Android

Generated by Doxygen 1.8.16

Chapter 1

TDF02-145 Remote

1.1 Introduction

This is the repository for Android remote app related matters for robot for Autism Spectrum Disorder therapy development funded through HEC TDF. The documentation folder contains all the code for Android remote division of HEC funded project TDF 02-145. There are multiple files contained in this folder. This code will run on most Android phones and Tablets.

1.2 Things to know

This section details the things one must know before using this code.

1.2.1 MAC Address

- You'll need the MAC Address of whatever device the remote will be connecting to. You have to update that strings.xml
- This code, for now, only works if the other device is paired with the remote device.
- It only supports Serial Communication.

1.2.2 What works how

- There are three section - Expression, Locomotion, and Speaking
- Expressions will change on short presses. Long presses play audio files. For now, these mp3s have to be present in the "face device".
- Locomotion simply moves the robot by a "1000" steps of the encoder.
- Speaking features custom text message sending, pitch control, and pre-written most used phrases.

1.2.2.1 Author

Taha Shaheen, Saifullah, Muhammad Wajahat Qureshi

1.2.2.2 Version

chotuX

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Package com

Packages

- package example

6.2 Package com.example

Packages

- package choturemote

6.3 Package com.example.choturemote

Packages

- package ui

Classes

- class MainActivity

  This is where everything important happens.
- class MyBluetoothService

  Handles everything Bluetooth.

6.4 Package com.example.choturemote.ui

Packages

- package main

6.5 Package com.example.choturemote.ui.main

Classes

- class ExpressionsFragment

  Expressions Fragment.
- class ExpressionsWordAdapter

  An ArrayAdapter Implementation.
- class LocomotionFragment

  Locomotion Fragment.
- class LocomotionWordAdapter

  An ArrayAdapter Implementation.
- class SectionsPagerAdapter

  Handles the tabbed "Whatsapp" look.
- class SpeakingFragment

  Speaking Fragment.
- class Word

  Represents a collection of resource IDs.

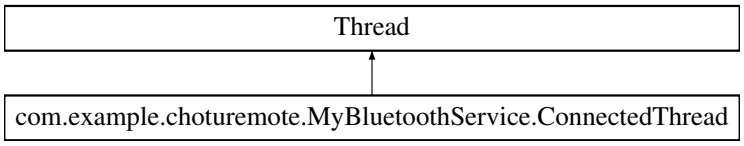
Chapter 7

Class Documentation

7.1 com.example.choturemote.MyBluetoothService.ConnectedThread Class Reference

Thread that manages a Bluetooth connection.

Inheritance diagram for com.example.choturemote.MyBluetoothService.ConnectedThread:



Public Member Functions

- ConnectedThread (BluetoothSocket socket)

  Constructor for the ConnectedThread class.
- void run ()

  Main code that runs in the Thread.
- void write (byte[ ] bytes)

  Sends data to the remote device.
- void cancel ()

  Closes the socket.

Private Attributes

- final BluetoothSocket mmSocket
- final InputStream mmInStream
- final OutputStream mmOutStream

7.1.1 Detailed Description

Thread that manages a Bluetooth connection.

- A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.
- This one manages a BluetoothSocket

Definition at line 248 of file MyBluetoothService.java.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 ConnectedThread()

```
com.example.choturemote.MyBluetoothService.ConnectedThread.ConnectedThread (
            BluetoothSocket socket )
```

Constructor for the ConnectedThread class.

Parameters

socket	RFCOMM Bluetooth Socket object

Manages a RFCOMM Bluetooth Socket

Definition at line 269 of file MyBluetoothService.java.

```
269                                                 {
270             Log.d(TAG, "ConnectedThread: Starting.");
271
272             mmSocket = socket;
273
274             // Temporary storage for  an input stream for reading bytes from this socket//
275             InputStream tmpIn = null;
276
277             // Temporary storage for  an output stream for writing bytes from this socket//
278             OutputStream tmpOut = null;
279
280             // dismiss the progressDialog when connection is established //
281             try {
282                 mProgressDialog.dismiss();
283             } catch (NullPointerException e) {
284                 e.printStackTrace();
285             }
286
287             // Get the InputStream and OutputStream that handle transmissions through the socket using
        getInputStream() and getOutputStream(), respectively. //
288             try {
289                 tmpIn = mmSocket.getInputStream();
290                 tmpOut = mmSocket.getOutputStream();
291             } catch (IOException e) {
292                 e.printStackTrace();
293             }
294
295             mmInStream = tmpIn;
296             mmOutStream = tmpOut;
297
298             mConnectedThreadRunning = true;
299         }
```

References com.example.choturemote.MyBluetoothService.mConnectedThreadRunning, com.example.↩choturemote.MyBluetoothService.ConnectedThread.mmInStream, com.example.choturemote.MyBluetooth↩Service.ConnectedThread.mmOutStream, com.example.choturemote.MyBluetoothService.ConnectedThread.↩mmSocket, com.example.choturemote.MyBluetoothService.mProgressDialog, and com.example.choturemote.↩MyBluetoothService.TAG.

7.1.3 Member Function Documentation

7.1.3.1 cancel()

```
void com.example.choturemote.MyBluetoothService.ConnectedThread.cancel ( )
```

Closes the socket.

Closes the client socket and causes the thread to finish. Called from the main activity to shut down the connection.

Definition at line 347 of file MyBluetoothService.java.

```
347                                 {
348             try {
349                 mmSocket.close();
350                 mConnectedThreadRunning = false;
351             } catch (IOException e) {
352             }
353         }
```

References com.example.choturemote.MyBluetoothService.mConnectedThreadRunning, and com.example.↩choturemote.MyBluetoothService.ConnectedThread.mmSocket.

Referenced by com.example.choturemote.MyBluetoothService.cancelThreads().

7.1.3.2 run()

```
void com.example.choturemote.MyBluetoothService.ConnectedThread.run ( )
```

Main code that runs in the Thread.

Try Catch to read data being sent through the connection with the remote device.

Definition at line 306 of file MyBluetoothService.java.

```
306                                 {
307             // buffer store for the stream //
308             byte[] buffer = new byte[1024];
309
310             // bytes returned from read() //
311             int bytes;
312
313             // Keep listening to the InputStream until an exception occurs //
314             while (true) {
315                 // Read from the InputStream //
316                 try {
317                     bytes = mmInStream.read(buffer);
318                     String incomingMessage = new String(buffer, 0, bytes);
319                     Log.d(TAG, "InputStream: " + incomingMessage);
320                 } catch (IOException e) {
321                     Log.e(TAG, "write: Error reading Input Stream. " + e.getMessage());
322                     break;
323                 }
324             }
325         }
```

References com.example.choturemote.MyBluetoothService.ConnectedThread.mmInStream, and com.example.↩choturemote.MyBluetoothService.TAG.

