JAVA LIBRARIES YOU CAN'T AFFORD TO MISS

ANDRES ALMIRAY
@AALMIRAY

THE CHALLENGE

Write an application that consumes a REST API.

Components must be small and reusable.

Say no to boilerplate code.

Behavior should be easy to test.

THE LIBRARIES

PRODUCTION

Guice | Spring

OkHttp

Retrofit

JDeferred

RxJava | Reactor

MBassador

Lombok

Slf4j | Log4j2

TEST

JUnitParams

Mockito

Jukito

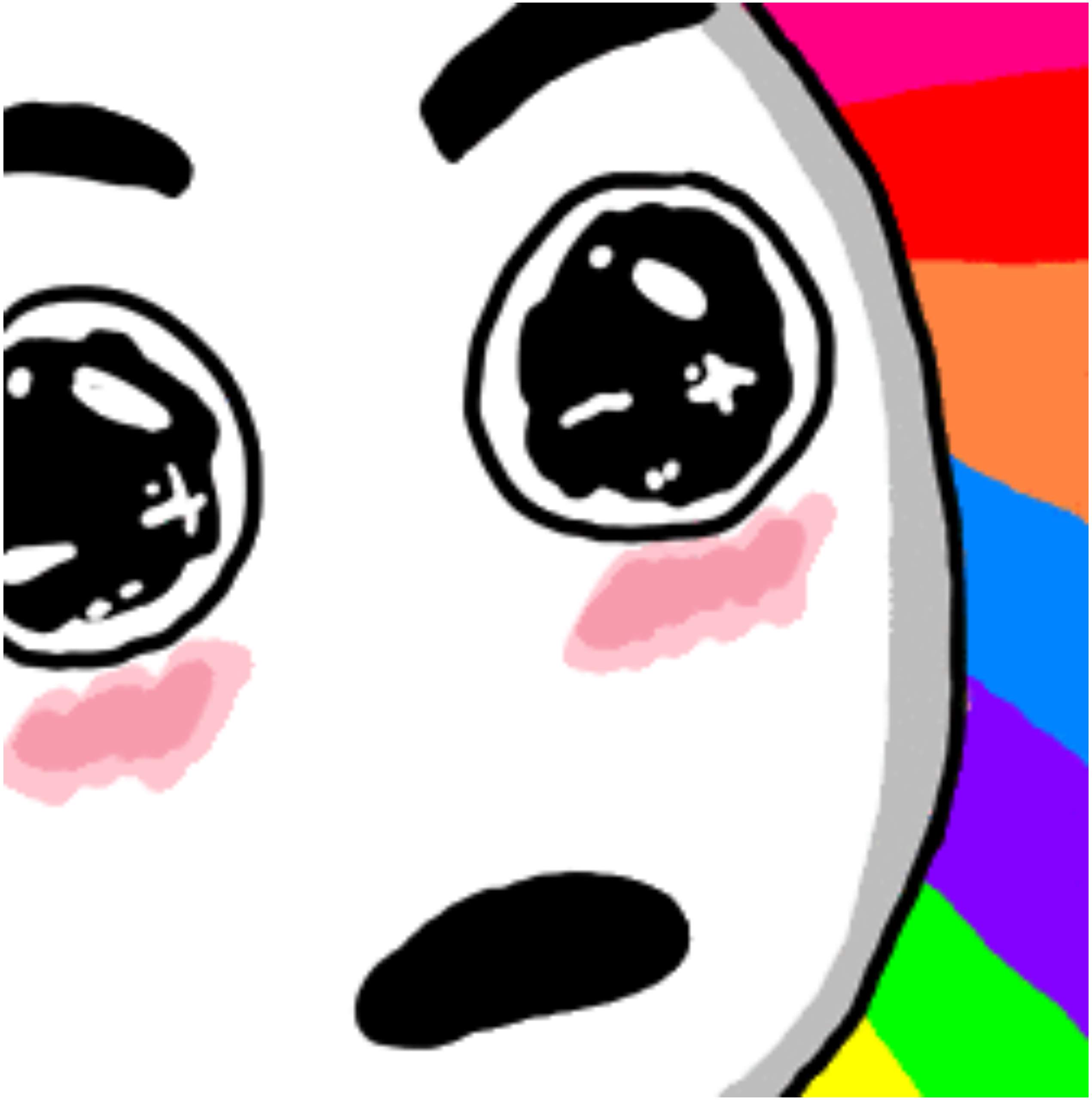
Awaitility

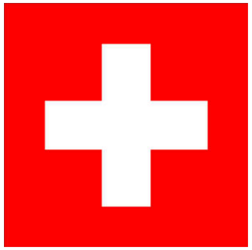
Spock

WireMock

DISCLAIMER

TM
open source

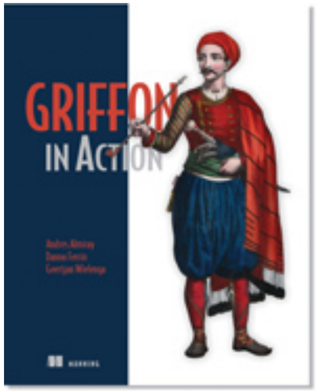




canoo

JAVA™
CHAMPIONS

HG
HACKERGARTEN



GRIFFON
IN ACTION

Andres Almiray
Danno Ferrin
Geertjan Wielenga

MANNING

THE CHALLENGE

Write an application that consumes a REST API.

Components must be small and reusable.

Say no to boilerplate code.

Behavior should be easy to test.

GITHUB API

Well documented REST API

Latest version located at

https://developer.github.com/v3/

We're interested in the repositories operation for now

QUERYING REPOSITORIES

API described at

https://developer.github.com/v3/repos/#list-organization-repositories

Given a query of the form

```
GET /orgs/${organization}/repos
```

QUERY RESULT

```
[
  {
  "id": 1296269,
  "owner": { /* omitted for brevity */ },
  "name": "Hello-World",
  "full_name": "octocat/Hello-World",
  "description": "This your first repo!",
  "html_url": "https://github.com/octocat/Hello-World",
  /* many other properties follow */
  },
  /* additional repositories if they exist */
]
```

WHAT WE’LL NEED

Dependency Injection

HTTP client & REST behavior

JSON mapping

Boilerplate buster

Handle concurrency

GET THE CODE

https://github.com/aalmiray/javatrove/

DEPENDENCY INJECTION

```
public interface Github {
    Promise<Collection<Repository>> repositories(String name);
}
```

```
public class AppController {
    @Inject private AppModel model;
    @Inject private Github github;

    public void loadRepositories() {
