coaxys

POEC Android

Pierre Duchemin (@pierreduchemin)

Système de debug

Le système de debug permet de faciliter le développement (désactiver le son, sélectionner un serveur...)

```
if (BuildConfig.DEBUG) {
        //...
}
```

Flavors

Flavor : permet de gérer un **système de déclinaisons de l'application**. Sert généralement à faire des versions payantes et gratuites à partir du même code source.

Une variable est définie dans le script de build, il faut ensuite la tester pour gérer les différences

```
if ("dev".equals(BuildConfig.FLAVOR)) {
        //...
}
```

Butterknife

- Outil couramment utilisé
- Permet de diminuer la quantité de code nécessaire
- Butterknife Zelezny permet de générer les annotations Butterknife : https://github.com/avast/android-butterknife-zelezny

Butterknife

- Ajouter la dépendance dans le fichier dans le build.gradle du module "app" :

```
compile 'com.jakewharton:butterknife:8.5.1'
annotationProcessor 'com.jakewharton:butterknife-compiler
```

Butterknife

```
class SimpleActivity extends Activity {
  @BindView(R.id.tvTitle) TextView title;
  @BindView(R.id.tvSubtitle) TextView subtitle;
  @BindView(R.id.tvFooter) TextView footer;

  @Override public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.activity_simple);
    ButterKnife.bind(this);
    // TODO Use fields...
  }
}
```

Butterknife

```
@OnClick(R.id.btnSubmit)
public void onClickSubmit(Button button) {
  button.setText("Hello!");
}
```

Remplace le OnClickListener !

Documentation : https://jakewharton.github.io/butterknife/

Exercice 8

- Reprendre le projet Quiz et utiliser ButterKnife

Gestion des permissions

- Android dispose d'un **système de sécurité basé sur des permissions**
- Des permissions sont requises quand des actions sensibles sont entreprises
- **Il faut les déclarer dans le Manifest.xml**
- L'utilisateur peut consulter la liste des permissions à l'installation

Communication réseau

- Nous allons utiliser le stack réseau de Square
- Il en existe d'autres, notamment Volley de Google
- Android utilise OkHttp de Square en interne

Retrofit, OkHttp, OkIo, et Gson

- OkIo : remplacement de JavaIo
- OkHttp : client Http basé sur OkIo
- Retrofit : client REST basé sur OkHttp
- Gson : Sérialisation / désérialisation json

Retrofit, OkHttp, OkIo, et Gson

- Ajouter la dépendance dans le fichier dans le build.gradle du module "app" :

```
compile 'com.squareup.retrofit2:retrofit:2.2.0'
compile 'com.squareup.retrofit2:converter-scalars:2.2.0'
compile 'com.squareup.okhttp3:logging-interceptor:3.6.0'
```

Note : vous pouvez aussi utiliser compile 'com.squareup.retrofit2:converter-scalars:2.2.0'

Communication réseau

- Dans un fichier GitHubService.java

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<String>> listRepos(@Path("user") String user)
}
```

Note : utiliser @Query au lieu de @Path pour passer le paramètre dans l'URL

Communication réseau

- Dans un fichier GitHubApplication.java

```
public class GitHubAPIService {

    private static GitHubService service;

    public static GitHubService get() {
        if (service == null) {
          Retrofit retrofit = new Retrofit.Builder()
          .baseUrl("https://api.github.com/")
          .addConverterFactory(GsonConverterFactory.creat
          .build();

          service = retrofit.create(GitHubService.class);
        }
        return service;
    }
}
```

Note : vous pouvez aussi utiliser :

.addConverterFactory(ScalarsConverterFactory.create())

Communication réseau

```
// Dans le onCreate()
GitHubAPIService.get().listRepos("octocat").enqueue(
    new Callback<List<String>>() {

    @Override
    public void onResponse(Call<List<String>> call,
        Response<List<String>> response) {
        if (response.isSuccessful()) {
            List<String> repositories = response.body();
        } else {
            Toast.makeText(this, "Erreur API",
                Toast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onFailure(Call<List<String>> call,
        Throwable t) {
        Toast.makeText(this, "Erreur réseau",
            Toast.LENGTH_LONG).show();
    }
});
```

Exercice 9

- Créer une page pour se connecter au PetStore
  (Vous pouvez utiliser n'importe quel login/password)

http://petstore.swagger.io/#!/user/loginUser

Exercice 10

- Afficher la liste des dépôts Github d'un utilisateur

Note : utiliser un RecyclerView

Note 2 : des exemples sont disponibles sur le site de Retrofit

Faire un menu

- On utilise la View Toolbar :

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/colorPrimary"
    />
```

Faire un menu

- Lier la toolbar à l'activity :

```
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
if (toolbar != null) {
    setSupportActionBar(toolbar);
}
```

Faire un menu

- Décrire les entrées du menu en xml :

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_settings"
        android:icon="@android:drawable/ic_menu_manage"
        android:title="@string/settings_menu"
        app:showAsAction="ifRoom"/>
</menu>
```

Faire un menu

- Lier le xml du menu à la toolbar :

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }
```

Service

Les **Services sont des tâches de fond** similaires aux crons Unix (analyse du réseau, de la batterie...). Ils ne disposent pas d'une IHM.
Les Services doivent être déclarés dans le fichier manifest.

```
public class MonService extends Service {
        //...
}
```

On les démarre avec :

```
// Dans une activity
startService(new intent(MonActvity.this,
        MonService.class));
```

Exercice 11

Faire une application qui affiche une alerte en cas de faible température depuis un service et pourra fonctionner en tâche de fond

http://weathers.co/api

Permissions dynamiques

- Depuis Android Marshmallow (API 23), **certaines actions requièrent l'utilisation de permission dynamiques** en plus des permissions classiques (accès à l'appareil photo, à la géolocalisation...)
- Nous utiliserons un wrapper qui va nous simplifier l'usage cette l'API : https://github.com/kayvannj/PermissionUtil

PermissionUtils (1)

- Ajouter la dépendance dans le fichier dans le build.gradle du module "app" :

```
compile 'com.github.kayvannj:PermissionUtils:1.0.3'
```

PermissionUtils (2)

```
private static final int REQUEST_CODE_STORAGE = 1234;
private PermissionUtil.PermissionRequestObject
        requestObject;

@Override
public void onRequestPermissionsResult(int requestCode,
        @NonNull String[] permissions,
        @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode,
        permissions, grantResults);
    requestObject.onRequestPermissionsResult(requestCode,
        permissions, grantResults);
}
```

PermissionUtils (3)

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

    requestObject = PermissionUtil.with(this).request(
        Manifest.permission.WRITE_EXTERNAL_STORAGE)
            .onAllGranted(new Func() {
                @Override
                protected void call() {
                    // Permission accordée
                }
            })
            .onAnyDenied(new Func() {
                @Override
                protected void call() {
                    // Permission refusée
                }
            })
            .ask(REQUEST_CODE_STORAGE);
}
```