7.1.3.3 write()

```
void com.example.choturemote.MyBluetoothService.ConnectedThread.write (
            byte[] bytes )
```

Sends data to the remote device.

Parameters

bytes	Array of bytes to be sent to the remote Bluetooth device

Called this from the main activity to send data to the remote device

Definition at line 333 of file MyBluetoothService.java.

```
333                                         {
334             String text = new String(bytes, Charset.defaultCharset());
335             Log.d(TAG, "write: Writing to output stream: " + text);
336             try {
337                 mmOutStream.write(bytes);
338             } catch (IOException e) {
339                 Log.e(TAG, "write: Error writing to output stream. " + e.getMessage());
340             }
341         }
```

References com.example.choturemote.MyBluetoothService.ConnectedThread.mmOutStream, and com.↩example.choturemote.MyBluetoothService.TAG.

Referenced by com.example.choturemote.MyBluetoothService.write().

7.1.4 Member Data Documentation

7.1.4.1 mmInStream

```
com.example.choturemote.MyBluetoothService.ConnectedThread.mmInStream  [private]
```

Holds a reference to an InputStream object, one of two types of streams. One that you can read data from

Definition at line 261 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.ConnectedThread.ConnectedThread(), and com.↩example.choturemote.MyBluetoothService.ConnectedThread.run().

7.1.4.2 mmOutStream

```
com.example.choturemote.MyBluetoothService.ConnectedThread.mmOutStream  [private]
```

Holds a reference to an OutputStream object, one of two types of streams. One that you can either write data to

Definition at line 262 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.ConnectedThread.ConnectedThread(), and com.↩example.choturemote.MyBluetoothService.ConnectedThread.write().

7.1.4.3 mmSocket

```
final BluetoothSocket com.example.choturemote.MyBluetoothService.ConnectedThread.mmSocket
[private]
```

Holds the RFCOMM Socket object

Definition at line 253 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.ConnectedThread.cancel(), and com.example.↩choturemote.MyBluetoothService.ConnectedThread.ConnectedThread().

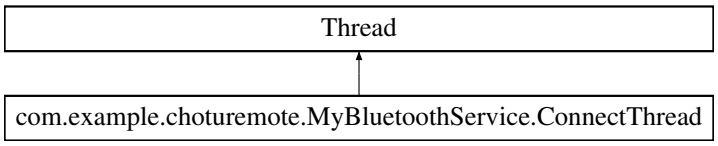
The documentation for this class was generated from the following file:

- MyBluetoothService.java

7.2 com.example.choturemote.MyBluetoothService.ConnectThread Class Reference

Client thread that initiates a Bluetooth connection.

Inheritance diagram for com.example.choturemote.MyBluetoothService.ConnectThread:



Public Member Functions

- ConnectThread (BluetoothDevice device, UUID uuid)

  Public constructor for the ConnectThread class.
- void run ()

  Main code that runs in the Thread.
- void cancel ()

  Closes the socket.

Private Attributes

- final BluetoothSocket mmSocket

  An instance of a Bluetooth socket.

7.2.1 Detailed Description

Client thread that initiates a Bluetooth connection.

- A Bluetooth client sends the connection request and the Bluetooth Server component accepts the request.
- A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.
- This one creates a BluetoothSocket

Definition at line 141 of file MyBluetoothService.java.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ConnectThread()

```
com.example.choturemote.MyBluetoothService.ConnectThread.ConnectThread (
            BluetoothDevice device,
            UUID uuid )
```

Public constructor for the ConnectThread class.

Parameters

device	BluetoothDevice object
uuid	UUID object

Creates a RFCOMM Bluetooth Socket

Definition at line 157 of file MyBluetoothService.java.

```
157                                                         {
158
159             // Use a temporary object that is later assigned to mmSocket because mmSocket is final //
160             BluetoothSocket tmp = null;
161             mmDevice = device;
162             try {
163                 // Get a BluetoothSocket to connect with the given BluetoothDevice. //
164                 tmp = device.createRfcommSocketToServiceRecord(uuid);
165             } catch (IOException e) {
166                 Log.e(TAG, "Socket's create() method failed", e);
167             }
168             mmSocket = tmp;
169         }
```

References com.example.choturemote.MyBluetoothService.mmDevice, com.example.choturemote.MyBluetooth↩Service.ConnectThread.mmSocket, and com.example.choturemote.MyBluetoothService.TAG.

7.2.3 Member Function Documentation

7.2.3.1 cancel()

```
void com.example.choturemote.MyBluetoothService.ConnectThread.cancel ( )
```

Closes the socket.

Closes the client socket and causes the thread to finish. Called from the main activity to shut down the connection.

Definition at line 217 of file MyBluetoothService.java.

```
217                                 {
218             try {
219                 mmSocket.close();
220             } catch (IOException e) {
221                 Log.e(TAG, "Could not close the client socket", e);
222             }
223         }
```

References com.example.choturemote.MyBluetoothService.ConnectThread.mmSocket, and com.example.↩choturemote.MyBluetoothService.TAG.

Referenced by com.example.choturemote.MyBluetoothService.cancelThreads().

7.2.3.2 run()

```
void com.example.choturemote.MyBluetoothService.ConnectThread.run ( )
```

Main code that runs in the Thread.

Try Catch to attempt a connection to the remote device.

Definition at line 175 of file MyBluetoothService.java.

```
175                             {
176
177             // Cancel discovery because it otherwise slows down the connection //
178             bluetoothAdapter.cancelDiscovery();
179
180             try {
181                 // Connect to the remote device through the socket. This call blocks until it succeeds
      or throws an exception //
182                 mmSocket.connect();
183             } catch (IOException connectException) {
184                 // Unable to connect; close the socket and return //
185
186                 Log.d(TAG, "R.string.CONNECTION_TO_DEVICE_UNSUCCESSFUL");
187                 showToast(R.string.CONNECTION_TO_DEVICE_UNSUCCESSFUL);
188
189                 try {
190                     mmSocket.close();
191                 } catch (IOException closeException) {
192                     Log.e(TAG, "Could not close the client socket", closeException);
193                 }
194
195                 // dismiss the progressDialog //
196                 try {
197                     mProgressDialog.dismiss();
198                 } catch (NullPointerException e) {
199                     e.printStackTrace();
200                 }
201
202                 return;
203             }
204
205             // The connection attempt succeeded. Perform work associated with the connection in a
      separate thread. //
206             Log.d(TAG, "R.string.CONNECTION_TO_DEVICE_SUCCESSFUL");
207             showToast(R.string.CONNECTION_TO_DEVICE_SUCCESSFUL);
208
209             // Start a Thread that'll manage the work associated with the connection //
210             connected(mmSocket, mmDevice);
211         }
```

References com.example.choturemote.MyBluetoothService.bluetoothAdapter, com.example.choturemote.↩MyBluetoothService.connected(), com.example.choturemote.MyBluetoothService.mmDevice, com.example.↩choturemote.MyBluetoothService.ConnectThread.mmSocket, com.example.choturemote.MyBluetoothService.m↩ProgressDialog, com.example.choturemote.MyBluetoothService.showToast(), and com.example.choturemote.↩MyBluetoothService.TAG.

