Научимся обращаться из кода к View­-элементам на экране и менять их свойства.

Для начала создадим проект:

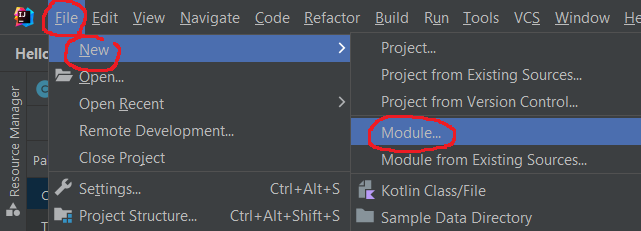


Рис. 1.

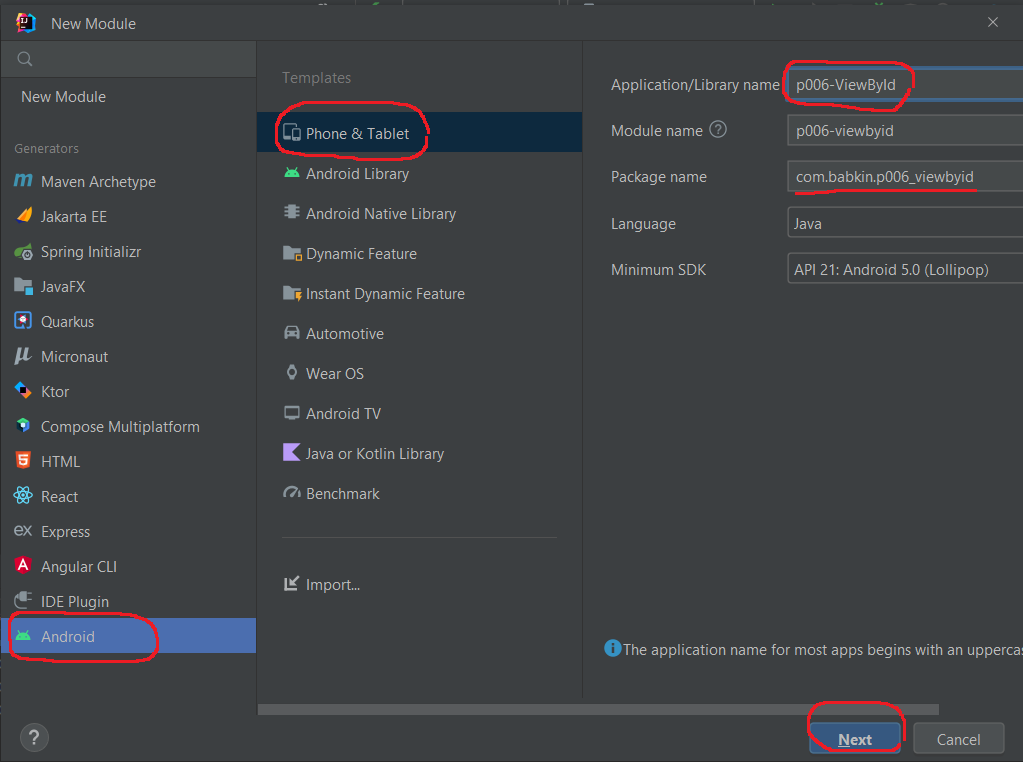


Рис. 2.

