JAVA BACKEND RESEARCH

19-9-2017

Sjoerd Brauer
Fontys Venlo

Dear FSG1,

In this document 10 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?

1. Spring needs an additional framework to connect with the database.
2. Spring has an http communication package: org.springframework.http
3. Spring is well documented, see website in sources.
4. Many tutorials can be found: https://www.tutorialspoint.com/spring/

Why use it?
Spring is a big time endured. It's very powerful and can do many many things.

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 also create a restful interface which can be extended later on by other programmers.
6. You could also use spring security, so that only certain people have access to the rest interface.
7. Easy to set up OATH2.

Cons:

1. Steep learning curve.
2. Very big.

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?

1. Strut2 needs an additional framework to connect with the database.
2. Strut2 uses servlets for http conenction: org.apache.struts2.ServletActionContext
3. Strut2 is well documented, see website in sources.
4. tutorials can easily be found: https://www.tutorialspoint.com/struts_2

Why use it?
Strut 2 is a time endured framework. It's very powerful and can do many many things.

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?

1. Hibernate is specially made to connect with database, see source for more information.
2. Hibernate does not support HTTP protocol, another framework needs to be used.
3. Hibernate is well documented, see website in sources.
4. Tutorials can easily be found: https://www.tutorialspoint.com/hibernate/

Why use it?
Hibernate is used to make database connections easier. For simple use it's perfect.

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
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?

1. Akka supports PostgreSQL https://github.com/AkkaNetContrib/Akka.Persistence.PostgreSql
2. Akka supports HTTP protocol: http://doc.akka.io/docs/akka-http/current/scala/http/
3. Akka is well documented, see sources and previous point.
4. Tutorials can easily be found: http://doc.akka.io/docs/akka/2.0.1/intro/getting-started-first-java.html

Why use it?
Compared to the other frameworks akka really has a benefit over them with their asynchronous ways. Most useful in a multi threaded environment.

Pros:

1. Multi-threaded behavior without the use of low-level concurrency constructs like atomics or locks, relieving you from even thinking about memory visibility issues.
2. Transparent remote communication between systems and their components, relieving you from writing and maintaining difficult networking code.
3. A clustered, high-availability architecture that is elastic, scales in or out, on demand, enabling you to deliver a truly reactive system.
4. Does not run on Java JEE server.

Cons:

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?

1. Vert.x supports PostgreSQL connection: http://vertx.io/docs/vertx-sql-common/java/
2. Vert.x supports HTTP protocol: http://vertx.io/blog/some-rest-with-vert-x/
3. Vert.x is well documented see sources.
4. Many tutorials can be found: http://vertx.io/blog/posts/introduction-to-vertx.html

Why use it?
Like Akka it's scalable, concurrent, non-blocking but with the exception that it runs on the JVM

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

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?

1. JDBC is specifically designed to communicate with databases.
2. JDBC does not support HTTP protocol.
3. JDBC is well documented: https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html
4. Many tutorials can be found, see url of previous link.

Why use it?
JDBC is used to connect to databases. It has everything to do what you need.

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.

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?

1. Play framework supports PostgreSQL: https://stackoverflow.com/questions/24124789/install-postgresql-with-play-framework-driver-not-found-org-postgresql-drive
2. Play framework supports HTTP: https://www.playframework.com/documentation/2.5.x/HTTPServer
3. Play framework has good documentation: https://www.playframework.com/documentation
4. There are good tutorials, see url above.

Why use it?
Fast multithreaded play work that covers a large area.

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

Jersey
Jersey framework is more than the JAX-RS Reference Implementation. Jersey provides it's own API that extend the JAX-RS toolkit with additional features and utilities to further simplify RESTful service and client development. Jersey also exposes numerous extension SPIs so that developers may extend Jersey to best suit their needs.

Goals of Jersey project can be summarized in the following points:

Track the JAX-RS API and provide regular releases of production quality Reference Implementations that ships with GlassFish; Provide APIs to extend Jersey & Build a community of users and developers; and finally Make it easy to build RESTful Web services utilising Java and the Java Virtual Machine.

Supports all criteria?

1. Jersey can communicate with PostgreSQL but best in combination with another technology.
2. Jersey supports HTTP. see source.
3. Jersey has proper documentation see source.
4. Jersey has tutorials see source.

Why use it?
Jersey implements the JAX-RS api with extra features. Jersey makes it easier to create a restful service on any java application server.

1. Fast and cutting edge
2. Smooth JUnit integration
3. Supports true asynchronous connections

Cons:

1. Jersey 2.0+ uses a somewhat messy dependency injection implemetation
2. A large amount of online resources (3rd party) are related to Jersey 1.X making them unsuitable for Jersey 2.X

Sources
https://jersey.github.io/ https://stackoverflow.com/questions/7052152/why-use-jax-rs-jersey http://www.gajotres.net/best-available-java-restful-micro-frameworks/

spark framework
Spark - A micro framework for creating web applications in Kotlin and Java 8 with minimal effort

Supports all criteria?

1. Spark has a function to communicate with a PostgreSQL: https://sparktutorials.github.io/2015/04/29/spa and-sql2o.html
2. Spark supports HTTP see source.
3. Spark has documentation see source.
4. Spark has tutorials see source.

Why use it?
A small lightweight framework with one purpose, to communicate with front-end

1. Fast and lightweight
2. Excellent for rapid prototyping
3. Easy to setup
4. Most commonly used with AngularJS
5. Real micro-framework

Cons:

1. Documentation could be better, it is not intended for beginners
2. Not suitable for large projects
3. Small community

Sources
http://sparkjava.com/ http://www.gajotres.net/best-available-java-restful-micro-frameworks/

OrmLite
Object Relational Mapping Lite (ORM Lite) provides some simple, lightweight functionality for persisting Java objects to SQL databases while avoiding the complexity and overhead of more standard ORM packages.

Supports all criteria?

1. ORMLite is specifically designed to communicate with databases.
2. ORMLite does not support HTTP.
3. ORMLite has documentation see sources.
4. ORMLite has tutorials see sources.

Why use it?
ORMLite is a lightweight ORM framework. Easy and accessible to use.

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

Pros:

1. Supports complicated database operations
2. no need to remember to SQL queries
3. prefer for big size application

Cons:

1. unnecessarily increase size of application

Sources
http://ormlite.com/ https://stackoverflow.com/questions/25424679/ormlite-or-sqlite-in-android