TABLE OF CONTENTS

APPLE HTTP CLIENT

PROJECT NAME

The project is titled as Apple HTTP Client.

PURPOSE

Apple HTTP client is developed for making Http/Https client calls to Wikipedia and gather statistics for the data set.

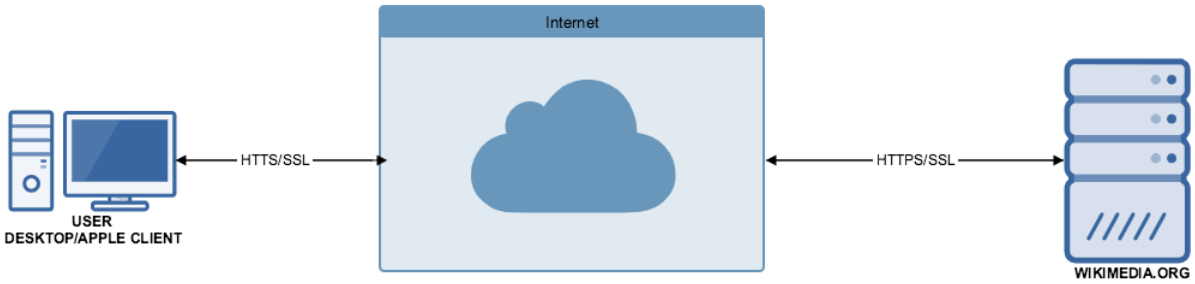
SCOPE

The scope of this document is to provide the technical design, details about design choices, edge cases, limitations, and provide implementation details.

ARCHITECTURE

Apple Client follows the client server architecture which is depicted as per the following diagram.

The Apple client is deployed at user's desktop or client machine, which communicates with Wikimedia.org using secure HTTPS/SSL communication via internet.



