Project 4 Task 2 – Interesting Picture App

Example, by Joe Mertz

Description:

My application takes a search string from the user, and uses it to fetch and display a photograph from Flickr.

Here is how my application meets the task requirements

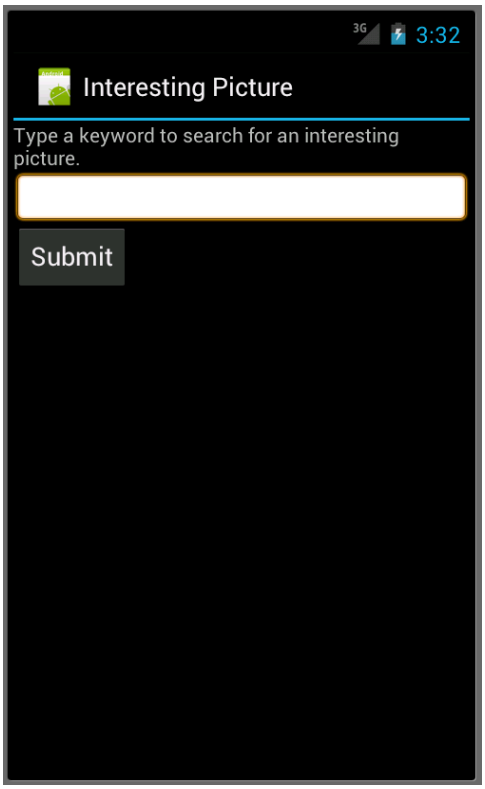
1. Implement a native Android application

The name of my native Android application project in Android Studio is: MyAndroidInterestingPicture

1.1. Has at least two different kinds of views in your Layout (TextView, EditText, ImageView, etc.)

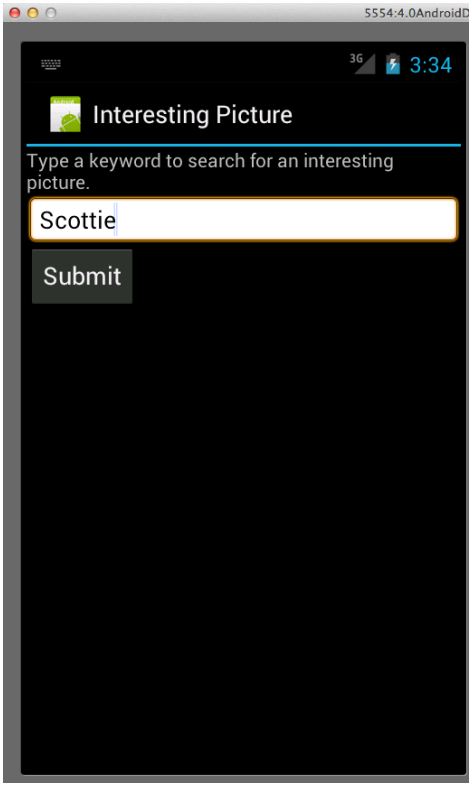
My application uses TextView, EditText, Button, and ImageView. See content_main.xml for details of how they are incorporated into the LinearLayout.

Here is a screenshot of the layout before the picture has been fetched.



1.2. Requires input from the user

Here is a screenshot of the user searching for a picture of a Scottie



1.3. Makes an HTTP request (using an appropriate HTTP method) to your web service

My application does an HTTP GET request in GetPicture.java. The HTTP request is:

"https://cool-hill-3131.herokuapp.com/myinterestingpictureserver?search="+search

where search is the user's search term.

The search method makes this request of my web application, parses the returned XML to find the picture URL, fetches the picture, and returns the bit image of the picture.

