captain-miao / **AndroidAgeraTutorial**

Home

yanlu edited this page on 11 Jun 2016 · 10 revisions

Agera Wiki 中文版

Google Agera Wiki

> https://github.com/google/agera/wiki

Agera(瑞典文的意思是"采取行动")是一个超轻量级的Android库，帮助Android应用中有生命周期的组件(比如:Activities)或者组件中的对象(比如:Views)预准备数据。通过加入函数式响应式编程，Agera可以在 *什么时机*, *什么线程* 和 *什么数据* 层面上更清晰的分离数据处理流程，并且使用一个接近自然语言的单个表达式就能编写一个复杂的异步流。

下面是一个Agera一些功能的示范例子，此文档(和javadoc一起)将介绍Agera各个部分是如何工作的。

```
public class AgeraActivity extends Activity
    implements Receiver<Bitmap>, Updatable {
  private static final ExecutorService NETWORK_EXECUTOR =
      newSingleThreadExecutor();
  private static final ExecutorService DECODE_EXECUTOR =
      newSingleThreadExecutor();
  private static final String BACKGROUND_BASE_URL =
      "http://www.gravatar.com/avatar/4df6f4fe5976df17deeea19443d4429d?s=";

  private Repository<Result<Bitmap>> background;
  private ImageView backgroundView;

  @Override
  protected void onCreate(final Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // Set the content view
    setContentView(R.layout.activity_main);

    // Find the background view
    backgroundView = (ImageView) findViewById(R.id.background);

    // 创建一个repository： 包括一个图片请求。Initially
    // 初始值：absent, 配置通过网络获取屏幕大小的图片。
     background = repositoryWithInitialValue(Result.<Bitmap>absent())
        .observe() // 在这个例子中不需要刷新背景图，也就是不需要被观察者
        .onUpdatesPerLoop() // 在这个例子中每次都更新
        .getFrom(new Supplier<HttpRequest>() {
          @NonNull
          @Override
          public HttpRequest get() {
            DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
            int size = Math.max(displayMetrics.heightPixels,
                displayMetrics.widthPixels);
```

Pages 11

Agera explained

- Reactive programming
  - Push event, pull data
- Observables and updatables
  - Activation lifecycle and event chain
  - UI lifecycle
  - Threading
- Repositories
  - Simple repositories
  - Complex repositories
- Compiled repositories
  - When, where, what
  - Data processing flow
  - Operators
  - Attempts and Result
  - Asynchronous programming
- Compiled functions
- Reservoirs and parallelism
- Custom observables
  - Proxy observables
  - BaseObservable
  - UpdateDispatcher
- Incrementally Agerifying legacy code
  - Upgrading legacy observer pattern
  - Exposing synchronous operations as repositories
  - Wrapping asynchronous calls in repositories

Clone this wiki locally

`https://github.com/capta:`

```
            return httpGetRequest(BACKGROUND_BASE_URL + size)
                .compile();
          }
        }) // 实现带有显示大小参数的Http请求
        .goTo(NETWORK_EXECUTOR) // 设置网络请求线程
        .attemptTransform(httpFunction())
        .orSkip() // 设置http请求，如果失败，跳过下面流程
        .goTo(DECODE_EXECUTOR) // 设置解码线程
        .thenTransform(new Function<HttpResponse, Result<Bitmap>>() {
          @NonNull
          @Override
          public Result<Bitmap> apply(@NonNull HttpResponse response) {
            byte[] body = response.getBody();
            return absentIfNull(decodeByteArray(body, 0, body.length));
          }
        }) // 解码图片, 如果失败返回absent
        .onDeactivation(SEND_INTERRUPT) // 中断线程
        .compile(); // 创建repository
  }

  @Override
  protected void onResume() {
    super.onResume();
    // 注册观察者，激活流程
    background.addUpdatable(this);
  }

  @Override
  protected void onPause() {
    super.onPause();
    // 注销观察者，去激活流程
    background.removeUpdatable(this);
  }

  @Override
  public void update() {
    // repository更新时 调用
    // 如果bitmap有效, 发送到下面的accept()方法
    background.get().ifSucceededSendTo(this);
  }

  @Override
  public void accept(@NonNull Bitmap background) {
    // 设置背景图片
    backgroundView.setImageBitmap(background);
  }
}
```

专有名词解释

> 由于一些专有的单词，翻译不出来，在这里统一解释，以便后文的理解。

Updatable(Observer)

> 翻译为：观察者
> 用途：接收事件通知，更新数据。说明：观察者模式中的Observer，在Agera中使用Updatable

```
public interface Updatable {
  void update();
}
```

Observable

> 翻译为：被观察者、事件源
> 用途：作为事件源，通知观察者更新数据，可以注册、注销观察者。说明：当事件发生的时候，调用dispatchUpdate()通知观察者。

```
public interface Observable {
  void addUpdatable(@NonNull Updatable updatable);
  void removeUpdatable(@NonNull Updatable updatable);
}
```

Supplier

> 翻译为：数据提供者
> 用途：get()新数据。

```
public interface Supplier<T> {
  @NonNull
  T get();
}
```

Repository

> 翻译为：Repository、数据仓库
> 用途：接收事件源、提供数据源的结合体。

```
public interface Repository<T> extends Observable, Supplier<T> {
}
```

Reactive programming

> 翻译为：响应式编程
> 说明：一种面向数据流和变化传播的编程范式。这意味着在编程语言中很方便地表达静态或动态的数据流，而相关的计算模型会自动将变化的值通过数据流进行传播。例如：在命令式编程环境中，a = b + c 表示将表达式的结果赋给a，而之后改变b或c的值不会影响a。但在响应式编程中，a的值会随着b或c的更新而更新。

Activation lifecycle

> 翻译为：激活生命周期
> 用途：注册观察者：数据仓库变为激活状态；注销观察者：数据仓库变为非激活状态。

Active

> 翻译为：激活状态、活动状态

Inactive

> 翻译为：非激活状态、非活动状态

worker Looper thread

> 翻译为：有Looper的线程
> 说明：Agera 内部消息是通过Handler来完成的，所有线程需要有Looper，更多参考：Android-Handler。

Data Processing Flow

> 翻译为：数据处理流程

When, Where, What

> 翻译为：什么时候(时机)，什么线程上[执行]，什么数据

compiled repository

> 翻译为：编译数据仓库