Cette étude va nous permettre de travailler avec plusieurs activités pout une même application ou de lancer une activité située dans une autre application.

- Communication entre activités au travers de messages : **Les intentions**.
- Lancer des **sous-activitées** au sein d'une même application.
- Utilisation des onglets pour passer d'une activité à une autre au sein d'une même application.
- Démarrer une activité et obtenir un retour
- Solliciter d'autres applications, filtrer les actions et embarquer des données supplémentaires.

✗ COMMUNICATION ENTRE APPLICATIONS : LA CLASSE INTENT

Le système Android repose sur un grand nombre de composants étroitement liés. Ce que vous pouvez obtenir dans une interface graphique via des boîtes de dialogue, des fenêtres filles, etc. est généralement traité par ***des activités indépendantes***. Bien que l'une d'elles puisse être "spécifique" puisqu'elle apparaît dans le lanceur, les autres doivent toutes être accessibles... d'une façon ou d'une autre.

*Elles le sont grâce aux **intentions**. Au cœur du système Android, les **intentions** forment un mécanisme sophistiqué et complet permettant **aux activités et aux applications d'interagir entre elles**.*

*La communication interne du système Android est basée sur **l'envoi et la réception de messages** exprimant l'intention d'une action. Représentant une description abstraite d'une opération à réaliser, chacun des messages peut être émis à destination d'un autre composant de la même application (**une activité, un service, etc.**) ou même celui d'une toute autre application.*

Issu de la classe **Intent**, ce **message** permet de véhiculer toutes les informations nécessaires à la réalisation de l'action :

- Informations à destination du composant qui le réceptionnera (actions à effectuer et les données avec lesquelles agir) ;
- Informations nécessaires au système pour son traitement (catégorie du composant cible du message et instructions d'exécution de l'action).

*Le démarrage des composants d'une application (**activités, services, etc.**) est réalisé au moyen d'un objet **Intent**. L'utilisation d'un composant externe peut ainsi être décidée au moment de l'exécution de l'application et non lors de la compilation de l'application. Ce mécanisme permet d'éviter les dépendances et de rendre beaucoup plus dynamique et pertinente l'utilisation des applications dans un contexte donné.*

- Vous pouvez envoyer des intentions au système de deux façons : soit **en ciblant un composant précis** d'une application (on parle alors de **mode explicite**), soit en laissant **le système déléguer le traitement** (ou non) de cette demande au composant le plus approprié (on parle alors de **mode implicite**).
- Un système de filtres permet à chaque application de filtrer et de gérer uniquement les intentions qui sont pertinents pour celle-ci. Une application peut ainsi être dans un état d'inactivité, tout en restant à l'écoute des intensions circulant dans le système.

✗ PRINCIPE DE FONCTIONNEMENT

Les objets **Intent** ont essentiellement trois utilisations : ils permettent de démarrer une activité au sein de l'application courante, de solliciter d'autres applications ou d'envoyer des informations.

- Le démarrage d'une activité au sein d'une même application est utilisée pour la navigation entre écrans d'une interface graphique et l'appel d'une boîte de dialogue. C'est d'ailleurs le seul cas où vous devrez démarrer une activité en mode explicite.
- Lorsqu'un besoin ne peut être satisfait par l'application elle-même, elle peut solliciter une autre application pour y répondre (le besoin peut être l'exécution d'une action ou la transmission d'informations). Elle peut se contenter de transmettre son intention au système qui, lui, va se charger de trouver l'application et le composant le plus approprié puis démarrer ce dernier et lui transmettre l'**Intent** correspondant. On parle alors de résolution de l'intention : à un contexte donné, le système choisit au mieux l'application correspondant à un objet **Intent** donné.

- Les intentions ont aussi d'autres utilisations, dont le démarrage d'un service. Le mécanisme relatif aux objets **Intent** et leur utilisation sont en effet indispensables pour les applications fonctionnant en arrière-plan (telles que les services que nous découvrirons dans un prochain chapitre) afin de recevoir des actions à effectuer (par exemple, pour un service de lecteur multimédia, les actions pourront être de lire une musique, passer à la chanson suivante ou encore mettre la lecture en pause en cas d'appel entrant) mais également pour pouvoir communiquer avec d'autres applications.
- Il est aussi possible de vouloir diffuser un objet **Intent** à plusieurs applications (par exemple pour informer l'ensemble des applications ouvertes que la batterie est défaillante). Le système pouvant très vite devenir verbeux, les applications peuvent mettre en place un filtre permettant de ne conserver que les **Intents** que l'application juge nécessaires.

L'OBJET INTENT

Comme nous venons de le découvrir, un objet **Intent** véhicule toutes les informations nécessaires à la réalisation d'une action (ou à la réception d'information) :

- **Le nom du composant ciblé** : Cette information facultative permet de spécifier de façon non ambigüe le nom du composant qui sera utilisé pour réaliser l'opération. Si le nom du composant n'est pas renseigné, le système déterminera la composant le plus approprié.
- **L'action** : Une chaîne de caractères définissant l'action à réaliser. Dans le cadre d'un récepteur d'intention, il s'agira de l'action qui s'est produite et pour laquelle le système ou l'application informe toutes les autres.
- **Les données** : Le type de contenu MIME sous la forme d'une chaîne de caractères et le contenu ciblé sous la forme d'un URI. Par exemple, si vous souhaitez afficher une page web, vous utiliserez l'action **ACTION_VIEW** et une URI de la forme ***http://<adresse du site>***.
- **Les données supplémentaires** : Sous la forme d'une collection de paire **clé/valeur**, vous pouvez spécifier des paramètres supplémentaires. Par exemple, si l'intention est une demande d'envoi d'un courriel, l'utilisation de la constante **EXTRA_EMAIL** permet de préciser les adresses de messagerie des destinataires ;
- **La catégorie** : Cette information complémentaire permet de cibler plus précisément qui devra gérer l'intention émise. Par exemple, l'utilisation de la constante **CATEGORY_BROWSABLE** demande le traitement par un navigateur alors que la constante **CATEGORY_LAUNCHER** spécifie que l'activité est le point d'entrée de l'application.
- **Les drapeaux** : Principalement utilisés pour spécifier comment le système doit démarrer une activité. Par exemple, l'utilisation de la constante **FLAG_ACTIVITY_NO_ANIMATION** spécifie que l'activité doit être démarrée sans jouer l'animation de transition entre écrans.

