参考代码 (/tutorial/a-simple-sample-for-android-development/repo)

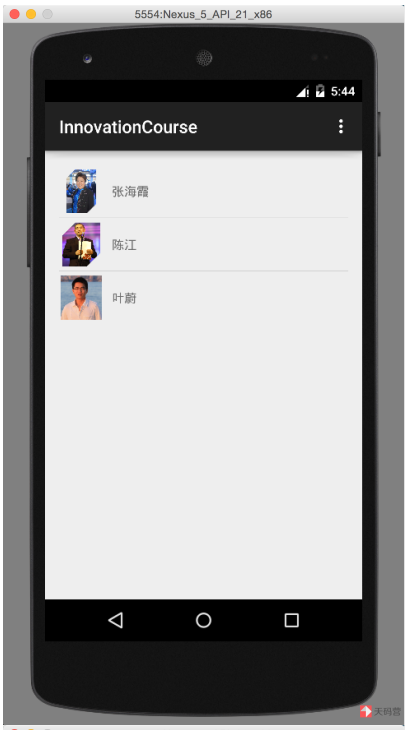
Android开发入门经典实例

David (/user/david) 发布于 2月16日 0评论 7405浏览

android (/tag/android/tutorials) android-studio (/tag/android-studio/tutorials) 1 4

开发实例概述

今天带大家做一个简单的Android App，这个App会显示创新工程实践老师们的照片和信息，不妨先看一看效果：



(http://assets.tianmaying.com/md-image/fec348f2aafa30d8aab40d8ee26134ba.png)

(http://assets.tianmaying.com/md-image/3521a11ffc26bb9e08ee099bde6bad52.png)

虽然这个App非常简单，但是涉及到了Android开发中的一些关键知识，比如：

- 配置开发环境
- App中一个屏幕的抽象： `Activity`
- 屏幕之间的跳转： `Intent`
- 构成屏幕展示的视图组件：显示图片的 `ImageView` ，显示文字的 `TextView` ，把组件组成一个列表的视图 `ListView`
- 通过 `Adapter` 来控制模型和视图组件之间通信，即如何在视图上展示特定的数据
- 通过事件来处理用户的交互： `OnClickListener`

让我们开始吧！

建立开发环境

工欲善其事，必先利其器

在编写代码之前，我们需要在我们的计算机中配置集成开发环境（IDE，Integrated Development Environment），它是用来开发和部署软件的工具集合。

如果你之前有过编程的经历，那么一定会知道最重要的工具是编译器——它将我们用高级语言编写的源代码转换成计算机可以执行的指令来完成指定的任务。

开发环境的安装和配置往往需要花费不少时间。对于不同的计算机以及操作系统平台，安装配置过程都不尽相同，会出现很多不可预测的错误—。新手见到这些错误往往会一头雾水、不知如何解决。

好的开始等于成功的一般，开始学习前最重要的一个步骤就是完成开发环境的配置，让我们开始吧。

配置Java开发环境

Android的开发使用Java语言，因此首先需要在开发设备上配置好Java的开发环境。

参考 Java开发环境安装与配置 (http://www.tianmaying.com/tutorial/java-dev-environment-installation-and-config) 。

Android Studio的安装

Android Studio是Google官方强烈推荐的集成开发环境。在Android官方网站下载 Android Studio (http://developer.android.com/sdk/index.html)，它包含了：

- 基于IntelliJ平台的Android IDE
- Android SDK工具（API、驱动、源码、样例等等）
- Android模拟器

由于众所周知的原因，可能需要“翻墙”才能够完整安装Android Studio，这里给出一个百度网盘的下载链接： http://pan.baidu.com/s/1i3rIfPz (http://pan.baidu.com/s/1i3rIfPz)

下载安装文件之后，普通软件的安装过程没有任何区别，不再赘述。

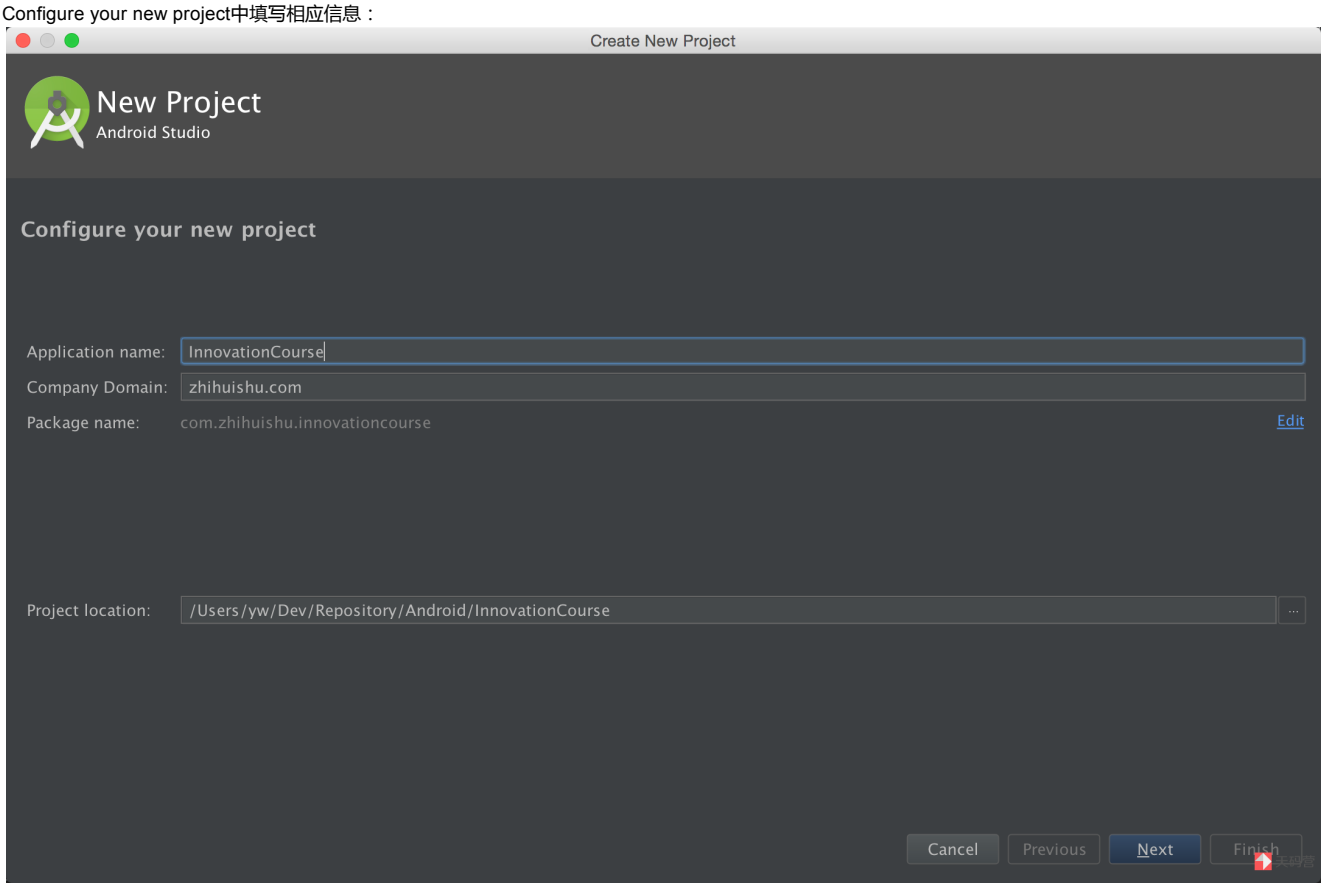
如果下载的不是最新版本，启动Android Studio后，可以通过**Check for Update**更新到最新的版本。

这里先告诉大家一个小招数，如果在编程过程中，遇到错误提示， `alt+Enter` (即按照 `alt` 键的同时按下回车键)可以自动进行错误修复。比如，在开发过程中我们经常需要引入一些包，就能自动帮我们引入。

