

```
1  /* Help.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: This activity displays all of the help information using a
8  recycler view.
9  */
10 package com.fbld.dulaney.fblyardsale;
11
12 import android.databinding.DataBindingUtil;
13 import android.os.Bundle;
14 import android.support.v7.app.AppCompatActivity;
15 import android.support.v7.widget.LinearLayoutManager;
16 import android.view.View;
17
18 import com.fbld.dulaney.fblyardsale.databinding.ActivityHelpBinding;
19
20 public class Help extends AppCompatActivity implements View.OnClickListener
21 {
22     ActivityHelpBinding mBinding;
23
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_help);
28
29         mBinding = DataBindingUtil.setContentView(this, R.layout.activity_help);
30         mBinding.done.setOnClickListener(this);
31         mBinding.listHelp.setLayoutManager(new LinearLayoutManager(this));
32         mBinding.listHelp.setAdapter(new HelpAdapter());
33         setSupportActionBar(mBinding.myToolbar);
34     }
35
36     @Override
37     public void onClick(View v)
38     {
39         switch(v.getId())
40         {
41             case R.id.done:
42                 this.finish();
43                 break;
44         }
45     }
46
47     @Override
48     public void onBackPressed()
49     {
50         this.finish();
51     }
52 }
53
```

```

1  /* MySales.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: This activity lists all of the items you have for sale. It allows
8  you to delete any item, or look at their comments.
9  */
10 package com.fbld.dulaney.fblyardsale;
11
12 import android.content.Intent;
13 import android.databinding.DataBindingUtil;
14 import android.os.Bundle;
15 import android.support.v7.app.AppCompatActivity;
16 import android.support.v7.widget.LinearLayoutManager;
17 import android.util.Log;
18 import android.view.View;
19
20 import com.fbld.dulaney.fblyardsale.controller.MySalesController;
21 import com.fbld.dulaney.fblyardsale.databinding.ActivityMysalesBinding;
22
23 public class MySales extends AppCompatActivity implements View.OnClickListener {
24
25     ActivityMysalesBinding mBinding;
26
27     protected void onCreate(Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_mysales);
30
31         mBinding = DataBindingUtil.setContentView(this, R.layout.activity_mysales);
32         mBinding.list.setLayoutManager(new LinearLayoutManager(this));
33         MySalesAdapter adapter = new MySalesAdapter(this, this);
34         MySalesController.AttachAdapter(adapter);
35         mBinding.list.setAdapter(adapter);
36         setSupportActionBar(mBinding.myToolbar);
37
38         Log.d("MySales", "onCreate");
39     }
40
41     @Override
42     public void onClick(View v) {
43         switch (v.getId()) {
44
45             case R.id.comments:
46                 this.startActivity(new Intent(this, Comments.class));
47                 break;
48             default:
49                 break;
50         }
51     }
52
53     @Override
54     public void onBackPressed()
55     {
56         this.finish();
57     }
58
59 }
60

```

```
1  /* AddSales.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: This activity is used to add a new sale item.
8  */
9  package com.fb\bla.dulaney.fb\blayardsale;
10
11  import android.Manifest;
12  import android.content.Intent;
13  import android.content.pm.PackageManager;
14  import android.databinding.DataBindingUtil;
15  import android.graphics.Bitmap;
16  import android.graphics.BitmapFactory;
17  import android.net.Uri;
18  import android.os.AsyncTask;
19  import android.os.Build;
20  import android.os.Bundle;
21  import android.provider.MediaStore;
22  import android.support.v4.app.ActivityCompat;
23  import android.support.v4.content.ContextCompat;
24  import android.support.v7.app.AppCompatActivity;
25  import android.util.Log;
26  import android.view.View;
27  import android.widget.Toast;
28
29  import com.fb\bla.dulaney.fb\blayardsale.controller.MySalesController;
30  import com.fb\bla.dulaney.fb\blayardsale.databinding.ActivityAddsalesBinding;
31
32  import java.io.InputStream;
33  import java.util.UUID;
34
35  import com.fb\bla.dulaney.fb\blayardsale.model.*;
36  import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
37
38  public class AddSales extends AppCompatActivity implements View.OnClickListener {
39      ActivityAddsalesBinding mBinding;
40
41      private MobileServiceTable<SaleItem> mSaleItemTable;
42
43      protected void onCreate(Bundle savedInstanceState) {
44          super.onCreate(savedInstanceState);
45          setContentView(R.layout.activity_addsales);
46
47          if (!FblaLogon.getLoggedOn()) {
48              Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
49                  LENGTH_LONG).show();
50              finish();
51              return;
52          }
53
54          mSaleItemTable = FblaLogon.getClient().getTable(SaleItem.class);
55
56          mBinding = DataBindingUtil.setContentView(this, R.layout.activity_addsales);
57          FblaPicture.setLayoutImage(mBinding.activityAddsales);
58          setSupportActionBar(mBinding.myToolbar);
59          mBinding.gallery.setOnClickListener(this);
60          mBinding.camera.setOnClickListener(this);
61          mBinding.back.setOnClickListener(this);
62          mBinding.finish.setOnClickListener(this);
63          mBinding.another.setOnClickListener(this);
```

```

63     }
64
65     @Override
66     public void onClick(View v) {
67         switch (v.getId()) {
68             case R.id.gallery:
69                 // Ask for permission first
70                 if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
71                     int permissionCheck = ContextCompat.checkSelfPermission(this, Manifest.
72 permission.READ_EXTERNAL_STORAGE);
73                     if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
74                         // Should we show an explanation?
75                         if (ActivityCompat.shouldShowRequestPermissionRationale(this,
76 Manifest.permission.READ_EXTERNAL_STORAGE)) {
77                             // Explain to the user why we need to read the contacts
78                             } else {
79                                 ActivityCompat.requestPermissions(this,
80                                     new String[]{Manifest.permission.READ_EXTERNAL_STORAGE}
81                                     , 0);
82                             }
83                             return;
84                         }
85                     }
86                     Intent i = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.
87 Images.Media.EXTERNAL_CONTENT_URI);
88                     Log.d("CameraFragment", "Starting GALLERY Intent");
89                     this.startActivityForResult(i, 1);
90                     break;
91             case R.id.camera:
92                 // Ask for permission first
93                 if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
94                     int permissionCheck = ContextCompat.checkSelfPermission(this, Manifest.
95 permission.CAMERA);
96                     if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
97                         // Should we show an explanation?
98                         if (ActivityCompat.shouldShowRequestPermissionRationale(this,
99 Manifest.permission.CAMERA)) {
100                             // Explain to the user why we need to read the contacts
101                             } else {
102                                 ActivityCompat.requestPermissions(this,
103                                     new String[]{Manifest.permission.CAMERA}, 0);
104                             }
105                             return;
106                         }
107                     }
108                     Intent j = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
109                     this.startActivityForResult(j, 2);
110                     break;
111             case R.id.another:
112                 addItem(v);
113                 this.finish();
114                 this.startActivity(new Intent(this, AddSales.class));
115                 break;
116             case R.id.finish:
117                 addItem(v);
118                 this.finish();
119                 break;
120             default:
121                 this.finish();
122                 break;
123         }
124     }

```

```

120     }
121
122     @Override
123     public void onBackPressed()
124     {
125         this.finish();
126     }
127
128     // Add a new item to the database.
129     private void addItem(View view) {
130         if (!FbblLogon.getLoggedOn()) return;
131
132         // Create a new item from the SaleItem model.
133         final SaleItem item = new SaleItem();
134         item.setId(UUID.randomUUID().toString());
135         item.setName(mBinding.editname.getText().toString());
136         item.setUserId(FbblLogon.getUserId());
137         item.setDescription(mBinding.editdesc.getText().toString());
138         String sPrice = mBinding.editprice.getText().toString();
139         if (sPrice == null || sPrice.equals("")) item.setPrice(0);
140         else item.setPrice(Float.parseFloat(mBinding.editprice.getText().toString()));
141         Bitmap b = FbblPicture.GetPictureFromView(mBinding.picture);
142         if (b != null) {
143             item.setPicture(b);
144         }
145
146         // Save the item to the database over the internet.
147         AsyncTask

```

```
183         }
184     } else if (requestCode == 1 && data != null) // Gallery
185     {
186         // Gallery
187         Log.d("AddSales", "Result from Gallery");
188
189         try {
190             Uri pickedImage = data.getData();
191             InputStream stream = getContentResolver().openInputStream(pickedImage);
192             Bitmap image = BitmapFactory.decodeStream(stream);
193             image = FblaPicture.ResizePicture(this(getApplicationContext()), image);
194             FblaPicture.LoadPictureOnView(mBinding.picture, image);
195         } catch (Exception ex) {
196             Log.e("AddSales:gallery", ex.getMessage());
197         }
198     }
199 }
200 } // onActivityResult
201 }
202
```

```

1  /* Comments.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: This activity allows you to both add new comments, review existing
8  comments posted on a sale item, and delete comments.
9  */
10 package com.fbld.dulaney.fblyardsale;
11
12 import android.content.Context;
13 import android.databinding.DataBindingUtil;
14 import android.os.AsyncTask;
15 import android.os.Bundle;
16 import android.support.v7.app.AppCompatActivity;
17 import android.support.v7.widget.LinearLayoutManager;
18 import android.util.Log;
19 import android.view.View;
20 import android.view.inputmethod.InputMethodManager;
21
22 import com.fbld.dulaney.fblyardsale.controller.CommentListController;
23 import com.fbld.dulaney.fblyardsale.databinding.ActivityCommentsBinding;
24 import com.fbld.dulaney.fblyardsale.model.ItemComment;
25 import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
26
27 import java.util.UUID;
28
29 public class Comments extends AppCompatActivity implements View.OnClickListener {
30
31     ActivityCommentsBinding mBinding;
32     MobileServiceTable<ItemComment> mCommentTable;
33
34     protected void onCreate(Bundle savedInstanceState) {
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.activity_comments);
37
38         mBinding = DataBindingUtil.setContentView(this, R.layout.activity_comments);
39         mBinding.post.setOnClickListener(this);
40         mBinding.list.setLayoutManager(new LinearLayoutManager(this));
41         CommentsAdapter adapter = new CommentsAdapter(this, this);
42         CommentListController.AttachAdapter(adapter);
43         mBinding.list.setAdapter(adapter);
44         setSupportActionBar(mBinding.myToolbar);
45
46         mCommentTable = FbldLogon.getClient().getTable(ItemComment.class);
47
48         Log.d("Comments", "onCreate");
49     }
50
51     @Override
52     public void onClick(View v) {
53         switch (v.getId()) {
54
55             case R.id.comments:
56                 this.finish();
57                 break;
58             case R.id.post:
59                 if (!mBinding.newcomment.getText().toString().equals("")) {
60                     addItem(v);
61                 }
62                 //this.finish();
63                 break;