7.2.4 Member Data Documentation

7.2.4.1 mmSocket

```
final BluetoothSocket com.example.choturemote.MyBluetoothService.ConnectThread.mmSocket  [private]
```

An instance of a Bluetooth socket.

- A socket is one endpoint of a two-way communication link
- The most common type of Bluetooth socket is RFCOMM, which is the type supported by the Android APIs

Definition at line 149 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.ConnectThread.cancel(), com.example.↩choturemote.MyBluetoothService.ConnectThread.ConnectThread(), and com.example.choturemote.MyBluetooth↩Service.ConnectThread.run().

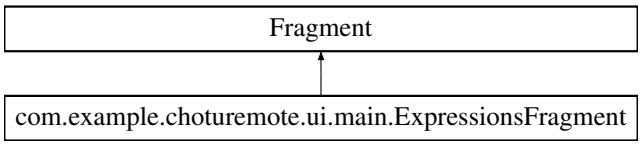
The documentation for this class was generated from the following file:

- MyBluetoothService.java

7.3 com.example.choturemote.ui.main.ExpressionsFragment Class Reference

Expressions Fragment.

Inheritance diagram for com.example.choturemote.ui.main.ExpressionsFragment:



Public Member Functions

- ExpressionsFragment ()

  Constructor.
- View onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)

  Called to have the fragment instantiate its user interface view.

7.3.1 Detailed Description

Expressions Fragment.

- A Fragment is a piece of an application's user interface or behavior that can be placed in an Activity
- This one deals with the Expressions aspect

Definition at line 32 of file ExpressionsFragment.java.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 ExpressionsFragment()

```
com.example.choturemote.ui.main.ExpressionsFragment.ExpressionsFragment ( )
```

Constructor.

Required empty public constructor

Definition at line 38 of file ExpressionsFragment.java.

```
38                                      {
39     }
```

7.3.3 Member Function Documentation

7.3.3.1 onCreateView()

```
View com.example.choturemote.ui.main.ExpressionsFragment.onCreateView (
             LayoutInflater inflater,
             ViewGroup container,
             Bundle savedInstanceState )
```

Called to have the fragment instantiate its user interface view.

- This fragment has a View inflated from the grid_view.xml file
- This View has a GridView with the resourceID "grid"
- This GridView is associated with an Adapter which is an ExpressionsWordAdapter object
- The ExpressionsWordAdapter object returns custom itemView object to populate the GridView

Parameters

inflater	The LayoutInflater object that can be used to inflate any views in the fragment
container	This is the parent view that the fragment's UI should be attached to
savedInstanceState	If non-null, this fragment is being re-constructed from a previous saved state as given here

Returns

The View for the fragment's UI, or null.

Definition at line 54 of file ExpressionsFragment.java.

```
55                                                  {
56         // Inflating grid_view.xml //
57         View rootView = inflater.inflate(R.layout.grid_view, container, false);
58 
59         // A list of Word objects //
60         final ArrayList<Word> words = new ArrayList<Word>();
61         words.add(new Word(R.string.happy, R.drawable.smile));
62         words.add(new Word(R.string.sad, R.drawable.sad));
63         words.add(new Word(R.string.angry, R.drawable.angre));
64         words.add(new Word(R.string.idle, R.drawable.idle));
65         words.add(new Word(R.string.surprised, R.drawable.surprise));
66 
67         // Command String variables for expressions //
68         final String EMOTION_CLASSIFIER = getString(R.string.EMOTION_CLASSIFIER);
69         final String TAIL_STRING = "000";
70         final String VERBALIZE_EMOTION = "V";
71 
72         // An ExpressionsWordAdapter whose data source is a list of Words //
73         // The adapter knows how to create custom itemViews for each item in the list //
74         ExpressionsWordAdapter adapter = new ExpressionsWordAdapter(getActivity(), words,
       R.color.category_expressions);
75 
76         // Find the GridView view hierarchy of the Activity with the ID "grid" //
77         GridView gridView = rootView.findViewById(R.id.grid);
78 
79         // Number of columns in the gridView //
80         gridView.setNumColumns(1);
81 
82         // Make gridView use the ExpressionsWordAdapter object we created above, so that the gridView
       will display items for each Word in the list.
83         gridView.setAdapter(adapter);
84 
85         // Set a click listener to send command String when the list item is clicked //
86         gridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
87             @Override
88             public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {
89                 Word word = words.get(position);
90                 MainActivity.writeToBluetooth(EMOTION_CLASSIFIER, getString(word.getStringResourceId()),
       TAIL_STRING);
91 
92             }
93         });
94 
95         // Set a click listener to play audio when the list item is long pressed //
96         gridView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
97             @Override
98             public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
99                 Word word = words.get(position);
100                 MainActivity.writeToBluetooth(EMOTION_CLASSIFIER, getString(word.getStringResourceId())
       + VERBALIZE_EMOTION, TAIL_STRING);
101                 return true;
102             }
103         });
104 
105         return rootView;
106     }
```

References com.example.choturemote.ui.main.Word.getStringResourceId(), and com.example.choturemote.↩ MainActivity.writeToBluetooth().

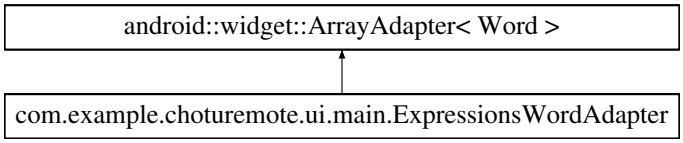
The documentation for this class was generated from the following file:

- ExpressionsFragment.java

7.4 com.example.choturemote.ui.main.ExpressionsWordAdapter Class Reference

An ArrayAdapter Implementation.

Inheritance diagram for com.example.choturemote.ui.main.ExpressionsWordAdapter:



Public Member Functions

- ExpressionsWordAdapter (Context context, ArrayList< Word > words, int colorResourceId)
  Constructor.
- View getView (int position, View convertView, ViewGroup parent)
  Return a View object.

Private Attributes

- int mColorResourceId

7.4.1 Detailed Description

An ArrayAdapter Implementation.

- This is an ArrayAdapter used to provide views for the AdapterView
- Returns a view for each object in a collection of Word objects provided

Definition at line 43 of file ExpressionsWordAdapter.java.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 ExpressionsWordAdapter()

```
com.example.choturemote.ui.main.ExpressionsWordAdapter.ExpressionsWordAdapter (
            Context context,
            ArrayList< Word > words,
            int colorResourceId )
```

Constructor.

Constructor for an ExpressionsWordAdapter object.