创建项目

1. 启动Android Studio
2. 如果Android Studio中尚未打开任何项目，在Welcome页面选择 `Start a new Android Studio project`
3. 如果Android Studio中已经打开了其他项目，从 `文件` 菜单中选择 `New Project`

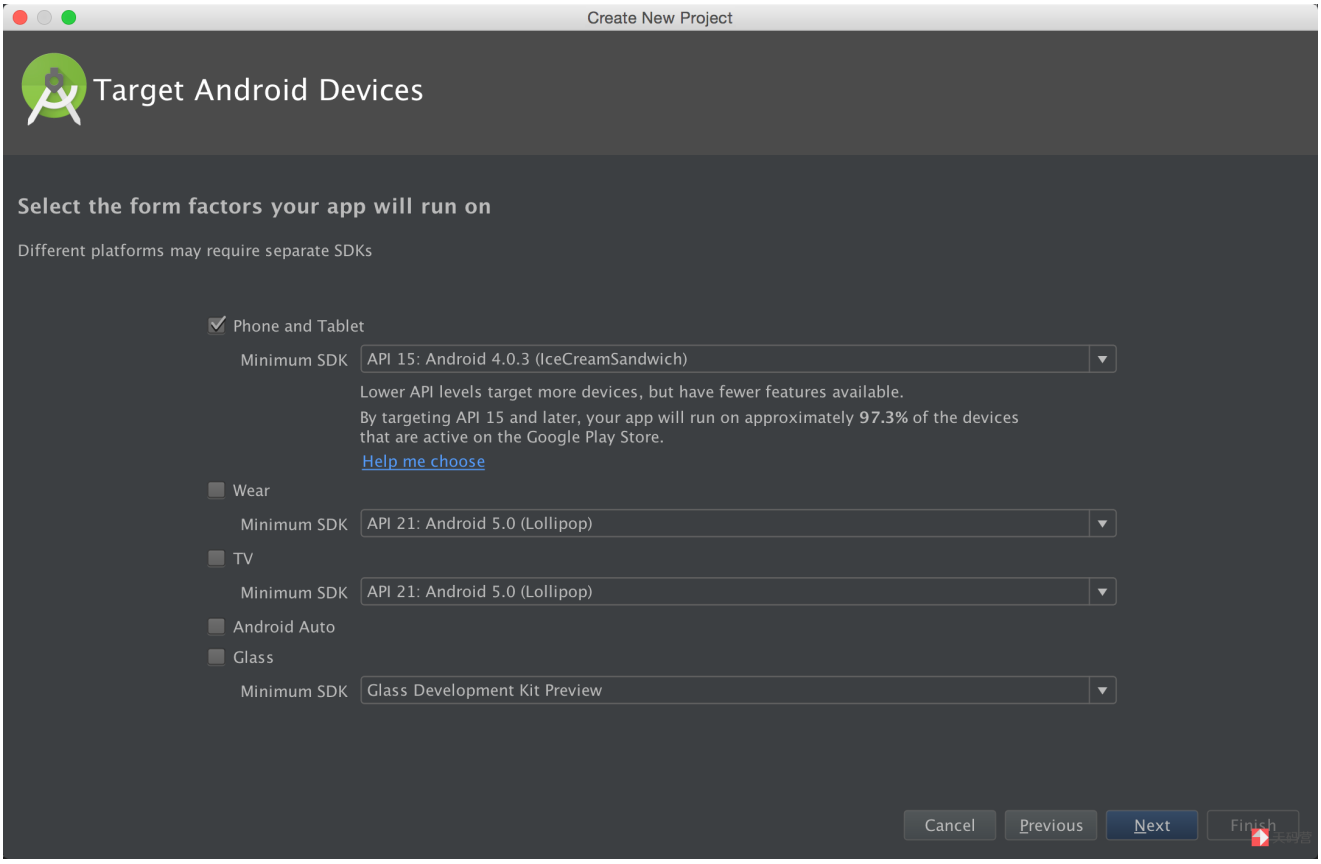
4. Configure your new project中填写相应信息：



(http://assets.tianmaying.com/md-image/a18d4ce55e74d4049d254fd86b63319f.png)

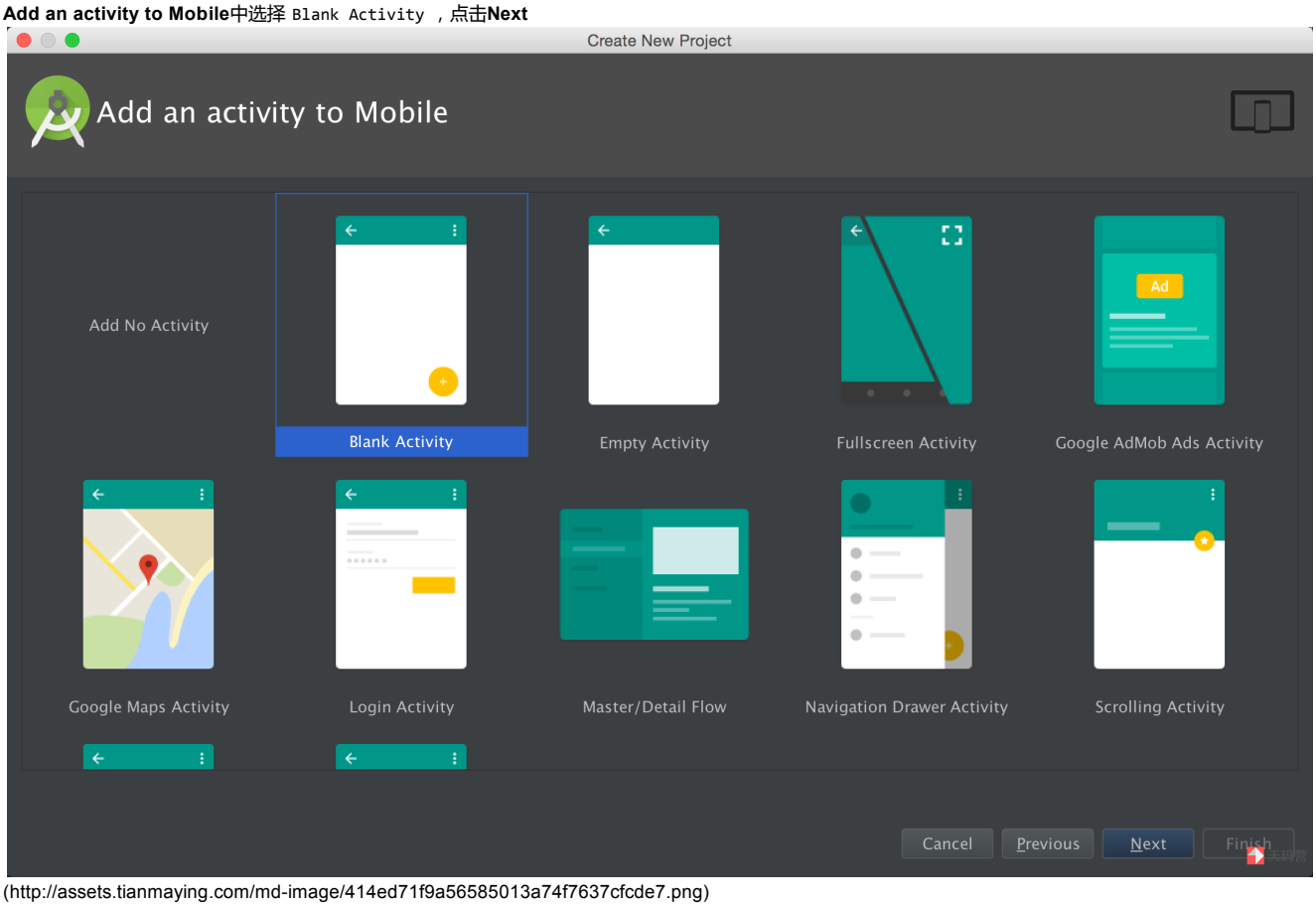
   - **Application Name**是我们项目的名称，我们填写为**InnovationCourse**
   - **Company Domain**是开发者所在的组织，一般是公司的域名，这里我们不妨填写为**zhihuishu.com**
   - **Project Location**是项目存在在本地的位置
   - 填写好后，点击**Next**