        model.setState(RUNNING);
        github.repositories(model.getOrganization())
            .done(model.getRepositories()::addAll)
            .fail(this::handleException)
            .always((state, resolved, rejected) -> model.setState(READY));
    }
}
```

Guice - https://github.com/google/guice

```
Injector injector = Guice.createInjector(new AbstractModule() {
    @Override
    protected void configure() {
        bind(Github.class)
            .to(GithubImpl.class)
            .in(Singleton.class);

        bind(AppModel.class)
            .in(Singleton.class);

        bind(GithubAPI.class)
            .toProvider(GithubAPIProvider.class)
            .in(Singleton.class);

        // additional bindings …
    }
});
```

Guice - https://github.com/google/guice

- Reference implementation for JSR-330.
- Can bind types, instances, constant values.
- Can provide lazily evaluated instances, i.e, providers.
- Extensible lifecycle.

BONUS

Guava - https://github.com/google/guava

- New Collections:
  - MultiSet
  - BiMap
  - MultiMap
  - Table
- Utility classes for Collections
- Utility classes for String
- Caches
- Reflection
- I/O
- Functional programming support (JDK6+)

Spring - http://projects.spring.io/spring-framework

- More than just dependency injection
- Supports JSR-330
- Assertions
- MessageSource + MessageFormat
- Serialization
- JDBC, JPA
- JMX
- Validation
- Scheduling
- Testing

REDUCING BOILERPLATE CODE

Lombok - https://projectlombok.org

```
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.Builder;
import lombok.Data;
import lombok.Setter;

@Data
@JsonIgnoreProperties(ignoreUnknown = true)
public class Repository implements Comparable<Repository> {
    private String name;
    private String description;
    @Setter(onMethod = @__({@JsonProperty("full_name")}))
    private String fullName;
    @Setter(onMethod = @__({@JsonProperty("html_url")}))
    private String htmlUrl;

    // continued
```