Parameters

context	the current context (i.e. Activity) that the adapter is being created in
words	the list of Word objects to be displayed
colorResourceId	the resource ID for the background color for this list of words

Definition at line 58 of file ExpressionsWordAdapter.java.

```
58                                                                                                  {
59         super(context, 0, words);
60         mColorResourceId = colorResourceId;
61     }
```

References com.example.choturemote.ui.main.ExpressionsWordAdapter.mColorResourceId.

7.4.3 Member Function Documentation

7.4.3.1 getView()

```
View com.example.choturemote.ui.main.ExpressionsWordAdapter.getView (
              int position,
              View convertView,
              ViewGroup parent )
```

Return a View object.

Parameters

position	specified position in the data set of the Word to be displayed
convertView	a View type. Here value is null. Instead expressions_grid_item.xml is inflated to create the View
parent	Parent View object of the View object being returned

Returns

a View that displays the data at the specified position in the data set

- Check if an existing view is being reused, otherwise inflate the view
- getView() was `@Override` in order to return a custom type View
- Called back automatically by Android

Definition at line 75 of file ExpressionsWordAdapter.java.

```
75                                                                      {
76
77         View itemView = convertView;
78
79         if (itemView == null) {
80             // Inflating expressions_grid_item.xml //
81             itemView = LayoutInflater.from(getContext()).inflate(
82                     R.layout.expressions_grid_item, parent, false);
83         }
```

```
84
85          // Get the Word object located at this position in the list //
86          Word currentWord = getItem(position);
87
88          // Find the TextView in the expressions_grid_item.xml layout with the ID "default_text_view" //
89          TextView defaultTextView = (TextView) itemView.findViewById(R.id.default_text_view);
90
91          // Get the text from the currentWord object and set this text on the default TextView //
92          defaultTextView.setText(currentWord.getStringResourceId());
93
94          // Find the ImageView in the expressions_grid_item.xml layout with the ID "image" //
95          ImageView imageView = (ImageView) itemView.findViewById(R.id.image);
96
97          if (currentWord.hasImage()) {
98              // If an image is available, display the provided image based on the resource ID //
99              imageView.setImageResource(currentWord.getImageResourceId());
100
101              // Make sure the view is visible //
102              imageView.setVisibility(View.VISIBLE);
103          } else {
104              // Otherwise hide the ImageView (set visibility to GONE) //
105              imageView.setVisibility(View.GONE);
106          }
107
108          // Set the background color of the text container View //
109          View textContainer = itemView.findViewById(R.id.text_container);
110          int color = ContextCompat.getColor(getContext(), mColorResourceId); // Find the color that the
     resource ID maps to //
111          textContainer.setBackgroundColor(color);
112
113          // Return the itemView object(containing one TextView and one ImageView) so that it can be shown
     in the GridView //
114          return itemView;
115      }
```

References com.example.choturemote.ui.main.Word.getImageResourceId(), com.example.choturemote.ui.↩ main.Word.getStringResourceId(), com.example.choturemote.ui.main.Word.hasImage(), and com.example.↩ choturemote.ui.main.ExpressionsWordAdapter.mColorResourceId.

7.4.4 Member Data Documentation

7.4.4.1 mColorResourceId

```
int com.example.choturemote.ui.main.ExpressionsWordAdapter.mColorResourceId  [private]
```

Resource ID for the background color for this list of Word objects

Definition at line 48 of file ExpressionsWordAdapter.java.

Referenced by com.example.choturemote.ui.main.ExpressionsWordAdapter.ExpressionsWordAdapter(), and com.example.choturemote.ui.main.ExpressionsWordAdapter.getView().

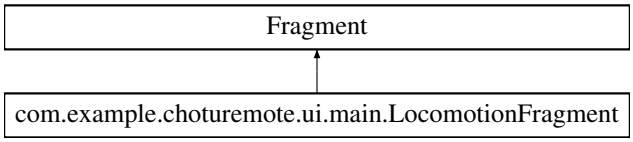
The documentation for this class was generated from the following file:

- ExpressionsWordAdapter.java

7.5 com.example.choturemote.ui.main.LocomotionFragment Class Reference

Locomotion Fragment.

Inheritance diagram for com.example.choturemote.ui.main.LocomotionFragment:



Public Member Functions

- LocomotionFragment ()

  Constructor.
- View onCreateView (LayutInflater inflater, ViewGroup container, Bundle savedInstanceState)

  Called to have the fragment instantiate its user interface view.

7.5.1 Detailed Description

Locomotion Fragment.

- A Fragment is a piece of an application's user interface or behavior that can be placed in an Activity.
- This one deals with the Locomotion aspect

Definition at line 31 of file LocomotionFragment.java.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 LocomotionFragment()

```
com.example.choturemote.ui.main.LocomotionFragment.LocomotionFragment ( )
```

Constructor.

Required empty public constructor

Definition at line 37 of file LocomotionFragment.java.

```
37                                  {
38     }
```

7.5.3 Member Function Documentation

7.5.3.1 onCreateView()

```
View com.example.choturemote.ui.main.LocomotionFragment.onCreateView (
            LayoutInflater inflater,
            ViewGroup container,
            Bundle savedInstanceState )
```

Called to have the fragment instantiate its user interface view.

Parameters

inflater	The LayoutInflater object that can be used to inflate any views in the fragment
container	This is the parent view that the fragment's UI should be attached to
savedInstanceState	If non-null, this fragment is being re-constructed from a previous saved state as given here

Returns

The View for the fragment's UI, or null.

- This fragment has a View inflated from the grid_view.xml file
- This View has a GridView with the resourceID "grid"
- This GridView is associated with an Adapter which is an LocomotionWordAdapter object
- The LocomotionWordAdapter object returns custom itemView object to populate the GridView

