TableLayout и TableRow

Разметка **TableLayout** (Табличная разметка) позиционирует свои дочерние элементы в строки и столбцы, как это привыкли делать веб-мастера в теге **table**. TableLayout не отображает линии обрамления для их строк, столбцов или ячеек. TableLayout может иметь строки с разным количеством ячеек. При формировании разметки таблицы некоторые ячейки при необходимости можно оставлять пустыми. При создании разметки для строк используются объекты **TableRow**, которые являются дочерними классами TableLayout (каждый TableRow определяет единственную строку в таблице). Строка может не иметь ячеек или иметь одну и более ячеек, которые являются контейнерами для других объектов. В ячейку допускается вкладывать другой TableLayout или LinearLayout.

TableLayout удобно использовать, например, при создании логических игр типа Судоку, Крестики-Нолики и т.п.

Вот несколько правил для TableLayout. Во-первых, ширина каждой колонки определяется по наиболее широкому содержимому в колонке. Дочерние элементы используют в атрибутах значение **match\_parent**. Атрибут TableRow для **layout\_height** всегда *wrap\_content*. Ячейки могут объединять колонки, но не ряды. Достигается слияние колонок через атрибут **layout\_span**.

Если атрибуту **android:stretchColumns** компонента TableLayout присвоить значение "\*", то содержимое каждого компонента TableRow может растягиваться на всю ширину макета.

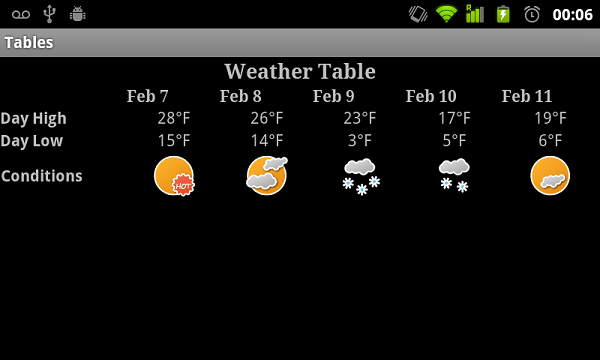
Создаем таблицу прогноза погоды

Например, для создания таблицы прогноза погоды можно использовать следующий дизайн:

* В первом ряду показать заголовок таблицы
* Во втором ряду показать даты по типу календаря
* В третьем ряду показать наибольшую температуру
* В четвёртом ряду показать наименьшую температуру
* В пятом ряду показать графику (дождь, снег, облачность, солнце)

Показать код (щелкните мышкой)

В альбомной ориентации наша разметка будет выглядеть следующим образом



Программное создание TableLayout

Показать код (щелкните мышкой)

Усадка, усушка, утруска

Если текст в ячейке таблицы слишком длинный, то он может растянуть ячейку таким образом, что часть текста просто выйдет за пределы видимости. Чтобы избежать данной проблемы, у контейнтера TableLayout есть атрибут **android:shrinkColumns**. Мы рассмотрим программное применение данного атрибута через метод **setColumnShrinkable()**.

Для начала смоделируем ситуацию и поместим в одну из ячеек очень длинный текст, который не поместится на экран:

<?xml version="1.0" encoding="utf-8"?>

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<LinearLayout

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TableLayout

android:id="@+id/tablelayout"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content">

<TableRow>

<TextView

android:text="Щелкните на кнопке Усадка, чтобы увидеть, как длинный текст может уместиться в ячейке таблицы"

android:padding="3dip" />

</TableRow>

</TableLayout>

<Button

android:id="@+id/toggle"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Усадка" />

</LinearLayout>

</ScrollView>

Теперь напишем код для усадки текста:

public class TableActivity extends Activity {

private boolean mShrink;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

final TableLayout table = (TableLayout) findViewById(R.id.tablelayout);

Button button = (Button) findViewById(R.id.toggle);

button.setOnClickListener(new Button.OnClickListener() {

public void onClick(View v) {

mShrink = !mShrink;

table.setColumnShrinkable(0, mShrink);

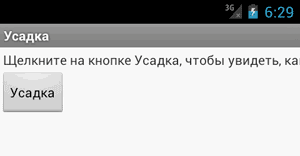
}

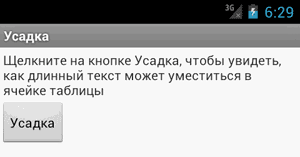
});

mShrink = table.isColumnShrinkable(0);

}

}





Книжная полка

Создадим подобие книжной полки. Нам понадобится изображение одной полки, которая послужит фоном и изображение книги (найдите сами).



Подготовим разметку.

<?xml version="1.0" encoding="utf-8"?>

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/scrollView"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" >

<LinearLayout

android:id="@+id/linearLayout"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<ImageView

android:id="@+id/imageView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="right"

android:src="@drawable/cat\_bottom" />

<TableLayout

android:id="@+id/tableLayout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" >

</TableLayout>

</LinearLayout>

</ScrollView>

Объявим две константы, которые отвечают за количество полок и количество книг на каждой полке.

package ru.alexanderklimov.test;

import android.app.Activity;

import android.os.Bundle;

import android.view.ViewGroup.LayoutParams;

import android.widget.ImageView;

import android.widget.TableLayout;

import android.widget.TableRow;

public class MainActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

int BOOKSHELF\_ROWS = 5;

int BOOKSHELF\_COLUMNS = 5;

TableLayout tableLayout = (TableLayout) findViewById(R.id.tableLayout);

for (int i = 0; i < BOOKSHELF\_ROWS; i++) {

TableRow tableRow = new TableRow(this);

tableRow.setLayoutParams(new LayoutParams(LayoutParams.MATCH\_PARENT,

LayoutParams.WRAP\_CONTENT));

tableRow.setBackgroundResource(R.drawable.shelf);

for (int j = 0; j < BOOKSHELF\_COLUMNS; j++) {

ImageView imageView = new ImageView(this);

imageView.setImageResource(R.drawable.book);

tableRow.addView(imageView, j);

}

tableLayout.addView(tableRow, i);

}

}

}