Lombok - https://projectlombok.org

```
    // continued

    @Builder
    public static Repository build(String name, String fullName, String
description, String htmlUrl) {
        Repository repository = new Repository();
        repository.name = name;
        repository.fullName = fullName;
        repository.description = description;
        repository.htmlUrl = htmlUrl;
        return repository;
    }
}
```

Lombok - https://projectlombok.org

```
public class ApplicationEvent {
}


@lombok.Data
@lombok.EqualsAndHashCode(callSuper = true)
@lombok.ToString(callSuper = true)
public class NewInstanceEvent extends ApplicationEvent {
    @javax.annotation.Nonnull
    private final Object instance;
}
```

Lombok - https://projectlombok.org

- Reduce boilerplate source code by generating bytecode.
- Relies on APT (Annotation Processing Tool).
- Extensible but not for the faint of heart.
- Common usages already covered, i.e, POJOs, builders.
- Don’t forget to enable annotation processing in your IDE!

BEHAVIOR

SLF4J - http://www.slf4j.org

- Wraps all other logging frameworks:
  - java.util.logging
  - Apache Commons Logging
  - Apache Log4j
- Provides varargs methods

HTTP

How many options are out there to build an HTTP client?

How many ways are there to build a REST client?

OkHttp - http://square.github.io/okhttp

```
public static final MediaType JSON
    = MediaType.parse("application/json; charset=utf-8");

OkHttpClient client = new OkHttpClient();

String post(String url, String json) throws IOException {
  RequestBody body = RequestBody.create(JSON, json);
  Request request = new Request.Builder()
      .url(url)
      .post(body)
      .build();
  Response response = client.newCall(request).execute();
  return response.body().string();
}
```

OkHttp - http://square.github.io/okhttp

- Basic HTTP/HTTP2 Client API.
- Configuration is extensible via factory classes.
- HTTP lifecycle can be decorated via interceptors.

ADDING JAVA ON TOP OF HTTP

What steps do we must ensure in order to build an HTTP/REST client?

How should HTTP error codes be handled?

How to handle connection errors?

Parsing results in format X or Y.

Retrofit - http://square.github.io/retrofit

```
public interface GithubAPI {
    @GET("/orgs/{name}/repos")
    Call<List<Repository>> repositories(@Path("name") String name);

    @GET
    Call<List<Repository>>> repositoriesPaginate(@Url String url);
}
```

Retrofit - http://square.github.io/retrofit

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com")
    .addConverterFactory(
        JacksonConverterFactory.create(objectMapper))
    .build();

return retrofit.create(GithubAPI.class);


githubApi.repositories("foo");
```

Retrofit - http://square.github.io/retrofit

- Wraps REST calls with Java interfaces.
- Extensible via factories.
- Relies on OkHttp.

MULTI-THREADED CODE

The golden rule of UI programming:

- Everything related to UI must be executed inside the UI thread (read/write UI properties, paint/repaint, etc).
- Everything else must be executed outside of the UI thread.

JDeferred - http://jdeferred.org

```
public interface Github {
    Promise<Collection<Repository>, Throwable, Void> repositories(String name);
}
```

JDeferred - http://jdeferred.org

```
public class GithubImpl implements Github {
    @Inject private GithubAPI api;
    @Inject private DeferredManager deferredManager;

    @Override
    public Promise<Collection<Repository>, Throwable, Void> repositories(final
String name) {
        return deferredManager.when(() -> {
            Response<List<Repository>> response = api.repositories(name).execute();
            if (response.isSuccess()) { return response.body(); }
            throw new IllegalStateException(response.message());
        });
    }
}
```

JDeferred - http://jdeferred.org

```
model.setState(RUNNING);
int limit = model.getLimit();
limit = limit > 0 ? limit : Integer.MAX_VALUE;

Promise<Collection<Repository>, Throwable, Repository> promise =
github.repositories(model.getOrganization(), limit);

promise.progress(model.getRepositories()::add)
    .fail(Throwable::printStackTrace)
    .always((state, resolved, rejected) -> model.setState(READY));
```

JDeferred - http://jdeferred.org

- Delivers the concept of Promises
- Promises can be chained
- Java8 friendly, i.e, lambda expressions can be used
- One shot execution.

REACTIVE PROGRAMMING

It’s time to embrace a new paradigm.

Reactive programming is a new name for old and well-known concepts:

events and streams

RxJava - http://reactivex.io