NAVIGUER ENTRE ÉCRANS AU SEIN D'UNE APPLICATION (CRÉATION D'INTENTION)

Une application est souvent composée de plusieurs écrans qui s'enchaînent les uns à la suite des autres en fonction de l'utilisateur et **chaque écran est représenté par une activité** définissant son interface utilisateur et sa logique.

La principale utilisation d'une intention est le démarrage de ces activités (une à la fois) permettant cet enchaînement. De façon plus générale, chaque composant de votre application nécessitera l'emploi d'une intention pour être démarré.

Pour démarrer une activité, il est nécessaire d'avoir une intention et de choisir comment la lancer. *Comme nous venons de le voir, les intentions encapsulent une requête pour une activité ou une demande adressée à un autre récepteur d'intention, afin qu'il réalise une certaine tâche.*

- Si l'activité que vous comptez lancez vous appartient, il est plus simple de créer une intention explicite, en nommant le composant à lancer, comme ici : ***Intent intention = new Intent(this, MonActivité.class);***
- Ici, le constructeur de la classe **Intent** prend deux paramètres. Le premier correspondant au contexte à partir duquel l'intention est créée et sera envoyée. Ce premier paramètre fait référence la plupart du temps à l'activité en cours. Le second paramètre fait référence à un type de classe Java héritant de la classe **Activity**.

Démarrer une activité *:* Maintenant que nous disposons de l'intention, il faut la fournir à Android afin de récupérer l'activité fille à lancer. Il existe deux méthodes pour démarrer une activité, en fonction de la logique de l'interface : parfois, nous avons besoin de savoir comment s'est déroulé l'activité (et obtenir un retour lors de son arrêt), parfois non.

- Le plus simple consiste à appeler la méthode ***startActivity()*** en lui passant l'intention. Android recherchera l'activité qui correspond le mieux et lui passera l'intention pour qu'elle la traite. Dans cette situation, votre activité ne sera jamais prévenue de la fin de l'activité fille, qui dans beaucoup de cas ne pose pas de problème.
- Vous pouvez également appeler ***startActivityForResult()*** en lui passant l'intention et un identifiant (unique pour l'activité appelante). Android recherchera l'activité qui correspond le mieux et lui passera l'intention. Cette fois-ci, votre activité sera prévenue par la méthode de rappel ***onActivityResult()*** de la fin de l'activité fille.

DÉMARRER UNE ACTIVITÉ FILLE SANS TENIR COMPTE DU RETOUR

Après tout ce préambule, je vous propose le premier projet de conversion entre les radians et les degrés qui possède plusieurs activités dans une même application. Ainsi, une activité principale lancera, au travers des intentions, plusieurs activités filles, dans un premier temps, sans attendre une valeur quelconque de leur part.

Attention, avant de pouvoir démarrer une activité, il est nécessaire au préalable de la déclarer dans le fichier de configuration **AndroidManifest.xml** de l'application. Sans cela, une erreur de type **ActivityNotFoundException** sera générée lors de l'appel de la méthode ***startActivity()***.

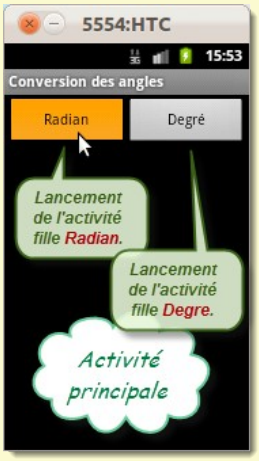
AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="fr.btsiris.radian"
        android:versionCode="1"
        android:versionName="1.0">
    <application android:label="Angles" android:icon="@drawable/icon">
        <activity android:name="Choix" android:label="Conversion des angles">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Radian" android:label="Recherche du radian" />
        <activity android:name="Degre" android:label="Recherche du degré" />
    </application>
</manifest>
```

Nota : Nous remarquons au travers de ce manifeste que l'activité principale de cette application se nomme **Choix**. Par ailleurs, deux autres activités sont déclarées, respectivement **Radian** et **Degre** qui sont les activités filles de Choix.

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="3px">
    <Button
        android:layout_width="0px"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Radian"
        android:onClick="calculRadian" />
    <Button
        android:layout_width="0px"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Degré"
        android:onClick="calculDegre" />
</LinearLayout>
```



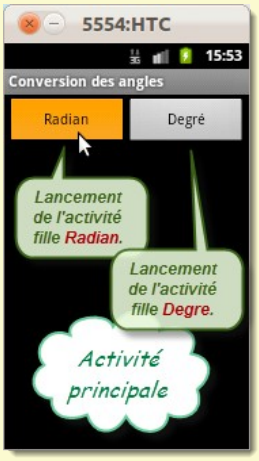
fr.btsiris.radian.Choix.java

```
package fr.btsiris.radian;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Choix extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void calculRadian(View vue) {
        startActivity(new Intent(this, Radian.class));
    }

    public void calculDegre(View vue) {
        startActivity(new Intent(this, Degre.class));
    }
}
```



- Cette activité principale est très simple. Elle possède juste les deux méthodes déjà déclarées dans le fichier de description principal et qui s'occupent de lancer les activités filles au travers de la méthode ***startActivity()*** et des Intentions respectives.
- Maintenant, pour chacune des activités filles, vous devez créer un fichier de description spécifique ainsi que la classe correspondante. Comme beaucoup d'éléments sont en commun, nous allons même créer une classe de base qui factorise tous ces éléments. Ainsi chacune de ces **sous-activités** héritera de cette classe.

res/layout/radian.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:padding="3px">
    <EditText
          android:id="@id/degre"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:inputType="number" />
    <Button
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:text="Radian"
          android:onClick="calcul" />
     <EditText
          android:id="@id/radian"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:editable="false" />
</LinearLayout>
```

res/layout/degre.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:padding="3px">
    <EditText
          android:id="@+id/radian"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:inputType="numberDecimal" />
    <Button
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:text="Degré"
          android:onClick="calcul" />
    <EditText
          android:id="@+id/degre"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:editable="false" />
</LinearLayout>
```