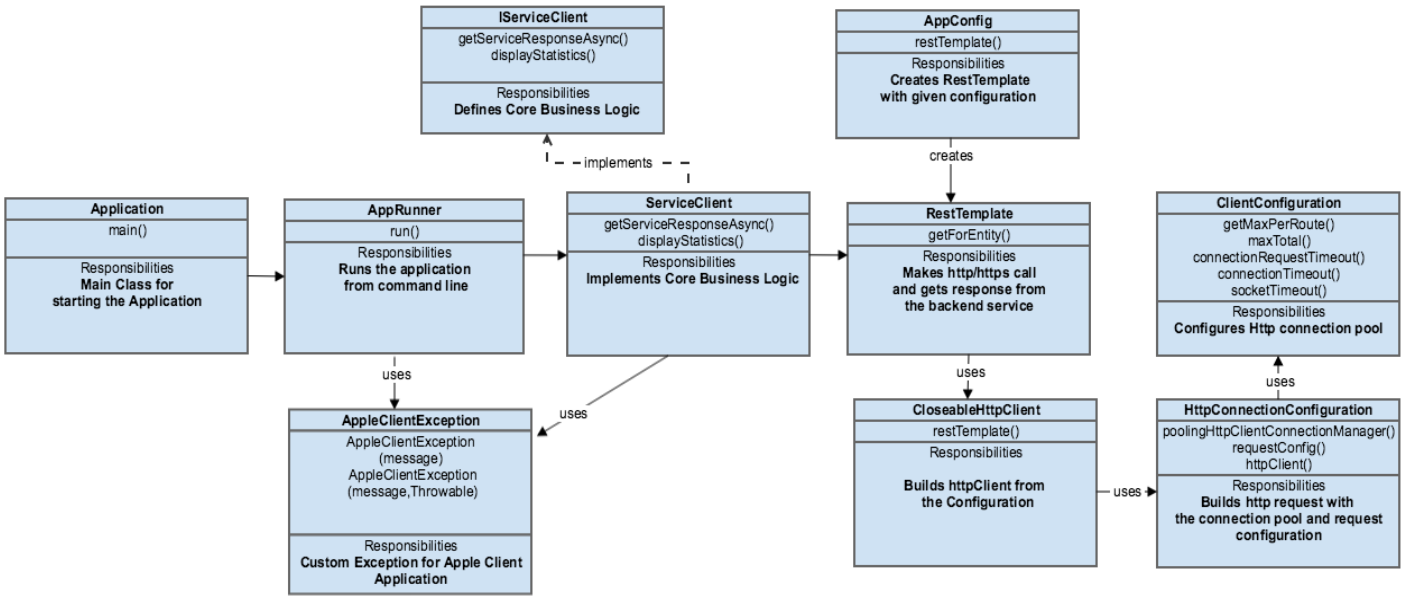
DESIGN

The following diagrams depict the design of the application using various diagrams – class, sequence, package, and deployment.

Class Diagram

The class diagram depicts the classes present in the application and their associations.

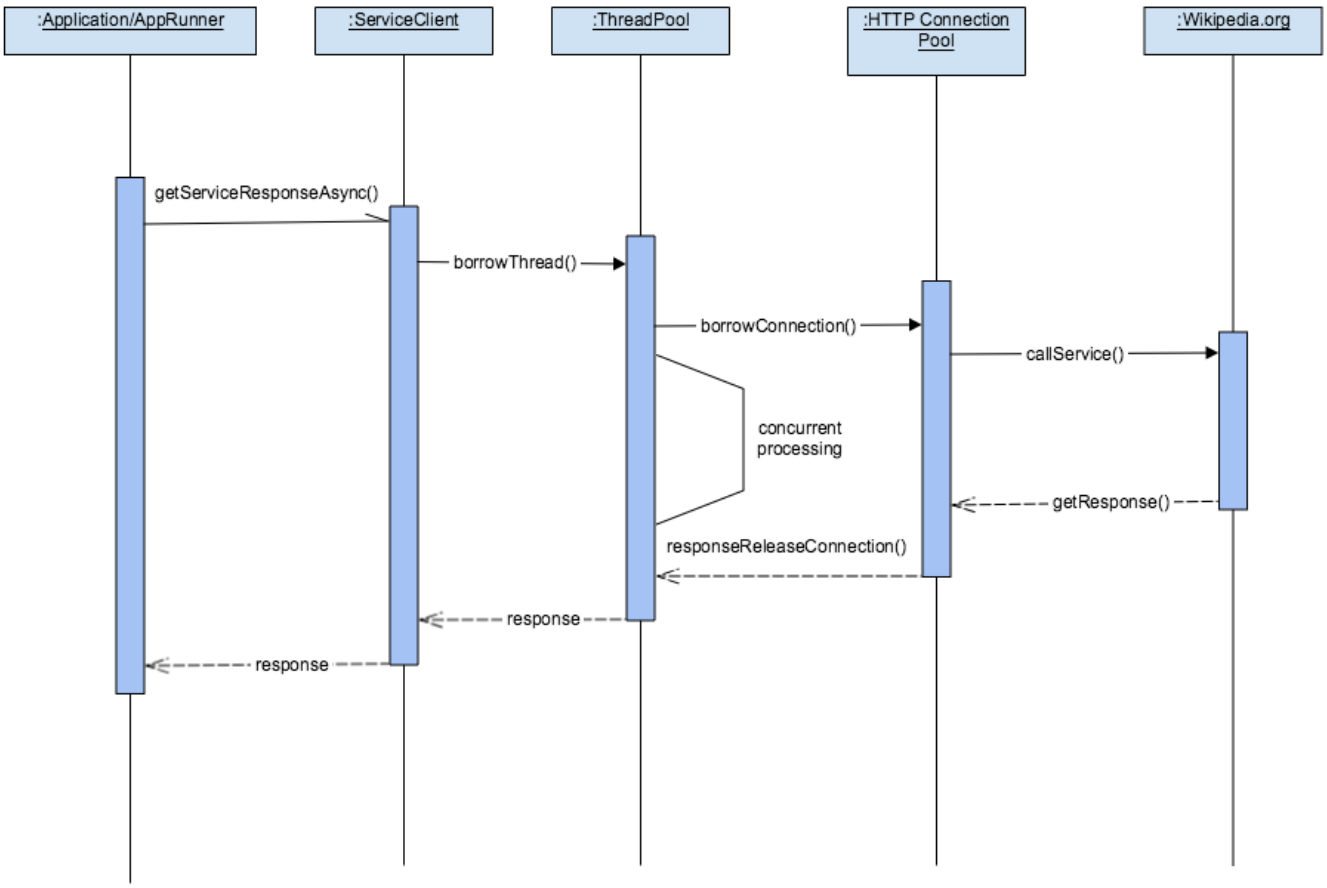
Application is the main class for Spring Application, which invokes AppRunner for command line processing. IServiceClient is business interface that defines core business logic methods, which are implemented using ServiceClient. AppRunner invokes ServiceClient business methods. RestTemplate was used for making Http calls to the backend service. RestTemplate was modified to use CloseableHttpClient with additional capabilities such as SSL, Retry, and Connection Pooling. The Http Client Configuration is defined in HttpConnectionConfiguration which gets the application level configuration properties from ClientConfiguration.