Definition at line 53 of file LocomotionFragment.java.

```
54                                                      {
55
56         // Inflating grid_view.xml //
57         View rootView = inflater.inflate(R.layout.grid_view, container, false);
58
59         // A list of Word objects //
60         final ArrayList<Word> words = new ArrayList<Word>();
61         words.add(new Word(R.string.empty, R.drawable.empty));
62         words.add(new Word(R.string.empty, R.drawable.empty));
63         words.add(new Word(R.string.empty, R.drawable.empty));
64         words.add(new Word(R.string.empty, R.drawable.empty));
65         words.add(new Word(R.string.empty, R.drawable.forward));
66         words.add(new Word(R.string.empty, R.drawable.empty));
67         words.add(new Word(R.string.empty, R.drawable.leftward));
68         words.add(new Word(R.string.empty, R.drawable.color_red));
69         words.add(new Word(R.string.empty, R.drawable.rightward));
70         words.add(new Word(R.string.empty, R.drawable.empty));
71         words.add(new Word(R.string.empty, R.drawable.backward));
72         words.add(new Word(R.string.empty, R.drawable.empty));
73         words.add(new Word(R.string.empty, R.drawable.empty));
74         words.add(new Word(R.string.empty, R.drawable.empty));
75         words.add(new Word(R.string.empty, R.drawable.empty));
76
77         // Command String variables for expressions //
78         final String FORWARD_MOVEMENT_CLASSIFIER = getString(R.string.FORWARD_MOVEMENT_CLASSIFIER);
79         final String BACKWARD_MOVEMENT_CLASSIFIER = getString(R.string.BACKWARD_MOVEMENT_CLASSIFIER);
80         final String LEFT_MOVEMENT_CLASSIFIER = getString(R.string.LEFT_MOVEMENT_CLASSIFIER);
81         final String RIGHT_MOVEMENT_CLASSIFIER = getString(R.string.RIGHT_MOVEMENT_CLASSIFIER);
82         final String STOP_MOVEMENT_CLASSIFIER = getString(R.string.STOP_MOVEMENT_CLASSIFIER);
83         final String MODIFIER = "_000_000";
84         final String distance = "1000";
85
86         // A LocomotionWordAdapter whose data source is a list of Words //
87         // The adapter knows how to create custom itemViews for each item in the list //
88         LocomotionWordAdapter adapter = new LocomotionWordAdapter(getActivity(), words,
       R.color.category_expressions);
89
90         // Find the GridView view hierarchy of the Activity with the ID "grid" //
91         GridView gridView = rootView.findViewById(R.id.grid);
92
93         // Number of columns in the gridView //
94         gridView.setNumColumns(3);
95
96         // Make gridView use the LocomotionWordAdapter object we created above, so that the gridView will
       display items for each Word in the list.
97         gridView.setAdapter(adapter);
98
99         // Set a click listener to send command String when the list item is clicked //
100         gridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
101             @Override
102             public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {
103
104                 switch (position) {
105                     case 4:
106                         MainActivity.writeToBluetooth(FORWARD_MOVEMENT_CLASSIFIER, distance, MODIFIER);
107                         break;
```

```
108                     case 6:
109                         MainActivity.writeToBluetooth(LEFT_MOVEMENT_CLASSIFIER, distance, MODIFIER);
110                         break;
111                     case 7:
112                         MainActivity.writeToBluetooth(STOP_MOVEMENT_CLASSIFIER, "", "");
113                         break;
114                     case 8:
115                         MainActivity.writeToBluetooth(RIGHT_MOVEMENT_CLASSIFIER, distance, MODIFIER);
116                         break;
117                     case 10:
118                         MainActivity.writeToBluetooth(BACKWARD_MOVEMENT_CLASSIFIER, distance, MODIFIER);
119                         break;
120                      default:
121                         break;
122                 }
123             }
124         });
125 
126         return rootView;
127     }
```

References com.example.choturemote.MainActivity.writeToBluetooth().

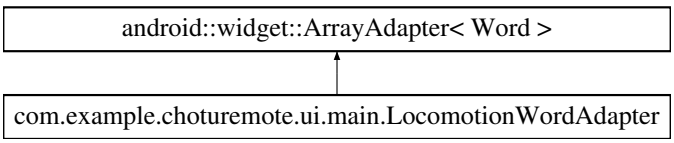
The documentation for this class was generated from the following file:

- LocomotionFragment.java

7.6 com.example.choturemote.ui.main.LocomotionWordAdapter Class Reference

An ArrayAdapter Implementation.

Inheritance diagram for com.example.choturemote.ui.main.LocomotionWordAdapter:



Public Member Functions

- LocomotionWordAdapter (Context context, ArrayList< Word > words, int colorResourceId)
  Constructor.
- View getView (int position, View convertView, ViewGroup parent)
  Return a View object.

Private Attributes

- int mColorResourceId

7.6.1 Detailed Description

An ArrayAdapter Implementation.

- This is an ArrayAdapter used to provide views for the AdapterView
- Returns a view for each object in a collection of Word objects provided

Definition at line 43 of file LocomotionWordAdapter.java.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 LocomotionWordAdapter()

```
com.example.choturemote.ui.main.LocomotionWordAdapter.LocomotionWordAdapter (
            Context context,
            ArrayList< Word > words,
            int colorResourceId )
```

Constructor.

Constructor for a LocomotionWordAdapter object.

Parameters

context	the current context (i.e. Activity) that the adapter is being created in
words	the list of Word objects to be displayed
colorResourceId	the resource ID for the background color for this list of words

Definition at line 58 of file LocomotionWordAdapter.java.

```
58                                                                                  {
59         super(context, 0, words);
60         mColorResourceId = colorResourceId;
61     }
```

References com.example.choturemote.ui.main.LocomotionWordAdapter.mColorResourceId.

7.6.3 Member Function Documentation

7.6.3.1 getView()

```
View com.example.choturemote.ui.main.LocomotionWordAdapter.getView (
            int position,
            View convertView,
            ViewGroup parent )
```

Return a View object.

Parameters

position	specified position in the data set of the Word to be displayed
convertView	a View type. Here value is null. Instead locomotion_grid_item.xml is inflated to create the View
parent	Parent View object of the View object being returned

Returns

a View that displays the data at the specified position in the data set

- Check if an existing view is being reused, otherwise inflate the view
- getView() was `@Override` in order to return a custom type View
- Called back automatically by Android

Check if an existing view is being reused, otherwise inflate the view

Definition at line 76 of file LocomotionWordAdapter.java.

```
76                                                                          {
79          View itemView = convertView;
80         if (itemView == null) {
81
82             // Inflating locomotion_grid_item.xml //
83             itemView = LayoutInflater.from(getContext()).inflate(
84                     R.layout.locomotion_grid_item, parent, false);
85         }
86
87         // Get the Word object located at this position in the list //
88         Word currentWord = getItem(position);
89
90         // Find the ImageView in the expressions_grid_item.xml layout with the ID "image" //
91         ImageView imageView = (ImageView) itemView.findViewById(R.id.image);
92
93         if (currentWord.hasImage()) {
94
95             // If an image is available, display the provided image based on the resource ID //
96             imageView.setImageResource(currentWord.getImageResourceId());
97
98             // Make sure the view is visible //
99             imageView.setVisibility(View.VISIBLE);
100         } else {
101
102             // Otherwise hide the ImageView (set visibility to GONE) //
103             imageView.setVisibility(View.GONE);
104         }
105
106         // Return the itemView object(one ImageView) so that it can be shown in the GridView //
107         return itemView;
108     }
```

References com.example.choturemote.ui.main.Word.getImageResourceId(), and com.example.choturemote.ui.↩main.Word.hasImage().

7.6.4 Member Data Documentation

7.6.4.1 mColorResourceId

```
int com.example.choturemote.ui.main.LocomotionWordAdapter.mColorResourceId  [private]
```

Resource ID for the background color for this list of Word objects

Definition at line 48 of file LocomotionWordAdapter.java.