5. **Select the form factors your app will run on**中选择 `Phone and Tablet` 并设置 `Mininum SDK` 为 `API 15: Android 4.0.3 (IceCreamSandwich)`。 `Mininum SDK` 表示所支持的Android的最低版本，版本越低支持的设备数量就会越多。我们可以选择不同的版本来看所支持设备的百分比。百分比越大，表示越多人可以使用你的APP。选择好后点击**Next**



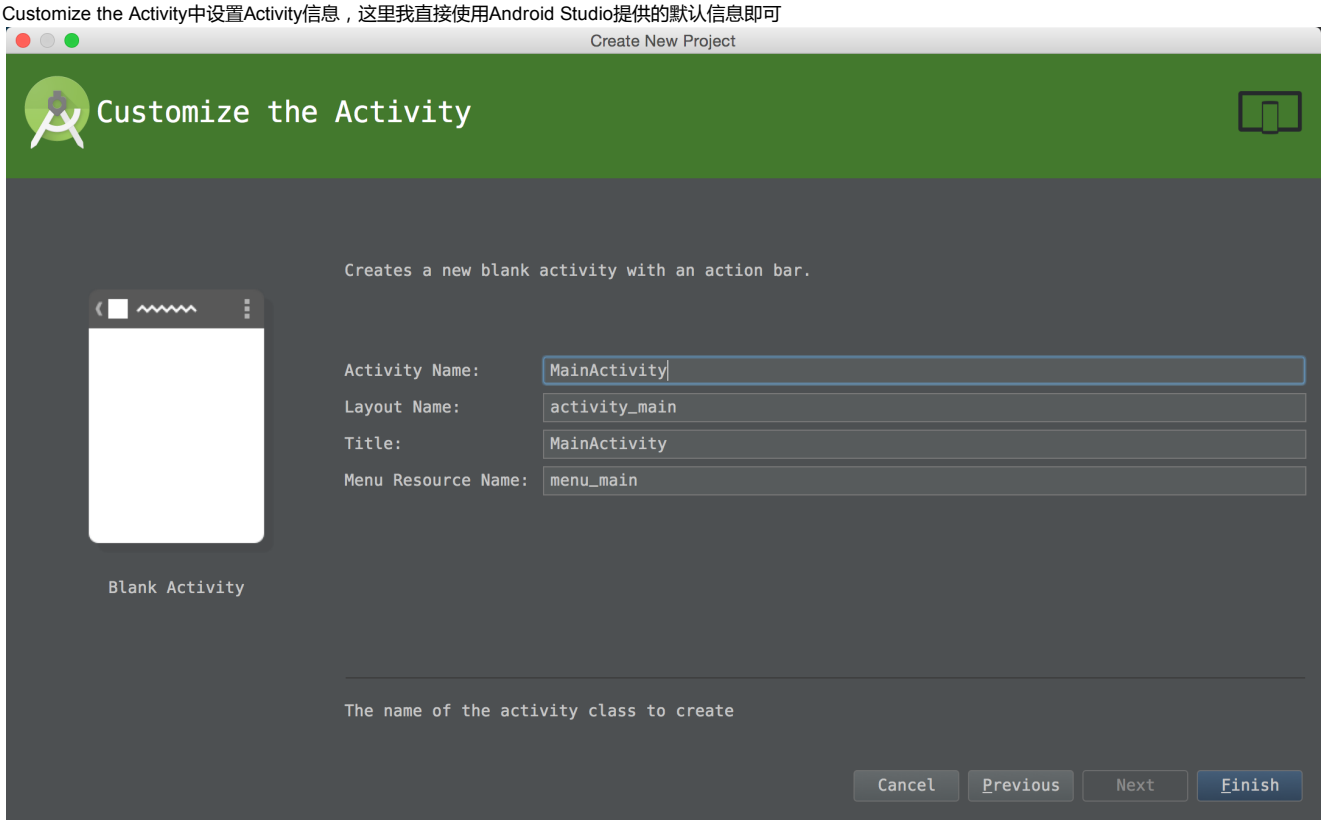
(http://assets.tianmaying.com/md-image/fdaca53fc4acc1f63a46023fe38c67cf.png)

6. **Add an activity to Mobile**中选择 `Blank Activity` ，点击**Next**



(http://assets.tianmaying.com/md-image/414ed71f9a56585013a74f7637cfcde7.png)

7. Customize the Activity中设置Activity信息，这里我直接使用Android Studio提供的默认信息即可



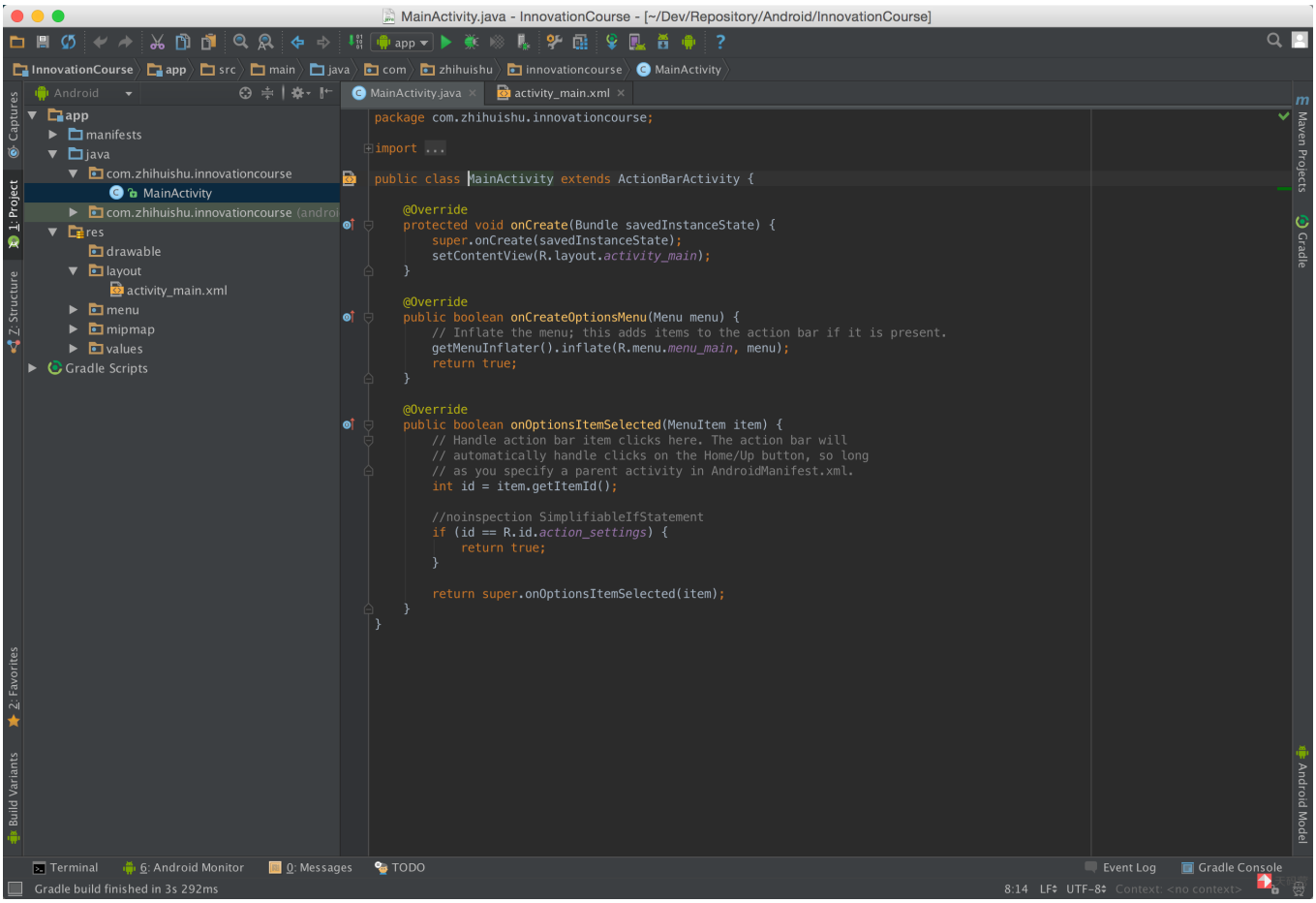
(http://tmy-course.oss-cn-beijing.aliyuncs.com/android-course/environment/studio-setup-2.png)

