International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 2, February - 2014

Polyglot Programming

C. Mritula
Christ University
Bangalore, India.

Arun. B. C
University of Madras
Chennai, India

***Abstract*—Polyglot programming is the technique of choosing and using the right tool that is more keenly shaped for the job that has to be performed. The advent of polyglot programming or polyglot programmer's focuses on solving a problem by using various best software's suitable for that job yielding the required result. The paper focuses on discussing the knits and knots of polyglot programming and few of its possible applications**.

Keywords- Polyglot programming; Framework; JVM;

I. INTRODUCTION (POLYGLOT PROGRAMMING)

Software is the most essential component of the computer which is written using any high-level programming language in a human interpretable form that makes life easier and performing tasks more efficient. A software programming paradigm is defined as a fundamental style of computer programming, a way of building the structure and basics of computer programs. A collection of coherent, often ideologically or theoretically based abstractions constitutes a programming paradigm. Often, a given programming technology is based on one particular paradigm [7]. There are five main paradigms: imperative, functional, logical and symbolic programming [4].Any given task to the programmers is handled completely using either of the paradigms mentioned. Though the purpose is served, the efficiency of the purpose and the duration of it make it highly questionable.

Knowing and using multiple programming languages for normal day-to-day development can yield significant benefits. No single language is a great fit for all problems [6].Polyglot programming is the technique of using several programming languages in a software system. The goal of this paper is to unfold the understanding of methods and various techniques for polyglot programming and incorporation, when these are appropriately applicable and advantages and disadvantages of polyglot programming.

II. WHAT IS POLYGLOT PROGRAMMING?

Polyglot programming is the activity of programming in more than one language within the same context [1]. Technically, it targets on rendering simpler solutions by combining best aspects from many programming languages that can help to complete a specific task more flexibly and easily. Comparing to the real world scenario, humans learn and use many languages for an effective communication ensuring that the goal of conveying the information is successfully accomplished.

Languages evolve to meet the changing environments in which they find themselves. [5] Mixing and matching various languages and platforms can be a flexible option to achieve a task more easily. This can be done by choosing a language that is most suitable for the problem. On using a tool that is more keenly shaped for the task keeps the job of a programmer much at ease. The idea of polyglot programming is not entirely new one, it is just the terminology. It has been quite in practice that many tools and applications run as a combination of two or more software's one in front end other for implementing business logics and other for database implementation, etc

III. WHY POLYGLOT PROGRAMMING?

Programming is a unique behavior, no matter how much we torture metaphors to compare it to other activities and professions. It combines engineering and craft in highly coupled ways, requiring good developers to exhibit a wide variety of skills: analytical thinking; extreme attention to detail on multiple levels and aesthetics; an awareness of macro and micro level concerns simultaneously; and a keen, fine-grained understanding of the subject matter we're writing software to assist.[7].There are many languages getting developed day by day, those which mostly can run on a similar virtual machine but provide an advanced flexibility to perform some specific task than other. Instance, other than java there are many more languages like Groovy, Ruby, Scala, Swing etc. that can run over the Java Virtual Machine (JVM). Using such languages makes the task of a programmer much flexible. On this basis the programming world is moving on to a different focus, polyglot programming, where a complete task is divided into smaller units then the most appropriate paradigm or language is used to develop the particular unit and at the end, every individual unit is combined and the whole task can be deployed over a single virtual machine. . The Polyglot framework is useful for domain-specific languages, exploration of language design, and for simplified versions of Java for pedagogical use [15].

For case in point, Java is a wonderful language to program but has a few pitfalls like designing User Interface (UI) becomes a tedious job using Java, whereas other languages like Groovy or Ruby can be used to build the UI and the underlying business logic can still be written using Java as all the software's mentioned are compatible to run over the JVM.

IV. PRODUCTIVE PROGRAMMER.

Productivity is defined as the amount of useful work performed over time. Someone who is more productive performs more effective work in a given time interval than someone less productive [6]. Customer requirements are highly volatile in nature. E.g., requirements crop up in various forms to developer community. Such as, requirements are received through short message text, Instant messages etc. Hence, choosing right technology using polyglot programming plays a vital role in delivering the requirements faster and thus saving cost too.

When a programmer is aware of multiple languages or paradigms and the advantage and disadvantages, the acquired knowledge can be used to achieve the requirement adhering to the parameters much flexibly. Thus, making himself a productive programmer whose widespread knowledge about various languages eases the application development.

For Instance, Compilers built on polyglot are themselves extensible; complex extensions such as JIF and PolyJ have themselves been extended.

