Szybciej (pisać)
Łatwiej (czytać)
Prościej (utrzymywać)

Marcin Wąsowski
Amsterdam Standard Sp. z o.o.

Co dostajemy na starcie ?

- pobieranie elementów widoku, rzutowanie: (np. findById)
- każde zdarzenie = nowa klasa listenera (często anonimowa)
- skomplikowany model wykonywania zadań w tle
- „rozwlekły” dostęp do zasobów, preferencji, serwisów systemowych

… i jeszcze kilka innych „wypełniaczy czasu”

Zacznijmy od UI


```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/txvName"
        android:gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />
    <ImageView
        android:id="@+id/ivPhoto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/txvName"
        android:layout_above="@+id/btnOthers"
        android:contentDescription="@string/example_image" />
    <Button
        android:id="@+id/btnOthers"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="@string/others" />
</RelativeLayout>
```

```java
public class SimpleActivity extends Activity {

    private TextView txvName;
    private ImageView ivPhoto;
    private Button btnOthers;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_simple);

        this.txvName = (TextView)findViewById(R.id.txvName);
        this.ivPhoto = (ImageView)findViewById(R.id.ivPhoto);
        this.btnOthers = (Button)findViewById(R.id.btnOthers);

        this.btnOthers.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View sender) {
                //Logika przejścia do listy
            }
        });

        this.ivPhoto.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View arg0) {
                //Pobranie następnego zdjęcia
                return false;
            }
        });

        Spanned photoTitleSpanned = Html.fromHtml(getString(R.string.photo_title));
        this.txvName.setText(photoTitleSpanned);
    }

}
```

```java
@Fullscreen
@WindowFeature({Window.FEATURE_NO_TITLE})
@EActivity(R.layout.activity_simple)
public class AASimpleActivity extends Activity {

    @ViewById(R.id.txvName)
    @FromHtml(R.string.photo_title)
    protected TextView txvName;

    @Click(R.id.btnOthers)
    protected void btnOthersClick(View sender){
        //Logika przejścia do listy

    }

    @LongClick(R.id.ivPhoto)
    protected void ivPhotoLongClick(View sender){
        //Pobranie następnego zdjęcia
    }

}
```

Jak więc dokonać takiej zmiany ?

Po kolei:

- mamy „adnotacje” (annotations) - należy je przetworzyć.
  Jak ? :

  1. Przetwarzanie w „runtime"-ie

  Wolne, narzut w wykonaniu.

  2. Przetwarzanie podczas kompilacji.

  Wydaje się być ok.

- jeśli podejście 2 to należy gdzieś zapisać rezultat.

  Generator kodu.

AndroidAnnotations

- Biblioteka ta działa właśnie w zaprezentowany sposób.
- Wprowadza elementy, które nazywa „rozszerzonymi komponentami”
- Obiekt taki deklarujemy poprzez zdefiniowanie dla adnotacji @E……..
  Mamy dostępne:
  @EActivity, @EApplication, @EBean, @EFragment, @EProvider, @EReceiver, @EIntentService, @EService, @EView, @EViewGroup, @EProvider,
- Generowana jest klasa odpowiedzialna za obsłużenie funkcjonalności „rozszerzonego komponentu”:

```
@EActivity(R.layout.activity_main)
public class MainActivity
    extends Activity {
```

```
public final class MainActivity_
    extends MainActivity
    implements HasViews, OnViewChangedListener {
```

Komponenty- subiektywny podział:

Wykorzystywane bezpośrednio przez środowisko Androida:

@EActivity	Actvity
@EProvider	ContentProvider
@EService	Service, IntentService
@EIntentService	IntentService
@EReceiver	BroadcastReceiver
@EProvider	ContentProvider

Wizualne:

@EFragment (?)	Fragment
@EView	View
@EViewGroup	ViewGroup

Wstrzykiwane:

@EApplication (?)	@Application	Application
@EBean	@Bean	Object

Wstrzykiwanie zależności (DI)

@AfterInject - po wstrzyknięciu wszystkie zależności.
Wykorzystywana w każdym z rodzajów komponentów

@AfterViews - po zabindowaniu wszystkich składowych widoku.
Wykorzystywana w:
`@EActivity, @EFragment, @EView, @EViewGroup, @EBean (?)`

Pobieranie elementów widoku

- @AfterViews - wszystkie elementy widoku zabindowane.