```
Observable<Repository> observable =
github.repositories(model.getOrganization());
if (model.getLimit() > 0) {
    observable = observable.take(model.getLimit());
}

Subscription subscription = observable
    .timeout(10, TimeUnit.SECONDS)
    .doOnSubscribe(() -> model.setState(RUNNING))
    .doOnTerminate(() -> model.setState(READY))
    .subscribeOn(Schedulers.io())
    .subscribe(
        model.getRepositories()::add,
        Throwable::printStackTrace);
model.setSubscription(subscription);
```

Retrofit + RxJava

```
public interface GithubAPI {
    @GET("/orgs/{name}/repos")
    Observable<Response<List<Repository>>> repositories(@Path("name") String
name);

    @GET
    Observable<Response<List<Repository>>> repositoriesPaginate(@Url String url);
}
```

Retrofit + RxJava

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com")
    .addConverterFactory(JacksonConverterFactory.create(objectMapper))
    .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
    .build();

return retrofit.create(GithubAPI.class);
```

Retrofit + RxJava

```
// concatenating multiple results into a single Observable

public Observable<Repository> repositories(String organization) {
    requireNonNull(organization, "Argument 'organization' must not be blank");

    return paginatedObservable(
        () -> {
            LOG.info("Querying /orgs/{}/repos", organization);
            return api.repositories(organization);
        },
        (Links links) -> {
            String next = links.next();
            LOG.info("Querying {}", next);
            return api.repositoriesPaginate(next);
        });
}
```

Retrofit + RxJava

```
// concatenating multiple results into a single Observable

private static <T> Observable<T> paginatedObservable(FirstPageSupplier<T>
firstPage, NextPageSupplier<T> nextPage) {
    return processPage(nextPage, firstPage.get());
}

private static <T> Observable<T> processPage(NextPageSupplier<T> supplier,
Observable<Response<List<T>>> items) {
    return items.flatMap(response -> {
        Links links = Links.of(response.headers().get("Link"));
        Observable<T> currentPage = Observable.from(response.body());
        if (links.hasNext()) {
            return currentPage.concatWith(processPage(supplier,
supplier.get(links)));
        }
        return currentPage;
    });
}
```

RxJava - http://reactivex.io

- Implements the Observable pattern.
- Delivers dozens of operations out of the box, e.g. zip, reduce, concat.
- Supports backpressure, i.e, when the data producer is faster than the data consumer.

COMPONENT COMMUNICATION

How do you keep two unrelated components communicated with one another?

How do you push data down the stream without forcing publishers to wait for consumers?

MBassador - https://github.com/bennidi/mbassador

```
// event publishing

@Inject private ApplicationEventBus eventBus;

model.setSubscription(observable
    .timeout(10, TimeUnit.SECONDS)
    .doOnSubscribe(() -> model.setState(RUNNING))
    .doOnTerminate(() -> model.setState(READY))
    .doOnError(throwable -> eventBus.publishAsync(new ThrowableEvent(throwable)))
    .subscribeOn(Schedulers.io())
    .subscribe(model.getRepositories()::add));
```

MBassador - https://github.com/bennidi/mbassador

```
// event consuming

import net.engio.mbassy.listener.Handler;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.inject.Inject;

public class ApplicationEventHandler {
    @Inject private ApplicationEventBus eventBus;

    @PostConstruct private void init()   { eventBus.subscribe(this); }
    @PreDestroy private void destroy() { eventBus.unsubscribe(this); }

    @Handler
    public void handleThrowable(ThrowableEvent event) {
        // handle event here !!
    }
}
```

MBassador - https://github.com/bennidi/mbassador

- Configurable event bus based on annotations.
- Faster implementation than Guava’s event bus.
- https://github.com/bennidi/eventbus-performance
- NOTE: project is no longer actively maintained, fixes and features will be slow to come.

TESTING

JUnitParams - https://github.com/Pragmatists/JUnitParams

```
@RunWith(JUnitParamsRunner.class)
public class SampleServiceTest {
    @Test
    @Parameters({",Howdy stranger!",
                        "Test, Hello Test"})
    public void sayHello(String input, String output) {
        // given:
        SampleService service = new SampleService();

        // expect:
        assertThat(service.sayHello(input), equalTo(output));
    }
}
```

JUnitParams - https://github.com/Pragmatists/JUnitParams

- Parameterize multiple methods with different argument cardinality.
- Different data provider strategies.