- Ces deux fichiers se ressemblent beaucoup. Toutefois, remarquez bien que dans le descripteur **degre.xml**, nous avons la déclaration de nouveaux identifiants, respectivement **radian** et **degre**, alors que dans le descripteur **radian.xml**, nous utilisons simplement ces déclarations, sachant que **degre.xml** va être compilé en premier (ordre alphabétique).
- Nous décrivons ensuite la classe de base abstraite **Conversion** qui représente la partie commune des deux activités filles, suivi des deux activités qui héritent de cette classe abstraite et qui redéfinissent la méthode ***calcul()***.

fr.btsiris.radian.Conversion.java

```
package fr.btsiris.radian;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import java.text.*;

public abstract class Conversion extends Activity {
    EditText degre, radian;
    NumberFormat formatRadian = new DecimalFormat("0.00");
    NumberFormat formatDegre= new DecimalFormat("0 °");
```

```
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        degre = (EditText) findViewById(R.id.degre);
        radian =  (EditText) findViewById(R.id.radian);
        degre.setText(formatDegre.format(0));
        radian.setText(formatRadian.format(0));
    }

    public abstract void calcul(View vue);
}
```

fr.btsiris.radian.Radian.java

```
package fr.btsiris.radian;

import android.os.Bundle;
import android.view.View;
import java.text.*;

public class Radian extends Conversion {

    @Override
    public void onCreate(Bundle icicle) {
        setContentView(R.layout.radian);
        super.onCreate(icicle);
    }

    @Override
    public void calcul(View vue) {
        try {
            Number valeur = formatDegre.parse(degre.getText().toString());
            radian.setText(formatRadian.format(valeur.doubleValue() * Math.PI / 360));
        }
        catch (ParseException ex) { }
    }
}
```

fr.btsiris.radian.Degre.java

```
package fr.btsiris.radian;

import android.os.Bundle;
import android.view.View;
import java.text.*;

public class Degre extends Conversion {

    @Override
    public void onCreate(Bundle icicle) {
        setContentView(R.layout.degre);
        super.onCreate(icicle);
    }

    @Override
    public void calcul(View vue) {
        try {
            Number valeur = formatRadian.parse(radian.getText().toString());
            degre.setText(formatDegre.format(valeur.doubleValue() * 360 / Math.PI));
        }
        catch (ParseException ex) { }
    }
}
```

x LANCEMENT SIMPLIFIÉE DES SOUS-ACTIVITÉS AVEC DES ONGLETS

Les onglets peuvent contenir une vue ou une activité. Dans le cas où chaque onglet représente une activité fille, c'est relativement simple à réaliser. Il suffit effectivement de fournir l'intention qui lancera l'activité souhaitée ; le framework de gestion des onglets placera alors automatiquement l'interface utilisateur de cette activité dans l'onglet.

Je vous propose de modifier le projet précédent, en supprimant la première page avec les deux boutons qui réalisent le choix de conversion, et de la remplacer par une gestion des onglets qui effectueront le même type de choix.

✗ Dans ce cas de figure, nous n'avons plus du tout besoin du fichier de description principal **main.xml**. Toute la mise œuvre des onglets se fait uniquement dans le code Java au travers de l'activité principale **Choix.java**.

fr.btsiris.radian.Choix.java

```
package fr.btsiris.radian;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TabHost;

public class Choix extends TabActivity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		TabHost onglets = getTabHost();
		onglets.addTab(onglets.newTabSpec("radian")
				.setIndicator("Radian")
				.setContent(new Intent(this, Radian.class)));
		onglets.addTab(onglets.newTabSpec("degre")
				.setIndicator("Degré")
				.setContent(new Intent(this, Degre.class)));
	}
}
```

✗ Comme vous pouvez le constater, notre classe hérite de **TabActivity** : nous n'avons donc pas besoin de créer de fichier de description ***XML*** car **TabActivity** s'en occupe pour nous. Nous nous contentons d'accéder au **TabHost** et de lui ajouter deux onglets, chacun précisant une intention qui fait directement référence à une autre classe. Ici, nos deux onglets hébergeront respectivement un **Radian** et un **Degre**.

✗ Vous remarquez également que le bouton de soumission se nomme maintenant tout simplement calcul puisque l'onglet nous précise sur quel traitement nous sommes. Du coup cette description du bouton est identique pour les deux fichiers de description. Je préfère du coup en faire un fichier à part entière que je nomme **calcul.xml**. Enfin, il n'est plus nécessaire de proposer un intitulé particulier pour chacune de ces sous-activité, je vais donc simplifier le manifeste en conséquence.

res/layout/calcul.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
	android:layout_width="0px"
	android:layout_height="wrap_content"
	android:layout_weight="1"
	android:text="Calcul"
	android:onClick="calcul"
/>
```

res/layout/radian.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:padding="3px">
    <EditText
          android:id="@id/degre"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:inputType="number" />
    <include layout="@layout/calcul" />
    <EditText
          android:id="@id/radian"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:editable="false" />
</LinearLayout>
```