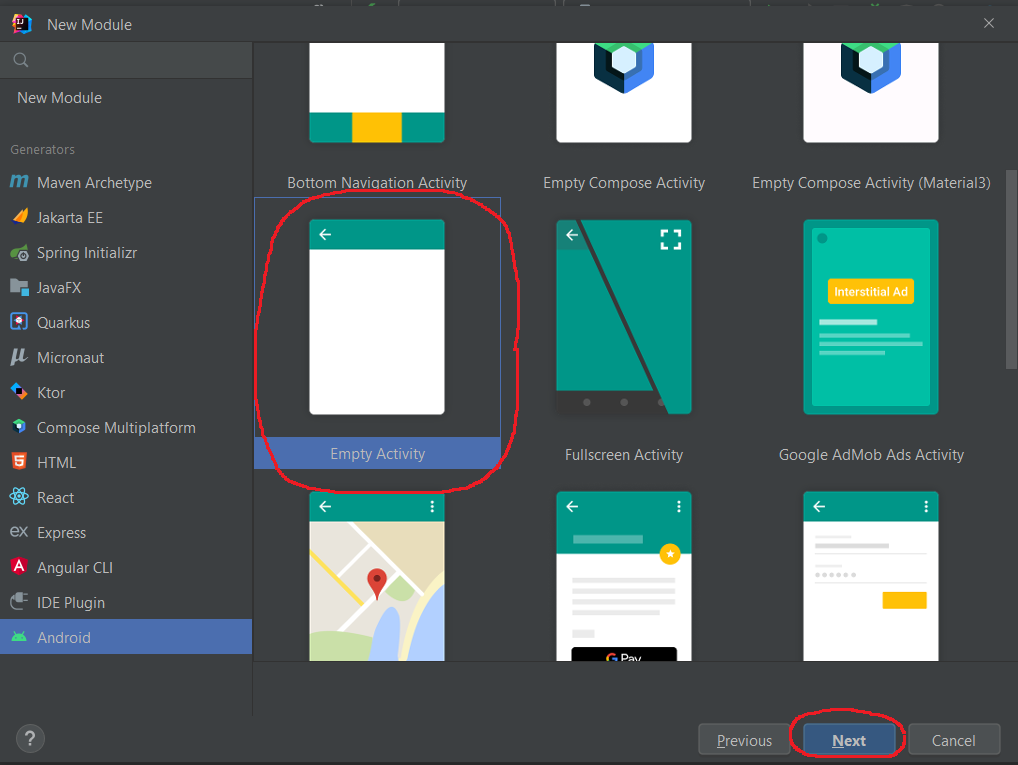


Рис. 3.

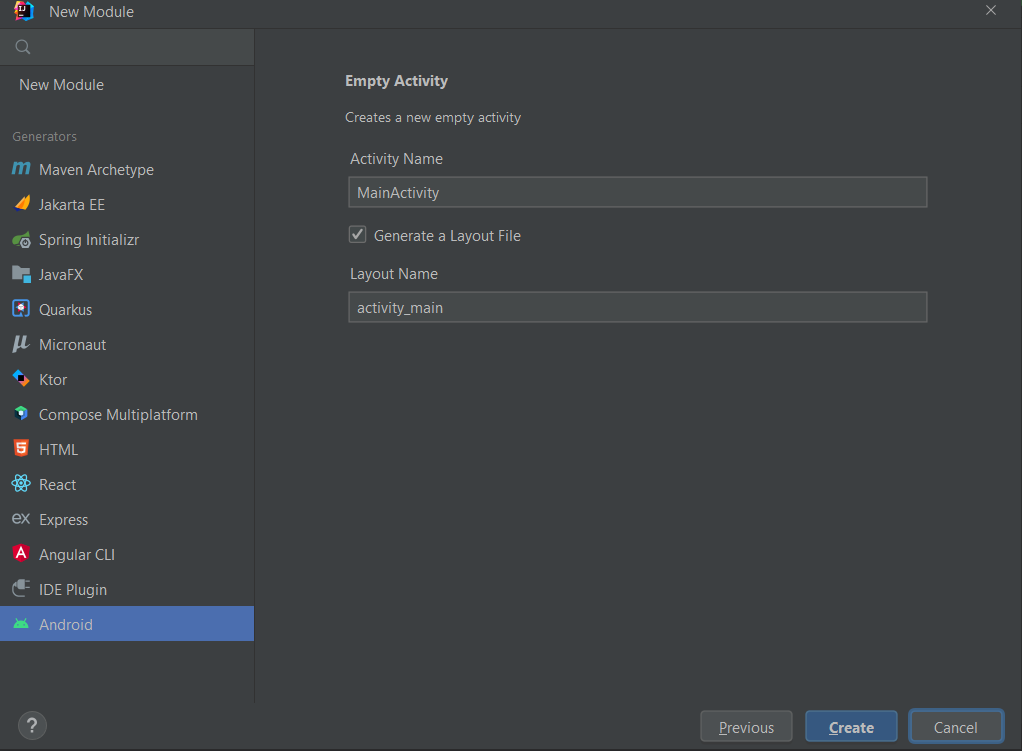


Рис. 4.