8. 点击 `Finish` ，一个项目就创建成功了！

项目结构

如果你急于开始项目的开发，你可以略过这一小节。

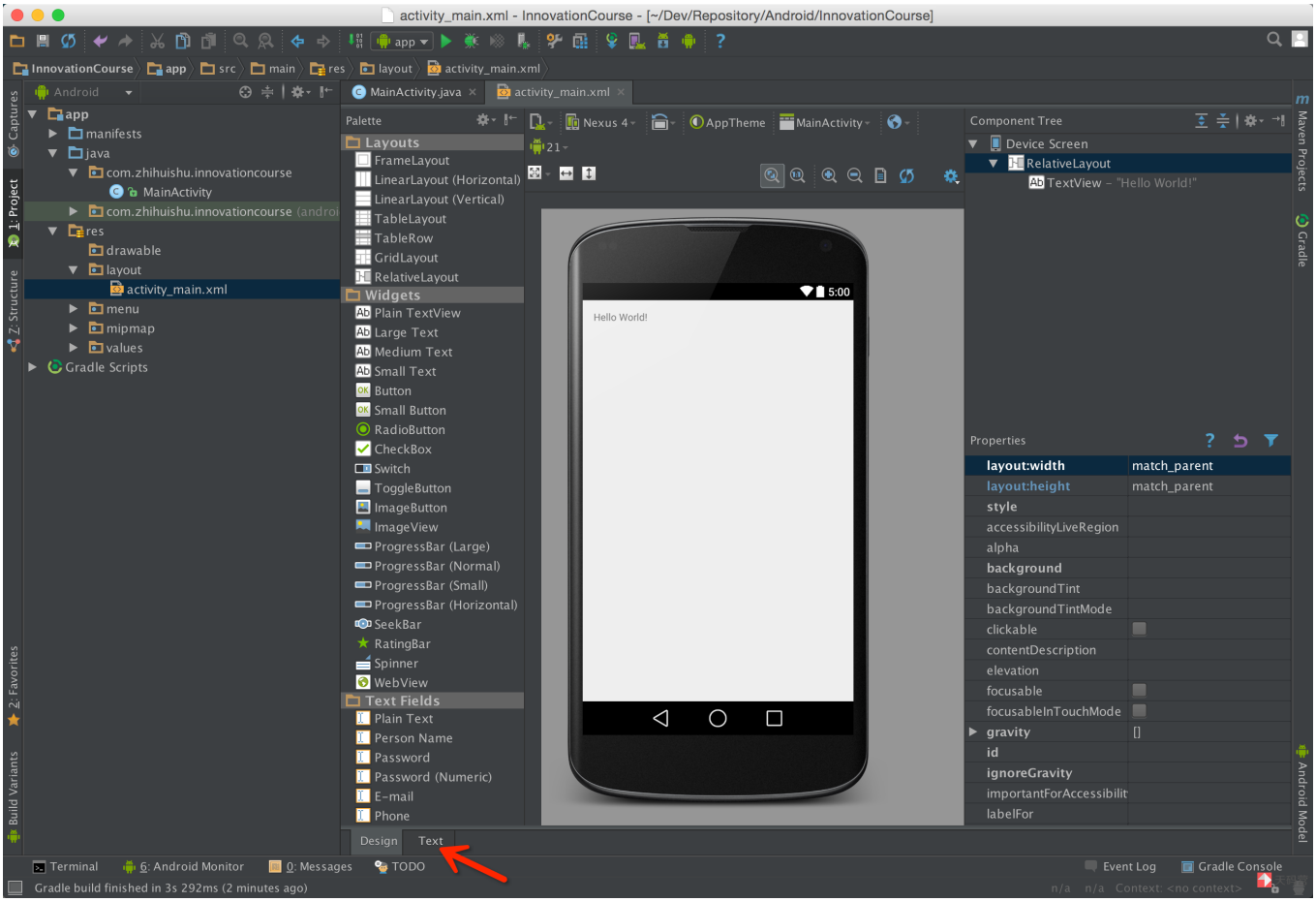
现在我们已经创建了一个基本的Android应用，它包含了Android Studio帮助我们生成的文件，在默认的 `Android` 视图中，文件结构如下图所示：



(http://assets.tianmaying.com/md-image/4d067eb517968e9295b15153119f3a8c.png)

这里我们看到的是Android Studio给我们生成 `MainActivity` 的代码骨架，这个Activity就代表App启动的时候，我们看到的那个屏幕。这个Java文件定义了一个 Activity (http://developer.android.com/guide/components/activities.html) ，当应用运行时， `MainActivity` 类启动一个 `Acitivty` 并加载 `activity_main.xml` 布局文件，将其显示在屏幕上。 `Acitvity` 和 `layout` 的关联是在 `MainActivity` 中 `onCreate()` 方法里完成的： `setContentView(R.layout.activity_main);`

我们再双击 `layout` 目录下的 `activity_main.xml` ，可以看到这个屏幕的预览：



(http://assets.tianmaying.com/md-image/65e7163a328cdaef8222c301fafbca2c.png)

点击下方的**Text**，我们可以以文本的形式看到这个文件的内容。这个文件就定义了屏幕中应该显示哪些组件。你如果学过HTML的话，可以类比一下，HTML描述了一个网页长什么样子，而这个XML文件则描述了App长什么样子。可以看到，这个布局简单的在屏幕上显示了一条消息——Hello world!

我们再来来具体看一下其它几个重要的文件：

- `app/src/main/AndroidManifest.xml`

  Android Manifest (http://developer.android.com/guide/topics/manifest/manifest-intro.html) 文件是描述Android应用的基本信息，并定义了应用中的各个组件（Activity是一种组件）。