Referenced by com.example.choturemote.ui.main.LocomotionWordAdapter.LocomotionWordAdapter().

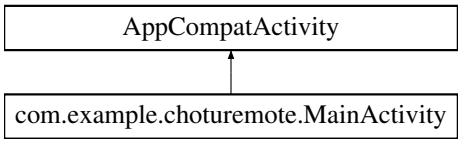
The documentation for this class was generated from the following file:

- LocomotionWordAdapter.java

7.7 com.example.choturemote.MainActivity Class Reference

This is where everything important happens.

Inheritance diagram for com.example.choturemote.MainActivity:



Public Member Functions

- void showToastMethod (final String message)
  Toasts for threads other than the main.

Static Public Member Functions

- static void writeToBluetooth (String classifier, String instruction, String modifier)
  Forms and sends command Strings through Bluetooth.

Static Public Attributes

- static MyBluetoothService myBluetoothService
  Handles Bluetooth stuff.

Protected Member Functions

- void onCreate (Bundle savedInstanceState)
  fires when the system first creates the activity.
- void onDestroy ()
  The final call you receive before your activity is destroyed.
- void onActivityResult (int requestCode, int resultCode, @Nullable Intent data)
  Called back after focus is returned from process started by startActivityForResult()

Static Package Attributes

- static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")
  Universally Unique Identifier (UUID)
- static String TERMINATOR
  Command instruction terminator.
- static String DISABLE

Private Member Functions

- void beginConnection ()
  Establishes a Bluetooth serial communication.

Private Attributes

- final boolean TESTING = false

  Debugging tool.
- SectionsPagerAdapter sectionsPagerAdapter

  Custom class.
- TabLayout tabs

  TabLayout provides a horizontal layout to display tabs.
- ViewPager viewPager

  Layout manager allowing left/right swipes to get access to more options.
- String MAC_ADDRESS
- BluetoothAdapter bluetoothAdapter

  A BluetoothAdapter object.
- int REQUEST_ENABLE_BT = 1

  An integer passed to startActivityForResult() and received by onActivityResult(). If it is greater >= 0, this code will be returned in onActivityResult() when the activity exits. Does nothing of significance at present.
- final BroadcastReceiver mBTStateBroadcastReceiver

  a BroadcastReceiver type object

Static Private Attributes

- static String POKE_SLEEP_CLASSIFIER

  Used to send sleep or poke instructions.
- static String SEPARATOR

  Command instruction separator.
- static String POKE

  Used as a modifier.
- static String ENABLE
- static String TAG = "MainActivity"

7.7.1 Detailed Description

This is where everything important happens.

This is the first screen to appear when the user launches the app. This handles everything and calls everything.

Definition at line 35 of file MainActivity.java.

7.7.2 Member Function Documentation

7.7.2.1 beginConnection()

```
void com.example.choturemote.MainActivity.beginConnection ( )  [private]
```

Establishes a Bluetooth serial communication.

- Fetches device info from paired devices.
-

Definition at line 281 of file MainActivity.java.

```
281                                 {
282
283         boolean deviceFound = false;
284
285         Set<BluetoothDevice> pairedDevices = bluetoothAdapter.getBondedDevices();
286         if (pairedDevices.size() > 0) {
287             // There are paired devices. Get the name and address of each paired device. //
288             for (BluetoothDevice device : pairedDevices) {
289                 String deviceHardwareAddress = device.getAddress(); // MAC address
290                 if (deviceHardwareAddress.equals(MAC_ADDRESS)) {
291                     // This bit matches the MAC address of your "face device" to a paired device's //
292                     // Then establishes a connection on a separate thread //
293                     deviceFound = true;
294                     myBluetoothService.startClient(device, MY_UUID);
295                     break;
296                 }
297             }
298             if (!deviceFound) {
299                 Toast.makeText(getBaseContext(), "ERROR: Device with MAC address " + MAC_ADDRESS + " not
    paired", Toast.LENGTH_LONG).show();
300                 this.finishAffinity();
301             }
302
303         } else {
304             Toast.makeText(getBaseContext(), "ERROR:" + getString(R.string.NO_PAIRED_DEVICES),
    Toast.LENGTH_LONG).show();
305             this.finishAffinity();
306         }
307     }
```

References com.example.choturemote.MainActivity.bluetoothAdapter, com.example.choturemote.MainActivity.M↩AC_ADDRESS, com.example.choturemote.MainActivity.MY_UUID, com.example.choturemote.MainActivity.my↩BluetoothService, and com.example.choturemote.MyBluetoothService.startClient().

Referenced by com.example.choturemote.MainActivity.onActivityResult(), and com.example.choturemote.Main↩Activity.onCreate().

7.7.2.2 onActivityResult()

```
void com.example.choturemote.MainActivity.onActivityResult (
             int requestCode,
             int resultCode,
             @Nullable Intent data )  [protected]
```

Called back after focus is returned from process started by startActivityForResult()

Parameters

requestCode	the requestCode passed as the second parameter to startActivityForResult(), here it is REQUEST_ENABLE_BT
resultCode	Possible values - RESULT_OK or RESULT_CANCELED
data	Optional parameter. An Intent, which can return result data to the caller. `@Nullable` denotes that a value can be null.

If enabling Bluetooth succeeds, this activity receives the RESULT_OK result code in the onActivityResult() callback. If Bluetooth was not enabled due to an error (or the user responded "No") then the result code is RESULT_CANCELED.

Definition at line 336 of file MainActivity.java.

```
336                                                                                 {
337         if (requestCode == REQUEST_ENABLE_BT && resultCode == RESULT_OK) {
338             beginConnection();
339         }
340         if (resultCode == RESULT_CANCELED)
341             Toast.makeText(this, "Unable to access Bluetooth", Toast.LENGTH_SHORT).show();
342     }
```

References com.example.choturemote.MainActivity.beginConnection(), and com.example.choturemote.MainActivity.REQUEST_ENABLE_BT.

7.7.2.3 onCreate()

```
void com.example.choturemote.MainActivity.onCreate (
             Bundle savedInstanceState )  [protected]
```

fires when the system first creates the activity.

Parameters

savedInstanceState	A Bundle object containing the activity's previously saved state. If the activity has never existed before, the value of the Bundle object is null. (Bundle is generally used for passing data between various activities of android.)

In the onCreate() method, you perform basic application startup logic that should happen only once for the entire life of the activity.