```

```

64         default:
65             break;
66     }
67 }
68
69 @Override
70 public void onBackPressed()
71 {
72     this.finish();
73 }
74
75 // Add a new item to the database.
76 private void addItem(View view) {
77     if (!FbblLogon.getLoggedOn()) return;
78
79     // Create a new comment from the ItemComment model.
80     final ItemComment comment = new ItemComment();
81     comment.setId(UUID.randomUUID().toString());
82     comment.setComment(mBinding.newcomment.getText().toString());
83     comment.setUserId(FbblLogon.getUserId());
84     comment.setItemId(CommentListController.getItem().getId());
85
86     // Save the item to the database over the internet.
87     AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
88         @Override
89         protected Void doInBackground(Void... params) {
90             try {
91                 mCommentTable.insert(comment);
92                 Log.d("Comments:insert", "Created comment " + comment.getComment());
93                 runOnUiThread(new Runnable() {
94                     @Override
95                     public void run() {
96                         comment.setAccount(FbblLogon.getAccount());
97                         CommentListController.addComment(comment);
98                     }
99                 });
100             } catch (Exception e) {
101                 Log.d("Comments:insert", e.toString());
102             }
103             return null;
104         }
105     };
106     task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
107     if (this.getCurrentFocus() != null) {
108         InputMethodManager imm = (InputMethodManager) getSystemService(Context.
INPUT_METHOD_SERVICE);
109         imm.hideSoftInputFromWindow(this.getCurrentFocus().getWindowToken(), 0);
110         mBinding.newcomment.setText("");
111     }
112 }
113
114 }
115

```



```

1  /* FblaLogon.java
2  =====
3          Josh Talley and Daniel O'Donnell
4          Dulaney High School
5          Mobile Application Development 2016-17
6  =====
7  Purpose: This class establishes the connection with the Azure Mobile App server
8  and the entire logon process. Part of the logon includes creating and/or fetching
9  the user's Account information. It's done with the logon to make sure communication
10 with the Azure server is working. This class extends AsyncTask because almost all
11 of the login processes must be done in the background.
12 */
13
14 package com.fb\bla.dulaney.fb\blayardsale;
15
16 import android.content.Context;
17 import android.content.SharedPreferences;
18 import android.os.AsyncTask;
19 import android.util.Base64;
20 import android.util.Log;
21 import android.webkit.CookieManager;
22 import android.webkit.ValueCallback;
23
24 import com.fb\bla.dulaney.fb\blayardsale.model.Account;
25 import com.google.common.util.concurrent.FutureCallback;
26 import com.google.common.util.concurrent.Futures;
27 import com.google.common.util.concurrent.ListenableFuture;
28 import com.microsoft.windowsazure.mobileservices.MobileServiceClient;
29 import com.microsoft.windowsazure.mobileservices.authentication.
    MobileServiceAuthenticationProvider;
30 import com.microsoft.windowsazure.mobileservices.authentication.MobileServiceUser;
31 import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
32
33 import org.json.JSONObject;
34
35 import java.net.URLDecoder;
36 import java.nio.charset.StandardCharsets;
37 import java.util.ArrayList;
38 import java.util.Date;
39
40 public class FblaLogon extends AsyncTask {
41     final public static String AZUREURL = "https://fb\bla-yardsale.azurewebsites.net";
42
43     private static boolean mLoggedIn = false;
44     private static String mUserId = null;
45     private static String mToken = null;
46     private static MobileServiceClient mClient = null;
47     private static Account mAccount = null;
48     private static MobileServiceTable<Account> mAccountTable = null;
49
50     private Context mContext;
51     private ArrayList<LogonResultListener> mListeners = new ArrayList<LogonResultListener>()
52 ;
53
54 // Initialize the MobileServiceClient
55 public FblaLogon(Context context) {
56     mContext = context;
57     if (getLoggedIn()) return;
58     // Clear cookies now to support being able to logout easily later.
59     clearCookies();
60     try {
61         mClient = new MobileServiceClient(AZUREURL, mContext);
62     } catch (Exception e) {

```

```

62         Log.d("FblaLogon:init", e.toString());
63         mClient = null;
64     }
65 }
66
67 @Override
68 protected Object doInBackground(Object[] params) {
69     doLogon();
70     return null;
71 }
72
73 @Override
74 protected void onPostExecute(Object result) {
75 }
76
77
78 public static boolean getLoggedOn() {
79     return mLoggedOn;
80 }
81 public void Logoff() {
82     mLoggedOn = false;
83     clearCookies();
84     setCache(mContext, null, null);
85     mAccountTable = null;
86     mAccount = null;
87 }
88
89 public static MobileServiceClient getClient() {
90     return mClient;
91 }
92
93 public static String getUserId() {
94     return mUserId;
95 }
96
97 public static Account getAccount() {
98     return mAccount;
99 }
100
101 public static void setAccount(Account account) {
102     mAccount = account;
103 }
104
105 // Once all of the logon processes are complete, notify any listeners
106 private void onLogonSuccess() {
107     mLoggedOn = true;
108     for (LogonResultListener listener : mListeners) {
109         listener.onLogonComplete(null);
110     }
111     Log.d("FblaLogon", "onLogonSuccess");
112 }
113
114 // Notify any listeners that the logon has failed.
115 private void onLogonFailure(Exception e) {
116     for (LogonResultListener listener : mListeners) {
117         listener.onLogonComplete(e);
118     }
119     Log.d("FblaLogon", "onLogonFailure");
120 }
121
122 // This starts the whole logon chain of asynchronous calls.
123 // So many things can happen in callbacks that I have to chain them all together.
124 private void doLogon() {

```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbld\ulaney\fblyardsale\FblaLogon.java

```
125         if (mClient == null) return;
126         getCache(mContext);
127         if (mUserId == null || mToken == null || isTokenExpired(mToken)) {
128             // Missing or expired token, so need to kick off the Google+ logon process.
129             googleLogon();
130         } else {
131             // The cached token seems to be good, so load the account with it.
132             loadAccount();
133         }
134     }
135
136     // This is the last part of the chain of asynchronous calls.
137     // A successfully loaded account means the token actually works and we can talk to Azure
138
139     // So the callback from this one will call either onLogonSuccess or onLogonFailure.
140     private void loadAccount() {
141         MobileServiceUser user = new MobileServiceUser(mUserId);
142         user.setAuthenticationToken(mToken);
143         mClient.setCurrentUser(user);
144
145         mAccountTable = mClient.getTable(Account.class);
146         ListenableFuture<Account> account = mAccountTable.lookup(mUserId);
147         Futures.addCallback(account, new FutureCallback<Account>() {
148             @Override
149             public void onFailure(Throwable exc) {
150                 // See what kind of exception it is
151                 if (exc.getMessage().equals("{\"error\":\"The item does not exist\"}")) {
152                     // The user is not in the table, so insert a new record for them.
153                     Account act = new Account();
154                     act.setId(mUserId);
155                     mAccountTable.insert(act);
156                     setAccount(act);
157                     Log.d("FblaLogon:account", "AccountEdit Created");
158                     onLogonSuccess();
159                 } else {
160                     // Something else bad happened.
161                     Log.d("FblaLogon:account", exc.toString());
162                     onLogonFailure((Exception) exc);
163                 }
164             }
165
166             @Override
167             public void onSuccess(Account result) {
168                 // Found the account record, so set it on the Data object.
169                 Log.d("FblaLogon:account", "onSuccess - "+result.getId());
170                 setAccount(result);
171                 clearCookies();
172                 onLogonSuccess();
173             }
174         });
175
176     // This is the longest chain of asynchronous processes.
177     // It seems to use WebKit to perform the Google+ OAuth authentication via Azure.
178     // If successful, chain it to loading Accounts. Otherwise call onLogonFailure to notify
179     // the listeners that it didn't work.
180     private void googleLogon() {
181         ListenableFuture<MobileServiceUser> mLogin = mClient.login(
182             MobileServiceAuthenticationProvider.Google);
183         Futures.addCallback(mLogin, new FutureCallback<MobileServiceUser>() {
184             @Override
185             public void onSuccess(MobileServiceUser mobileServiceUser) {
186                 Log.d("FblaLogon:login", "Logged On");
187             }
188         });
189     }
190 }
```

```

186         setCache(mContext, mobileServiceUser.getUserId(), mobileServiceUser.
getAuthenticationToken());
187         loadAccount();
188     }
189
190     @Override
191     public void onFailure(Throwable throwable) {
192         Log.d("FblaLogon:login", throwable.toString());
193         onLogonFailure((Exception)throwable);
194     }
195     });
196 }
197
198 private void clearCookies() {
199     // Clear cookies and cache before logging on
200     CookieManager.getInstance().removeAllCookies(new ValueCallback<Boolean>() {
201         @Override
202         public void onReceiveValue(Boolean value) {
203             Log.d("FblaLogon:cookies", "Cookies cleared");
204         }
205     });
206 }
207
208 // Helper functions to set and clear Shared Preferences cache.
209 private void getCache(Context context) {
210     SharedPreferences prefs = context.getSharedPreferences("auth", Context.MODE_PRIVATE
);
211     mUserId = prefs.getString("usr", null);
212     mToken = prefs.getString("tkn", null);
213 }
214 private void setCache(Context context, String userId, String token) {
215     SharedPreferences prefs = context.getSharedPreferences("auth", Context.MODE_PRIVATE
);
216     SharedPreferences.Editor editor = prefs.edit();
217     if (userId == null) editor.remove("usr");
218     else editor.putString("usr", userId);
219     if (token == null) editor.remove("tkn");
220     else editor.putString("tkn", token);
221     editor.commit();
222     mUserId = userId;
223     mToken = token;
224 }
225
226 // Need to make sure the token has not expired.
227 // https://blogs.msdn.microsoft.com/writingdata_services/2015/04/27/check-for-expired-
azure-mobile-services-authentication-tokens/
228 private boolean isTokenExpired(String token) {
229     // The expiration is embedded in the token. This will rip apart the token to get
the expiration.
230     try {
231         // Get the string after the period
232         String jwt = token.split("\\.")[1];
233         // Decode the URL encoding
234         jwt = URLDecoder.decode(jwt, "UTF-8");
235         // Decode from Base64
236         byte[] bytes = Base64.decode(jwt, Base64.DEFAULT);
237         // Convert the byte array into a string, which is JSON
238         String jsonString = new String(bytes, StandardCharsets.UTF_8);
239         // Create the JSON object
240         JSONObject json = new JSONObject(jsonString);
241         // Extract the expiration from the JSON object
242         jwt = json.getString("exp");
243         // The value is the number of seconds since 1/1/1970 UTC (epoch)

