RxJava (http://developer.alexanderklimov.ru/android/rx/)

Советы (http://developer.alexanderklimov.ru/android/tips-android.php)

Статьи (http://developer.alexanderklimov.ru/android/articles-android.php)

Книги (http://developer.alexanderklimov.ru/android/books.php)

Java (http://developer.alexanderklimov.ru/android/java/java.php)

Kotlin (http://developer.alexanderklimov.ru/android/kotlin/)

Дизайн (http://developer.alexanderklimov.ru/android/design/)

Отладка (http://developer.alexanderklimov.ru/android/debug/)

Open Source (http://developer.alexanderklimov.ru/android/opensource.php)

Полезные ресурсы (http://developer.alexanderklimov.ru/android/links.php)

Fragment (Фрагменты). Часть вторая

Поняв, как создавать фрагмент, можно переходить к следующей части - как взаимодействовать с фрагментами.

Поговорим о важном моменте. Вы можете установить связь между двумя фрагментами напрямую, чтобы при нажатии кнопки в первом фрагменте менялось содержимое во втором фрагменте. Но это неправильный подход, так как теряется смысл модульности фрагментов. Фрагменты ничего не должны знать о существовании друг друга. Любой фрагмент существует только в активности и только активность через свой специальный менеджер фрагментов должна управлять ими. А сами фрагменты должны реализовать необходимые интерфейсы, которые активность будет использовать в своих целях.

Начнём издалека. В первом фрагменте имеются кнопки. Добавим обработчик нажатий кнопок (такой же код вы могли использовать в активности, всё знакомо):

```
public class Fragment1 extends Fragment implements View.OnClickListener {
    ...

    @Override
        public void onClick(View view) {

        }
}
```

Подключаем кнопки в методе **onCreateView()**. Код будет похож на код, который мы обычно используем в методе **onCreate()** у активности, только метод **findViewById()** будет относиться уже не к классу **Activity** (обычно, мы опускали это), а к корневому элементу разметки фрагмента, в нашем случае **rootView**.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
    View rootView =
            inflater.inflate(R.layout.fragment1, container, false);

    Button button1 = (Button) rootView.findViewById(R.id.button1);
    Button button2 = (Button) rootView.findViewById(R.id.button2);
    Button button3 = (Button) rootView.findViewById(R.id.button3);

    button1.setOnClickListener(this);
    button2.setOnClickListener(this);
    button3.setOnClickListener(this);

    return rootView;
}
```

Для начала просто выведем сообщение, что кнопка нажата.

```
@Override
public void onClick(View view) {
    Toast.makeText(getActivity(), "Вы нажали на кнопку",
            Toast.LENGTH_SHORT).show();
}
```

Запустите пример и проверьте. Но у нас три кнопки. Надо написать код, который бы получал информацию о нажатой кнопке, чтобы активность могла использовать эту информацию и использовать её для управления вторым фрагментом. Для удобства создадим отдельный метод, который на основании идентификатора кнопки создаст нужный индекс:

```
int translateIdToIndex(int id) {
        int index = -1;
        switch (id) {
        case R.id.button1:
                index = 1;
                break;
        case R.id.button2:
                index = 2;
                break;
        case R.id.button3:
                index = 3;
                break;
        }
        return index;
}
```

Каждой кнопке соответствует свой индекс от 1 до 3.

Фрагмент всегда может узнать, в какой активности он находится, через метод **getActivity()**. В методе **makeText()** мы уже воспользовались данным методом, так как в фрагментах нет метода **getApplicationContext()**.

Перепишем код для щелчка кнопки, чтобы узнать индекс нажатой кнопки.

```
@Override
public void onClick(View view) {
    int buttonIndex = translateIdToIndex(v.getId());

    // Временный код для получения индекса нажатой кнопки
    Toast.makeText(getActivity(), String.valueOf(buttonIndex),
            Toast.LENGTH_SHORT).show();
}
```

Теперь мы умеем определять индекс нажатой кнопки. Но пока эта информация доступна только самому фрагменту. Наша задача - передать эту информацию активности, которая затем передаст её другой активности.

Для этой цели используются интерфейсы.

Открываем код первого фрагмента **Fragment1** и объявляем интерфейс с единственным методом:

```
public interface OnSelectedButtonListener {
        void onButtonSelected(int buttonIndex);
}
```

Интерфейс не определяет работу метода, а только даёт ему имя. Класс, который будет использовать данный интерфейс, должен придумать, что делать в методе с данным именем.

У нас интерфейс будет использовать класс активности.

Переходим в класс активности и добавляем интерфейс **OnSelectedButtonListener**, который следует реализовать.

```
public class MainActivity extends AppCompatActivity implements Fragment1.OnSelectedButtonList
ener { ...
}
```

Среда разработки поможет создать заготовку для необходимого метода:

```
@Override
public void onButtonSelected(int buttonIndex) {

}
```

В этом методе надо написать такой код, чтобы активность получила индекс нажатой кнопки и передала информацию другому фрагменту, которая должна выполнить свою работу.

Но сначала подготовим второй фрагмент к работе.

Объявим ссылки на компоненты, которые есть в разметке второго фрагмента. А также загрузим массив строк из ресурсов, который будем использовать для описания котов.