Definition at line 166 of file MainActivity.java.

```
166                                                 {
167
168         super.onCreate(savedInstanceState);
169         setContentView(R.layout.activity_main);
170
171         // String code //
172         POKE_SLEEP_CLASSIFIER = getString(R.string.POKE_SLEEP_CLASSIFIER);
173         SEPARATOR = getString(R.string.SEPARATOR);
174         TERMINATOR = getString(R.string.TERMINATOR);
175         POKE = getString(R.string.POKE_INSTRUCTION);
176         DISABLE = getString(R.string.DISABLE);
177         ENABLE = getString(R.string.ENABLE);
178
179         // ViewPager and connected TabLayout setup //
180         sectionsPagerAdapter = new SectionsPagerAdapter(this, getSupportFragmentManager());
181         viewPager = findViewById(R.id.view_pager);
182         viewPager.setAdapter(sectionsPagerAdapter);
183         tabs = findViewById(R.id.tabs);
184         tabs.setupWithViewPager(viewPager);
185
186         // FAB //
187         final FloatingActionButton fab = findViewById(R.id.fab);
188         fab.setOnClickListener(new View.OnClickListener() {
189             boolean toggle = true;
190
191             @Override
192             public void onClick(View view) {
193                 // Toggles between POKE ENABLED and POKE DISABLED //
194                 toggle = !toggle;
195                 if (toggle) {
196                     // Changes the FAB icon //
197                     fab.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
       android.R.drawable.ic_menu_view));
```

```
198                     writeToBluetooth(POKE_SLEEP_CLASSIFIER, POKE, ENABLE);
199                     Snackbar.make(view, POKE + " " + ENABLE + "ED", Snackbar.LENGTH_LONG)
200                             .setAction("Action", null).show();
201                 } else {
202                     fab.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
    android.R.drawable.ic_lock_idle_lock));
203                     writeToBluetooth(POKE_SLEEP_CLASSIFIER, POKE, DISABLE);
204                     Snackbar.make(view, POKE + " " + DISABLE + "ED", Snackbar.LENGTH_LONG)
205                             .setAction("Action", null).show();
206                 }
207             }
208         });
209         fab.setOnLongClickListener(new View.OnLongClickListener() {
210             @Override
211             public boolean onLongClick(View view) {
212                 // Triggers sleep animation and turns off screen of the "face device" //
213                 writeToBluetooth(POKE_SLEEP_CLASSIFIER, "SLEEP", "000");
214                 Snackbar.make(view, "Chotu is now asleep", Snackbar.LENGTH_LONG)
215                         .setAction("Action", null).show();
216                 return true;
217             }
218         });
219
220         //Bluetooth code //
221         if (TESTING)
222             MAC_ADDRESS = getString(R.string.TESTING_MAC_ADDRESS_2); // MAC_ADDRESS IN USE - EASIER TO
    CHANGE HERE THAN ALL OVER THE PLACE //
223         else
224             MAC_ADDRESS = getString(R.string.MAC_ADDRESS); // MAC_ADDRESS IN USE - EASIER TO CHANGE HERE
    THAN ALL OVER THE PLACE //
225
226         myBluetoothService = new MyBluetoothService(MainActivity.this) {
227             @Override
228             public void showToast(int resourceID) {
229                 // Will receive a resourceID, convert it into a String, and send it to showToastMethod()
    //
230                 showToastMethod(getString(resourceID));
231             }
232
233         };
234
235         // Get a handle/reference to the default local Bluetooth adapter of the device being used //
236         bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
237         if (bluetoothAdapter == null)
238             Toast.makeText(this, "Error: This device device does not support Bluetooth",
    Toast.LENGTH_LONG).show();
239         else {
240             if (!bluetoothAdapter.isEnabled()) {
241
242                 // Creating an Intent //
243                 Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
244
245                 // A dialog appears requesting user permission to enable Bluetooth. //
246                 // If the user responds "Yes", the system begins to enable Bluetooth //
247                 // Focus returns to your application once the process completes (or fails) //
248                 // onActivityResult() that gets called upon return of focus //
249                 startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
250
251
252             } else {
253                 beginConnection();
254             }
255
256             // Informs us when BT condition changes //
257             // Registers a BroadcastReceiver to be run in the main activity thread. //
258             // The receiver will be called with any broadcast Intent that matches filter //
259             // The Broadcast Receiver implementation is outside onCreate() //
260             registerReceiver(mBTStateBroadcastReceiver, new
    IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED));
261         }
262     }
```

References com.example.choturemote.MainActivity.beginConnection(), com.example.choturemote.MainActivity.↩ bluetoothAdapter, com.example.choturemote.MainActivity.DISABLE, com.example.choturemote.MainActivity.↩ ENABLE, com.example.choturemote.MainActivity.MAC_ADDRESS, com.example.choturemote.MainActivity.↩ mBTStateBroadcastReceiver, com.example.choturemote.MainActivity.myBluetoothService, com.example.↩ choturemote.MainActivity.POKE, com.example.choturemote.MainActivity.POKE_SLEEP_CLASSIFIER, com.↩ example.choturemote.MainActivity.REQUEST_ENABLE_BT, com.example.choturemote.MainActivity.sections↩ PagerAdapter, com.example.choturemote.MainActivity.SEPARATOR, com.example.choturemote.MainActivity.↩ showToastMethod(), com.example.choturemote.MainActivity.tabs, com.example.choturemote.MainActivity.TER↩ MINATOR, com.example.choturemote.MainActivity.TESTING, com.example.choturemote.MainActivity.viewPager, and com.example.choturemote.MainActivity.writeToBluetooth().

7.7.2.4 onDestroy()

```
void com.example.choturemote.MainActivity.onDestroy ( )  [protected]
```

The final call you receive before your activity is destroyed.

This opportunity is used to unregister the BroadcastReceiver, mBTStateBroadcastReceiver

Definition at line 269 of file MainActivity.java.

```
269                                 {
270         super.onDestroy();
271         // Unregistering broadcast listener to free up resources
272         unregisterReceiver(mBTStateBroadcastReceiver);
273     }
```

References com.example.choturemote.MainActivity.mBTStateBroadcastReceiver.

7.7.2.5 showToastMethod()

```
void com.example.choturemote.MainActivity.showToastMethod (
              final String message )
```

Toasts for threads other than the main.

Parameters

message	Message to be shown in Toast

Toasts can only be displayed on the main thread. To call Toasts from other threads, a public method in the Activity running on the main thread can be used.

Definition at line 349 of file MainActivity.java.

```
349                                                 {
350         runOnUiThread(new Runnable() {
351             public void run() {
352                 Toast.makeText(getBaseContext(), message, Toast.LENGTH_SHORT).show();
353             }
354         });
355     }
```

Referenced by com.example.choturemote.MainActivity.onCreate().

7.7.2.6 writeToBluetooth()

```
static void com.example.choturemote.MainActivity.writeToBluetooth (
              String classifier,
              String instruction,
              String modifier )  [static]
```

Forms and sends command Strings through Bluetooth.

Parameters

classifier	Possible values F, B, R, L, E, etc.
instruction	Parameterizes or explains the classifier. This can be a number or can be something else entirely.
modifier	Modifies the parameter definition.

Constructs command Strings using SEPARATOR and TERMINATOR then sends command String through Bluetooth. This makes the command construction process easily programmable and editable as it is all in one place.

