JAVA BACKEND RESEARCH

19-9-2017

Sjoerd Brauer
Fontys Venlo

Dear FSG1,

In this document 7 Java frameworks will be described. This document will outline the pros and cons of each framework. Based on the criteria and pro's and cons a decision will be made.

Criteria for our project are:

1. Framework must work with PostgreSQL.
2. Framework must support http protocol.
3. Framework must have documentation.
4. Framework must provide tutorials on how to use the framework.

Sincerely,

Sjoerd Brauer
E: s.brauer@student.fontys.nl

Spring MVC
Spring is a full blown MVC java framework.

Supports all criteria?
Yes

Pros:

1. Simplified injection of test data through the use of Plain old Java object .
2. Enhanced modularity, resulting in better code readability.
3. Loose coupling between different modules.
4. Dependency Injection (DI) flexible use.
5. you can avoid using SQL scripts but still retrieve data from your database.
6. You can also create a restful interface which can be extended later on by other programmers.
7. You could also use spring security, so that only certain people have access to the rest interface.
8. Easy to set up OATH2.

Cons:

1. Steep learning curve.
2. Consequences of a conflict to an organization.
3. Types of conflict.
4. Identifying in which state a conflict is according to the conflict model somewhere in the sources.

(Sources)
https://spring.io/

Strut 2
Struts 2 is a pull-MVC framework. i.e. the data that is to be displayed to user has to be pulled from the Action.

Supports all criteria?
Doesn't support criteria 1
Strut 2 needs an additional (ORM) framework to connect with a database.

Pros:

1. Configurable MVC components, which are stored in struts.xml file. If you want to change anything, you can easily do it in the xml file.
2. POJO based actions. Struts 2 action class is Plain Old Java Object, which prevents developers to implement any interface or inherit any class.
3. Support for Ajax, which is used to make asynchronous request. It only sends needed field data rather than providing unnecessary information, which at the end improves the performance.
4. Whether you want to use JSP, freemarker, velocity or anything else, you can use different kinds of result types in Struts 2.

Cons:

1. Compatibility
2. Limited Documentation.
3. UI driven framework.

Sources
https://struts.apache.org/

Hibernate
Hibernate ORM (Hibernate in short) is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate handles object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

Supports all criteria?
Doesn't support criteria 2
You need to use something else to handle HTTP communication.

Pros:

1. Caching mechanism to bug database with similar queries.
2. N+1 or Lazy loading support.
3. Inheritance, encapsulation

Cons:

1. Does not permit multiple inserts
2. Supports less queries then JDBC.
3. Not a good choice for small projects.

textbfSources
http://hibernate.org/

akka
Welcome to Akka, a set of open-source libraries for designing scalable, resilient systems that span processor cores and networks. Akka allows you to focus on meeting business needs instead of writing low-level code to provide reliable behavior, fault tolerance, and high performance.

Supports all criteria?
yes

Pros:

1. Multi-threaded behavior without the use of low-level concurrency constructs like atomics or locks âĂŤ relieving you from even thinking about memory visibility issues.
2. Transparent remote communication between systems and their components âĂŤ relieving you from writing and maintaining difficult networking code.
3. A clustered, high-availability architecture that is elastic, scales in or out, on demand âĂŤ enabling you to deliver a truly reactive system.

Cons:

1. Does not run on Java JEE server

Sources
http://akka.io/

Vert.x
Eclipse Vert.x is a polyglot event-driven application framework that runs on the Java Virtual Machine.

Supports all criteria? **yes**

1. Non-blocking, event driven runtime
2. Simple to use concurrency and scalability
3. Polyglot (multiple languages can use vert.x)
4.

Cons:

1. less well known.

Sources
http://vertx.io/blog/posts/introduction-to-vertx.html

JDBC
Java Database Connectivity (JDBC) is an application program interface (API) specification for connecting programs written in Java to the data in popular databases.

Supports all criteria? **Doesn't support criteria 2** You need to use something else to handle HTTP communication.

Pros:

1. You have complete control.
2. Can manipulate large data sets.
3. Great for handling large data transferring.

Cons:

1. Hard to implement MVC concept.
2. Large programming overhead.
3. UI driven framework.

(Sources)
https://www.slideshare.net/rajkrrsingh/proscons-jdbc-hibernate-ejb
https://stackoverflow.com/questions/35955020/hibernate-orm-framework-vs-jdbc-pros-and-cons

playframework
Java Database Connectivity (JDBC) is an application program interface (API) specification for connecting programs written in Java to the data in popular databases.

Supports all criteria? **yes**

1. Reactive. Play is built on Netty, so it supports non-blocking I/O. This means it's very easy and inexpensive to make remote calls in parallel, which is important for high performance apps in a service oriented architecture. It also makes it possible to use server push technologies such as Comet and WebSockets. More info here: Play Framework: async I/O without the thread pool and callback hell
2. Amazing error handling. Play has beautiful error handling in dev mode: for both compile and runtime errors, it shows the error message, the file path, line number, and relevant code snippet right in the browser. No more digging through random log files (Tomcat) and far fewer incomprehensible, gigantic stacktraces (Spring).
3. Java (and Scala). Use reliable, type-safe languages and leverage JVM performance to scale to many users and many developers. Also, you can leverage the huge Java community, strong IDE/tooling support, and tons of open source libraries.
4.

Cons:

1. not well known.
2. Java + Async. Play is built around async I/O, which means writing code that "executes later". Unlike Scala, Java lacks key language features, such as closures, to keep async code clean. There are patterns and tools that make it tolerable, but you end up with lots of anonymous inner classes. For streaming functionality (iteratees, enumeratees), you can't really use Java at all. Also worth mentioning is that lots of existing Java libraries are synchronous/blocking, so you have to be careful with which ones you use in a an async/non-blocking environment like Netty. However, if necessary, you can always configure Play's thread pool to use lots of threads and behave just like any other blocking server.
3. Memory Hog

sources https://www.playframework.com/ https://www.quora.com/What-are-the-pros-and-cons-of-the-Play-Framework-2-for-a-Java-developer