V. WHEN TO USE POLYGLOT PROGRAMMING?

In the world of computing, software products or applications are created mostly using high level languages. Languages like C, C++, Java, Groovy, Scala, and Ruby are used to create software products or applications. Programs can be laid out along a spectrum with predominantly symbolic programs at one end and predominantly numerical programs at the other [8].It's the fact that the programming languages were created with a purpose in mind and later enhanced based on the needs. Also, there were concepts beyond which some of these languages were not enhanced. Its best known that difficulties, limitations etc. in one programming language has led to the creation of another. Normally, when a computing problem is concerned, one tries to solve it by thinking and applying the strategies using the programming language which that individual is familiar off. This is natural and there is nothing erroneous in that. For instance considering hardware as an example here, if a need arises to buy a laptop and there is an option to choose the processor one prefers, the operating system, the RAM, etc. as per the requirements and individuals choice. This method is followed because that is what can help us to assist individuals problem or help to face ones requirements in a better manner. Polyglot system has two essential aspects, the platform used for the integration and the programming languages supported. The recent rise of non-Java programming languages running on the Java Virtual Machine has created a favorable environment for polyglot programming [12]. Knowledge to choose which fits in appropriately for accomplishing the individual's task is most important aspect to be considered. Relatively in software, a product is developed using one language which is more suitable for completing the requirement and when the programmer is aware that there is another programming language that is robust and has easy to use concurrent API's and if on using both these languages can make development faster and easier provided one has the awareness of how to technically to interoperate between these two languages then the best of both the worlds can be used and our software and the result can be attained successfully. When changes are confined to the implementation, a preconceived set of application invariants may serve as a basis for preserving system integrity [15].

On considering few other use cases, we can classify software languages into two categories, one is the language which requires compilation and other is the language which is interpreted. Building a compiler is both science and black art and demands an intimate knowledge of data structures, algorithms, high-level programming languages, and processor architectures and their instruction sets [9].It is merely same with interpreters too. Both approaches have their own advantages and disadvantages. Instance considering a software product which is developed using Java Enterprise Edition. The source code is compiled, tested, packaged and then deployed. Once it is deployed, any change to the software has to go through the big cycle of compilation, testing, packaging and deployment again. Considering a simple use case to explain the problem, let us develop a product which will do the invoicing and sales tax calculation of products. When this product is sold to different countries, they may have their own sales tax calculation logic. If we have to create sales tax of all the countries in the world and package it as part of our application then the problem look like solved, but it is not. Because, these sales tax calculations may be revised over a period of time. We can say that sales tax calculation is dynamic and may require changes at different times in a period and it may be different for different countries. We can always extend our object oriented sales tax calculation class, but again there is an overhead of compiling, testing, packaging and deployment. Otherwise the best solution can be foreseen as we develop our enterprise application using Java Enterprise Edition and have the sales tax calculation logic implemented using an interpreter based language which can interoperate with Java using the JVM, then it can be used to easily modified the sales tax calculation based on the needs, tested and pushed it over to production. This approach made software more extendible and easier to use. Thus polyglot programming, using more than one programming language to create software serves the purpose.

VI. IMPLEMENTATION

This validates the scope of implementation of polyglot programming and its possibility. For Instance Consider JVM, Java Virtual Machine. The compiled java code is executed by JVM. Similarly there are other programming language which can run on JVM. For example, Ruby (interpreted language) flavor which runs on JVM is called as JRuby. Groovy is interpreted language and it is designed to run on JVM. Scala is compiler based language which is has very good concurrency API that can also run on JVM. Clojure and many more languages that can run on JVM are available.

Considering the Microsoft world, there is a Common Language Runtime which is similar to JVM and it has support for various programming languages like Visual C#, Visual Basic.Net, Visual F# and again there is a growing big list.

Seeing Java space to clarify the fact of how easy it is make these languages interoperate, a sample code snippet is provided below. To make this understanding simple, the frameworks like Spring is used, which takes case of inner workings of tying or facilitating the interoperability. Software creation is an art, how well it can be thought through, how simple and easy to maintain and how well it solves the problem counts to its success. Everything today is fast, dynamic and hence creating software has become really a challenging task. Using the techniques which was helpful in the past may be good but not apt for the current market trends and conditions. Hence using appropriate programming languages to create software plays an important part in the industry. Example or Code Snippet:

Step 1: Considering interface creation in Java (may be in enterprise application)

```
package.org.ployglot.programming.sample;

import
org.springframework.stereotype.Component;

@Component
public interface InterestCalculator{

   public void setRate(double rate);

      public double calculate(double amount,
                double year);}
```

Step 2: Create a Ruby class which has the calculation logic. This ruby code is like a script or it is interpreted, hence any changes to the file are immediately reflected in the programming execution

```
class SimpleInterestCalculator

def setRate(rate)

end

def calculate(amount, year)

end

SimpleInterestCalculator.new

@rate = rate

amount * year * @rate
```