Sequence Diagram

The flow of the application is depicted using the following sequence diagram.

Application/AppRunner invokes ServiceClient with a batch of Jobs for processing, ServiceClient has business logic implementation which creates a Thread Pool using Spring's implementation of ThreadPoolExecutor implicitly. All the submitted jobs are processed asynchronously by the in-built ThreadPoolExecutor. The ThreadPoolExecutor configuration is defined inside the application.properties under section – TaskExecutionProperties such as core thread pool size, max thread pool size, and thread name prefix etc.,



Package Diagram

The package diagram shows the organization of the code in to different packages and the associations, interactions between the packages.

com.app.client – contains the base packages for the application from which the application is invoked.

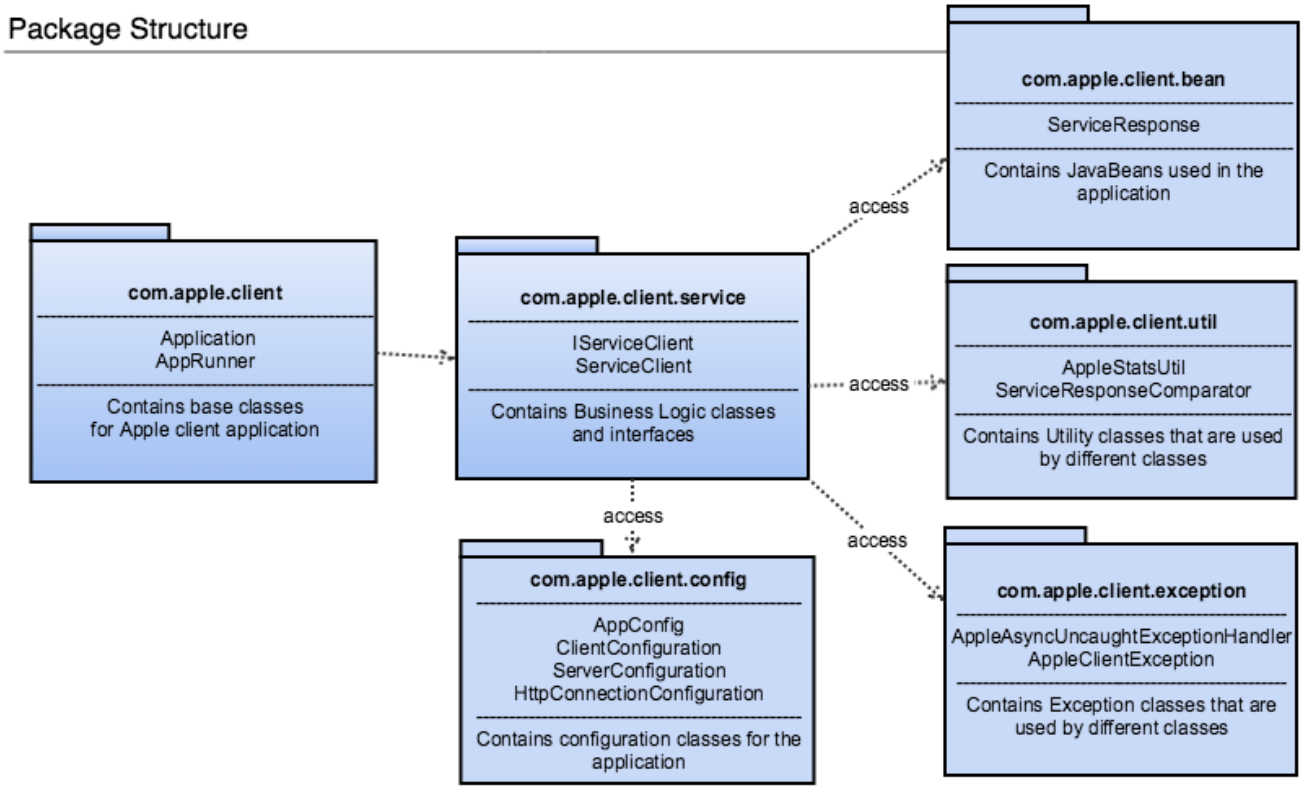
com.app.client.service – contains the business logic classes and interfaces.

