while ($row = $result->fetch_assoc()) {
    // do something with $row
}
```

Figure 4.71 SQL injection

Using MySQLi

```
$stmt = $pdo->prepare ('SELECT * FROM employees WHERE name =: name');
$stmt->execute (array('name' => $name));
foreach ($stmt as $row) {
    // do something with $row
}
```

Figure 4.72 SQL injection using MySQLi

Java has inbuilt security architecture through the Java Authentication and Authorisation Service (JAAS). JAAS enables services to enforce access control upon users. Java applications do not fall to these security risks mainly for its use of APIs. For example, JDBC API for accessing relational databases is robust enough such that SQL injection can rarely scale through. In addition, Java transaction API (JTA) makes information exchange between heterogeneous enterprise systems much safer than without the APIs. One of the goals of the security architecture was to specify the component's security requirement using the deployment descriptors. This gives the deployer the freedom to specify/modify the security parameters which are consistent with the deployment environment. In addition, Java provides both declarative and programmatic security.

V. Conclusion

The enterprise applications do require lots features and consideration because they are: business sensitive, on-demand from desperate locations, mission critical and security conscious. PHP and Java have dominated the implementation of enterprise applications, however with a varying degree of efficiency. The PHP is easy to learn and apply by starters in web development. Its LIBXML2 XML parser performs better than the Java's Woodstox. This makes information transfer between applications much easier. However, Java is more secured, strongly typed, support multi-threading and has better APIs for enterprise environment such as Java Transaction API.

Therefore, when security is a primary concern for the enterprise, Java should be considered for the enterprise software. This does not mean that PHP is not secured, but the inbuilt security features of Java does not require any additional overhead to make the application much secured unlike PHP. This applies to multi-threading as well.

On the overall, Java shows more promising in the enterprise environment than the PHP.

References

[1]. Rima, P. S., Gerald, B., & Micah, S. (2006). Mastering Enterprise JavaBeans 3.0. Canada: Wiley Publishing Incorpation.
[2]. James, G., Bill, J., Guy, S., Gilad, B., & Alex, B. (2013). The Java Language Specification . California: Oracle America, Inc.
[3]. Inderjeet, S., Beth, S., Mark, J., & Team, a. E. (2002). Designing Enterprise Applications with the J2EE Platform, Second Edition. California : Sun Microsystems, Inc.
[4]. Zacharewicz, G., Agostinho, C., Ducq, Y., Sarraipa, J., Lampathaki, F., Poler, R., & Jardim-Goncalves, R. (2016). Towards a sustainable interoperability in networked enterprise information systems: trends of knowledge and model-driven technology. Computers in Industry, 79, 64-76.
[5]. Nicholas, K., & Team, a. t. (2000). Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition. California: Sun Microsystems, Inc.
[6]. George, S. (2004). Advanced PHP Programming. Indiana: Sams Publishing.
[7]. Derek, B. (2014). VP Product Development. Addepar.
[8]. Ericson C., (2009) Object Oriented Programming Atomic Object LLC. (Online) available at https://atomicobject.com/uploadedImages/archive/files/ObjectOrientedProgramming.pdf retrieved 15 May 2017
[9]. Syme, D., Battocchi, K., Takeda, K., Malayeri, D., Fisher, J., Hu, J., Taveggia, M. (2012). Strongly-typed language support for internet-scale information sources. Technical Report MSR-TR-2012–101, Microsoft Research,
[10]. Shannon, J. P.Enterprise edition (java EE) specification, v5, may 8, 2006, and 2006 sun microsystems.
[11]. Deshmukh, V. M., & Bamnote, G. R. (2015, February). An Empirical Study of XML Parsers across Applications. In Computing Communication Control and Automation (ICCUBEA), 2015 International Conference on (pp. 396-401). IEEE.
[12]. Woodstox (2009) Woodstox available at http://www.woodstockstory.com/woodstock2009.html accessed 15 July 2017.
[13]. Oyemade, O., Odiagbe, J., & Buhari, B. (2015). A survey of security vulnerabilities in PHP applications among IT professionals in nigeria. African Journal of Computing & ICT, 8(3)
[14]. Jovanovic, N., Kruegel, C., & Kirda, E. (2006, May). A static analysis tool for detecting web application vulnerabilities. In Security and Privacy, 2006 IEEE Symposium on (pp. 6-pp). IEEE.
[15]. Walden, J., Doyle, M., Lenhof, R., & Murray, J. (2010). Idea: Java vs. PHP: Security implications of language choice for web applications. International Symposium on Engineering Secure Software and Systems, 61-69.
[16]. Siddiqui, M. S., & Verma, D. (2011). Cross site request forgery: A common web application weakness. Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, 538-543.

Muhammad Bello Aliyu. “Analysis of PHP and Java Languages for Enterprise Applications.” International Journal of Engineering Science Invention (IJESI), vol. 6, no. 9, 2017, pp. 65–74.