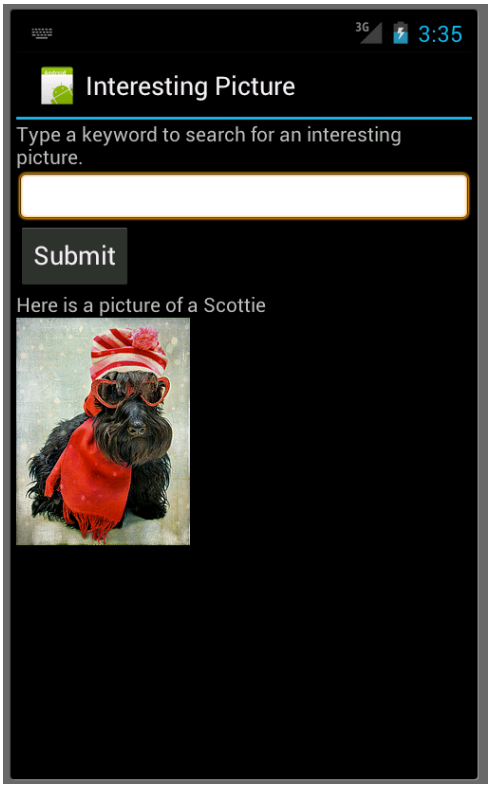
1.4. Receives and parses an XML or JSON formatted reply from the web service

An example of the XML reply is:

```
<interestingresponse>
    <photo tag="sand"
imageURL="http://farm8.static.flickr.com/7061/6831566406_d43314d9fd_m.jpg"
        />
</interestingresponse>
```

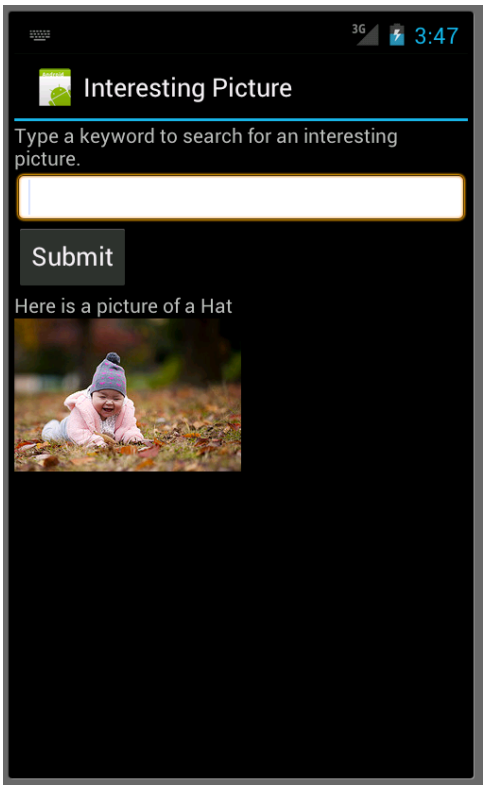
1.5. Displays new information to the user

Here is the screen shot after the picture has been returned.



1.6. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

The user can type in another search term and hit Submit. Here is an example of having typed in "Hat".



2. Implement a web application, deployed to Heroku

The name of the Heroku project in Android Studio is: MyInterestingPictureServer

2.1. Using an HttpServlet to implement a simple (can be a single path) API

In my web app project:

Model: InterestingPictureModelUsingWS.java

View: result.jsp

Controller: MyInterestingPictureServerServlet.java

2.2. Receives an HTTP request from the native Android application

MyInterestingPictureServerServlet.java receives the HTTP GET request with the argument "search". It passes this search string on to the model.

2.3. Executes business logic appropriate to your application

InterestingPictureModelUsingWS.java makes an HTTP request to:

http://api.flickr.com/services/rest/?method=flickr.photos.search.

It then parses the XML response and extracts the parts it needs to respond to the Android application.

2.4. Replies to the Android application with an XML or JSON formatted response.

Response.jsp formats the response to the mobile application in a simple XML format of my own design:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<%@ page contentType="text/xml;charset=ISO-8859-1" %>
<interestingresponse>
    <photo tag="<%= request.getAttribute("pictureTag")%>"
    imageURL="<%= request.getAttribute("pictureURL")%>" />
</interestingresponse>
```