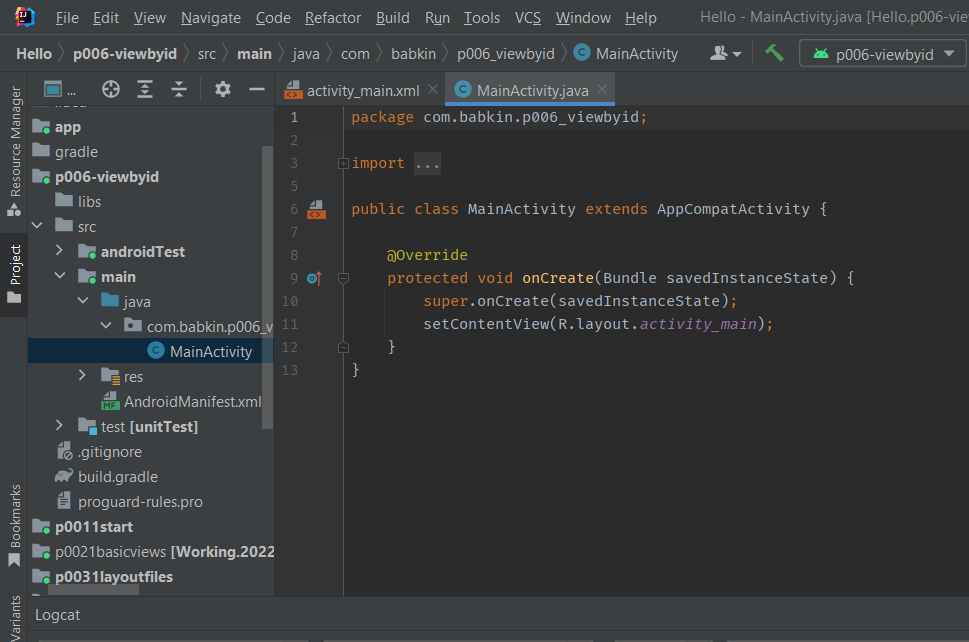


Рис. 5.

Чтобы обратиться к элементу экрана из кода, нам нужен его ID. Он прописывается либо в Properties, либо в layout-­файлах, как вам удобнее. Для ID существует четкий формат ­ @+id/name, где + означает, что это новый ресурс и он должен добавиться в R.java класс, если он там еще не существует.

Давайте откроем main.xml, для TextView укажем ID = @+id/myText и сохраним

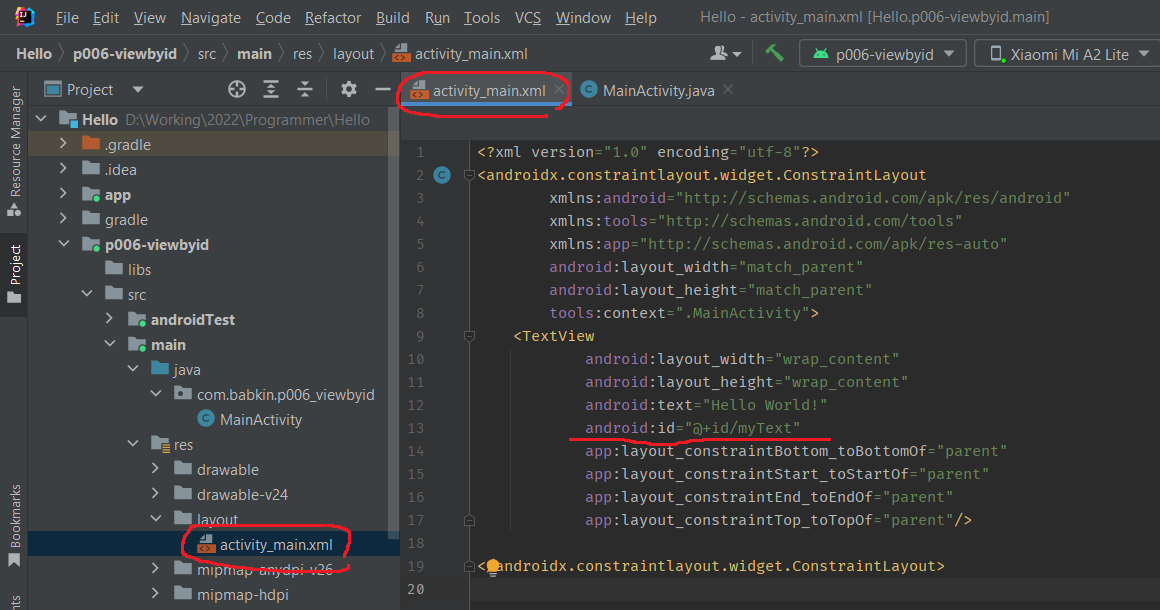


Рис. 6.