res/layout/degre.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:padding="3px" >
    <EditText
          android:id="@+id/radian"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:inputType="numberDecimal" />
    <include layout="@layout/calcul" />
    <EditText
          android:id="@+id/degre"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:editable="false" />
</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="fr.btsiris.radian"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:label="Angles" android:icon="@drawable/icon">
        <activity android:name="Choix" android:label="Conversion des angles">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Radian" />
        <activity android:name="Degre" />
    </application>
</manifest>
```

x DÉMARRER UNE ACTIVITÉ ET OBTENIR UN RETOUR

Comme nous venons de le voir, la méthode ***startActivity()*** de la classe **Activity** permet de démarrer une autre activité "***activité enfant***". Cette méthode, bien que très utile dans son usage courant, ne propose cependant aucun mécanisme de retour d'information vers l'activité "***parent***" qui a démarré l'activité enfant.

Prenons comme exemple une activité enfant proposant à l'utilisateur un formulaire de réponse sur le nombre de chiffres après la virgule pour le calcul des radians. Comment récupérer la valeur saisie par l'uilisateur dans l'activité enfant depuis l'activité principale ?

*Pour gérer ce type de scénario, à savoir lancer une activité enfant et en connaître sa valeur de retour, il est possible d'utiliser la méthode **startActivityForResult()** prévue à cet effet. Avec cette méthode, lorsque l'activité enfant aura terminé sa tâche, elle en avertira l'activité parent.*

- **Spécificité de la méthode d'appel** : La méthode ***startActivityForResult()*** a besoin de deux paramètres. Le premier correspond à l'intention, le deuxième à une valeur numérique correspondant au code de l'activité parente qui est à votre libre choix. Si la valeur est inférieure à ***0***, l'activité parent n'attend pas de retour, ce qui équivaut à utiliser la méthode ***startActivity()***.
- **Renvoyer une valeur de retour** : Pour renvoyer la valeur de retour à l'activité principale, appelez la méthode ***setResult()*** de la classe Activity en indiquant en paramètre le code de retour. Android prévoit plusieurs valeurs par défaut telles que **RESULT_OK** et **RESULT_CANCELED**, etc.
- **Arrêter l'activité enfant** : Pour arrêter explicitement une activité enfant afin de revenir sur l'activité parent qui l'a démarré, il suffit de faire appel à la méthode ***finish()*** de la classe Activity.
- **Récupérer la valeur de retour** : Pour récupérer la valeur de retour envoyé par une activité enfant, utiliser la méthode ***onActivityResult()*** de l'activité parent, depuis laquelle vous avez appelé la méthode ***startActivityForResult()***.

*La méthode **onActivityResult()** utilise trois paramètres pour identifier l'activité et ses valeurs de retour :*

- **int requestCode** : valeur identifiant quelle activité a appelé la méthode. Cette même valeur a été spécifiée comme deuxième paramètre de la méthode ***startActivityForResult()*** lors du démarrage de la sous-activité.
- **int resultCode** : représente la valeur de retour envoyée par la sous-activité pour signaler normalement son état à la fin de la transaction. C'est une constante définie dans la class Activity (**RESULT_OK, RESULT_CANCELED, etc.**) ou par le développeur.
- **Intent data** : cet objet permet d'échanger des données comme nous le verrons bientôt.

Encore une fois, je vous propose de modifier le projet précédent. Cette fois-ci, nous disposons d'une activité principale qui permet de faire la conversion des angles dans les deux sens et dispose d'un bouton supplémentaire qui permet de choisir le nombre de chiffres après la virgule pour la visualisation des radians. Ce réglage est effectué au moyen d'une activité enfant spécifique :

- Dans ce cas de figure, nous disposons maintenant de deux fichiers de descriptions, respectivement **main.xml** et **reglage.xml**. Par ailleurs, les deux activités sont représentées par les classes **Conversion** pour ***l'activité principale*** et **Reglage** pour ***l'activité enfant***. Vous devez donc modifier le manifeste afin de respecter tous ces changements.

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="3px">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:padding="2px">
        <EditText
            android:id="@+id/radian"
            android:layout_width="0px"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:inputType="numberDecimal"/>
        <Button
            android:layout_width="0px"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Degré"
            android:onClick="calculDegre" />
    </LinearLayout>
```

```
    <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:padding="2px">
            <EditText
                android:id="@+id/degre"
                android:layout_width="0px"
                android:layout_height="wrap_content"
                android:layout_weight="2"
                android:inputType="number" />
            <Button
                android:layout_width="0px"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Radian"
                android:onClick="calculRadian" />
    </LinearLayout>
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Nombre de décimales"
        android:onClick="reglage" />
</LinearLayout>
```

res/layout/reglage.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="3px">
    <EditText
        android:id="@+id/reglage"
        android:layout_width="0px"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="2"
        android:selectAllOnFocus="true"
        android:inputType="number" />
    <Button
        android:layout_width="0px"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Valider"
        android:onClick="valider"   />
</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="fr.btsiris.radian"
        android:versionCode="1"
        android:versionName="1.0">
    <application android:label="Angles" android:icon="@drawable/icon">
        <activity android:name="Conversion" android:label="Conversion des angles">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Reglage"  android:label="Nombre de décimales (Radian)" />
    </application>
</manifest>
```

fr.btsiris.radian.Conversion.java

```
package fr.btsiris.radian;

import android.app.Activity;
public class Conversion extends Activity {
    EditText degre, radian;
```

```
    NumberFormat formatRadian = new DecimalFormat("0.00");
    NumberFormat formatDegre= new DecimalFormat("0 °");
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        degre = (EditText) findViewById(R.id.degre);
        radian =  (EditText) findViewById(R.id.radian);
        degre.setText(formatDegre.format(0));
        radian.setText(formatRadian.format(0));
    }

    public void calculRadian(View vue) {
        try {
            Number valeur = formatDegre.parse(degre.getText().toString());
            radian.setText(formatRadian.format(valeur.doubleValue() * Math.PI / 360));
        }
        catch (ParseException ex) { }
    }

    public void calculDegre(View vue) {
        try {
            Number valeur = formatRadian.parse(radian.getText().toString());
            degre.setText(formatDegre.format(valeur.doubleValue() * 360 / Math.PI));
        }
        catch (ParseException ex) { }
    }

    public void reglage(View vue) {
        startActivityForResult(new Intent(this, Reglage.class), 1); // 1 correspond au numéro d'identification de l'activité
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode>0 && resultCode<=5) {
            try {
                Number valeur = formatRadian.parse(radian.getText().toString());
                String motif = "0";
                for (int i=1; i<resultCode; i++) motif +=  "0";
                formatRadian = new DecimalFormat("0."+motif);
                radian.setText(formatRadian.format(valeur));
            }
            catch (ParseException ex) { }
        }
    }
}
```

fr.btsiris.radian.Reglage.java