```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbld\ulaney\fbldayardsale\FblaLogon.java

```
244         long exp = Long.parseLong(jwt) * 1000;
245         boolean isExpired = System.currentTimeMillis() >= exp;
246         Log.d("FblaLogon:Token", isExpired ? "Expired " : "Valid " + (new Date(exp)).
toString());
247         return isExpired;
248     } catch (Exception e) {
249         Log.d("FblaLogon:Token", e.toString());
250     }
251     return false;
252 }
253
254 // Add a listener to call after logon is complete
255 public void setLogonListener(LogonResultListener listener) {
256     mListeners.add(listener);
257 }
258
259 // This is the interface to use on the logon callbacks.
260 public interface LogonResultListener {
261     void onLogonComplete(Exception e);
262 }
263 }
264
```

```

1  /* AccountEdit.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: This activity is used to display and edit account information. When
8  a user first logs it, you are forwarded directly to this activity.
9  */
10 package com.fbld.dulaney.fblyardsale;
11
12 import android.databinding.DataBindingUtil;
13 import android.os.AsyncTask;
14 import android.os.Bundle;
15 import android.support.v7.app.AppCompatActivity;
16 import android.util.Log;
17 import android.view.View;
18 import android.widget.AdapterView;
19 import android.widget.Toast;
20
21 import com.fbld.dulaney.fblyardsale.controller.LocalController;
22 import com.fbld.dulaney.fblyardsale.databinding.ActivityAccountBinding;
23 import com.fbld.dulaney.fblyardsale.model.Account;
24 import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
25
26 public class AccountEdit extends AppCompatActivity implements View.OnClickListener {
27
28     ActivityAccountBinding mBinding;
29     ArrayAdapter<CharSequence> mStateAdapter;
30
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_account);
34
35         if (!FbldLogon.getLoggedOn()) {
36             Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
LENGTH_LONG).show();
37             finish();
38             return;
39         }
40
41         mBinding = DataBindingUtil.setContentView(this, R.layout.activity_account);
42         mBinding.save.setOnClickListener(this);
43         mBinding.cancel.setOnClickListener(this);
44
45         // Load the states onto the spinner from the resource file
46         mStateAdapter = ArrayAdapter.createFromResource(this, R.array.states_list, android.R
.layout.simple_spinner_item);
47         mStateAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
;
48
49         mBinding.state.setAdapter(mStateAdapter);
50         setSupportActionBar(mBinding.myToolbar);
51
52         Account account = FbldLogon.getAccount();
53         int spinnerPosition = mStateAdapter.getPosition(account.getState());
54         mBinding.name.setText(account.getName());
55         mBinding.state.setSelection(spinnerPosition);
56         mBinding.address.setText(account.getAddress());
57         mBinding.chapter.setText(account.getChapter());
58         mBinding.region.setText(account.getRegion());
59         mBinding.zip.setText(account.getZipCode());
60     }

```

```

61     @Override
62     public void onClick(View v) {
63         switch (v.getId()) {
64
65             case R.id.save:
66                 if (FblaLogon.getLoggedOn()) {
67                     Account account = FblaLogon.getAccount();
68                     account.setName(mBinding.name.getText().toString());
69                     account.setAddress(mBinding.address.getText().toString());
70                     account.setChapter(mBinding.chapter.getText().toString());
71                     account.setRegion(mBinding.region.getText().toString());
72                     // If zip code changed, refresh the LocalController
73                     if (!account.getZipCode().equals(mBinding.zip.getText().toString())) {
74                         account.setZipCode(mBinding.zip.getText().toString());
75                         LocalController.Refresh();
76                     }
77                     int statePosition = mBinding.state.getSelectedItemPosition();
78                     account.setState(mStateAdapter.getItem(statePosition).toString());
79
80                     // Save the item to the database over the internet.
81                     AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
82                         @Override
83                         protected Void doInBackground(Void... params) {
84                             try {
85                                 MobileServiceTable<Account> mAccountTable = FblaLogon.
getClient().getTable(Account.class);
86                                 mAccountTable.update(FblaLogon.getAccount());
87                                 Log.d("AccountEdit:onClick", "AccountEdit Saved");
88                                 runOnUiThread(new Runnable() {
89                                     @Override
90                                     public void run() {
91                                         finish();
92                                     }
93                                 });
94                             } catch (Exception e) {
95                                 Log.d("AddItem", e.toString());
96                             }
97                             return null;
98                         }
99                     };
100                     task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
101                 }
102                 break;
103             case R.id.cancel:
104                 this.finish();
105                 break;
106             default:
107                 break;
108         }
109     }
110
111     @Override
112     public void onBackPressed()
113     {
114         this.finish();
115     }
116 }
117

```

```

1  /* FblaPicture.java
2  =====
3          Josh Talley and Daniel O'Donnell
4          Dulaney High School
5          Mobile Application Development 2016-17
6  =====
7  Purpose: This class contains a bunch of helper methods that make it easy to
8  manage pictures.
9  */
10 package com.fbla.dulaney.fbayardsale;
11
12 import android.content.Context;
13 import android.graphics.Bitmap;
14 import android.graphics.BitmapFactory;
15 import android.graphics.Point;
16 import android.graphics.drawable.BitmapDrawable;
17 import android.graphics.drawable.Drawable;
18 import android.util.Base64;
19 import android.util.Log;
20 import android.view.Display;
21 import android.view.WindowManager;
22 import android.widget.ImageView;
23 import android.widget.LinearLayout;
24
25 import java.io.ByteArrayOutputStream;
26
27 public class FblaPicture {
28     private static LinearLayout mLayoutImage;
29
30     public static void setLayoutImage(LinearLayout layout)
31     {
32         mLayoutImage = layout;
33     }
34     public static int getImageHeight()
35     {
36         if (mLayoutImage.getHeight() > 0)
37             return mLayoutImage.getHeight() / 2;
38         else return 0;
39     }
40
41     // Returns dimensions of phone in pixels
42     public static Point GetSize(Context c)
43     {
44         WindowManager wm = (WindowManager) c.getSystemService(Context.WINDOW_SERVICE);
45         Display display = wm.getDefaultDisplay();
46         Point size = new Point();
47         if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.HONEYCOMB_MR2
48         )
49         {
50             display.getSize(size);
51         }
52         else // Old Version
53         {
54             size.set(display.getWidth(), display.getHeight());
55         }
56         return size;
57     }
58
59     // Loads a bitmap picture onto the ImageView item on the layout.
60     public static void LoadPictureOnView(ImageView view, Bitmap original) {
61         int vh = getImageHeight();
62         view.setMinimumHeight(vh);
63         view.setMaxHeight(vh);

