Android – Parceler

Introducción

Como se comento en el último post de Java para android en adelante trataremos librerías y ejemplos de novedades, para cursos de android podéis encontrar enlaces en el post anterior. En dicho post en la parte de serialización comentábamos que en android existían parcelables y el porqué debíamos utilizarlos. Bien pues hoy presentamos una librería que facilita mucho su uso Parceler

- Configuración
- Utilizando Parceler en nuestra clase
- Observaciones

Configuración

Para la instalación, tenemos que añadir el plugin android-apt a nuestro classpath en el fichero build.gradle que se encuentra en la raíz de nuestro proyecto.

```
dependencies{
  classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
}
```

Dentro del archivo app/build.gradle, debemos añadir el plugin antes de añadir las dependencias de Parceler.

```
plugin : 'com.neenbedankt.android-apt'
```

```
dependencies {
  compile 'org.parceler:parceler-api:1.1.5'
  apt 'org.parceler:parceler:1.1.5'
}
```

Utilizando Parceler a nuestra clase

Incluiremos un constructor vacío, las variables de clase deben ser public y finalmente utilizaremos la anotación **@Parcel**. Bien esto nos ahorra mucho código y al mismo tiempo este se hace mucho más legible. Pero tiene un inconveniente, asume como entorno toda la clase User. Para evitar encapsular variables que no son necesaria utilizaremos la anotación **@Transient**

```
@Parcel
public class User {
    // class vars must be public
    public String name;
    public String lastName;
    public String job;
    @Transient
    public boolean hasCar;

    // empty constructor needed by the Parceler library
    public User() {
    }

     public User(String name, String lastName, String job,
boolean hasCar) {
        this.name = name;
        this.lastName = lastName;
        this.job = job;
        this.hasCar = hasCar;
    }

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getJob() {
        return job;
    }

    public void setJob(String job) {
        this.job = job;
    }

    public boolean isHasCar() {
        return hasCar;
    }

    public void setHasCar(boolean hasCar) {
        this.hasCar = hasCar;
    }
}
```

Para crear un parcelable utilizaremos **Parcels.wrap**:

```
User user = new User("Joan", "Miquel", "Android Developer",
true);
Intent intent = new Intent(getApplicationContext(),
SecondActivity.class);
intent.putExtra("user", Parcels.wrap(user));
```

Para recuperar el objeto user utilizaremos **Parcels.unwrap**:

```
user = Parcels.unwrap(getIntent().getParcelableExtra("user"));
```

Observaciones

Como podéis comprobar el uso de esta librería es muy útil y ademas muy fácil incluir en nuestros proyectos. Pero no todo es perfecto, como toda librería que utilicemos, mejorará con el tiempo y solucionara bugs. A priori esto es bueno pero no disponemos de un sistema de upgrade automático. Debemos de estar detrás de las novedades y preocuparnos notros de actualizarla. Tenemos que tener en cuenta que utilizar esta librería tiene sus ventajas e inconvenientes. Nos facilita mucho el uso de parcelabe pero al mismo tiempo es menos eficiente que implementarlo nosotros mismos. De echo el uso de esta librería es un buen balance entre eficiencia y facilidad de uso en comparación con una serialización y una implementación de parcelabe. Cabe destacar que si nuestra aplicación requiere de la máxima eficiencia lo mejor será implementar parcelable. A continuación os dejo el enlace a la página de la librería, un simple ejemplo en GitHub y un artículo en android developers de como usar parcelables

- Parceler
- ParcelerTest
- Parcelable (Android Developer)