```
package fr.btsiris.radian;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class Reglage extends Activity {
    private EditText reglage;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.reglage);
        reglage = (EditText) findViewById(R.id.reglage);
    }
    public void valider(View vue) {
        setResult(Integer.parseInt(reglage.getText().toString()));
        finish();
    }
}
```

*Comme nous venons de l'évoquer, au travers de la méthode **setResult()**, nous devons spécifier si l'utilisateur accepte les réglages ou pas en donnant une valeur entière prédéfinie (**RESULT_OK, RESULT_CANCELED, etc.**). Comme, nous pouvons proposer une valeur entière avec cette méthode, je m'en sers pour envoyer le nombre de chiffres après la virgule pour les radians.*

✗ SOLLICITER D'AUTRES APPLICATIONS

Nous avons déjà utilisé plusieurs fois l'objet **Intent** pour démarrer des activités au sein d'une même application. Cependant, qu'en est-il si vous désirez faire appel à un composant d'une autre application, par exemple pour ouvrir une page web ? La réponse est : toujours en utilisant une intention.

*En effet, l'envoi d'une intention permet également de demander à un composant d'une autre application que la vôtre de traiter l'action que vous souhaitiez réaliser. C'est le système qui décide alors de l'application à utiliser pour accomplir votre souhait. Pour décider du composant le plus approprié, le système se base sur les informations que vous spécifiez dans votre objet **Intent** : action, données, catégorie, etc.*

✗ Ainsi, vous exprimez l'intention au système et le système se chargera de résoudre votre intention pour vous proposer le composant de l'application le plus approprié. Ce mécanisme permet d'éviter les dépendances vers des applications puisque l'association entre votre application et le composant nécessaire se fait au moment de l'exécution et non de la compilation. Cela permet de lancer des activités en fonction du contexte et de ne pas se soucier de l'application qui sera réellement utilisée.

✗ Ce système d'utilisation d'**Intent** implicite, puisque le système doit résoudre celle-ci en fonction de son environnement, recourt à des filtres (voir plus loin) comme points d'entrée pour distribuer les intentions aux composants les plus appropriés. Les informations qui sont utilisées pour la résolution sont : ***l'action, les données*** (L'**URI** et le type de contenu **MIME**) et ***la catégorie***. Les autres données ne jouent pas de rôle dans la résolution et la distribution des **Intent** aux applications.

Les actions natives

Action	Définition
ACTION_ANSWER	Prend en charge un appel entrant. Ceci est couramment géré par l'écran natif des appels.
ACTION_CALL	Appeler un numéro de téléphone. Cette action lance une activité affichant l'interface pour composer un numéro puis appelle le numéro contenu dans l'URI spécifié en paramètre. On considère généralement qu'il est préférable d'utiliser plutôt ACTION_DIAL lorsque c'est possible.
ACTION_DELETE	Démarrer une activité permettant de supprimer une donnée identifiée par l'URI spécifiée en paramètre.
ACTION_DIAL	Afficher l'interface de composition des numéros. Celle-ci peut être pré-remplie par des données contenues dans l'URI spécifiée en paramètre. Le composeur de numéros Android natif est utilisé par défaut. Il peut normaliser la plupart des schémas de numérotation : tel:01 56 60 12 34 et tel:+33 1 56 60 12 34 sont par exemple tous deux valides.
ACTION_EDIT	Editer une donnée.
ACTION_INSERT	Ouvre une activité capable d'insérer de nouvelles entrées dans le Cursor spécifié dans l'URI de l'Intent. Lorsqu'elle est appelée comme sous-activité, elle doit envoyer une URI vers la nouvelle entrée insérée.
ACTION_PICK	Lance une nouvelle sous-activité qui vous permet de choisir un élément du Content Provider (voir dans un prochain chapitre) spécifié par l'URI de l'Intent. Une fois fermée, elle doit renvoyer une URI vers l'élément choisi. L'activité lancée dépend des données à choisir :content://contacts/people invoquera par exemple la liste des contacts natifs.
ACTION_SEARCH	Démarre une activité de recherche. L'expression de recherche devra être spécifiée au moyen de la clé SearchManager.QUERY envoyé en extra de l'action.
ACTION_SEND	Envoyer des données textuelles ou binaires par courriel ou SMS. Les paramètres dépendront du type d'envoi. Les données elles-mêmes doivent être stockées comme des compléments à l'aide d'EXTRA_TEXT ou EXTRA_STREAM en fonction de leur type. Dans le cas d'un e-mail, les applications Android acceptent également les compléments via les clés EXTRA_EMAIL, EXTRA_CC, EXTRA_BCC etEXTRA_SUBJECT. N'utilisez l'action ACTION_SEND que pour envoyer des données à un destinataire externe et non à une autre application sur l'appareil.
ACTION_SENDTO	Lance une activité capable d'envoyer un message au contact défini par l'URI spécifié en paramètre.
ACTION_VIEW	Démarre une action permettant de visualiser l'élément identifié par l'URI spécifié en paramètre. C'est l'action la plus commune. Par défaut, les adresses commençant par http: lanceront un navigateur web, celles commençant par tel: lanceront l'interface de composition de numéro et celles débutant par geo: lanceront Google Map.
ACTION_WEB_SEARCH	Effectue une recherche sur Internet avec l'URI spécifiée en paramètre comme requête.

✗ EXEMPLE DE MISE EN OEUVRE

Dans ce type de scénario, l'activité lancée est plutôt un "pair" de l'activité qui l'a lancée. Elle sera donc lancée comme une activité classique. Votre activité ne sera pas informée de la fin de sa "fille" mais, encore une fois, elle n'a pas vraiment besoin de le savoir.

Afin de mieux comprendre le lancement d'une activité paire, je vous propose de réaliser un projet qui affiche la carte d'un lieu en spécifiant sa localisation à l'aide de la latitude et de la longitude.

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      android:padding="3px">
      <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="wrap_content">
          <TextView
              android:layout_width="0dp"
              android:layout_height="wrap_content"
              android:text="Lieu : "
              android:textStyle="bold"
              android:layout_weight="1" />
          <EditText
              android:id="@+id/latitude"
              android:layout_width="0dp"
              android:layout_height="wrap_content"
              android:layout_weight="2"
              android:inputType="numberDecimal" />
          <EditText
              android:id="@+id/longitude"
              android:layout_width="0dp"
              android:layout_height="wrap_content"
              android:layout_weight="2"
              android:inputType="numberDecimal" />
      </LinearLayout>
      <Button
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="Visualiser !"
          android:onClick="visualiser" />
</LinearLayout>
```

- Ce fichier de description est assez simple puisqu'il contient deux champs pour la latitude et la longitude, ainsi qu'un bouton pour lancer l'activité paire.
- Lorsque nous cliquons sur le bouton de visualisation, l'activité de cartographie intégrée à Android est automatiquement lancée avec les coordonnées de géolocalisation (latitude et longitude). Pour cette application, nous n'avons pas eu besoin de créer une activité fille pour afficher cette carte. Nous utilisons les compétences de l'environnement.

fr.btsiris.geo.Geolocalisation.java

```
package fr.btsiris.geo;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;


public class Geolocalisation extends Activity {
    private EditText latitude, longitude;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        latitude = (EditText) findViewById(R.id.latitude);
        longitude = (EditText) findViewById(R.id.longitude);
    }

    public void visualiser(View vue) {
        String lat = latitude.getText().toString();
        String lon = longitude.getText().toString();
        Uri coord = Uri.parse("geo:"+lat+","+lon);
        startActivity(new Intent(Intent.ACTION_VIEW, coord));
    }
}
```

FILTRER LES ACTIONS

Android permet aux applications de spécifier quelles sont les actions qu'elles gèrent : le système peut ainsi choisir le composant le mieux adapté au traitement d'une action (véhiculée dans un objet **Intent**).