Теперь в режиме дизайна видим, что для класса id появилась константа myText. Т.е. чтобы к ней обратиться, надо написать R.id.myText

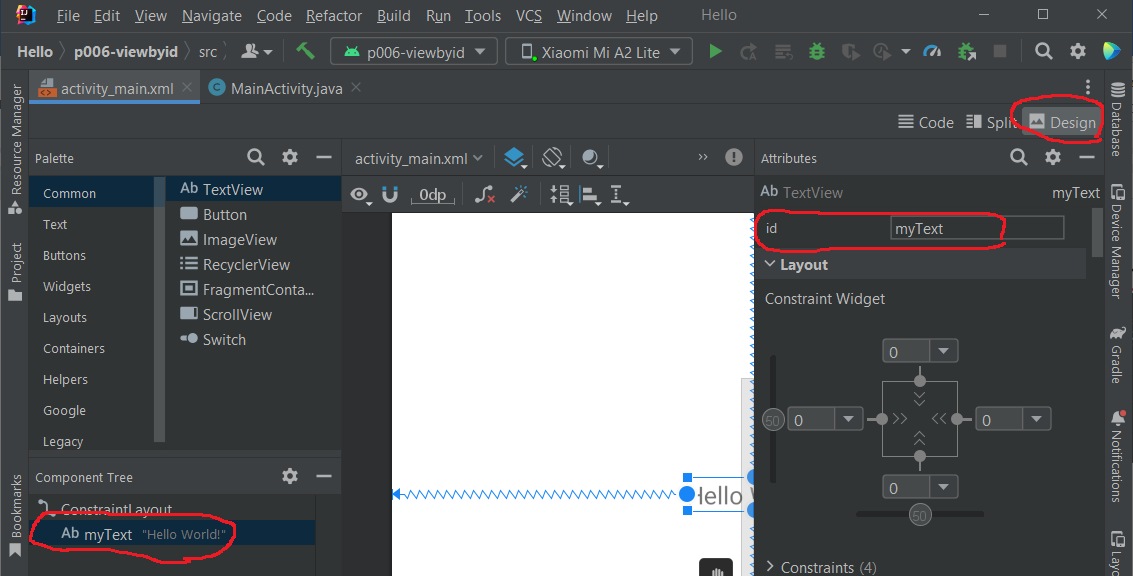


Рис. 7.

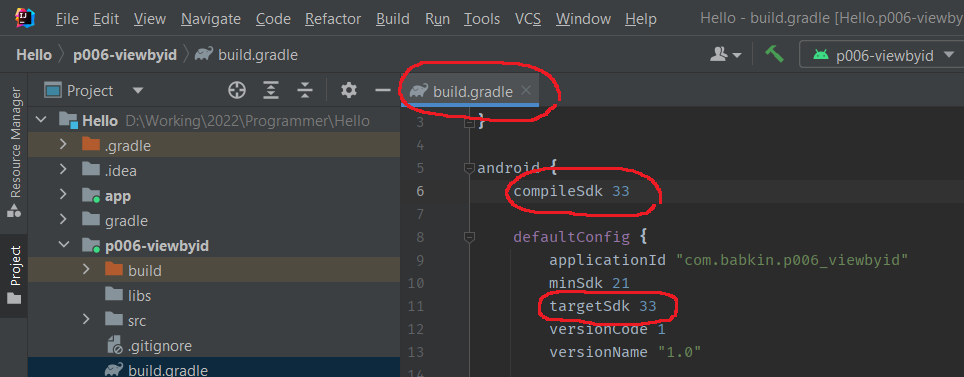


Рис. 8.



Рис. 9.

Она связана с элементом TextView и мы можем ее использовать, чтобы обратиться к элементу программно. Для этого нам понадобится метод findViewById

Он по ID возвращает View. Давайте напишем вызов этого метода.

Напомню, что пока мы пишем наш код в методе onCreate. Это метод, который вызывается при создании Activity.

