bit.ly/2f9NVnz

These slides, so you can follow along!

Friendly Architecture

Building performant apps

Mike Nakhimovich

Brian Plummer

What are performant apps

Fast to load

Work offline

Testable

Scalable

Doesn’t crash or run out of memory

Let’s Build something

Reddit Client

Hits API

Show Top 50 posts

- Images
- Title

How?

What we need to make

Network Client

Data Models

Memory & Disk Caching

Presenters - Good Architecture and Separation of Concerns

RecyclerView

Image loader

Don’t reinvent the wheel

Retrofit - network client

Immutables - data modeling

Picasso - image loading

Dagger - Dependency Injection

RxJava - Concurrency and data transformation

First we show you how

Then we all build something together

bit.ly/2f9NVnz

These slides, so you can follow along!

The finished product: bit.ly/2fqxbsY

friendlyrobotnyc/FriendlyDemo

Today’s starting point: bit.ly/2fdE9E8

```
git checkout startingPoint
```

reddit.com/r/aww.json

Data Modeling

With Immutables

Post.java

```
@Nullable
public abstract Preview preview();
public abstract String title();
public abstract String url();

@Nullable
public abstract Integer height();

@Nullable
public abstract Integer width();
```

Post.java Helper Method

```
@Value.Derived

public Optional<Image> nestedThumbnail() {

   if (preview() == null || preview().images() == null ||
preview().images().get(0).source() == null)

      return Optional.absent();

   return Optional.of(preview().images().get(0).source());

}
```

Data Fetching

Retrofit Networking Client

Define an interface for endpoint

```
@GET("r/{subredditName}/new/.json")
Observable<RedditData> fetchSubreddit(
                               @Path("subredditName") String subredditName,
                               @Query("limit") String limit,
                               @Header("fresh") String fresh);
```

Make a store

This is where our memory cache lives

Create Your Store

```java
public class RedditStore extends BaseStore<RedditData, String> {


}
```

Hook Store into API

```
@Inject
Lazy<Api> api;

@Inject
public RedditStore() {
}

protected Observable<RedditData> fetch(String redditName,String forceNetwork) {
   return api.get().fetchSubreddit(redditName, "50", forceNetwork);
}}
```

Whats with the Force Network Param?

Cache Interceptor Demo

Create OKHTTP Instance

With built in disk caching

NetworkModule.java

```java
@Provides @Singleton
public OkHttpClient providClient(@ClientCache File cacheDir, CacheInterceptor interceptor) {
  final OkHttpClient.Builder okHttpBuilder = new OkHttpClient.Builder();
  Cache cache = new Cache(cacheDir, 20 * 1024 * 1024);
  okHttpBuilder.cache(cache);
  okHttpBuilder.interceptors().add(interceptor); //needed for force network
  okHttpBuilder.networkInterceptors().add(interceptor); //needed for offline mode
  return okHttpBuilder.build();
}
```

Network Module continued

```
@Singleton @Provides
File provideCacheFile(Application context)    {
  return new File(context.getCacheDir(), "cache_file");
}
```

Create Picasso Gson & API Instance

Inside of App Module

Gson

```
@Provides @Singleton
Gson provideGson() {
   return new GsonBuilder()
           .registerTypeAdapterFactory(new GsonAdaptersModel())
           .registerTypeAdapter(Date.class, new DateDeserializer())
           .setDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'")
           .create();
}
```

Api implementation (Retrofit)

```
@Provides @Singleton
public Api provideApi(OkHttpClient okHttpClient, Gson gson) {
  return new Retrofit.Builder()
          .baseUrl("http://reddit.com/")
          .client(okHttpClient)
          .addConverterFactory(GsonConverterFactory.create(gson))
          .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
          .build()
          .create(Api.class);
}
```

Async Image Loader (Picasso)

```
@Provides @Singleton
public Picasso providePicasso(Application application, OkHttpClient okHttpClient)
{
   return new Picasso.Builder(application)
           .downloader(new OkHttp3Downloader(okHttpClient))
           .build();
}
```

BaseActivity.java

```
protected ActivityComponent activityComponent;

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(getLayout());
  setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
}
```

BaseActivity.java

```java
public ActivityComponent getActivityComponent() {
   if (activityComponent == null) {
       activityComponent = ActivityComponentFactory.create(this);
   }
   return activityComponent;
}

protected abstract int getLayout();
```