*Tous les composants Android qui souhaitent être prévenus par des intentions doivent déclarer des filtres d'intention afin qu'Android sache quelles intentions doivent aller vers quel composant. Pour ce faire, vous devez ajouter des éléments **<intent-filter>** au manifeste de l'application.*

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="fr.btsiris.tache"
        android:versionCode="1"
        android:versionName="1.0">
    <application android:label="@string/app_name" android:icon="@drawable/icon">
        <activity android:name="Tache" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

- Notez la présence de l'élément **<intent-filter>** sous l'élément **<activity>**.
- Il annonce que cette activité est l'activité principale de cette application.
- De plus, elle appartient à la catégorie ***LAUNCHER***, ce qui signifie qu'elle aura une icône dans le menu principal d'Android.

Cette activité étant l'activité principale de l'application, Android sait qu'elle est le composant qu'il devra lancer lorsqu'un utilisateur choisit cette application à partir du menu principal.

- Vous pouvez bien sûr indiquer plusieurs actions ou catégories dans vos filtres d'intention afin de préciser que le composant associé (l'activité) gère plusieurs sortes d'intentions différentes.
- Il est fort probable que vous voudrez également que vos activités secondaires (non principales) précisent le type MIME des données qu'elles manipulent. Ainsi, si une intention est destinée à ce type **MIME**, directement ou indirectement via une URI référençant une ressource de ce type, Android saura que le composant sait gérer ces données. Chaque élément de la balise **<intent-filter>** est important puisqu'il détermine le niveau de filtrage :

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="demo" />
    </intent-filter>
</activity>
```

- **action** : identifiant unique sous forme de chaîne de caractères. Il est d'usage d'utiliser la convention de nom Java. Utilise l'attribut **android:name** pour spécifier le nom de l'action à laquelle répondre.
- **category** : premier niveau de filtrage de l'action. Cette balise indique dans quelle circonstance l'action va s'effectuer ou non. Il est possible d'ajouter plusieurs balises de catégorie. Utilise l'attribut **android:name** pour spécifier dans quelles circonstances une réponse doit être donnée à l'action.
- **data** : filtre l'objet **Intent** au niveau des données elles-mêmes. Par exemple, en jouant avec l'attribut **android:host**, nous pouvons répondre à une action comportant un nom de domaine particulier, comme **www.site.com**. Cette balise vous permet ainsi de spécifier sur quels types de données vos composants peuvent agir. Vous pouvez inclure plusieurs balises **<data>** selon vos besoin.

Grâce à la définition de ce filtre, l'activité déclarée par le fichier de configuration de l'application ci-dessus réagira à l'action du code suivant :

```
Uri valeurs = Uri.parse("demo://Ceci est une chaîne de caractères");
Intent composerNuméro = new Intent(Intent.ACTION_VIEW, valeurs);
startActivity(composerNuméro);
```

***Nota** : Encore une fois, nous n'avons pas spécifié la classe du composant à utiliser pour cette action, ni explicitement quelle activité démarrer. C'est le système qui détermine seul le composant le mieux adapté en interrogeant les filtres définis dans les différents fichiers de configuration des applications installées.*

Toutes les catégories associées à la balise <category>

- **ALTERNATIVE** : Cette catégorie spécifie que l'action doit être disponible comme une alternative à l'action par défaut effectuée sur un élément de ce type de données. Par exemple, si l'action par défaut sur un contact est de l'affiché, l'alternative pourrait être de l'éditer.
- **SELECTED_ALTERNATIVE** : Semblable à la catégorie précédente mais, alors que cette dernière effectuera toujours la résolution en

une seule action en utilisant la résolution de l'**Intent** décrite à la suite, **SELECTED_ALTERNATIVE** sera utilisée lorsque plusieurs possibilités seront requises. Comme nous le verrons ultérieurement, l'un des usages est d'aider à remplir dynamiquement les menus contextuels à l'aide d'actions.

- **BROWSABLE** : Spécifie une action disponible dans le navigateur. Lorsqu'un **Intent** est déclenché depuis celui-ci, il comprend toujours cette catégorie. Vous devez inclure cette catégorie si vous voulez que votre application réponde aux actions déclenchées dans le navigateur (intercepter les liens vers un site web particulier, par exemple).
- **DEFAULT** : Fait d'un composant l'action par défaut pour le type de données spécifié. Ceci est nécessaire pour les activités lancées en utilisant un **Intent** explicite.
- **GADGET** : En utilisant cette catégorie, vous spécifiez que l'activité peut être exécutée au sein d'une autre activité.
- **HOME** : En utilisant cette catégorie sans spécifier d'action, vous la présentez comme une alternative à l'écran d'accueil natif.
- **LAUNCHER** : L'utilisation de cette catégorie fait apparaître l'activité dans le lanceur d'applications.

Tous les attributs possibles associés à la balise <data>

- **android:host** : Spécifie un nom d'hôte valide (**www.site.fr**, par exemple).
- **android:mimetype** : Vous permet de spécifier le type de données que votre composant est capable de prendre en charge.
- **android:path** : Spécifie les valeurs du chemin valides pour l'**URI**.
- **android:port** : Spécifie les ports valides pour l'hôte.
- **android:scheme** : En utilisant cette catégorie, vous spécifiez que l'activité peut être exécutée au sein d'une autre activité.

Exploiter l'objet Intent de l'activité

*Une fois le filtre de l'**Intent** défini et opérationnel, quelques lignes de code suffiront pour traiter l'**Intent** transmis. Si le composant n'est pas démarré, le système s'occupe de créer automatiquement une nouvelle instance du composant pour traiter l'intention. La classe **Intent** possède deux méthodes intéressantes pour récupérer les données envoyées sous forme d'**URI** :*

- La méthode ***getDataString()*** où nous pouvons consulter la chaîne de caractères au complet représentant l'**URI**.
- La méthode ***getData()*** qui nous renvoie cette fois-ci un objet de type **Uri**. Cette classe est généralement plus intéressante à utiliser puisqu'elle même dispose de méthodes plus spécifique pour récupérer une partie de l'**URI** soumise. Elle ressemble à ce sujet à la classe **URL** du Java standard que nous connaissons bien.

***Exemple de mise en œuvre** : Afin de valider cette notion de filtre, je vous propose d'en créer un sur une sous-activité. Nous allons reprendre le projet sur le calcul des angles et nous allons élaborer un filtre uniquement sur l'activité Radian. Cette sous-activité sera donc automatiquement affichée avec la valeur en degré soumise par une autre application qui servira au calcul et qui sera également automatiquement lancée.*

fr.btsiris.test.LancerAutreActivite.java