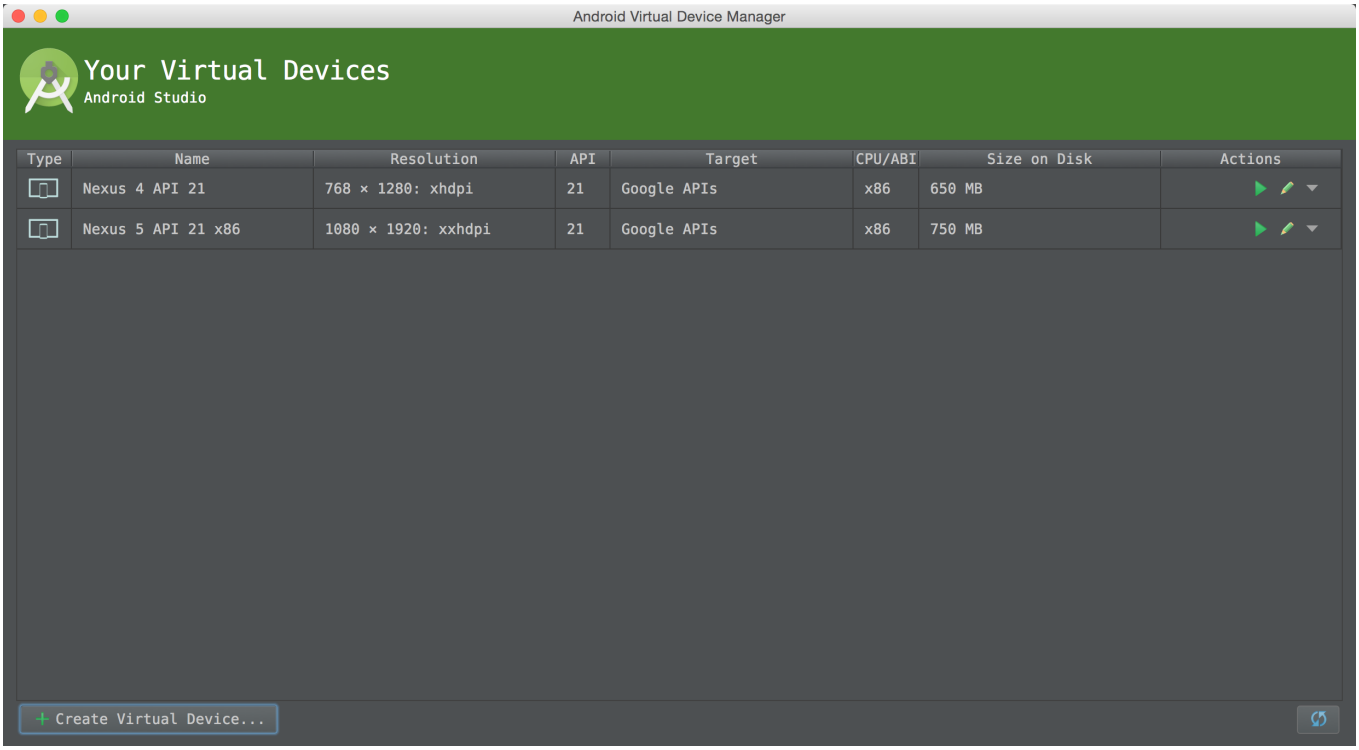
- ） `app/src/main/res` 目录下包含了应用所需要的资源文件：
  - `drawable<density>/` - 图片资源文件
  - `layout/` - 用户界面布局描述文件
  - `menu/` - 应用的菜单布局
  - `values/` 常量值例如字符串、颜色数值等
  - `strings/` 国际化数据

运行应用

在模拟器上运行应用

- 在Android Studio的菜单中打开： **Tools > Android > AVD Manager**
- 选择 `Create Virtual Device` 创建模拟器

创建完成后，AVD Manager如下图：



(http://tmy-course.oss-cn-beijing.aliyuncs.com/android-course/environment/studio-run-1.png)

模拟器创建完成后，回到Android Studio的项目中，在工具栏里点击 `Run` 按钮，接下来会弹出一个 `Choose Device` 窗口，选择 `Launch emulator` 并设置好需要使用的模拟器。

接下来等待模拟器启动，然后就可以看见刚刚创建的应用运行在模拟器窗口中了（刚启动的机器可能需要在模拟器中解锁屏幕）

在真机上运行应用

在真机上运行应用需要先进行一些设置：

- 将真机通过USB线缆连接至开发机器。如果在Windows上开发，可能需要选择合适的USB驱动，可以参考 OEM USB Drivers (http://developer.android.com/tools/extras/oem-usb.html) 文档
- 在真机系统中打开USB调试选项

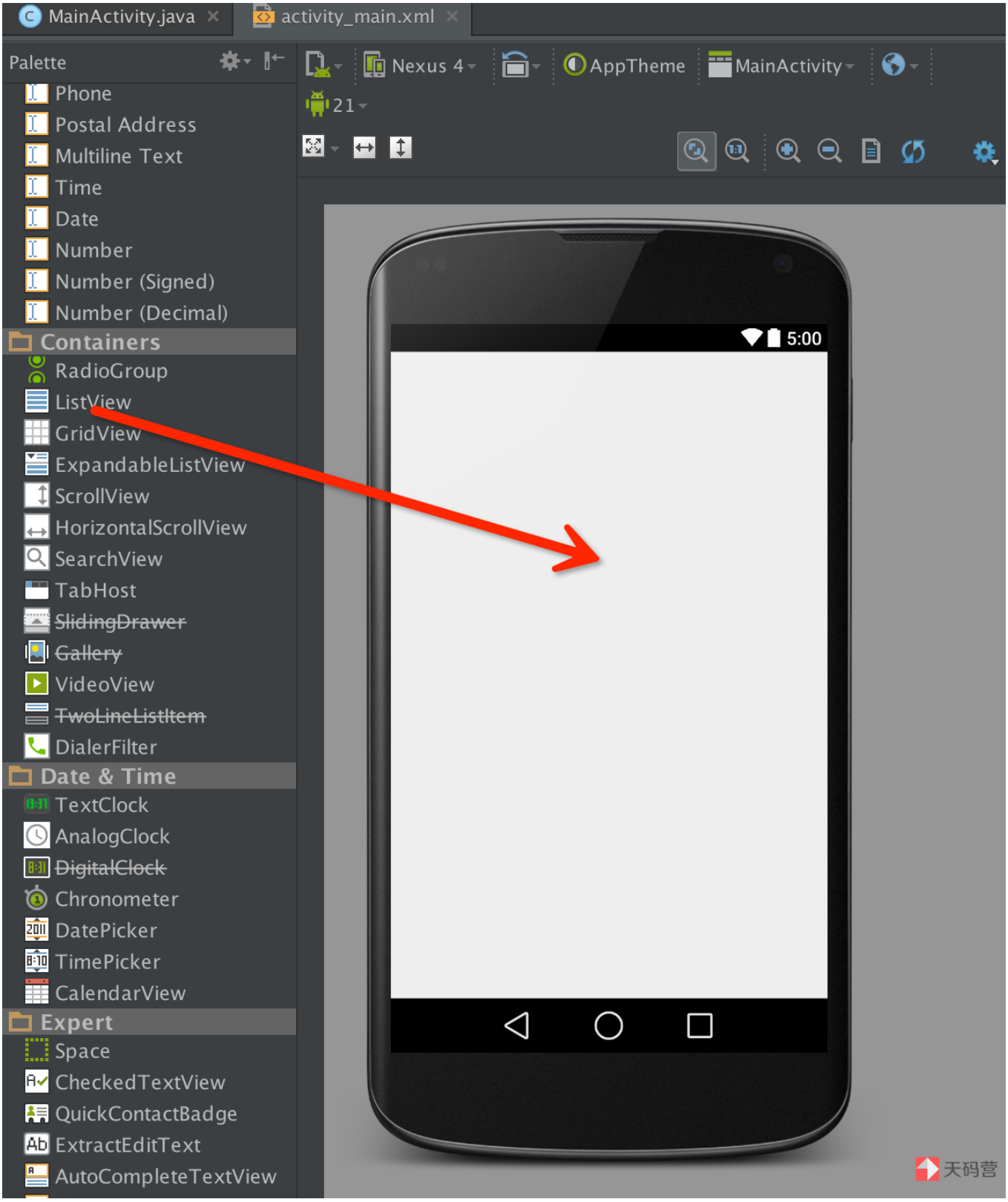
接下来就可以在Android Studio运行应用了，方法和在模拟器相同，只是在 `Choose Device` 窗口中需要选择USB连接的真机。

这里我们以模拟器为例进行讲解。

展示列表

打开 `activity_main.xml` 布局文件，在Design视图下，我们将上面的HelloWorld标签删掉。

我们准备在这个屏幕中显示老师的列表，而一个列表是一个ViewGroup，内部包含了其他的视图。我们在显示手机屏幕的预览区的左侧可以看到大量的视图组件，这些就是用来构建我们App模样的基本元素。我们我们选中ListView，将其拖入预览区内



(http://assets.tianmaying.com/md-image/ee17fa8ce42745dd5818feb01e7f2c01.png)

我们前后左右拖动一下，使其布满屏幕。也可以切换到**Text**视图来编辑XML，XML进行了修改之后，切回**Design**视图可以马上看到效果。

最终 `acitivity_main.xml` 中的内容为：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.zhihuishu.innovationcourse.MainActivity">

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/teacher_listView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

注意，这里面 `android:id="@+id/teacher_listView"` 定义了这个 `ListView` 的ID为 `teacher_listView` ，这是需要大家手动去修改的。当然你也可以全部复制后替换掉整个XML文件的内容。