RedditActivity.java

```
protected int getLayout() {
  return R.layout.activity_main;
}
```

activity_main.xml

```
<android.support.design.widget.CoordinatorLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:fitsSystemWindows="true">

</android.support.design.widget.CoordinatorLayout>
```

activity_main.xml

```xml
<android.support.design.widget.AppBarLayout
  android:layout_height="wrap_content"
  android:layout_width="match_parent"
  android:theme="@style/AppTheme.AppBarOverlay">
  <android.support.v7.widget.Toolbar
     android:id="@+id/toolbar"
     android:layout_width="match_parent"
     android:layout_height="?attr/actionBarSize"
     android:background="?attr/colorPrimary"
     app:popupTheme="@style/AppTheme.PopupOverlay"
     app:layout_scrollFlags="scroll|enterAlways"/>
</android.support.design.widget.AppBarLayout>

<include layout="@layout/content_main"/>
```

content_main.xml

```xml
<nyc.friendlyrobot.demo.ui.reddit.RedditRecyclerView
  android:id="@+id/postRecyclerView"
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@color/gray70"/>
```

RedditRecyclerView.java

```java
@Inject
PostAdapter postAdapter;

@Inject
RedditPresenter presenter;
```

RedditRecyclerView.java

```
public RedditRecyclerView(Context context) {
   this(context, null);
}
public RedditRecyclerView(Context context, AttributeSet attrs) {
   this(context, attrs, 0);
}
public RedditRecyclerView(Context context, AttributeSet attrs, int defStyle) {
   super(context, attrs, defStyle);
   ((BaseActivity) context)
           .getActivityComponent()
           .inject(this);
}
```

RedditRecyclerView.java

```
@Override
protected void onFinishInflate() {
  super.onFinishInflate();
  presenter.attachView(this);
  presenter.loadPosts();
  setLayoutmanager();
  setAdapter(postAdapter);
}
```

RedditRecyclerView.java

```java
@Override
public void showPosts(List<Post> posts) {
  postAdapter.setPosts(posts);
}
@Override
public void showError() {
  // do something
}
@Override
protected void onDetachedFromWindow() {
  super.onDetachedFromWindow();
  presenter.detachView();
}
```

Row layouts

PostViewHolder.java

```
public PostViewHolder(View itemView) {
   super(itemView);
   performInjection(itemView);
   findViews(itemView);
   setMaxDimensions(itemView);
}
```

PostViewHolder.java

```
public void onBind(Post article) {
   title.setText(article.title());
   if (article.nestedThumbnail().isPresent()) {
       showImage(article);
   }
}
```

PostViewHolder.java

```
private void showImage(Post article) {
   Image image = transformPostToImage(article);
   BitmapTransform bitmapTransform =
                                  new BitmapTransform(maxWidth, maxHeight, image);
   configureLayoutItems(bitmapTransform);
   Picasso.with(itemView.getContext())
          .load(image.url()).transform(bitmapTransform)
          .resize(bitmapTransform.targetWidth, bitmapTransform.targetHeight)
          .centerInside().placeholder(R.color.gray80)
          .into(thumbnail);
}
```

RedditMVPView.java

```
void showError();

void showPosts(List<Post> posts);
```

RedditPresenter.java

```java
private final CompositeSubscription compositeSubscription;
private final RedditStore redditStore;
@Inject
public RedditPresenter(RedditStore redditStore) {
   this.redditStore = redditStore;
   compositeSubscription = new CompositeSubscription();
}
@Override
public void detachView() {
   compositeSubscription.clear();
   super.detachView();
}
```

RedditPresenter.java

```
public void loadPosts() {
   checkViewAttached();
   compositeSubscription.add(redditStore.get("aww")
           .flatMap(this::sanitizeData)
           .toList()
           .compose(FriendlyScheduler.schedule())
           .subscribe(posts -> getMvpView().showPosts(posts),
                   throwable -> getMvpView().showError()));
}
```

RedditPresenter.java

```
@NonNull
private Observable<Post> sanitizeData(RedditData redditData) {
  return Observable.from(redditData.data().children())
          .map(Children::data)
          .filter(post -> post.nestedThumbnail().isPresent());
}
```