```



```

63         view.setImageBitmap(original);
64     }
65
66     public static Bitmap GetPictureFromView(ImageView view) {
67         Drawable d = view.getDrawable();
68         if (d == null) return null;
69         return ((BitmapDrawable)d).getBitmap();
70     }
71
72     // Resizes a picture selected from the gallery or taken by the camera so they are a
    common size.
73     public static Bitmap ResizePicture(Context c, Bitmap original) {
74         int w = original.getWidth();
75         int h = original.getHeight();
76         Point screen = GetSize(c);
77         // Force everything to be 500 pixels long
78         int screenL = 500;
79         int originL = (w > h) ? w : h;
80         int originS = (w > h) ? h : w;
81
82         int newS = (int)((float)screenL * ((float)originS / (float)originL));
83         if (w > h)
84         {
85             Log.d("Picture:ResizePicture", "Screen " + screen.x + "x" + screen.y + " From "
+ w + "x" + h + " to " + screenL + "x" + newS);
86             return Bitmap.createScaledBitmap(original, screenL, newS, true);
87         }
88         else
89         {
90             Log.d("Picture:ResizePicture", "Screen " + screen.x + "x" + screen.y + " From "
+ w + "x" + h + " to " + newS + "x" + screenL);
91             return Bitmap.createScaledBitmap(original, newS, screenL, true);
92         }
93     }
94
95     public static String EncodeToBase64(Bitmap image) {
96         if (image == null) return "";
97         // http://stackoverflow.com/questions/9768611/encode-and-decode-bitmap-object-in-
base64-string-in-android
98         ByteArrayOutputStream baos = new ByteArrayOutputStream();
99         image.compress(Bitmap.CompressFormat.PNG, 100, baos);
100         byte[] b = baos.toByteArray();
101         return Base64.encodeToString(b, Base64.DEFAULT);
102     }
103
104     public static Bitmap DecodeFromBase64(String image) {
105         if (image == null || image.equals("")) return null;
106         byte[] decodedImage = Base64.decode(image, Base64.DEFAULT);
107         return BitmapFactory.decodeByteArray(decodedImage, 0, decodedImage.length);
108     }
109 }
110

```

```
1  /* HelpAdapter.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: This is the recycler view adapter for the Help activity.
8  */
9  package com.fb\la\dulaney.fb\layardsale;
10
11  import android.support.v7.widget.RecyclerView;
12  import android.view.LayoutInflater;
13  import android.view.View;
14  import android.view.ViewGroup;
15
16  public class HelpAdapter extends RecyclerView.Adapter<HelpAdapter.ViewHolder> {
17
18      @Override
19      public int getItemViewType(int position) {
20          return position;
21      }
22
23      @Override
24      public HelpAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
25          View v;
26          switch (viewType)
27          {
28              case 0:
29                  v = LayoutInflater.from(parent.getContext())
30                      .inflate(R.layout.help01_register, parent, false);
31                  break;
32              case 1:
33                  v = LayoutInflater.from(parent.getContext())
34                      .inflate(R.layout.help02_login, parent, false);
35                  break;
36              case 2:
37                  v = LayoutInflater.from(parent.getContext())
38                      .inflate(R.layout.help03_switch, parent, false);
39                  break;
40              case 3:
41                  v = LayoutInflater.from(parent.getContext())
42                      .inflate(R.layout.help04_editaccount, parent, false);
43                  break;
44              case 4:
45                  v = LayoutInflater.from(parent.getContext())
46                      .inflate(R.layout.help05_addsales, parent, false);
47                  break;
48              case 5:
49                  v = LayoutInflater.from(parent.getContext())
50                      .inflate(R.layout.help06_viewsales, parent, false);
51                  break;
52              case 6:
53                  v = LayoutInflater.from(parent.getContext())
54                      .inflate(R.layout.help07_deletesales, parent, false);
55                  break;
56              case 7:
57                  v = LayoutInflater.from(parent.getContext())
58                      .inflate(R.layout.help08_comment, parent, false);
59                  break;
60              case 8:
61                  v = LayoutInflater.from(parent.getContext())
62                      .inflate(R.layout.help09_deletcomment, parent, false);
63                  break;
```

```
64         case 9:
65             v = LayoutInflater.from(parent.getContext())
66                 .inflate(R.layout.help10_logout, parent, false);
67             break;
68         default:
69             v = null;
70             break;
71     }
72     if (v == null) return null;
73     return new HelpAdapter.ViewHolder(v);
74 }
75
76 @Override
77 public void onBindViewHolder(HelpAdapter.ViewHolder holder, int position) {
78
79 }
80
81 @Override
82 public int getItemCount() {
83     return 10;
84 }
85
86 class ViewHolder extends RecyclerView.ViewHolder {
87
88     public ViewHolder(View itemView) {
89         super(itemView);
90     }
91 }
92 }
93
```

```

1  /* HomeFragment.java
2  =====
3          Josh Talley and Daniel O'Donnell
4          Dulaney High School
5          Mobile Application Development 2016-17
6  =====
7  Purpose: This is the first fragment loaded on YardSaleMain. It shows the
8  application icon and is used like a menu. Buttons take you other activities.
9  You can also swipe left to get to the Local Sales fragment.
10 */
11 package com.fbldulaney.fblayardsale;
12
13 import android.content.Context;
14 import android.content.Intent;
15 import android.databinding.DataBindingUtil;
16 import android.support.v4.app.Fragment;
17 import android.support.v4.app.FragmentActivity;
18 import android.os.Bundle;
19 import android.view.LayoutInflater;
20 import android.view.View;
21 import android.view.ViewGroup;
22
23 import com.fbldulaney.fblayardsale.databinding.FragmentHomeBinding;
24
25 public class HomeFragment extends Fragment implements View.OnClickListener {
26
27     private OnFragmentInteractionListener mListener;
28     private FragmentActivity mParent;
29     FragmentHomeBinding mBinding;
30
31     @Override
32     public void onClick(View v) {
33         switch (v.getId()) {
34
35             case R.id.account:
36                 if (FbldLogon.getLoggedOn()) {
37                     getActivity().startActivity(new Intent(getActivity(), AccountEdit.class))
38                 };
39                 break;
40             case R.id.add:
41                 if (FbldLogon.getLoggedOn()) {
42                     getActivity().startActivity(new Intent(getActivity(), AddSales.class));
43                 }
44                 break;
45             case R.id.my:
46                 if (FbldLogon.getLoggedOn()) {
47                     getActivity().startActivity(new Intent(getActivity(), MySales.class));
48                 }
49                 break;
50             case R.id.help:
51                 getActivity().startActivity(new Intent(getActivity(), Help.class));
52                 break;
53             case R.id.logout:
54                 YardSaleMain parent = (YardSaleMain) getActivity();
55                 parent.Logoff();
56                 break;
57             default:
58                 break;
59         }
60     }
61
62     public interface OnFragmentInteractionListener {

```

```

63     public void onHomeInteraction(View v);
64 }
65
66 // Implementation of Fragment
67 public static HomeFragment newInstance(String param1, String param2) {
68     HomeFragment fragment = new HomeFragment();
69     Bundle args = new Bundle();
70     fragment.setArguments(args);
71     return fragment;
72 }
73
74 public HomeFragment() {
75     // Required empty public constructor
76 }
77
78 @Override
79 public void onAttach(Context context) {
80     super.onAttach(context);
81     try {
82         mListener = (OnFragmentInteractionListener) context;
83     } catch (ClassCastException e) {
84         throw new ClassCastException(context.toString()
85             + " must implement OnFragmentInteractionListener");
86     }
87 }
88
89 @Override
90 public void onDetach() {
91     super.onDetach();
92     mListener = null;
93 }
94
95 @Override
96 public void onCreate(Bundle savedInstanceState) {
97     super.onCreate(savedInstanceState);
98 }
99
100 @Override
101 public View onCreateView(LayoutInflater inflater, ViewGroup container,
102     Bundle savedInstanceState) {
103
104     mBinding = DataBindingUtil.inflate(
105         inflater, R.layout.fragment_home, container, false);
106     mBinding.account.setOnClickListener(this);
107     mBinding.add.setOnClickListener(this);
108     mBinding.my.setOnClickListener(this);
109     mBinding.help.setOnClickListener(this);
110     mBinding.logout.setOnClickListener(this);
111     View view = mBinding.getRoot();
112
113     // Inflate the layout for this fragment
114     //View v = inflater.inflate(R.layout.fragment_home, container, false);
115     //mParent = getActivity();
116     return view;
117 }
118
119 // Initializes layout items
120 @Override
121 public void onActivityCreated(Bundle bundle) {
122     super.onActivityCreated(bundle);
123     mParent = getActivity();
124 }
125

```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbldulaney\fbayardsale\HomeFragment.java

126 }

127

```

1  /* LocalAdapter.java
2  =====
3          Josh Talley and Daniel O'Donnell
4          Dulaney High School
5          Mobile Application Development 2016-17
6  =====
7  Purpose: This is the recycler view adapter for the Local fragment.
8  */
9  package com.fb\la\dulaney.fb\layardsale;
10
11 import android.databinding.DataBindingUtil;
12 import android.graphics.Bitmap;
13 import android.support.v7.widget.RecyclerView;
14 import android.util.Log;
15 import android.view.LayoutInflater;
16 import android.view.View;
17 import android.view.ViewGroup;
18
19 import com.fb\la\dulaney.fb\layardsale.controller.CommentListController;
20 import com.fb\la\dulaney.fb\layardsale.controller.LocalController;
21 import com.fb\la\dulaney.fb\layardsale.databinding.ListItemsBinding;
22 import com.fb\la\dulaney.fb\layardsale.model.Account;
23 import com.fb\la\dulaney.fb\layardsale.model.SaleItem;
24
25 public class LocalAdapter extends RecyclerView.Adapter<LocalAdapter.ViewHolder> implements
    View.OnClickListener {
26     private View.OnClickListener mParentListener;
27     private ListItemsBinding mBinding;
28
29     public LocalAdapter (View.OnClickListener onClickListener) {
30         mParentListener = onClickListener;
31     }
32
33     @Override
34     public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
35         ListItemsBinding mBinding = DataBindingUtil.inflate(
36             LayoutInflater.from(parent.getContext()), R.layout.list_items, parent, false
37         );
38         mBinding.sold.setOnClickListener(this);
39         mBinding.sold.setVisibility(View.GONE); //this will be variable based on account info
40         mBinding.comments.setOnClickListener(this);
41         View view = mBinding.getRoot();
42
43         return new ViewHolder(view, mBinding);
44     }
45
46     @Override
47     public void onBindViewHolder(ViewHolder holder, int position) {
48         if (!Fb\laLogon.getLoggedOn()) return;
49         SaleItem item = LocalController.getItem(position);
50         if (item != null) {
51             mBinding = holder.getBinding();
52             Log.d("LocalAdapter", "onBindViewHolder");
53             mBinding.comments.setTag(position);
54             mBinding.name.setText(item.getName());
55             mBinding.price.setText(String.format("$%.2f", item.getPrice()));
56             mBinding.description.setText(item.getDescription());
57             Account account = item.getAccount();
58             if (account != null) {
59                 mBinding.address.setText(account.getAddress());
60                 mBinding.chapter.setText(account.getChapter());
61                 mBinding.zipcode.setText(account.getZipCode());
62             }
63         }
64     }
65 }