```
package fr.btsiris.test;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;

public class LancerAutreActivite extends Activity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		Uri commande = Uri.parse("radian://?degre=60");
		Intent radian = new Intent(Intent.ACTION_EDIT, commande);
		startActivity(radian);
	}
}
```

- Lorsque je soumets ma nouvelle intention dans cette application au travers de l'activité **LancerAutreActivite**, je demande à lancer l'action d'édition à l'aide de la constante **Intent.ACTION_EDIT**.
- J'aurais tout-à-fait pu demander plutôt une action de simple visualisation à l'aide de la constante **Intent.ACTION_VIEW**.
- L'**Uri** ***radian://?degre=60*** proposée pour soumettre les données est à la fois standard, mais également personnalisée. Effectivement le schéma est un schéma personnel et se nomme **"radian"**. Pour le reste de l'**Uri,** nous retrouvons une écriture plus classique avec le point d'interrogation, un paramètre (***degre***) ainsi que sa valeur (***60***).
- La particularité dans cette **Uri**, c'est que nous n'avons aucune action proposée avant le point d'interrogation. Il est tout-à-fait possible de proposer alors l'Uri suivante en ayant exactement la même réaction au niveau de la sous-activité : ***radian://calcul?degre=60***.
- A ce sujet, nous aurions pu prendre un autre exemple où c'est l'activité principale, le calcul des angles, qui prend en compte l'intention et qui redirige la requête vers la bonne sous-activité pour réaliser le traitement souhaité. Dans ce cas, il aurait été judicieux de proposer une **Uri** de la forme suivante pour effectuer un calcul qui permet de connaître une valeur en radian en partir d'un angle en degré : ***angle://radian?degre=60***.
- Cela vous montre toutes les possibilités. Android est vraiment très riche à ce sujet et ouvre plein de perspectives intéressantes.

*Voici toutes les modifications à apporter, d'une part dans le manifeste de l'application concernant le calcul des angles et dans le code Java propre à la sous-activité **Radian**.*

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="fr.btsiris.radian"
        android:versionCode="1"
        android:versionName="1.0">
    <application android:label="Angles" android:icon="@drawable/icon">
        <activity android:name="Choix" android:label="Conversion des angles">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Radian" android:label="Recherche du radian">
            <intent-filter>
                <action android:name="android.intent.action.EDIT" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:scheme="radian" />
            </intent-filter>
        </activity>
        <activity android:name="Degre"  />
    </application>
</manifest>
```

*Remarquez bien la présence du filtre d'intention sur la sous-activité **Radian**. Ce filtre stipule que la sous-activité sera automatiquement démarrée qu'à partir d'action de type **Edit** avec une catégorie par défaut (vous devez proposer impérativement une catégorie) et un schéma personnalisé nommé **radian**.*

fr.btsiris.radian.Radian.java

```
package fr.btsiris.radian;

public class Radian extends Conversion {

    @Override
    public void onCreate(Bundle icicle) {
        setContentView(R.layout.radian);
        super.onCreate(icicle);
        Uri requete = getIntent().getData();
        if (requete!=null) {
            String valeur = requete.getQueryParameter("degre");
            degre.setText(valeur+" °");
            calcul(degre);
        }
    }

    @Override
    public void calcul(View vue) {
        try {
            Number valeur = formatDegre.parse(degre.getText().toString());
            radian.setText(formatRadian.format(valeur.doubleValue() * Math.PI / 360));
        }
        catch (ParseException ex) { }
    }
}
```

- Cette activité peut être appelée normalement par son activité principale ou même par une autre activité d'une autre application. Il est donc nécessaire de savoir comment cette sous-activité a été générée.
- Il faut donc connaître l'intention de l'activité, au moyen de la méthode ***getIntent()*** et aussi savoir si une **Uri** a été proposée, au moyen de la méthode ***getData()***. Dans l'affirmative, c'est une autre application extérieure qui réclame son service.
- Dans ce dernier cas de figure, il faut récupérer la valeur de l'angle en degré donné par l'**Uri** au moyen de la méthode ***getQueryParameter()*** et soumettre cette valeur pour le calcul, après avoir formaté correctement la zone de saisie des degrés.

EMBARQUER DES DONNÉES SUPPLÉMENTAIRES

Lorsque vous demandez à un autre composant de réaliser une action, vous devrez bien souvent spécifier d'autres données supplémentaires. Par exemple, si vous souhaitez ouvrir une activité contenant un navigateur web, il y a de grande chance que cette dernière attende de vous une adresse web.