com.app.client.config – contains the configuration classes for the application.

com.app.client.bean – contains the JavaBeans used in the application.

com.app.client.exception – contains exception handlers and exception classes.

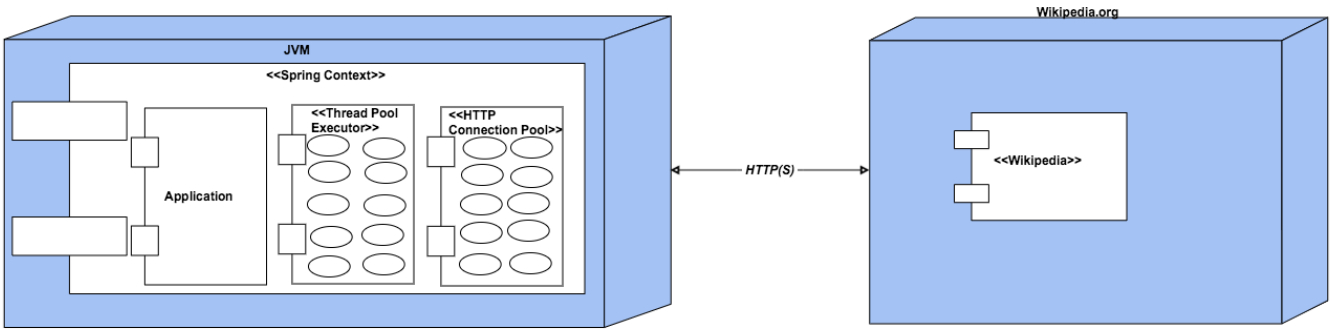
com.app.client.util – contains utility classes used in the application.



Deployment Diagram

The deployment diagram shows the deployment model for the application. The Spring Boot Application is deployed in a JVM at the client. The spring context holds the Thread Pool executor

and Http Connection pool that are required for the application to execute. On execution of the client application request/s are made to Wikipedia.org and the response are fetched and processed at the client.



QUESTIONS AND ANSWERS

How much time did you spend on this project?

I spent around 2 days on developing this project, that includes design, development, and unit testing.

What challenges did you face?

Faced minor issues with the configuration of the different tools, when testing the application with multiple JDK versions i.e., on JDK8 and JDK 10.