Откроем MainActivity.java и после строки с вызовом метода setContentView напишем:

View myTextView = findViewById(R.id.myText);

Если View подчеркнуто красным, то скорей всего этот класс не добавлен в секцию import

Нажмите CTRL+SHIFT+O для автоматического обновления импорта.  
Теперь myTextView (типа View) – это наш TextView на экране. Но тип View (http://developer.android.com/reference/android/view/View.html) – это предок для TextView (и остальных View-элементов). И он нам не подходит, если мы хотим проделывать операции соответствующие TextView. Поэтому нам необходимо приведение View в TextView. Изменим наш код на следующий:

Теперь myTextView имеет тип TextView, а результат метода findViewById мы приводим от View к TextView. Теперь мы можем применять к myTextView методы класса TextView. Для примера возьмем метод setText. Сейчас отображаемый текст = Hello World, MainActivity!. Мы его программно поменяем на New text in TextView

Сохраняем, запускаем (CTRL+F11) и видим, что текст изменился

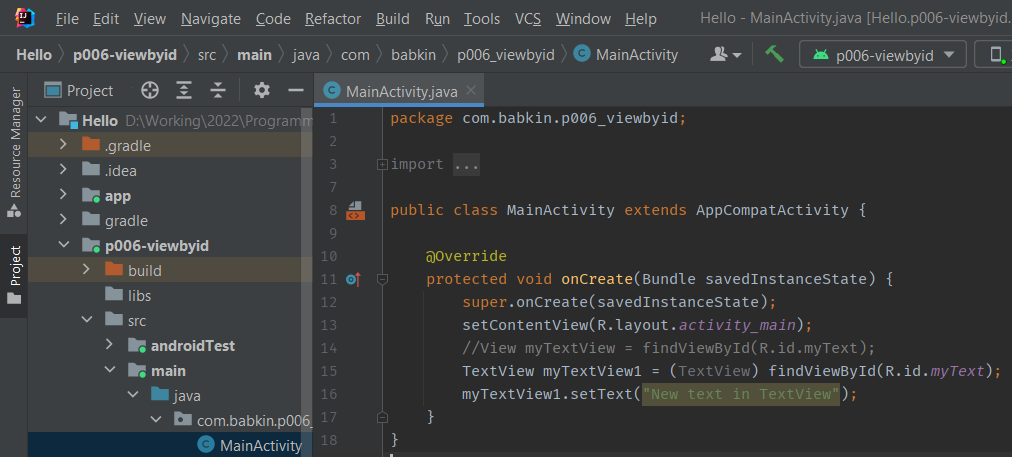


Рис. 10.

Добавим на экран кнопку (Button), Id = @+id/myBtn, текст оставим по умолчанию. Сохраняем ­ CTRL+SHIFT+S (если не сохранить, то в R.java не появитсяID).

Пишем код:

Button myBtn = (Button) findViewById(R.id.myBtn);

Обратите внимание, что у меня совпадает имя объекта и ID. Они друг другу не мешают и так делать даже логичнее. Это остается на ваше усмотрение. Так, кнопку мы нашли, теперь давайте изменим ее текст:

myBtn.setText("My button");

Запустим приложение.

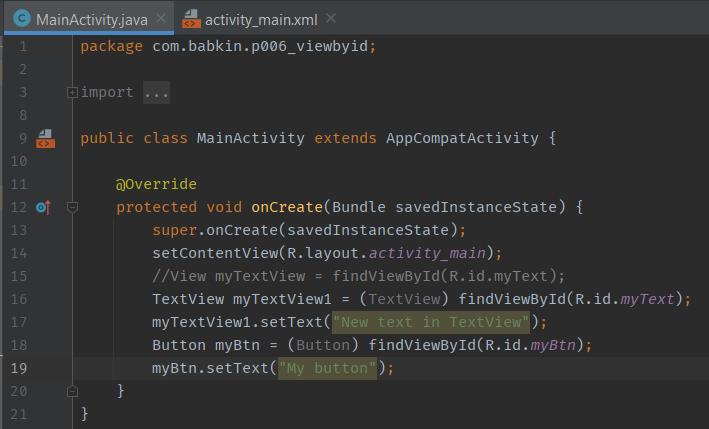


Рис. 11.

Рис. 12.

Текст на кнопке поменялся, на кнопку можно понажимать, но ничего происходить не будет. Т.к. мы нигде не указывали, что надо делать при нажатии. Этим займемся в следующий раз. А пока давайте сделаем кнопку неактивной.

myBtn.setEnabled(false);

Мы поменяли параметр Enabled.