```

```
62         Bitmap image = item.getPicture();
63         if (image != null) {
64             FblaPicture.setLayoutImage(mBinding.layoutPicture);
65             FblaPicture.LoadPictureOnView(mBinding.picture, image);
66         }
67     }
68 }
69
70 @Override
71 public int getItemCount() {
72     return LocalController.getItemCount();
73 }
74
75 @Override
76 public void onClick(View v) {
77     switch (v.getId()) {
78         case R.id.comments:
79             if (FblaLogon.getLoggedOn()) {
80                 int position = (int)v.getTag();
81                 CommentListController.setItem(LocalController.getItem(position));
82                 CommentListController.Refresh();
83                 mParentListener.onClick(v);
84             }
85             break;
86     }
87 }
88
89 public class ViewHolder extends RecyclerView.ViewHolder {
90     private ListItemsBinding mBinding;
91
92     public ViewHolder(View itemView, ListItemsBinding binding) {
93         super(itemView);
94         mBinding = binding;
95     }
96
97     public ListItemsBinding getBinding() {
98         return mBinding;
99     }
100 }
101 }
102
```



```

1  /* YardSaleMain.java
2  =====
3          Josh Talley and Daniel O'Donnell
4          Dulaney High School
5          Mobile Application Development 2016-17
6  =====
7  Purpose: This is the main startup activity. It uses the FblaPagerAdapter to manage 2
different
8  activity fragments. This activity has the title bar and navigation buttons.
9  The ViewPager fills the center, which holds each page fragment. It automatically handles
10 swipes and smooth transitions between each page. Most navigation is handled by this
activity.
11 This activity will also execute the initial login using a Google+ account. The login
12 uses Azure Mobile Apps to get the user's Google ID and Token, which is cached and used
13 by all other database calls to Azure Mobile Apps.
14 */
15
16 package com.fbld.dulaney.fblyardsale;
17
18 import android.Manifest;
19 import android.content.Intent;
20 import android.content.pm.PackageManager;
21 import android.databinding.DataBindingUtil;
22 import android.net.Uri;
23 import android.os.AsyncTask;
24 import android.os.Build;
25 import android.support.v4.app.ActivityCompat;
26 import android.support.v7.app.AppCompatActivity;
27 import android.os.Bundle;
28
29 import android.view.View;
30 import android.widget.Toast;
31
32 import com.fbld.dulaney.fblyardsale.controller.LocalController;
33 import com.fbld.dulaney.fblyardsale.controller.MySalesController;
34 import com.fbld.dulaney.fblyardsale.databinding.ActivityYardsaleBinding;
35 import com.google.android.gms.appindexing.Action;
36 import com.google.android.gms.appindexing.AppIndex;
37 import com.google.android.gms.appindexing.Thing;
38 import com.google.android.gms.common.api.GoogleApiClient;
39
40 public class YardSaleMain extends AppCompatActivity implements View.OnClickListener,
41     HomeFragment.OnFragmentInteractionListener,
42     LocalFragment.OnFragmentInteractionListener,
43     FblaLogon.LogonResultListener{
44
45     // Class Variables
46     FblaPagerAdapter mPagerAdapter;
47     int mCurrentPage;
48     ActivityYardsaleBinding mBinding;
49     FblaLogon mLogon;
50     public void Logoff() {
51         mLogon.Logoff();
52         //finish();
53         mLogon = new FblaLogon(this);
54         mLogon.setLogonListener(this);
55         mLogon.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
56     }
57     //private MobileServiceTable<AccountEdit> mAccountTable;
58
59     /**
60      * ATTENTION: This was auto-generated to implement the App Indexing API.
61      * See https://g.co/AppIndexing/AndroidStudio for more information.

```

```

62     */
63     private GoogleApiClient client;
64     //private static final String TEMP_FILENAME = "Camera";
65
66     @Override
67     protected void onCreate(Bundle savedInstanceState) {
68         super.onCreate(savedInstanceState);
69         setContentView(R.layout.activity_yardsale);
70
71         mLogon = new FblaLogon(this);
72
73         mBinding = DataBindingUtil.setContentView(this, R.layout.activity_yardsale);
74         mPagerAdapter = new FblaPagerAdapter(getSupportFragmentManager(), this);
75         mBinding.pager.setAdapter(mPagerAdapter);
76         mBinding.home.setOnClickListener(this);
77         mBinding.local.setOnClickListener(this);
78         setSupportActionBar(mBinding.myToolbar);
79
80         mLogon.setLogonListener(this);
81         if (!FblaLogon.isLoggedOn())
82             mLogon.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
83
84         // ATTENTION: This was auto-generated to implement the App Indexing API.
85         // See https://g.co/AppIndexing/AndroidStudio for more information.
86         client = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
87     }
88
89     // @Override
90     public void onClick(View v) {
91         // Perform page changes so they transition just like a swipe.
92         int pg;
93         switch (v.getId()) {
94             case R.id.home:
95                 pg = 0;
96                 break;
97             case R.id.local:
98                 pg = 1;
99                 break;
100             default:
101                 pg = 2;
102                 break;
103         }
104         mBinding.pager.setCurrentItem(pg, true);
105     }
106
107     /**
108     * ATTENTION: This was auto-generated to implement the App Indexing API.
109     * See https://g.co/AppIndexing/AndroidStudio for more information.
110     */
111     public Action getIndexApiAction() {
112         Thing object = new Thing.Builder()
113             .setName("YardSaleMain Page") // TODO: Define a title for the content shown
114
115             // TODO: Make sure this auto-generated URL is correct.
116             .setUrl(Uri.parse("http://[ENTER-YOUR-URL-HERE]"))
117             .build();
118         return new Action.Builder(Action.TYPE_VIEW)
119             .setObject(object)
120             .setActionStatus(Action.STATUS_TYPE_COMPLETED)
121             .build();
122     }
123     @Override

```

```
124     public void onStart() {
125         super.onStart();
126
127         // ATTENTION: This was auto-generated to implement the App Indexing API.
128         // See https://g.co/AppIndexing/AndroidStudio for more information.
129         client.connect();
130         AppIndex.AppIndexApi.start(client, getIndexApiAction());
131     }
132
133     @Override
134     public void onStop() {
135         super.onStop();
136
137         // ATTENTION: This was auto-generated to implement the App Indexing API.
138         // See https://g.co/AppIndexing/AndroidStudio for more information.
139         AppIndex.AppIndexApi.end(client, getIndexApiAction());
140         client.disconnect();
141     }
142
143     public void onHomeInteraction(View v)
144     {
145
146     }
147
148     public void onLocalInteraction(View v)
149     {
150
151     }
152
153     @Override
154     public void onLogonComplete(Exception e) {
155         if (e != null) {
156             Toast.makeText(this, "Unable to connect to Azure. Please try again.", Toast.
LENGTH_LONG).show();
157             finish();
158         } else {
159             MySalesController.Refresh();
160             LocalController.Refresh();
161             String name = FblaLogon.getAccount().getName();
162             if (name == null || name.equals("")) {
163                 startActivity(new Intent(this, AccountEdit.class));
164             }
165         }
166     }
167 }
168
```

```

1  /* LocalFragment.java
2  =====
3          Josh Talley and Daniel O'Donnell
4          Dulaney High School
5          Mobile Application Development 2016-17
6  =====
7  Purpose: This is the second fragment loaded on YardSaleMain. It will display
8  all sale items that are in your same zip code, excluding any of your own
9  sale items. You can also swipe right to get to the Home fragment.
10 */
11 package com.fbld.dulaney.fblyardsale;
12
13 import android.content.Context;
14 import android.content.Intent;
15 import android.databinding.DataBindingUtil;
16 import android.support.v4.app.Fragment;
17 import android.support.v4.app.FragmentActivity;
18 import android.os.Bundle;
19 import android.support.v7.widget.LinearLayoutManager;
20 import android.view.LayoutInflater;
21 import android.view.View;
22 import android.view.ViewGroup;
23
24 import com.fbld.dulaney.fblyardsale.controller.LocalController;
25 import com.fbld.dulaney.fblyardsale.databinding.FragmentLocalBinding;
26
27 public class LocalFragment extends Fragment implements View.OnClickListener {
28
29     private LocalFragment.OnFragmentInteractionListener mListener;
30     private FragmentActivity mParent;
31     FragmentLocalBinding mBinding;
32
33     @Override
34     public void onClick(View v) {
35         switch (v.getId()) {
36             case R.id.comments:
37                 getActivity().startActivity(new Intent(getActivity(), Comments.class));
38                 break;
39             default:
40                 break;
41         }
42     }
43
44     public interface OnFragmentInteractionListener {
45         public void onLocalInteraction(View v);
46     }
47
48     // Implementation of Fragment
49     public static LocalFragment newInstance(String param1, String param2) {
50         LocalFragment fragment = new LocalFragment();
51         Bundle args = new Bundle();
52         fragment.setArguments(args);
53         return fragment;
54     }
55
56     public LocalFragment() {
57         // Required empty public constructor
58     }
59
60     @Override
61     public void onAttach(Context context) {
62         super.onAttach(context);
63         try {