这个ID非常重要，我们在Java程序中，将需要通过这个ID来找到这个视图组件，进行相应的设置和操作。

视图组件还有其他一些属性，比如这里其它四个属性都是跟布局相关的属性，比如 `android:layout_alignParentLeft="true"` 就表示与包含它的父视图是左侧对齐的。其它属性我们就不一一解释了。

简化的老师列表

准备数据：实现模型

我们这里先实现一个简化的老师列表，只显示老师的名字，图片暂时不管。

首先我们需要准备老师的数据，我们来创建一个 `Teacher` 类。

选中 `java` 目录下的 `com.zhihuishu.innovationcourse` 包，【右键】->【New】->【Java Class】，在弹出的窗口中输入类的名称**Teacher**。

给 `Teacher` 类增加一个获取所有老师姓名的方法。这个 `Teacher` 类，正是我们所说的MVC中M，即Model。

```
package com.zhihuishu.innovationcourse;

import java.util.ArrayList;
import java.util.List;

public class Teacher {

    public static List<String> getAllTeachers() {
        List<String> teachers = new ArrayList<String>();
        teachers.add("张海霞");
        teachers.add("陈江");
        teachers.add("叶蔚");

        return teachers;
    }
}
```

这里我们采用了硬编码的形式，通常情况下数据是从数据库中取出来的，而且是不断变化的。不过这里我们不涉及数据库访问，简化处理了。大家知道有数据库的存在就可以了。

设置Adapter

用来将数据传递给 `ListView` 的适配器是 `ArrayAdapter` 。这里我们传入的是一个字符串的数组，因此我们创建一个 `ArrayAdapter<String>` 类。

这部分代码添加到 `MainActivity.java` 的 `onCreate` 方法中：

```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //初始化一个Adapter
        ArrayAdapter<String> teacherAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, Teacher.getAllTeachers()

        //通过ID获取listView
        ListView listView = (ListView) findViewById(R.id.teacher_listView);

        //设置listView的Adapter
        listView.setAdapter(teacherAdapter);
    }

    ...
}
```

注意初始化Adapter的代码 `ArrayAdapter<String> teacherAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, Teacher.getAllTeachers());` 三个参数的含义：

- 第一个参数是 `this` ，表示传入的是当前的Activity
- 第二个参数是 `android.R.layout.simple_list_item_1` ，这是Android系统自带的一个列表元素（即列表中的每一行）布局，只显一串简单的文字
- 第三个参数是需要显示的所有数据构成的List，即数据源

后面我们会传入自己定义的元素布局（因为我们还希望在每一行中显示老师的图片），数据源也需要修改，因为此时不仅仅包含字符串了，还需要包含图片。

运行效果

我们已经可以看到一个简单的老师列表的效果了：

(http://assets.tianmaying.com/md-image/45704ec64a27f4e09cca7c1f80fd9d6b.png)

优化模型

老师的信息不仅仅包括姓名，还包括图片和介绍。因此我们需要给 `Teacher` 类增加一些属性，分别为 `name` 、 `imageId` 和 `desc` 。

老师的图片我们实现准备好了，大家只需要拷贝进入 `drawable` 文件夹就能在Java代码中引用了。我们之前说过 `drawable` 文件夹就是用来存放图片文件的。

增加属性之后我们还需要做三件事情：

1. 增加一个构造函数，这个构造有三个参数。即我们创建一个 `Teacher` 类的实例时，需要告诉系统这个老师的名字、图片和介绍。
2. 给每个属性增加 `getter` 和 `setter` ，面向对象编程讲究信息的隐藏，因此老师们的属性都是 `private` 的，为了让外部能够访问老师的属性，比如获取老师的图片来进行展示，那么就需要有访问属性的公共函数。 `getter` 和 `setter` 就是扮演这样的角色。
3. 修改 `getAllTeachers()` 方法，这个方法此时返回的是 `Teacher` 对象构成的List，而不是字符串构成的List了。在这个方法的实现中，我们就通过Teacher的构造函数生成了三个 `Teacher` 对象。

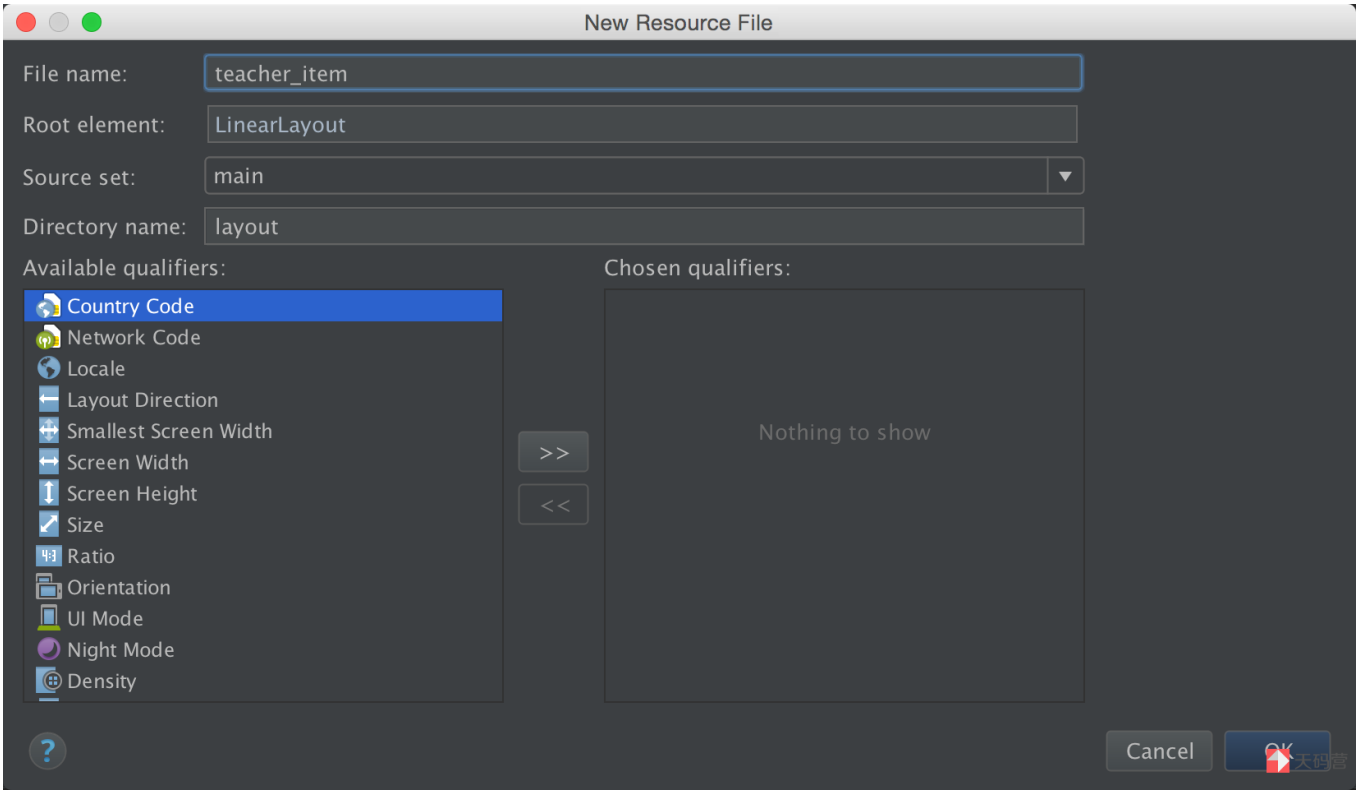
```
package com.zhihuishu.innovationcourse;

import java.util.ArrayList;
import java.util.List;

public class Teacher {

    private String name;
    private int imageId;
    private String desc;

    //构造函数
    public Teacher(String name, int imageId, String desc) {
        this.name = name;
        this.imageId = imageId;
        this.desc = desc;
    }

    // 返回一个Teacher的列表
    public static List<Teacher> getAllTeachers() {
        List<Teacher> teachers = new ArrayList<Teacher>();
        teachers.add(new Teacher("张海霞", R.drawable.zhx, "张海霞，北京大学教授国际大学生iCAN创新创业大赛发起人、主席北京大学信息科学技术学院教授
        teachers.add(new Teacher("陈江", R.drawable.cj, "陈江，北京大学信息科学技术学院副教授，高等学校电路和信号系统教学与教材研究会常务理事，中国
        teachers.add(new Teacher("叶蔚", R.drawable.yw, "叶蔚，北京大学软件工程国家工程研究中心副研究员，创办了技术学习服务平台天码营(http://tian

        return teachers;
    }

    // 以下都是访问内部属性的getter和setter

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getImageId() {
        return imageId;
    }

    public void setImageId(int imageId) {
        this.imageId = imageId;
    }

    public String getDesc() {
        return desc;
    }

    public void setDesc(String desc) {
        this.desc = desc;
    }
}
```

