[**(转)Android EditText限制输入字符的5种实现方式**](http://www.cnblogs.com/1925yiyi/p/7567205.html)

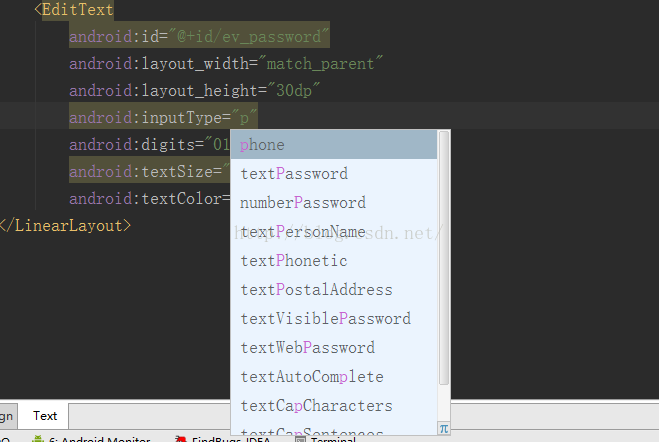
  最近项目要求限制密码输入的字符类型， 例如不能输入中文。   现在总结一下EditText的各种实现方式，  以比较各种方法的优劣。

 第一种方式：  设置EditText的inputType属性，可以通过xml或者java文件来设置。假如我要设置为显示密码的形式，可以像下面这样设置：

在xml中，   android:inputType="textPassword"

在java文件中，可以用 ev.setInputType(InputType.TYPE\_TEXT\_VARIATION\_PASSWORD);

      inputType参数有phone、textPasswrod等等属性， 有兴趣的话可以测试一下。



第二种： 设置EditText的android:digits 属性， 这种方式可以指出要支持的字符。比如要限制只能输入数字和字母，可以这样：

        android:digits="0123456789abcdefghijklmnopqrstuvwxyz"。 PS：吐槽一下，写着好累，不支持区间操作符~或-

第三种： 使用TextWather监听EditText的字符变化， 当内容改变时删掉不想要的字符。说白了就是用户可以随便按键盘，在代码里马上删掉非法字符。 PS： 常见的搜索功能输入框就是用TextWatcher监测关键词变化，然后查数据并显示出来了。示例代码：只能输入字母或者数字， 如果要支持其它字符可以修改regEx正则表达式。

[复制代码](javascript:void(0);)

evPwd = (EditText) findViewById(R.id.ev\_password);

evPwd.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

String editable = evPwd.getText().toString();

String regEx = "[^a-zA-Z0-9]"; //只能输入字母或数字

Pattern p = Pattern.compile(regEx);

Matcher m = p.matcher(editable);

String str = m.replaceAll("").trim(); //删掉不是字母或数字的字符

if(!editable.equals(str)){

evPwd.setText(str); //设置EditText的字符

evPwd.setSelection(str.length()); //因为删除了字符，要重写设置新的光标所在位置

}

}

[复制代码](javascript:void(0);)

第四种：通过InputFilter来实现。实现InputFilter过滤器， 需要覆盖一个叫filter的方法。  
public abstract CharSequence filter (   
    CharSequence source,  //输入的文字   
    int start,  //开始位置   
    int end,  //结束位置   
    Spanned dest, //当前显示的内容   
    int dstart,  //当前开始位置   
    int dend //当前结束位置   
);  
注意：**IntentFilter是个数组， 也就是说可以写多个过滤条件！**  
下面的实现使得EditText只接收字符（数字、字母），Character.isLetterOrDigit会把中文也当做Letter， 所以要在写个正则判断是否中文。

[复制代码](javascript:void(0);)

evPwd.setFilters(new InputFilter[]{

new InputFilter() {

@Override

public CharSequence filter(CharSequence charSequence, int i, int i1, Spanned spanned, int i2, int i3) {

String regex = "^[\u4E00-\u9FA5]+$";

boolean isChinese = Pattern.matches(regex, charSequence.toString());

if (!Character.isLetterOrDigit(charSequence.charAt(i)) || isChinese) {

return "";

}

return null;

}

}

});

[复制代码](javascript:void(0);)

第五种：    使用EditText的InputConnection属性限制输入字符。 新建个类继承于EditText并覆盖onCreateInputConnection函数， 在xml里使用LimitText替换EditText。

[复制代码](javascript:void(0);)

public class LimitEditText extends EditText {

public LimitEditText(Context context) {

super(context);

}

public LimitEditText(Context context, AttributeSet attrs) {

super(context, attrs);

}

public LimitEditText(Context context, AttributeSet attrs, int defStyleAttr) {

super(context, attrs, defStyleAttr);

}

/\*\*

\* 输入法

\* @param outAttrs

\* @return

\*/

@Override

public InputConnection onCreateInputConnection(EditorInfo outAttrs) {

return new InnerInputConnecttion(super.onCreateInputConnection(outAttrs),

false);

}

class InnerInputConnecttion extends InputConnectionWrapper implements InputConnection {

public mInputConnecttion(InputConnection target, boolean mutable) {

super(target, mutable);

}

/\*\*

\* 对输入的内容进行拦截

\*

\* @param text

\* @param newCursorPosition

\* @return

\*/

@Override

public boolean commitText(CharSequence text, int newCursorPosition) {

// 只能输入字母或者数字

if (!Character.isLetterOrDigit(charSequence.charAt(i)) || isChinese) {

return false;

}

return super.commitText(text, newCursorPosition);

}

@Override

public boolean sendKeyEvent(KeyEvent event) {

return super.sendKeyEvent(event);

}

@Override

public boolean setSelection(int start, int end) {

return super.setSelection(start, end);

}

}

}