Mockito - http://mockito.org

```
@Test @Parameters({",Howdy stranger!", "Test, Hello Test"})
public void sayHelloAction(String input, String output) {
    // given:
    SampleController controller = new SampleController();
    controller.setModel(new SampleModel());
    controller.setService(mock(SampleService.class));

     // expectations
    when(controller.getService().sayHello(input)).thenReturn(output);

    // when:
    controller.getModel().setInput(input);
    controller.sayHello();

    // then:
    assertThat(controller.getModel().getOutput(), equalTo(output));
    verify(controller.getService(), only()).sayHello(input);
}
```

Mockito - http://mockito.org

- Fluid DSL based on static methods.
- Provides support for Stubs, Mocks, and Spies.
- Mock interfaces, abstract classes, and concrete classes.

Jukito - https://github.com/ArcBees/Jukito

```
@RunWith(JukitoRunner.class)
public class SampleControllerJukitoTest {
    @Inject private SampleController controller;

    @Before
    public void setupMocks(SampleService sampleService) {
        when(sampleService.sayHello("Test")).thenReturn("Hello Test");
    }

    @Test
    public void sayHelloAction() {
        controller.setModel(new SampleModel());
        controller.getModel().setInput("Test");
        controller.sayHello();
        assertThat(controller.getModel().getOutput(),
                        equalTo("Hello Test"));
        verify(controller.getService(), only()).sayHello("Test");
    }
}
```

Jukito - https://github.com/ArcBees/Jukito

- Combines JUnit, Guice, and Mockito
- Bind multiple values to the same source type.
- Can be used to parameterize test methods.

Spock- http://spockframework.org

```
@spock.lang.Unroll
class SampleControllerSpec extends spock.lang.Specification {
    def "Invoke say hello with #input results in #output"() {
        given:
            SampleController controller = new SampleController()
            controller.model = new SampleModel()
            controller.service = Mock(SampleService) {
                sayHello(input) >> output
            }
        when:
            controller.model.input = input
            controller.sayHello()
        then:
            controller.model.output == output
        where:
            input   << ['', 'Test']
            output << ['Howdy, stranger!', 'Hello Test']
    }
}
```

Spock- http://spockframework.org

- Groovy based DSL.
- Parameterize multiple methods with different argument cardinality.
- Parameterize test method names.
- JUnit friendly (can use extensions and rules).

Awaitility - https://github.com/awaitility/awaitility

```
@Test
public void happyPath(Github github) {
    // given:
    Collection<Repository> repositories = createSampleRepositories();
    when(github.repositories(ORGANIZATION))
        .thenReturn(Observable.from(repositories));

    // when:
    model.setOrganization(ORGANIZATION);
    controller.load();
    await().timeout(2, SECONDS).until(model::getState, equalTo(State.READY));

    // then:
    assertThat(model.getRepositories(), hasSize(10));
    assertThat(model.getRepositories(), equalTo(repositories));
    verify(github, only()).repositories(ORGANIZATION);
}
```

Awaitility - https://github.com/awaitility/awaitility

- DSL for testing multi-threaded code.
- Extensions available for Java8, Groovy, and Scala.
- Conditions can be customized with Hamcrest matchers.

WireMock- http://wiremock.org/

```
import static com.github.tomakehurst.wiremock.client.WireMock.*;

String nextUrl = "/organizations/1/repos?page=2";
List<Repository> repositories = createSampleRepositories();
stubFor(get(urlEqualTo("/orgs/" + ORGANIZATION + "/repos"))
    .willReturn(aResponse()
        .withStatus(200)
        .withHeader("Content-Type", "text/json")
        .withHeader("Link", "<http://localhost:8080" + nextUrl + ">; rel=\"next\"")
        .withBody(repositoriesAsJSON(repositories.subList(0, 5), objectMapper))));
stubFor(get(urlEqualTo(nextUrl ))
    .willReturn(aResponse()
        .withStatus(200)
        .withHeader("Content-Type", "text/json")
        .withBody(repositoriesAsJSON(repositories.subList(5, 10), objectMapper))));
```

WireMock- http://wiremock.org/

- Supply custom HTTP response payloads.
- DSL for matching HTTP requests.
- Supports record and playback.

KEEP
CALM
AND
OPEN
SOURCE

THANK YOU!

ANDRES ALMIRAY
@AALMIRAY