改进列表

创建新的布局

为了在列表的一行中，同时显示老师的图片和名称，我们需要创建自定义的布局。

选中 `res/layout` 目录，【右键】->【New】->【Layout Resource File】，在弹出对话框中，输入如下信息：



(http://assets.tianmaying.com/md-image/50ed8d8db63e9664526821bea6e78835.png)

File Name为布局文件的名称，我们输入**teacher_item**。其他的输入框使用默认信息即可。其中**Root Element**表示布局的方式，这里我们无需修改，使用默认的 `LinearLayout` ，即视图组件通过线性的方式来进行布局。

新的布局中，我们需要在左侧显示一张图片，右侧显示老师姓名，因此我们往预览区拖入一个 `ImageView` 和 `TextView` ，然后切换到**Text**视图进行编辑，最终的布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginBottom="5dp"
        android:layout_marginTop="5dp"
        android:id="@+id/teacher_small_imageView" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:text="New Text"
        android:id="@+id/teacher_name_textView"
        android:layout_gravity="center_vertical" />
</LinearLayout>
```

注意我们将根元素 `<LinearLayout>` 的 `android:orientation` 改为了 `horizontal` ，因为我们希望老师的图片和老师的姓名水平线性布局。

ImageView命名为 `teacher_small_imageView` ，长宽都为50dp，与上下的间距为5dp。dp是一种长度的单位，也有其他类型的单位，dp能够比较好的兼容各种分辨率的设备。

TextView命名为 `teacher_name_textView` ，长宽为根据内容自适应，与左侧图片间距为10dp，垂直居中。

创建自定义的ArrayAdapter

我们创建一个自定义的ArrayAdapter。我们创建一个命名为 `TeacherAdapter` 的类，让其继承 `ArrayAdapter<Teacher>` ，同时提供一个构造函数。

```
package com.zhihuishu.innovationcourse;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.List;
import java.util.zip.Inflater;

public class TeacherAdapter extends ArrayAdapter<Teacher> {
    public TeacherAdapter(Context context, int resource, List<Teacher> objects) {
        super(context, resource, objects);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        // 获取老师的数据
        Teacher teacher = getItem(position);

        // 创建布局
        View oneTeacherView = LayoutInflater.from(getContext()).inflate(R.layout.teacher_item, parent, false);

        // 获取ImageView和TextView
        ImageView imageView = (ImageView) oneTeacherView.findViewById(R.id.teacher_small_imageView);
        TextView textView = (TextView) oneTeacherView.findViewById(R.id.teacher_name_textView);

        // 根据老师数据设置ImageView和TextView的展现
        imageView.setImageResource(teacher.getImageId());
        textView.setText(teacher.getName());

        return oneTeacherView;
    }
}
```

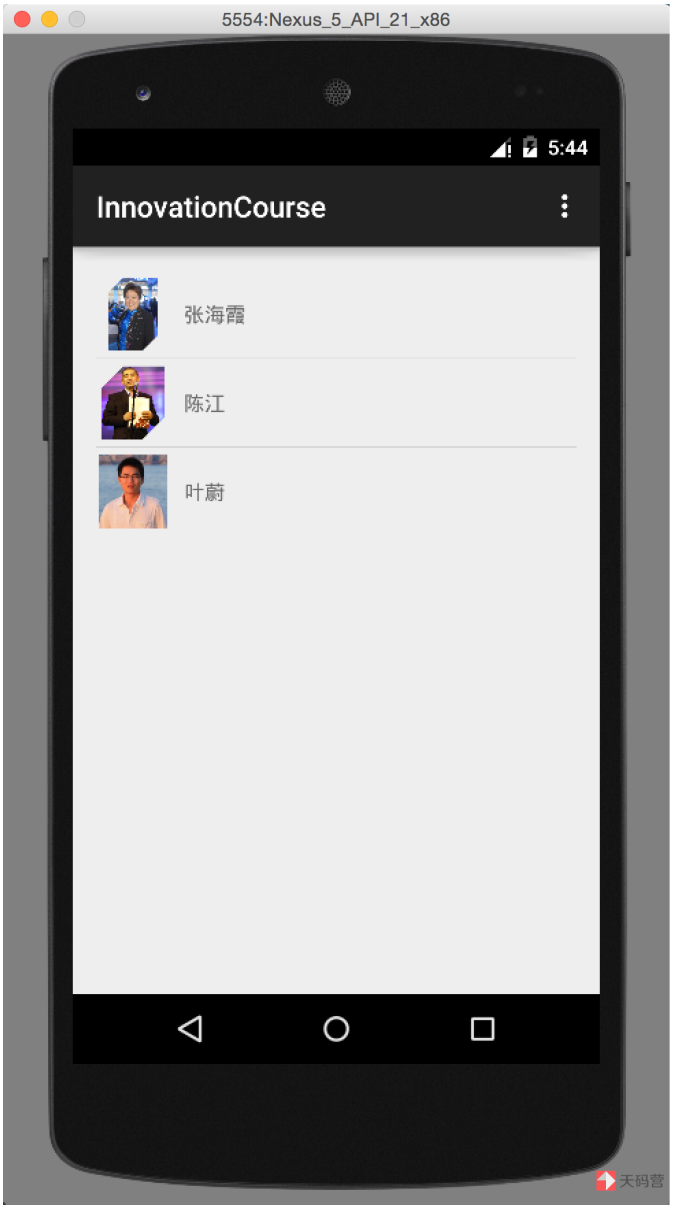
这段代码的主要功能就是，提供了一个 `getView()` 方法的重载实现，我们通过重载这个方法就能够让listView根据我们的要求来生成每一个列表元素了。而这个方法做了四件事情：

1. 获取老师的数据， `getItem(position)` 会把 `poistion` 位置的 `Teacher` 对象返回给你，这件事情Adapter会帮你处理好，你只管调用就好了。
2. 创建布局， `View oneTeacherView = LayoutInflater.from(getContext()).inflate(R.layout.teacher_item, parent, false);` 这条语句大家照着写就好了，你只需要知道这是根据Layout文件生成一个布局（布局也是一个View的子类）。
3. 获取ImageView和TextView
4. 最后根据老师数据设置ImageView和TextView的展现

在 `MainActivity.java` 中的代码也需要做相应的修改，此时我们要创建一个 `TeacherAdapter` 的对象，并将其设置为 `listView` 的Adapter。

```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TeacherAdapter teacherAdapter = new TeacherAdapter(this, R.layout.teacher_item, Teacher.getAllTeachers());