```

```
64         mListener = (LocalFragment.OnFragmentInteractionListener) context;
65     } catch (ClassCastException e) {
66         throw new ClassCastException(context.toString()
67             + " must implement OnFragmentInteractionListener");
68     }
69 }
70
71 @Override
72 public void onDetach() {
73     super.onDetach();
74     mListener = null;
75 }
76
77 @Override
78 public void onCreate(Bundle savedInstanceState) {
79     super.onCreate(savedInstanceState);
80 }
81
82 @Override
83 public View onCreateView(LayoutInflater inflater, ViewGroup container,
84     Bundle savedInstanceState) {
85     mBinding = DataBindingUtil.inflate(
86         inflater, R.layout.fragment_local, container, false);
87     //mBinding.comments.setOnClickListener(this);
88     View view = mBinding.getRoot();
89     return view;
90 }
91
92 @Override
93 public void onActivityCreated(Bundle bundle) {
94     super.onActivityCreated(bundle);
95     // Setup the RecyclerView here because the data changes.
96     mBinding.list.setLayoutManager(new LinearLayoutManager(mParent));
97     LocalAdapter adapter = new LocalAdapter(this);
98     LocalController.AttachAdapter(adapter);
99     LocalController.Refresh();
100    mBinding.list.setAdapter(adapter);
101 }
102
103 }
104
```

```

1  /* MySalesAdapter.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: This is the recycler view adapter for the MySales fragment.
8  */
9  package com.fbla.dulaney.fbayardsale;
10
11  import android.content.DialogInterface;
12  import android.databinding.DataBindingUtil;
13  import android.graphics.Bitmap;
14  import android.os.AsyncTask;
15  import android.support.v7.app.AlertDialog;
16  import android.support.v7.widget.RecyclerView;
17  import android.util.Log;
18  import android.view.LayoutInflater;
19  import android.view.View;
20  import android.view.ViewGroup;
21  import android.widget.TextView;
22
23  import com.fbla.dulaney.fbayardsale.controller.CommentListController;
24  import com.fbla.dulaney.fbayardsale.controller.MySalesController;
25  import com.fbla.dulaney.fbayardsale.databinding.ListItemsBinding;
26  import com.fbla.dulaney.fbayardsale.model.SaleItem;
27  import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
28
29  public class MySalesAdapter extends RecyclerView.Adapter<MySalesAdapter.ViewHolder>
    implements View.OnClickListener {
30      private View.OnClickListener mParentListener;
31      ListItemsBinding mBinding;
32      MySales mContext;
33
34      public MySalesAdapter (MySales context, View.OnClickListener onClickListener) {
35          mContext = context;
36          mParentListener = onClickListener;
37      }
38
39      @Override
40      public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
41          ListItemsBinding mBinding = DataBindingUtil.inflate(
42              LayoutInflater.from(parent.getContext()), R.layout.list_items, parent, false
43          );
44          mBinding.sold.setOnClickListener(this);
45          mBinding.comments.setOnClickListener(this);
46          mBinding.layoutAddress.setVisibility(View.GONE);
47          mBinding.layoutChapter.setVisibility(View.GONE);
48          mBinding.layoutZipcode.setVisibility(View.GONE);
49          View view = mBinding.getRoot();
50
51          return new ViewHolder(view, mBinding);
52      }
53
54      @Override
55      public void onBindViewHolder(ViewHolder holder, int position) {
56          if (!FblaLogon.getLoggedOn()) return;
57          SaleItem item = MySalesController.getItem(position);
58          if (item != null) {
59              mBinding = holder.getBinding();
60              Log.d("MySalesAdapter", "onBindViewHolder");
61              mBinding.comments.setTag(position);
62              mBinding.name.setText(item.getName());

```

```

62         mBinding.price.setText (String.format ("%$.2f", item.getPrice()));
63         mBinding.description.setText (item.getDescription());
64         mBinding.sold.setTag (position);
65         Bitmap image = item.getPicture();
66         if (image != null) {
67             FbldPicture.setLayoutImage (mBinding.layoutPicture);
68             FbldPicture.LoadPictureOnView (mBinding.picture, image);
69         }
70     }
71 }
72
73 @Override
74 public int getItemCount () {
75     return MySalesController.getItemCount ();
76 }
77
78 @Override
79 public void onClick (View v) {
80     switch (v.getId()) {
81         case R.id.comments:
82             if (FbldLogon.getLoggedOn ()) {
83                 int position = (int)v.getTag();
84                 SaleItem item = MySalesController.getItem (position);
85                 CommentListController.setItem (item);
86                 CommentListController.Refresh ();
87                 Log.d ("MySalesAdapter", "Refreshed for " + position);
88                 mParentListener.onClick (v);
89             }
90             break;
91         case R.id.sold:
92             final int position = (int)v.getTag();
93             AlertDialog.Builder builder = new AlertDialog.Builder (mContext);
94             builder.setTitle ("Are You Sure?");
95             final TextView info = new TextView (mContext);
96             info.setText ("By Pressing Confirm, The Item Will Be Deleted.");
97             info.setPadding (30, 0, 0, 0);
98             builder.setView (info);
99
100             builder.setPositiveButton ("Confirm", new DialogInterface.OnClickListener () {
101
102                 @Override
103                 public void onClick (DialogInterface dialog, int which) {
104                     Log.d ("MySalesAdapter", "delete");
105                     deleteItem (position);
106                     dialog.dismiss ();
107                 }
108             });
109
110             builder.setNegativeButton ("Cancel", new DialogInterface.OnClickListener () {
111
112                 @Override
113                 public void onClick (DialogInterface dialog, int which) {
114                     dialog.cancel ();
115                 }
116             });
117
118             builder.show ();
119             break;
120         default:
121             break;
122     }
123 }
124
125 public class ViewHolder extends RecyclerView.ViewHolder {

```

```

124     private ListItemsBinding mBinding;
125
126     public ViewHolder(View itemView, ListItemsBinding binding) {
127         super(itemView);
128         mBinding = binding;
129     }
130
131     public ListItemsBinding getBinding() {
132         return mBinding;
133     }
134 }
135
136 private void deleteItem(int position) {
137     if (!FblaLogon.getLoggedOn()) return;
138
139     final int pos = position;
140     final SaleItem item = MySalesController.getItem(position);
141     final MobileServiceTable<SaleItem> mSaleItemTable = FblaLogon.getClient().getTable(
SaleItem.class);
142     // Delete the comment from the database.
143     AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
144         @Override
145         protected Void doInBackground(Void... params) {
146             try {
147                 mSaleItemTable.delete(item);
148                 Log.d("MySales:delete", "Deleted item " + item.getName());
149                 mContext.runOnUiThread(new Runnable() {
150                     @Override
151                     public void run() {
152                         MySalesController.removeItem(pos);
153                     }
154                 });
155             } catch (Exception e) {
156                 Log.d("MySales:delete", e.toString());
157             }
158             return null;
159         }
160     };
161     task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
162 }
163 }
164

```



```

1  /* CommentsAdapter.java
2  =====
3          Josh Talley and Daniel O'Donnell
4          Dulaney High School
5          Mobile Application Development 2016-17
6  =====
7  Purpose: This adapter is used by the Comments activity to manage the list of
8  comments. It makes use of the CommentListController.
9  */
10 package com.fbld.dulaney.fblyardsale;
11
12 import android.content.DialogInterface;
13 import android.databinding.DataBindingUtil;
14 import android.os.AsyncTask;
15 import android.support.v7.app.AlertDialog;
16 import android.support.v7.widget.RecyclerView;
17 import android.util.Log;
18 import android.view.LayoutInflater;
19 import android.view.View;
20 import android.view.ViewGroup;
21 import android.widget.TextView;
22
23 import com.fbld.dulaney.fblyardsale.controller.CommentListController;
24 import com.fbld.dulaney.fblyardsale.databinding.ListCommentsBinding;
25 import com.fbld.dulaney.fblyardsale.model.ItemComment;
26 import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
27
28 public class CommentsAdapter extends RecyclerView.Adapter<CommentsAdapter.ViewHolder>
29 implements View.OnClickListener {
30     private View.OnClickListener mParentListener;
31     ListCommentsBinding mBinding;
32     Comments mContext;
33
34     public CommentsAdapter (Comments context, View.OnClickListener onClickListener) {
35         mContext = context;
36         mParentListener = onClickListener;
37     }
38
39     @Override
40     public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
41         ListCommentsBinding mBinding = DataBindingUtil.inflate(
42             LayoutInflater.from(parent.getContext()), R.layout.list_comments, parent,
43             false);
44         View view = mBinding.getRoot();
45         mBinding.delete.setOnClickListener(this);
46
47         Log.d("CommentsAdapter", "onCreateViewHolder");
48         return new ViewHolder(view, mBinding);
49     }
50
51     @Override
52     public void onBindViewHolder(ViewHolder holder, int position) {
53         if (!FbldLogon.getLoggedOn()) return;
54         ItemComment comment = CommentListController.getComment(position);
55         if (comment != null) {
56             mBinding = holder.getBinding();
57             Log.d("CommentsAdapter", "onBindViewHolder");
58             mBinding.comments.setText(comment.getComment());
59             if (comment.getAccount() == null) mBinding.username.setText("{Unknown}");
60             else mBinding.username.setText(comment.getAccount().getName());
61             mBinding.delete.setTag(position);
62         }
63     }
64 }