Step 3: Create a spring beans.xml configuration file. This is how interpretability is achieved. Here the beans.xml file in enterprise application will have the hook to Ruby class which can be either in an external location or on classpath based on requirement and will be invoked when the java interface is called.

```
<?xml version="1.0" encoding="UTF-8"?>

<beans
xmlns=http://www.springframework.org/schema/bea
ns

xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns:context="http://www.springframework.org/s
chema/context"

xmlns:lang="http://www.springframework.org/sche
ma/lang"

xsi:schemaLocation="http://www.springframework.
org/schema/beans
http://www.springframework.org/schema/beans/spr
ing-beans-3.2.xsd

<!-- Scans the given directory for Spring
components (java classes with @component
annotation) -->

<context:component-scan base-
package="org.ployglot.programming.sample" />

<!-- Ruby class declaration, this class will be
called by Java code -->

<!– [eg. file in external location] script-
source="file:C:/Users/arun.bc/SimpleInterestCal
culator.rb" -->

<lang:jruby id="interestCalculator"

http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/s
pring-context-3.2.xsd

http://www.springframework.org/schema/lang
http://www.springframework.org/schema/lang/spri
ng-lang-3.2.xsd">
[1]
    script-
    zsource="classpath:SimpleInterestCalculator
    .rb"

script-
interfaces="org.ployglot.programming.sample.Int
erestCalculator">

<!-- setRate() method in
SimpleInterestCalculator.rb will be called
automatically -->
```

VII. POLYGLOT PERSISTENCE.

The era of Big Data has begun. Computer scientists, physicists, economists, mathematicians, political scientists, bio-informaticists, sociologists, and many others are clamoring for access to the massive quantities of information produced by and about people, things, and their interactions [13].Polyglot technique is not confined or restricted to programming world alone; it also marks its existence in data handling in the database field too. Polyglot persistence uses different data stores in different circumstances. To support multiple use cases in a single application is currently being studied as part of an emerging movement, named polyglot persistence [11]. In terms of polyglot the relational databases is also considered as one of the form of data storage along with various other forms. This point of view is often referred to as Polyglot Persistence- using different data stores in different circumstances [3]. As a customary practice the relational database is considered to be the primary medium for data storage. But the emerging trend has paved way to consider data on the basis of its nature and usage purpose. A polyglot persistence database is used when it is necessary to solve a complex problem by breaking that problem into segments and applying different database models. It is then necessary to aggregate the results into a hybrid data storage and analysis solution [10]

Polyglot programming can enhance web development too, because different programming languages and frameworks promise an increase in productivity, reduced amount of code and improved code quality that together promote better maintainability. Although polyglot programming has a steep learning curve that effects required knowledge, maintainability and tool support. In data analytics different analytics in the workflow can be written in different languages enabling a simple polyglot programming model [14].

VIII. ACKNOWLEDGMENT

First of all I would like to thank Mr. Clarence J M Tauro, whose help has been most essential for this paper. Would also extend my gratitude to the faculty of the Department of Computer Science, Christ University for encouraging this experience. For excellent proof editing I would like to thank Ms.Vinutha Kini H. The case study could not have been done without some willing interview subjects. For that, I would extend my thanks Mr. Raj Rajendran, Mr. Sivaraman. R.

REFERENCES

[1] Fjeldberg, Hans-Christian. "Polyglot Programming." PhD diss., Master thesis, Norwegian University of Science and Technology, Trondheim/Norway, 2008.

[2] Nystrom, Nathaniel, Michael R. Clarkson, and Andrew C. Myers. "Polyglot: An extensible compiler framework for Java." In Compiler Construction, pp. 138-152. Springer Berlin Heidelberg, 2003

[3] Sadalage, Pramod J., and Martin Fowler. NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Addison-Wesley, 2012

[4] Gupta, Amit, Shaji Chempath, Martin J. Sanborn, Louis A. Clark, and Randall Q. Snurr. "Object-oriented

programming paradigms for molecular modeling." Molecular Simulation 29, no. 1 (2003): 29-46.

[5] Ford, Neal, The Productive Programmer, O'Reilly Media, Inc.,2008

[6] Vinoski, Steve. "Multilanguage programming." Internet Computing, IEEE 12, no. 3 (2008): 83-85.

[7] Wampler, Dean, and Tony Clark. "Guest Editors' Introduction: Multiparadigm Programming." Software, IEEE 27, no. 5 (2010): 20-24.

[8] Halstead Jr, Robert H. "Parallel symbolic computing." Computer ;(United States) 19, no. 8 (1986).

[9] Muchnick, Steven. "Advanced compiler design and implementation." (1997).

[10] Hurwitz, Judith, Alan Nugent, Fern Halper, and Marcia Kaufman. Big Data for Dummies. Wiley. com, 2013.

[11] Castrejón, Juan, Genoveva Vargas-Solar, Christine Collet, and Rafael Lozano. "Model-Driven Cloud Data Storage." Proceedings of CloudMe (2012).

[12] Harmanen, Juhana. "Polyglot Programming in Web Development." (2013).

[13] Boyd, Danah, and Kate Crawford. "Six provocations for big data." (2011).Harmanen, Juhana. "Polyglot Programming in Web Development." (2013).

[14] Mohindra, Sanjeev, Daniel Hook, Andrew Prout, Ai-Hoa Sanh, An Tran, and Charles Yee. "Big Data Analysis using Distributed Actors Framework."

[15] Oreizy, Peyman, Nenad Medvidovic, and Richard N. Taylor. "Architecture-based runtime software evolution." In Proceedings of the 20th international conference on Software engineering, pp. 177-186. IEEE Computer Society, 1998.