        ListView listView = (ListView) findViewById(R.id.teacher_listView);

        listView.setAdapter(teacherAdapter);
    }
    ...
}
```

运行效果

此时已经可以看到更好看的老师列表界面了： 

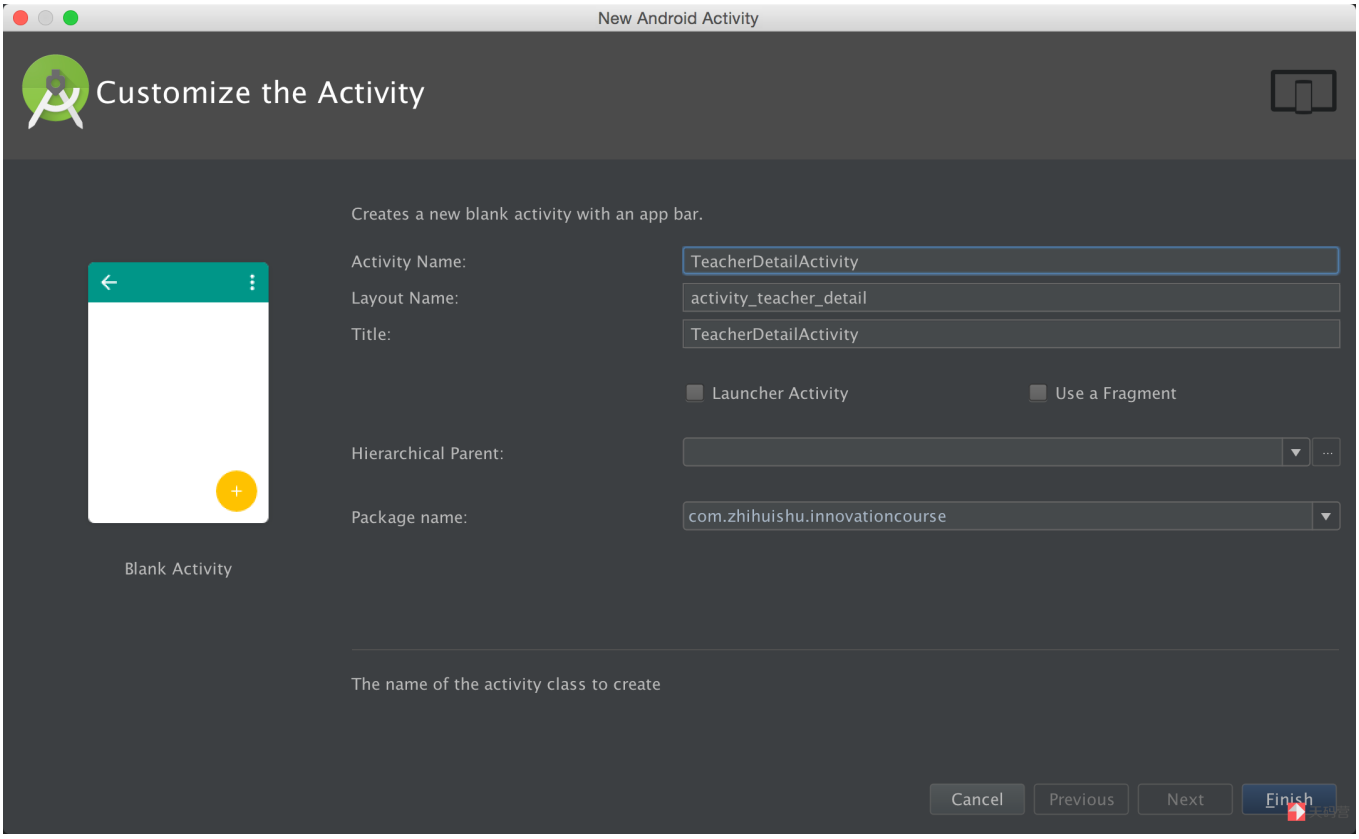
(http://assets.tianmaying.com/md-image/fec348f2aafa30d8aab40d8ee26134ba.png)

创建第二个Activity

一切进展顺利，我们已经了解如何创建Activity中的视图，如何通过Adapter给视图传递数据。接下来我们要做一个功能，当点击列表中的每一项时，会进入第二个Activity，显示老师的大图片以及详细介绍。

所以我们先来创建第二个Activity吧。

选中 `java` 目录下的 `com.zhihuishu.innovationcourse` 包，【右键】->【New】->【Activity】->【Blank Activity】，在弹出的窗口中输入Activity名称 **TeacherDetailActivity**，点击【Finish】。



(http://assets.tianmaying.com/md-image/9d7c54b8fb1a3253a9568f8f5e1790aa.png)

类似于 `teacher_item.xml` 布局的编辑，我们往预览区中拖入一个 `ImageView` 和 `TextView` ，然后在**Text**视图中进行布局的编辑。最终的布局配置如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.zhihuishu.innovationcourse.TeacherDetailActivity">

    <ImageView
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:id="@+id/teacher_large_imageView"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Text"
        android:id="@+id/teacher_desc_textView"
        android:layout_below="@+id/teacher_large_imageView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp" />
</RelativeLayout>
```

很多属性我们之前已经遇到过了。这里我们使用了另外一种布局 `RelativeLayout` ，顾名思义，就是通过视图元素的相对位置来进行布局。其中关键的一行代码是 `android:layout_below="@+id/teacher_large_imageView"` ，这表示 `teacher_large_imageView` 在 `teacher_large_imageView` 的下方。

实现行为：事件处理

这个Activity的布局已经创建好了，如何跳转到这个Activity呢？这时我们需要识别出用户的点击行为，当用户点击发生时才能进行跳转。

我们回到 `TeacherAdapter` 的 `getView()` 方法，在最后一条 `return` 语句前加入：

```
        oneTeacherView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 这里进行跳转
            }
        });
```

这段代码就是在返回 `oneTeacherView` 之前，设置一个 `OnClick` 点击事件的监听器，当事件发生的时候，就会执行 `public void onClick(View v)` 内部的代码。

Activity的跳转：Intent

接下来我们就要进行真正的跳转了， `Intent` 终于排上用场了，代码如下：

```
        oneTeacherView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //  初始化一个准备跳转到TeacherDetailActivity的Intent
                Intent intent = new Intent(getContext(), TeacherDetailActivity.class);
                // 准备跳转
                getContext().startActivity(intent);
            }
        });
```

增加两行代码就能跳转到第二个Activity了。但是运行起来之后，在第二个Activity中我们并没有看到数据。所以让我们来做最后一步吧，你马上就要成功了。

在进行跳转的时候，我们需要把老师的数据传递给 `TeacherDetailActivity` ，并在 `TeacherDetailActivity` 中进行适当的设置。

通过Intent传递参数

设置Intent的Extra数据

`Intent` 的 `putExtra` 方法就是用来传递参数的，我们只需在初始化 `Intent` 对象之后把老师的数据传递进去即可。

需要注意的一点是，为了访问 `teacher` 变量，需要在申明的时候加上 `final` 修饰符。注意代码中哪些地方发生了变化，完整的代码如下：

```
package com.zhihuishu.innovationcourse;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.List;

public class TeacherAdapter extends ArrayAdapter<Teacher> {
    public TeacherAdapter(Context context, int resource, List<Teacher> objects) {
        super(context, resource, objects);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        // 获取老师的数据
        final Teacher teacher = getItem(position);

        // 创建布局
        View oneTeacherView = LayoutInflater.from(getContext()).inflate(R.layout.teacher_item, parent, false);

        // 获取布局中的ImageView和TextView
        ImageView imageView = (ImageView) oneTeacherView.findViewById(R.id.teacher_small_imageView);
        TextView textView = (TextView) oneTeacherView.findViewById(R.id.teacher_name_textView);

        // 根据老师数据设置ImageView和TextView的展现
        imageView.setImageResource(teacher.getImageId());
        textView.setText(teacher.getName());

        oneTeacherView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                //  初始化一个准备跳转到TeacherDetailActivity的Intent
                Intent intent = new Intent(getContext(), TeacherDetailActivity.class);

                // 往Intent中传入Teacher相关的数据，供TeacherDetailActivity使用
                intent.putExtra("teacher_image", teacher.getImageId());
                intent.putExtra("teacher_desc", teacher.getDesc());

                //  初始化一个准备跳转到TeacherDetailActivity的Intent
                getContext().startActivity(intent);
            }
        });

        return oneTeacherView;
    }
}
```

根据Intent数据展示内容

最后我们修改 `TeacherDetailActivity` 的 `onCreate()` 方法，加入从 `Intent` 获取数据并设置视图展现的代码。