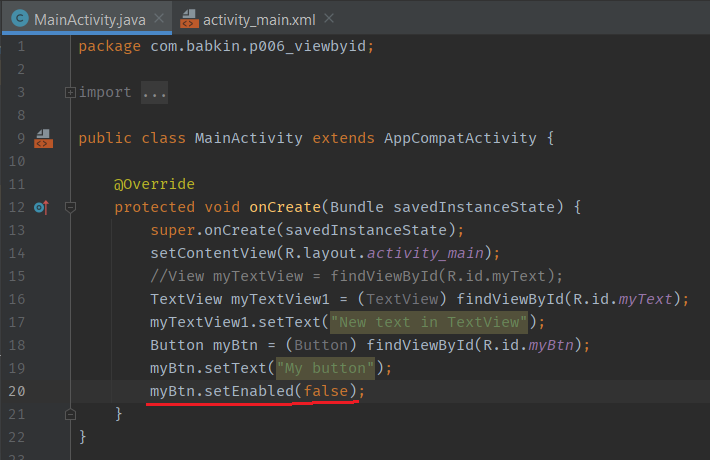


Рис. 13.

Теперь на кнопку нельзя нажать. Сохраним, запустим и убедимся.

Рис. 14

Добавим CheckBox, id = @+id/myChb. По умолчанию галочка не стоит. Давайте поставим ее программно, для этого используется метод setChecked, который меняет параметр Checked.

CheckBox myChb = (CheckBox) findViewById(R.id.myChb);

myChb.setChecked(true);