```

```

62
63     @Override
64     public int getItemCount() {
65         return CommentListController.getCommentCount();
66     }
67
68     @Override
69     public void onClick(View v) {
70         switch (v.getId()) {
71             case R.id.delete:
72                 final int position = (int)v.getTag();
73                 AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
74                 builder.setTitle("Are You Sure?");
75                 final TextView info = new TextView(mContext);
76                 info.setText("By Pressing Confirm, The Comment Will Be Deleted.");
77                 info.setPadding(30, 0, 0, 0);
78                 builder.setView(info);
79
80                 builder.setPositiveButton("Confirm", new DialogInterface.OnClickListener()
81 {
82                     @Override
83                     public void onClick(DialogInterface dialog, int which) {
84                         deleteComment(position);
85                         dialog.dismiss();
86                     }
87                 });
88                 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
89                     @Override
90                     public void onClick(DialogInterface dialog, int which) {
91                         dialog.cancel();
92                     }
93                 });
94
95                 builder.show();
96                 break;
97             default:
98                 break;
99         }
100     }
101
102     private void deleteComment(int position) {
103         if (!FblaLogon.getLoggedOn()) return;
104
105         final int pos = position;
106         final ItemComment comment = CommentListController.getComment(position);
107         final MobileServiceTable<ItemComment> mCommentTable = FblaLogon.getClient().
108             getTable(ItemComment.class);
109         // Delete the comment from the database.
110         AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
111             @Override
112             protected Void doInBackground(Void... params) {
113                 try {
114                     mCommentTable.delete(comment);
115                     Log.d("Comments:delete", "Deleted comment " + comment.getComment());
116                     mContext.runOnUiThread(new Runnable() {
117                         @Override
118                         public void run() {
119                             CommentListController.removeComment(pos);
120                         }
121                     });
122                 } catch (Exception e) {
123                     Log.d("Comments:delete", e.toString());
124                 }
125             }
126         };
127     }

```

```
123         }
124         return null;
125     }
126     };
127     task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
128 }
129
130 public class ViewHolder extends RecyclerView.ViewHolder {
131     private ListCommentsBinding mBinding;
132
133     public ViewHolder(View itemView, ListCommentsBinding binding) {
134         super(itemView);
135         mBinding = binding;
136     }
137
138     public ListCommentsBinding getBinding() {
139         return mBinding;
140     }
141 }
142 }
143
```

```
1  /* FblaPagerAdapter.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: This simply loads the appropriate fragment onto the YardSaleMain activity.
8  */
9  package com.fbldulaney.fblyardsale;
10
11  import android.content.Context;
12  import android.os.Bundle;
13  import android.support.v4.app.Fragment;
14  import android.support.v4.app.FragmentManager;
15  import android.support.v4.app.FragmentStatePagerAdapter;
16
17  public class FblaPagerAdapter extends FragmentStatePagerAdapter {
18      protected Context mContext;
19
20      public FblaPagerAdapter(FragmentManager fm, Context context)
21      {
22          super(fm);
23          mContext = context;
24      }
25
26      @Override
27      public Fragment getItem(int position) {
28          Fragment fragment;
29          switch (position)
30          {
31              case 0:
32                  fragment = new HomeFragment();
33                  break;
34              default:
35                  fragment = new LocalFragment();
36                  break;
37          }
38          Bundle args = new Bundle();
39          args.putInt("page_position", position);
40
41          fragment.setArguments(args);
42
43          return fragment;
44      }
45
46      @Override
47      public int getCount() {
48          return 2;
49      }
50  }
51
```

```
1  /* Account.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: Model of the Azure database table for user account information.
8  */
9  package com.fbld.dulaney.fblyardsale.model;
10
11 public class Account {
12     /*
13     User Id
14     */
15     @com.google.gson.annotations.SerializedName("id")
16     private String mId;
17
18     /*
19     User Name
20     */
21     @com.google.gson.annotations.SerializedName("name")
22     private String mName;
23
24     /*
25     State
26     */
27     @com.google.gson.annotations.SerializedName("state")
28     private String mState;
29
30     /*
31     Region
32     */
33     @com.google.gson.annotations.SerializedName("region")
34     private String mRegion;
35
36     /*
37     FBLA Chapter
38     */
39     @com.google.gson.annotations.SerializedName("chapter")
40     private String mChapter;
41
42     /*
43     Address
44     */
45     @com.google.gson.annotations.SerializedName("address")
46     private String mAddress;
47
48     /*
49     Zip Code
50     */
51     @com.google.gson.annotations.SerializedName("zipcode")
52     private String mZipCode;
53
54     public Account() {
55         mId = "";
56         mName = "";
57         mAddress = "";
58         mChapter = "";
59         mRegion = "";
60         mState = "";
61         mZipCode = "";
62     }
63 }
```

```
64     @Override
65     public String toString() {
66         return getId();
67     }
68
69     // Getters and Setters
70     public String getId() { return mId; }
71     public final void setId(String id) { mId = id; }
72     public String getName() { return mName; }
73     public final void setName(String name) { mName = name; }
74     public String getState() { return mState; }
75     public final void setState(String state) { mState = state; }
76     public String getRegion() { return mRegion; }
77     public final void setRegion(String region) { mRegion = region; }
78     public String getChapter() { return mChapter; }
79     public final void setChapter(String chapter) { mChapter = chapter; }
80     public String getAddress() { return mAddress; }
81     public final void setAddress(String address) { mAddress = address; }
82     public String getZipCode() { return mZipCode; }
83     public final void setZipCode(String zipCode) { mZipCode = zipCode; }
84
85     @Override
86     public boolean equals(Object o) {
87         return o instanceof Account && ((Account)o).mId == mId;
88     }
89 }
90
```

```
1  /* SaleItem.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: Model of the Azure database table for sale item information.
8  */
9  package com.fbldulaneyfbayardsale.model;
10
11  import android.graphics.Bitmap;
12
13  import com.fbldulaneyfbayardsale.FblaPicture;
14
15  public class SaleItem {
16      /*
17       Item Id
18       */
19      @com.google.gson.annotations.SerializedName("id")
20      private String mId;
21
22      /*
23       User Id
24       */
25      @com.google.gson.annotations.SerializedName("userid")
26      private String mUserId;
27
28
29      /*
30       User Id
31       */
32      @com.google.gson.annotations.SerializedName("name")
33      private String mName;
34
35      /*
36       Item Description
37       */
38      @com.google.gson.annotations.SerializedName("description")
39      private String mDescription;
40
41      /*
42       Item Price
43       */
44      @com.google.gson.annotations.SerializedName("price")
45      private float mPrice;
46
47      /*
48       Picture (Base64 Encoded String)
49       */
50      @com.google.gson.annotations.SerializedName("picture")
51      private String mPictureBase64;
52
53      @com.google.gson.annotations.Expose(serialize = false)
54      private Account mAccount;
55
56      public SaleItem() {
57          mAccount = null;
58          mName = "";
59          mId = "";
60          mUserId = "";
61          mDescription = "";
62          mPictureBase64 = "";
63          mPrice = 0;
```

```
64     }
65
66     @Override
67     public String toString() {
68         return getId();
69     }
70
71     // Getters and Setters
72     public String getId() { return mId; }
73     public final void setId(String id) { mId = id; }
74     public String getUserId() { return mUserId; }
75     public final void setUserId(String userId) { mUserId = userId; }
76     public String getName() { return mName; }
77     public final void setName(String name) { mName = name; }
78     public String getDescription() { return mDescription; }
79     public final void setDescription(String description) { mDescription = description; }
80     public float getPrice() { return mPrice; }
81     public final void setPrice(float price) { mPrice = price; }
82     public Bitmap getPicture() {
83         return FbPicture.DecodeFromBase64(mPictureBase64);
84     }
85     public final void setPicture(Bitmap image) {
86         mPictureBase64 = FbPicture.EncodeToBase64(image);
87     }
88     public Account getAccount() { return mAccount; }
89     public final void setAccount(Account account) { mAccount = account; }
90
91     @Override
92     public boolean equals(Object o) {
93         return o instanceof SaleItem && ((SaleItem)o).mId == mId;
94     }
95 }
96
```



```

1  /* ItemComment.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: Model of the Azure database table for item comment information.
8  */
9  package com.fbld.dulaney.fblyardsale.model;
10
11 public class ItemComment {
12     /*
13     Item Id
14     */
15     @com.google.gson.annotations.SerializedName("id")
16     private String mId;
17
18     /*
19     User Id
20     */
21     @com.google.gson.annotations.SerializedName("userid")
22     private String mUserId;
23
24     /*
25     Item Id
26     */
27     @com.google.gson.annotations.SerializedName("itemid")
28     private String mItemId;
29
30     /*
31     Comment
32     */
33     @com.google.gson.annotations.SerializedName("comment")
34     private String mComment;
35
36     @com.google.gson.annotations.Expose(serialize = false)
37     private Account mAccount;
38
39     public ItemComment() {
40         mAccount = null;
41         mId = "";
42         mUserId = "";
43         mItemId = "";
44         mComment = "";
45     }
46
47     @Override
48     public String toString() {
49         return getId();
50     }
51
52     // Getters and Setters
53     public String getId() { return mId; }
54     public final void setId(String id) { mId = id; }
55     public String getUserId() { return mUserId; }
56     public final void setUserId(String userId) { mUserId = userId; }
57     public String getItemId() { return mItemId; }
58     public final void setItemId(String itemId) { mItemId = itemId; }
59     public String getComment() { return mComment; }
60     public final void setComment(String comment) { mComment = comment; }
61     public Account getAccount() { return mAccount; }
62     public final void setAccount(Account account) { mAccount = account; }
63