How would you improve the code if you had more time?

I would convert the code in to a toolkit or service so that others can benefit from the client I developed as part of this project., also the Apache HTTP client configuration, Thread Pool configuration are available to them beforehand. If given more time I would want to make the HTTP Client selection also configurable so that users can choose between different HTTP vendor implementations (e.g., Apache Http Client or OKHttpClient or Netflix ribbon client library etc.,).

Design Choices

I made the following design choices while developing this application.

a) Framework selection

Used Spring Boot as the Framework for development as it provides the following benefits.

- Reduces lots of development time and increases productivity.
- Avoids writing lots of boilerplate Code, Annotations and XML Configuration.
- Follows "Opinionated Defaults Configuration" Approach to reduce Developer effort.
- Provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
- Provides CLI (Command Line Interface) tool to develop and test Applications from command prompt very easily and quickly.
- Integrates well with Build tools like Maven and Gradle.
- Provides implementation for Thread Pool Executor which is configurable.
- Provides RestTemplate (HttpClient) which is configurable with the given choice of HTTP Vendor implementation (e.g. Apache, OK HTTP, Google Http Client).

b) Design patterns

Builder pattern to build the restTemplate with the given http client selection, and connection pooling.

The connection pooling and http client configuration is controlled from application.properties configuration file.

c) Application configuration (make the application configurable)

The connection pooling, and HTTP client configurations are read from the application.properties(configuration file).

The application.properties file provides the base configuration, this can be overwritten for different environments using files application-<environment-name>.properties.

c) Logging

Used Logging Framework for logging, the logs files are rolled after they reach the threshold.

The logging is configured from the resources/logback-spring.xml file.

The log file is located under project root folder/logs/

d) Build Framework

Used Maven with pom.xml file for building the Spring Framework application.

All the frameworks used in the application are controlled from the pom.xml

e) Unit Tests

Created Unit Tests for the application using Mockito, as Mockito lets us test the application without having to call the actual backend services. Objects are mocked, and behavior is verified using Mockito with JUnit.

The Unit Test classes are suffixed with ***Test** and are picked during the Test Phase of the Build. The execution can be enabled or disabled using the configuration flags in pom.xml.

f) Integration Tests

The Integration Test classes are suffixed with ***IT** and are picked during the Test Phase of the Build. The execution can be enabled or disabled using the configuration flags in pom.xml.

g) Code Coverage

Jacoco is used for code coverage. Once the build is executed, and the test phase is completed, the code coverage reports are available at the path - target/site/jacoco.

h) Javadoc

Javadoc are generated as part of the Build process and are available under target/apidocs.

Javadoc provide detailed information about the classes, methods, and their behavior.

i) Checkstyle

Used a check style xml configuration file and named it as checkstyle_apple.xml.

The code adheres to the checkstyle configuration. The checkstyle configuration can be controlled from the pom.xml file.

Known Limitations and Edge Cases.

#1 Edge case

The Wikipedia Service could go down, or could be under load, and may not respond within the Timeout then Timeout Exception may occur.

Mitigation Plan#

I made the Http Client to check the Wikipedia Service Status for a retry count of 3 times in case of initial failure response.

Each attempt will fall under 3 seconds SLA, this way We can reduce the timeout exceptions.

In case the Wikipedia Service does not respond even after 3 seconds, Will show an Information

message to the Customer saying, "Contact Customer Service Phone Number" or "Retry the service call at a later time, as there is an error in Processing"

#2 Edge case

Security is of paramount importance to the Application to prevent compromise of the system.

Mitigation Plan#

I took care of the Open Web Application Security Project Vulnerabilities (OWASP) as follows.

Encryption - The Channel is Secured between client, and server using SSL/HTTPS, so that any attacker like eavesdropper, Man in the middle attacks are mitigated.

Log Forging - Did not to write any user input into log files or to the console.

Took most the inputs from the application.properties(configuration file).