- @ViewById - pobranie elementów widoku

```
//Deklaracje
private ImageView ivPhoto;
private Button btnOthers;

//A później w onCreate
this.ivPhoto = (ImageView)findViewById(R.id.ivPhoto);
this.btnOthers = (Button)findViewById(R.id.btnOthers);
```

```
@ViewById(R.id.btnOthers)
protected Button someButton;

@ViewById
protected ImageView ivPhoto;
```

- FragmentById oraz @FragmentByTag

```
//Deklaracje ....
protected ExampleFragment frgExample1;
protected ExampleFragment frgTag3;

//....a następnie
frgExample1 = ((ExampleFragment) getSupportFragmentManager().
    findFragmentById(R.id.frgExample1));
frgTag3 = ((ExampleFragment) getSupportFragmentManager().
    findFragmentByTag("frgTag3"));
```

```
@FragmentById(R.id.frgExample1)
protected AAExampleFragment frgExample1;
@FragmentByTag("frgTag3")
protected AAExampleFragment frgTag3;
```

Konfiguracja fragmentów i aktywności

@Extra

```
Bundle extras = this.getIntent().getExtras();
this.exampleExtra = extras.containsKey("exampleExtra") ?
        extras.getString("exampleExtra") : "default_value";


@Extra
protected String exampleExtra = "default_value";
```

@FragmentArg

```
Bundle args = this.getArguments();
        if(args!=null){
            this.exampleArgument = args.containsKey("exampleArgument") ?
                    args.getString("exampleArgument") : null;
        }

@FragmentArg("exampleArgument")
protected String exampleArgument;
```

@NonConfigurationInstance

Wstrzykiwanie własnych zasobów

@Bean

```
@EBean
public class AAMyBean {

@EBean(scope=Scope.Singleton)
public class AAMySingletonBean {
```

```
@Bean
protected AAMyBean exampleBean;

@Bean
protected AAMySingletonBean exampleSingletonBean;
```

@App

```
@EApplication
public class AAExampleApplication
    extends Application {

    @Override
    public void onCreate(){
        super.onCreate();
        this.initializeTaskInBg();
    }

    @Background
    protected void initializeTaskInBg(){
        //Operacje
    }
}
```

```
<application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:name=".aa.AAExampleApplication_">


@App
protected AAExampleApplication app;
```

Wstrzykiwanie zasobów systemowych

@RootContext

```
public class AASimpleActivity
    extends Activity {

    @Bean
    protected AAMyBean exampleBean;
```

```
@EBean
public class AAMyBean {

    @RootContext
    protected Context context;

    @RootContext
    protected Activity anyActivityContext;

    @RootContext
    protected AASimpleActivity exactActvityContext;

}
```

@SystemService

```
//Deklaracja ...
private NotificationManager nManager;

// ... i pobranie
this.nManager = (NotificationManager)
    this.getSystemService(Context.NOTIFICATION_SERVICE);
```

```
@SystemService
protected NotificationManager nManager;
```

Obsługa zdarzeń

- Standardowo mamy:

```
//Deklaracja ...
private Button btnCancel;

// ... pobranie ...
this.btnCancel = (Button)findViewById(R.id.btnCancel);

// ... a następnie przypisanie klasy "listenera"
this.btnCancel.setOnClickListener(
    new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //[..]
```

- w przypadku AndroidAnnotations:

```
@Click(R.id.btnCancel)
protected void btnCancelClick(){
    //[...]
```

- Dostępne zdarzenia:

- @Click, @Touch, @LongClick
- @SeekBarProgressChange, @SeekBarTouchStart, @SeekBarTouchStop
- @ItemClick, @ItemLongClick, @ItemSelect
- @OptionsItem
- @FocusChanges, @CheckedChange