```

File - C:\Users\josh\AndroidStudioProjects\FBLAYardSale\app\src\main\java\com\fbla\dulaney\fblayardsale\model\ItemComment.jav

```
64     @Override
65     public boolean equals(Object o) {
66         return o instanceof ItemComment && ((ItemComment)o).mId == mId;
67     }
68 }
69
```

```

1  /* LocalController.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: Used by LocalFragment to control access to the list of Sale Items in
8  the user's local area (by zip code). Attaching a recycler view to the class
9  so that when the list of items is refreshed or changed, the recycler view is
10 notified of that change.
11 */
12 package com.fbld.dulaney.fblyardsale.controller;
13
14 import android.os.AsyncTask;
15 import android.support.v7.widget.RecyclerView;
16 import android.util.Log;
17
18 import com.fbld.dulaney.fblyardsale.FbldLogon;
19 import com.fbld.dulaney.fblyardsale.model.Account;
20 import com.fbld.dulaney.fblyardsale.model.SaleItem;
21 import com.microsoft.windowsazure.mobileservices.MobileServiceList;
22 import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
23
24 import java.util.ArrayList;
25
26 public class LocalController {
27     private static ArrayList<SaleItem> mSaleItems = new ArrayList<>();
28     private static ArrayList<RecyclerView.Adapter> mAdapters = new ArrayList<>();
29
30     public static void AttachAdapter(RecyclerView.Adapter adapter) {
31         mAdapters.add(adapter);
32     }
33
34     public static int getItemCount() {
35         return mSaleItems.size();
36     }
37
38     public static SaleItem getItem(int position) {
39         if (mSaleItems.size() > position) return mSaleItems.get(position);
40         else return null;
41     }
42
43     public static void addItem(SaleItem item) {
44         mSaleItems.add(item);
45         for (RecyclerView.Adapter adapter : mAdapters) {
46             adapter.notifyDataSetChanged();
47         }
48     }
49
50     public static void removeItem(int position) {
51         mSaleItems.remove(position);
52         for (RecyclerView.Adapter adapter : mAdapters) {
53             adapter.notifyDataSetChanged();
54         }
55     }
56
57     private static MobileServiceTable<Account> mAccountTable;
58     private static MobileServiceTable<SaleItem> mSaleItemTable;
59     public static void Refresh() {
60         if (!FbldLogon.getLoggedOn()) return;
61         mSaleItems.clear();
62
63         mAccountTable = FbldLogon.getClient().getTable(Account.class);

```

```

64         mSaleItemTable = FbldLogon.getClient().getTable(SaleItem.class);
65         new AsyncTask<Void, Void, Void>() {
66             @Override
67             protected Void doInBackground(Void... params) {
68                 try {
69                     Account myAccount = FbldLogon.getAccount();
70                     // First get all of the accounts in the same zip code.
71                     final MobileServiceList<Account> accounts =
72                         mAccountTable.where().field("zipcode").eq(myAccount.getZipCode(
73 ))).execute().get();
74                     for (Account account : accounts) {
75                         // Now get all the items for those accounts (excluding your own)
76                         if (!account.getId().equals(myAccount.getId())) {
77                             final MobileServiceList<SaleItem> result =
78                                 mSaleItemTable.where().field("userid").eq(account.getId
79 ())).execute().get();
80                             for (SaleItem item : result) {
81                                 item.setAccount(account);
82                                 mSaleItems.add(item);
83                             }
84                         } catch (Exception exception) {
85                             Log.e("MySalesController", exception.toString());
86                         }
87                     return null;
88                 }
89             }
90             @Override
91             protected void onPostExecute(Void v) {
92                 for (RecyclerView.Adapter adapter : mAdapters) {
93                     adapter.notifyDataSetChanged();
94                 }
95             }.execute();
96         }
97     }
98

```

```

1  /* MySalesController.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: Used by MySalesFragment to control access to the list of Sale Items owned
8  by the user. Attaching a recycler view to the class so that when the list of
9  items is refreshed or changed, the recycler view is notified of that change.
10 */
11 package com.fbld.dulaney.fblyardsale.controller;
12
13 import android.os.AsyncTask;
14 import android.support.v7.widget.RecyclerView;
15 import android.util.Log;
16
17 import com.fbld.dulaney.fblyardsale.FbldLogon;
18 import com.fbld.dulaney.fblyardsale.model.SaleItem;
19 import com.microsoft.windowsazure.mobileservices.MobileServiceList;
20 import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
21
22 import java.util.ArrayList;
23
24 public class MySalesController {
25     private static ArrayList<SaleItem> mSaleItems = new ArrayList<>();
26     private static ArrayList<RecyclerView.Adapter> mAdapters = new ArrayList<>();
27
28     public static void AttachAdapter(RecyclerView.Adapter adapter) {
29         mAdapters.add(adapter);
30     }
31
32     public static int getItemCount() {
33         return mSaleItems.size();
34     }
35
36     public static SaleItem getItem(int position) {
37         if (mSaleItems.size() > position) return mSaleItems.get(position);
38         else return null;
39     }
40
41     public static void addItem(SaleItem item) {
42         mSaleItems.add(item);
43         for (RecyclerView.Adapter adapter : mAdapters) {
44             adapter.notifyDataSetChanged();
45         }
46     }
47
48     public static void removeItem(int position) {
49         mSaleItems.remove(position);
50         for (RecyclerView.Adapter adapter : mAdapters) {
51             adapter.notifyDataSetChanged();
52         }
53     }
54
55     private static MobileServiceTable<SaleItem> mSaleItemTable;
56     public static void Refresh() {
57         if (!FbldLogon.getLoggedOn()) return;
58         mSaleItems.clear();
59
60         mSaleItemTable = FbldLogon.getClient().getTable(SaleItem.class);
61         new AsyncTask<Void, Void, Void>() {
62             @Override
63             protected Void doInBackground(Void... params) {

```

```
64         try {
65             final MobileServiceList<SaleItem> result =
66                 mSaleItemTable.where().field("userid").eq(FblaLogon.getUserId()
67         ).execute().get();
68             for (SaleItem item : result) {
69                 mSaleItems.add(item);
70             } catch (Exception exception) {
71                 Log.e("MySalesController", exception.toString());
72             }
73             return null;
74         }
75         @Override
76         protected void onPostExecute(Void v) {
77             for (RecyclerView.Adapter adapter : mAdapters) {
78                 adapter.notifyDataSetChanged();
79             }
80         }
81     }.execute();
82 }
83 }
84 }
```

```

1  /* CommentListController.java
2  =====
3      Josh Talley and Daniel O'Donnell
4      Dulaney High School
5      Mobile Application Development 2016-17
6  =====
7  Purpose: Used by CommentList to control access to the list of comments for
8  a selected item. Attaching a recycler view to the class so that when the list of
9  items is refreshed or changed, the recycler view is notified of that change.
10 */
11 package com.fbld.dulaney.fblyardsale.controller;
12
13 import android.os.AsyncTask;
14 import android.support.v7.widget.RecyclerView;
15 import android.util.Log;
16
17 import com.fbld.dulaney.fblyardsale.FbldLogon;
18 import com.fbld.dulaney.fblyardsale.model.Account;
19 import com.fbld.dulaney.fblyardsale.model.ItemComment;
20 import com.fbld.dulaney.fblyardsale.model.SaleItem;
21 import com.microsoft.windowsazure.mobileservices.MobileServiceList;
22 import com.microsoft.windowsazure.mobileservices.table.MobileServiceTable;
23
24 import java.util.ArrayList;
25
26 public class CommentListController {
27     private static ArrayList<ItemComment> mComments = new ArrayList<>();
28     private static ArrayList<RecyclerView.Adapter> mAdapters = new ArrayList<>();
29     private static MobileServiceTable<ItemComment> mItemCommentTable;
30     private static SaleItem mItem;
31
32     public static void AttachAdapter(RecyclerView.Adapter adapter) {
33         mAdapters.add(adapter);
34     }
35     public static void RemoveAdapter(RecyclerView.Adapter adapter) { mAdapters.remove(
36         adapter); }
37
38     public static int getCommentCount() {
39         return mComments.size();
40     }
41
42     public static ItemComment getComment(int position) {
43         if (mComments.size() > position) return mComments.get(position);
44         else return null;
45     }
46
47     // Add a comment and notify the adapter of the change
48     public static void addComment(ItemComment comment) {
49         mComments.add(comment);
50         for (RecyclerView.Adapter adapter : mAdapters) {
51             adapter.notifyDataSetChanged();
52         }
53     }
54
55     // Remove a comment and notify the adapter of the change
56     public static void removeComment(int position) {
57         mComments.remove(position);
58         for (RecyclerView.Adapter adapter : mAdapters) {
59             adapter.notifyDataSetChanged();
60         }
61     }
62
63     public static SaleItem getItem() { return mItem; }

```

```

63     public static void setItem(SaleItem item) { mItem = item; }
64
65     // Refresh all comments and notify the adapter of the change
66     public static void Refresh() {
67         if (!FblaLogon.getLoggedOn()) return;
68         mComments.clear();
69
70         mItemCommentTable = FblaLogon.getClient().getTable(ItemComment.class);
71         final MobileServiceTable<Account> mAccountTable = FblaLogon.getClient().getTable(
Account.class);
72         new AsyncTask<Void, Void, Void>() {
73             @Override
74             protected Void doInBackground(Void... params) {
75                 try {
76                     final MobileServiceList<ItemComment> result =
77                         mItemCommentTable.where().field("itemid").eq(mItem.getId()).
execute().get();
78                     for (ItemComment comment : result) {
79                         Account account = mAccountTable.lookup(comment.getUserId()).get();
80                         comment.setAccount(account);
81                         mComments.add(comment);
82                     }
83                 } catch (Exception exception) {
84                     Log.e("CommentListController", exception.toString());
85                 }
86                 return null;
87             }
88             @Override
89             protected void onPostExecute(Void v) {
90                 for (RecyclerView.Adapter adapter : mAdapters) {
91                     adapter.notifyDataSetChanged();
92                 }
93             }
94         }.execute();
95     }
96 }
97

```