```
public class Fragment2 extends Fragment {
    private TextView mInfoTextView;
    private ImageView mCatImageView;
    private String[] mCatDescriptionArray;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView =
                inflater.inflate(R.layout.fragment2, container, false);

        mInfoTextView = (TextView) rootView.findViewById(R.id.textView);
        mCatImageView = (ImageView) rootView.findViewById(R.id.imageView);

        // загружаем массив из ресурсов
        mCatDescriptionArray = getResources().getStringArray(R.array.cats);

        return rootView;
    }
}
```

Массив задаём в ресурсах (файл **res/values/strings.xml**). Так как первый элемент массива идёт под индексом 0, то добавим нейтральный текст:

```
<string-array name="cats">
    <item>Просто кот</item>
    <item>Рыжик - рыжий кот</item>
    <item>Барсик болеет за Барселону</item>
    <item>Мурзик выписывает Мурзилку</item>
</string-array>
```

Подготовим метод, который будет менять содержимое фрагмента в зависимости от индекса нажатой кнопки:

```
public void setDescription(int buttonIndex) {
    String catDescription = mCatDescriptionArray[buttonIndex];
    mInfoTextView.setText(catDescription);

    switch (buttonIndex) {
        case 1:
            mCatImageView.setImageResource(R.drawable.cat_yellow);
            break;
        case 2:
            mCatImageView.setImageResource(R.drawable.cat_white);
            break;
        case 3:
            mCatImageView.setImageResource(R.drawable.cat_green);
            break;

        default:
            break;
    }
}
```

Осталось только получить информацию от активности (не от фрагмента) об индексе.

Опять возвращаемся в активность и напишем код для пустого метода **onButtonSelected()**, который будет получать от первого фрагмента индекс нажатой кнопки и передавать его второму фрагменту:

```
@Override
public void onButtonSelected(int buttonIndex) {
    // подключаем FragmentManager
    FragmentManager fragmentManager = getSupportFragmentManager();

    // Получаем ссылку на второй фрагмент по ID
    Fragment2 fragment2 = (Fragment2) fragmentManager
            .findFragmentById(R.id.fragment2);

    // Выводим нужную информацию
    if (fragment2 != null)
        fragment2.setDescription(buttonIndex);
}
```

Активность получает доступ к своим фрагментам через специальный менеджер фрагментов (коты называют его манагером). Менеджер есть у любой активности, поэтому мы его не создаём через конструкцию **new FragmentManager**, а получаем через метод **getSupportFragmentManager()** или для новых версий **getFragmentManager()**.

Так было не всегда, менеджеры появились у активностей только в Android 3.0. А старые активности понятия не имели о них. По этой причине и добавили библиотеку совместимости, в которой переписали код, добавив новые классы в пакетах **android.support.xxx**, чтобы новый тип активностей работал на старых устройствах. По сути, библиотека является патчем к операционной системе.

Менеджер фрагментов держит в руках все нити управления над своими фрагментами. Найти нужный фрагмент можно по идентификатору через метод **FragmentManager.findFragmentById()**, который похож на метод **findViewById()** для поиска кнопки, метки и т.д. У менеджера есть ещё один метод для поиска фрагмента по тегу **findFragmentByTag()**.

В созданной заготовке вызываем менеджер фрагментов, получаем ссылку на второй фрагмент через его идентификатор и вызываем его метод **setDescription()**.

Во втором фрагменте у нас нет надобности создавать интерфейс, так как фрагменту не нужно ничего сообщать активности. Он исполняет пассивную роль и ему нужно только получить данные для работы.

Опять возвращаемся к классу первого фрагмента. Сейчас нажатие кнопки приводит к появлению всплывающего сообщения об индексе кнопки.

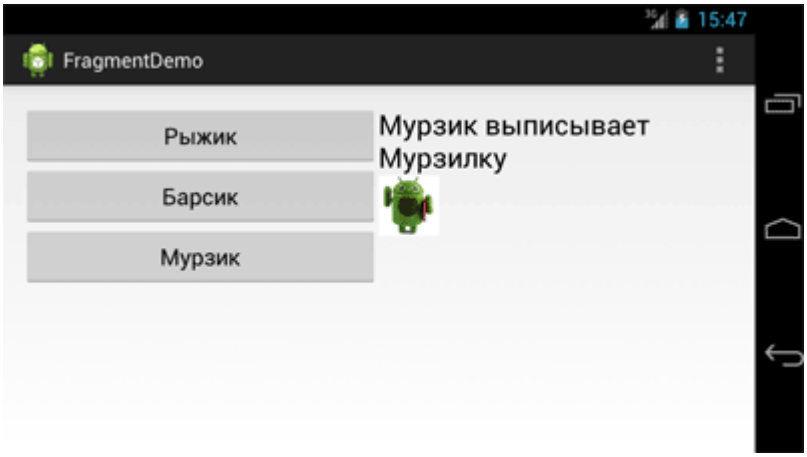
Но теперь в методе **onClick()** мы можем получить доступ к слушателю активности.

```
@Override
public void onClick(View view) {
        int buttonIndex = translateIdToIndex(view.getId());

        OnSelectedButtonListener listener = (OnSelectedButtonListener) getActivity();
        listener.onButtonSelected(buttonIndex);

        // Можно закомментировать
        Toast.makeText(getActivity(), String.valueOf(buttonIndex),
                        Toast.LENGTH_SHORT).show();
}
```

По цепочке мы передаём информацию от первого фрагмента в активность, а затем активность передаёт информацию во второй фрагмент.

Если посмотреть на код двух фрагментов, то увидим, что они полностью независимы и не обращаются ни конкретно к друг другу, ни к определённой активности. Принцип модульности соблюдён. Вы можете добавить любой из этих фрагментов в любую новую активность и при этом вам не придётся менять код в самих фрагментах. Весь необходимый функционал в фрагментах уже прописан.

Запустите проект и проверьте на работоспособность. Для данного случая мы пока не получили никаких преимуществ в использовании фрагментов. Но сейчас главное для вас - понять основные принципы создания и взаимодействия фрагментов.



Часть третья. Продолжение (fragment3.php)

Дополнительное чтение

Обсуждение статьи (http://forum.alexanderklimov.ru/viewtopic.php?id=31) на форуме.