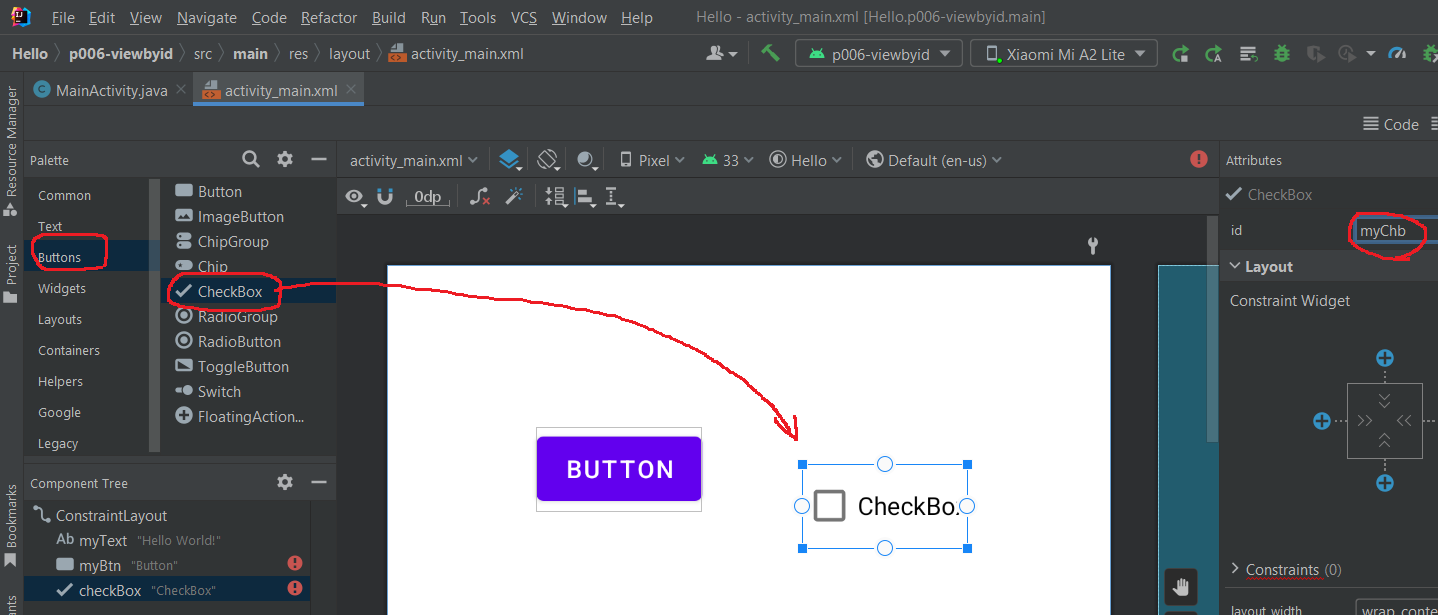


Рис. 15

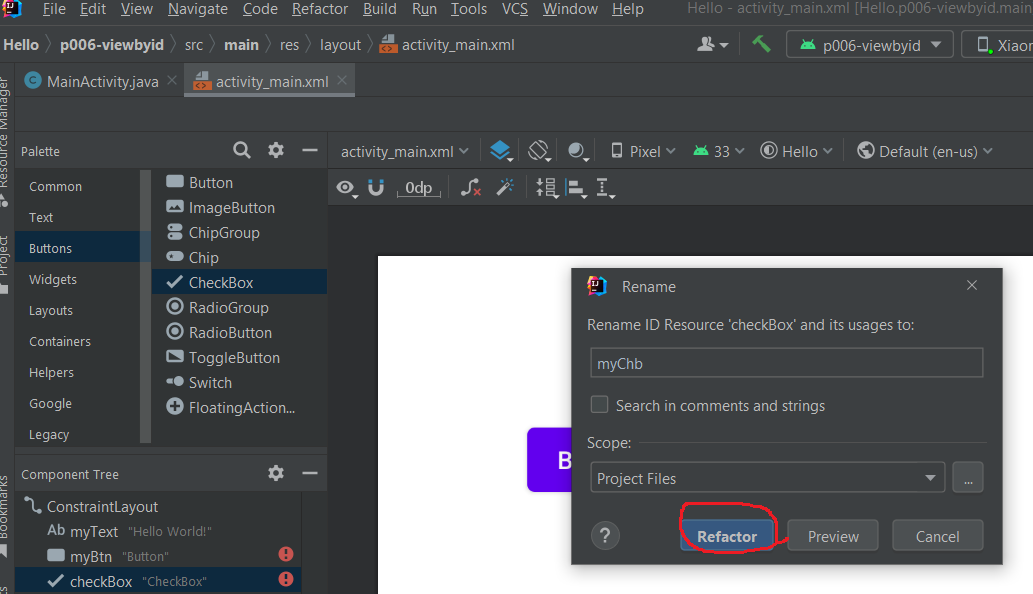


Рис. 16.

Запустив приложение видим, что код сработал.



Как видите – все несложно. Используем метод findViewById, чтобы по ID получить объект соответствующий какому-либо View-элементу (Button, TextView, CheckBox) и далее вызываем необходимые методы объектов (setText, setEnabled, setChecked).

В итоге должен получиться такой код:

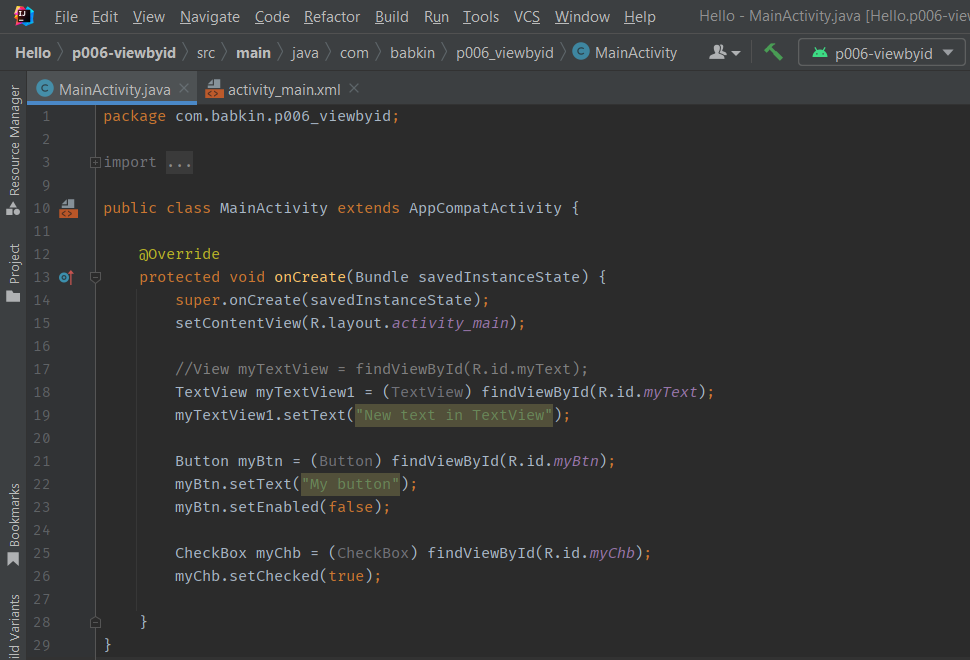


Рис. 17.