Praca z wątkami


```
public class ThreadActivity extends Activity {

    private TextView txvProgress;
    private Button btnStart;
    private Button btnCancel;
    private OperationTask task;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_thread);

        this.btnCancel = (Button)findViewById(R.id.btnCancel);
        this.btnStart = (Button)findViewById(R.id.btnStart);
        this.txvProgress = (TextView)findViewById(R.id.txvProgress);

        this.btnStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(task!=null &&
                    task.getStatus() == AsyncTask.Status.RUNNING){
                    task.cancel(true);
                }
                task = new OperationTask();
                task.execute(100L);

            }
        });

        this.btnCancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(task!=null && task.getStatus() == Status.RUNNING){
                    task.cancel(true);
                }
            }
        });
    }

    private void showFinish(String message){
        this.txvProgress.setText(message);
        Toast.makeText(this, message, Toast.LENGTH_LONG).show();
    }

    private void showProgress(String message){
        this.txvProgress.setText(message);
    }

    private class OperationTask extends AsyncTask<Long, Integer, Long>
    {/*[...]*/}
}
```

```
private class OperationTask
     extends AsyncTask<Long, Integer, Long>
{
    @Override
    protected Long doInBackground(Long... params) {
        long time = 0;
        try{
            for(int i = 0; i < 100; i++){
                Thread.sleep(params[0]);
                time += params[0];
                publishProgress(i);
                if (isCancelled()) break;
            }
        }catch(InterruptedException e){
            return -1L;
        }
        return time;

    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        showProgress(getString(R.string.progress, values[0]));
    }

    @Override
    protected void onPostExecute(Long result) {
        showFinish(getString(R.string.finished, (long)result));
    }
}
```

Praca z wątkami (AA)

- ~~AsyncTask~~
- @Background
  - id
  - serial
  - delay
- @UiThread
  - delay


```
@EActivity(R.layout.activity_thread)
public class AAThreadActivity extends Activity {

    @ViewById(R.id.txvProgress)
    protected TextView txvProgress;

    @Click(R.id.btnStart)
    protected void btnStartClick(){
        doOperation(100);
    }

    @Click(R.id.btnCancel)
    protected void btnCancelClick(){
        BackgroundExecutor.cancelAll("operation", true);
    }

    @Background(id="operation")
    protected void doOperation(int timeStep){
        long time = 0;
        try{
            for(int i = 0; i < 100; i++){
                Thread.sleep(timeStep);
                time += timeStep;
                showProgress(getString(R.string.progress, i));
            }
        }catch(InterruptedException e){
            showFinish(getString(R.string.finished, -1L));
        }
        showFinish(getString(R.string.finished, time));
    }

    @UiThread
    protected void showProgress(String message){
        this.txvProgress.setText(message);
    }

    @UiThread
    protected void showFinish(String message){
        this.txvProgress.setText(message);
        Toast.makeText(this, message, Toast.LENGTH_LONG).show();
    }

}
```

Zasoby

- Aby skorzystać z jakiegoś zasobu:

```
//Deklaracja ...
private String applicationName;

// .... i pobranie
this.applicationName = getResources().
    getString(R.string.app_name);
```

- W przypadku wykorzystania AA:

```
@StringRes(R.string.app_name)
protected String applicationName;
```

- Dostępna jest cała rodzina adnotacji @XXXRes:

  @StringRes, @AnimationRes, @ColorRes, @DimensionPixelOffsetRes, @DimensionPixelSizeRes, @DimensionRes, @BooleanRes, @ColorStateListRes, @DrawableRes, @IntArrayRes, @IntegerRes, @LayoutRes, @MovieRes, @StringArrayRes, @TextArrayRes, @TextRes, @HtmlRes

SharedPreferences

- Wykorzystanie standardowe:

```
SharedPreferences prefs = this.getSharedPreferences(
        "pl.amsard.aa3examples", Context.MODE_PRIVATE);
prefs.edit().putLong("timeKey", new Date().getTime()+3600).commit();
long time = prefs.getLong("timeKey", new Date().getTime());
```

- W przypadku użycia AA:

```
@SharedPref
public interface AAPrefs {
    long time();
}


long time = prefs.time().getOr(new Date().getTime());
this.prefs.edit().time().put(new Date().getTime()+3600).apply();
this.prefs.clear();
```

Klient RESTowego API

- Oparty o Spring Android (core, RestTemplate)
- Dodatkowo wymaga serializera do określonego, używanego formatu danych (np. Jackson - JSON)
- Jak używać:

```
@Rest(converters = { MappingJacksonHttpMessageConverter.class, FormHttpMessageConverter.class })
public interface WebserviceClientInterface {

    @Get("/api/news?lang={lang}&api={api}")
    NewsResponse getNews(String lang, int api);

    @Post("/api/auth")
    AuthorizeResponse authorizeUser(UserPostParams params);

}

@RestService
protected WebserviceClientInterface restClient;
```

Jeszcze kilka przydatnych dodatków

- Integracja z biblioteką RoboGuice
- Integracja z biblioteką ORMLite

Q&A

- mail: marcin.wasowski@amsterdam-standard.pl
- przykłady: http://github.com/marcinwasowski/aa_examples
- więcej ?: Zapraszamy:

AMSTERDAM STANDARD