*A cette fin, la classe **Intent** dispose de méthodes grâce auxquelles un objet de type **Bundle** véhiculera vos données d'une activité à l'autre. Ainsi, l'insertion des données dans ce conteneur se fait au moyen de la méthode **putExtra()** et l'extraction au moyen de la méthode **getExtras()**.*

- L'échange des données se fait par l'association d'une **clé** (représentée par une chaîne de caractères) à une donnée à échanger.
- Pour ajouter chaque donnée à échanger, il faut appeler la méthode ***putExtra()*** avec la **clé** et la **donnée** à échanger. La méthode ***putExtra()*** dispose d'une surcharge pour pratiquement tous les types de données primitifs : **int, double, String, etc**.
- Ainsi, pour définir les données à échanger lors du démarrage d'une activité, la méthode ***putExtra()*** permet d'ajouter une information à l'action qui sera ensuite transmise à l'activité cible :

```
Intent intention = new Intent(this, ServiceDemarrer.class);
intention.putExtra("maclé", valeur);
startService(intention);
```

- Une fois l'intention reçue par l'application, via une instance de la classe **Intent**, vous pourrez récupérer les données contenues dans l'objet **Bundle** transmis à l'aide de la méthode ***getExtras()***.
- Une fois l'objet **Bundle** récupéré, appelez l'une des méthodes ***getXXX()*** correspondant au type de la donnée recherchée - ***getString(), getInt(), getDouble(), etc.*** - en spécifiant le nom de la **clé** associée pour récupérer les informations :

```
Bundle données = getIntent().getExtras();
if (données!=null) {
    int valeur = données.getInt("maclé");
    ...
}
```

- Il existe des méthodes encore plus rapides (***getIntExtra(), getDoubleExtra(), getStringExtra(), etc***.) qui permettent de récupérer une valeur à partir d'une **clé** sans passer par la classe Bundle, et qui permettent en même temps de proposer une valeur par défaut dans le cas où la valeur n'a pas été envoyée :

```
int valeur = getIntent().getIntExtra("maclé", 18);
```

Je vous propose de reprendre un des projets sur le calcul des angles en faisant en sorte que lorsque nous sortons d'un type de calcul, par exemple les radians, nous récupérons automatiquement la valeur calculée pour la soumettre à l'autre type de calcul, par exemple les degrés :

- Beaucoup de fichiers ne change pas. Ceux qui sont à prendre en compte pour respecter ce nouveau cahier des charges concerne l'activité principale **Choix** et les deux sous-activités permettant de réaliser les conversions, savoir **Radian** et **Degre**.

fr.btsiris.radian.Choix.java

```
package fr.btsiris.radian;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Choix extends Activity {
    private Intent radian, degre;
    private final int CALCUL_RADIAN =  1;
    private final int CALCUL_DEGRE = 2;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        radian = new Intent(this, Radian.class);
        degre = new Intent(this, Degre.class);
     }

     public void calculRadian(View vue) {
         radian.putExtra("degre", degre.getIntExtra("degre", 0));
         startActivityForResult(radian, CALCUL_RADIAN);
     }

     public void calculDegre(View vue) {
         degre.putExtra("radian", radian.getDoubleExtra("radian", 0.0));
         startActivityForResult(degre, CALCUL_DEGRE);
     }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode==RESULT_OK)
            switch (requestCode) {
                case CALCUL_RADIAN : radian = data; break;
                case CALCUL_DEGRE  : degre = data;  break;
            }
    }
}
```

- Dans les méthodes ***calculRadian()*** et ***calculDegre()***, nous envoyons dans l'intention concernée la valeur supplémentaire, au moyen de la méthode ***putExtra()***, qui tient compte du calcul effectué dans l'autre type de calcul, au moyen des méthodes respectives ***getIntExtra()*** et ***getDoubleExtra()***.
- Pour ces deux dernières méthodes, si aucun calcul antérieur n'a pas encore été proposé, une valeur par défaut est alors fournie.
- Toujours à l'intérieur de ces deux méthodes de calcul, le lancement de la sous-activité requise est ensuite effectué avec une attente d'un résultat possible, au moyen de la méthode ***startActivityForResult()***.
- La méthode ***onActivityResult()*** est alors redéfinie afin de permettre la récupération de la valeur de retour de la sous-activité qui se traduit tout simplement par la prise en compte de la nouvelle intention qui sera générée dans la sous-activité.

fr.btsiris.radian.Radian.java

```
package fr.btsiris.radian;

import android.os.Bundle;
import android.view.View;
import java.text.*;

public class Radian extends Conversion {

    @Override
    public void onCreate(Bundle icicle) {
        setContentView(R.layout.radian);
        super.onCreate(icicle);
    }

    @Override
    protected void onResume() {
        super.onResume();
        int valeur =getIntent().getIntExtra("degre", 0);
        degre.setText(formatDegre.format(valeur));
        radian.setText(formatRadian.format(valeur * Math.PI / 360));
    }
```

```
    @Override
    public void calcul(View vue) {
        try {
            Number valeur = formatDegre.parse(degre.getText().toString());
            double resultat = valeur.doubleValue() * Math.PI / 360;
            radian.setText(formatRadian.format(resultat));
            getIntent().putExtra("radian", resultat);
            setResult(RESULT_OK, getIntent());
        }
        catch (ParseException ex) { setResult(RESULT_CANCELED);  }
    }
}
```

fr.btsiris.radian.Degre.java

```
package fr.btsiris.radian;

import android.os.Bundle;
import android.view.View;
import java.text.*;

public class Degre extends Conversion {

    @Override
    public void onCreate(Bundle icicle) {
        setContentView(R.layout.degre);
        super.onCreate(icicle);
    }

    @Override
    protected void onResume() {
        super.onResume();
        double valeur = getIntent().getDoubleExtra("radian", 0.0);
        radian.setText(formatRadian.format(valeur));
        degre.setText(formatDegre.format(valeur * 360 / Math.PI));
    }

    @Override
    public void calcul(View vue) {
        try {
            Number valeur = formatRadian.parse(radian.getText().toString());
            int resultat = (int) (valeur.doubleValue() * 360 / Math.PI);
            degre.setText(formatDegre.format(resultat));
            getIntent().putExtra("degre", resultat);
            setResult(RESULT_OK, getIntent());
        }
        catch (ParseException ex) { setResult(RESULT_CANCELED); }
    }
}
```

- A chaque fois que la sous-activité doit repassée en premier plan, la méthode ***onResume()***, intégrée dans le cycle de vie d'une activité, est appelée.
- Nous profitons de l'occasion pour récupérer la valeur supplémentaire proposée avec l'intention, au moyen des méthodes respectives ***getDoubleExtra()*** et ***getIntExtra()***.
- A chaque fois qu'un calcul est demandé, nous mettons à jour l'intention avec la prise en compte du nouveau calcul au moyen de la méthode ***putExtra()***, et nous soumettons cette intention comme résultat de la sous-activité au moyen de la méthode ***setResult()***.