Definition at line 316 of file MainActivity.java.

```
316                                                                                                    {
317
318         if (MyBluetoothService.connectedThreadRunning()) {
319
320             String fullMessage = classifier + SEPARATOR + instruction + SEPARATOR + modifier +
    TERMINATOR;
321
322             // Converts the String to an Array of byte type //
323             byte[] bytes = fullMessage.getBytes(Charset.defaultCharset());
324             myBluetoothService.write(bytes);
325         }
326     }
```

References com.example.choturemote.MyBluetoothService.connectedThreadRunning(), com.example.choturemote.↩MainActivity.myBluetoothService, com.example.choturemote.MainActivity.SEPARATOR, com.example.choturemote.↩MainActivity.TERMINATOR, and com.example.choturemote.MyBluetoothService.write().

Referenced by com.example.choturemote.MainActivity.onCreate(), com.example.choturemote.ui.main.Locomotion↩Fragment.onCreateView(), com.example.choturemote.ui.main.ExpressionsFragment.onCreateView(), and com.↩example.choturemote.ui.main.SpeakingFragment.onCreateView().

7.7.3 Member Data Documentation

7.7.3.1 bluetoothAdapter

```
BluetoothAdapter com.example.choturemote.MainActivity.bluetoothAdapter  [private]
```

A BluetoothAdapter object.

A BluetoothAdapter lets you perform fundamental Bluetooth tasks, such as initiate device discovery, query a list of bonded (paired) devices, instantiate a BluetoothDevice using a known MAC address, and create a Bluetooth↩ServerSocket to listen for connection requests from other devices, and start a scan for Bluetooth LE devices.

Definition at line 121 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.beginConnection(), and com.example.choturemote.Main↩Activity.onCreate().

7.7.3.2 DISABLE

com.example.choturemote.MainActivity.DISABLE [static], [package]

Disables poking

Definition at line 104 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate().

7.7.3.3 ENABLE

com.example.choturemote.MainActivity.ENABLE [static], [private]

Enables poking

Definition at line 104 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate().

7.7.3.4 MAC_ADDRESS

String com.example.choturemote.MainActivity.MAC_ADDRESS [private]

MAC ADDRESS of the Bluetooth device to establish communication with

Definition at line 115 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.beginConnection(), and com.example.choturemote.Main↩
Activity.onCreate().

7.7.3.5 mBTStateBroadcastReceiver

```
final BroadcastReceiver com.example.choturemote.MainActivity.mBTStateBroadcastReceiver  [private]
```

Initial value:

```
= new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {

            final String action = intent.getAction();
            if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)) {
                final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, BluetoothAdapter.ERROR);
                switch (state) {
                    case BluetoothAdapter.STATE_OFF:
                        Toast.makeText(context, R.string.BT_STATE_OFF_TEXT, Toast.LENGTH_SHORT).show();
                        break;
                    case BluetoothAdapter.STATE_TURNING_OFF:
                        Toast.makeText(context, R.string.BT_STATE_TURNING_OFF_TEXT,
       Toast.LENGTH_SHORT).show();
                        break;
                    case BluetoothAdapter.STATE_ON:
                        Toast.makeText(context, R.string.BT_STATE_ON_TEXT, Toast.LENGTH_SHORT).show();
                        break;
                    case BluetoothAdapter.STATE_TURNING_ON:
                        Toast.makeText(context, R.string.BT_STATE_TURNING_ON_TEXT,
       Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        }
    }
```

a BroadcastReceiver type object

- Receives and handles broadcast intents sent by `Context.sendBroadcast(Intent)`.
- This is an implementation of BroadcastReceiver registered to be run in the main activity thread. Its receiver is called with any broadcast Intent that matches filter, in this case is **BluetoothAdapter.ACTION_STATE_↩ CHANGED** (in other words, when the state of the local Bluetooth adapter has been changed).

Definition at line 134 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate(), and com.example.choturemote.MainActivity.↩ onDestroy().

7.7.3.6 MY_UUID

```
final UUID com.example.choturemote.MainActivity.MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9↩
B34FB")  [static], [package]
```

Universally Unique Identifier (UUID)

- Creating a UUID which represents a 128-bit value.
- More information on UUIDs by the Internet Engineering Task Force can be found `here`.
- UUIDs are not tied to particular devices. They identify software services. You just need both sides to use the same one.
- "00001101-0000-1000-8000-00805F9B34FB" is the one and only UUID for SPP (serial port profile). Check out `the Android Developer's page`.

Definition at line 51 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.beginConnection().

7.7.3.7 myBluetoothService

```
MyBluetoothService com.example.choturemote.MainActivity.myBluetoothService [static]
```

Handles Bluetooth stuff.

Custom class. Handles Bluetooth stuff.

Definition at line 57 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.beginConnection(), com.example.choturemote.Main↩ Activity.onCreate(), and com.example.choturemote.MainActivity.writeToBluetooth().

7.7.3.8 POKE

```
com.example.choturemote.MainActivity.POKE [static], [private]
```

Used as a modifier.

Used as a modifier to distinguish a poke command from a sleep command.

Definition at line 103 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate().

7.7.3.9 POKE_SLEEP_CLASSIFIER

```
com.example.choturemote.MainActivity.POKE_SLEEP_CLASSIFIER [static], [private]
```

Used to send sleep or poke instructions.

Command Strings pertaining to sleep or poke instructions classified by this classifier

Definition at line 101 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate().

7.7.3.10 REQUEST_ENABLE_BT

```
int com.example.choturemote.MainActivity.REQUEST_ENABLE_BT = 1 [private]
```

An integer passed to startActivityForResult() and received by onActivityResult(). If it is greater >= 0, this code will be returned in onActivityResult() when the activity exits. Does nothing of significance at present.

Definition at line 126 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onActivityResult(), and com.example.choturemote.Main↩ Activity.onCreate().

7.7.3.11 sectionsPagerAdapter

```
SectionsPagerAdapter com.example.choturemote.MainActivity.sectionsPagerAdapter [private]
```

Custom class.

SectionsPagerAdapter extends FragmentPagerAdapter which is an implementation of the PagerAdapter class. It is needed for the ViewPager object, viewPager.

Definition at line 63 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate().

7.7.3.12 SEPARATOR

```
com.example.choturemote.MainActivity.SEPARATOR [static], [private]
```

Command instruction separator.

Is used to visually distinguish between parts of a String instruction.

Definition at line 102 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate(), and com.example.choturemote.MainActivity.↩ writeToBluetooth().

7.7.3.13 tabs

```
TabLayout com.example.choturemote.MainActivity.tabs [private]
```

TabLayout provides a horizontal layout to display tabs.

- The class method setupWithViewPager() links the given ViewPager and this TabLayout together so that changes in one are automatically reflected in the other. This includes scroll state changes and clicks. The tabs displayed in this layout will be populated from the ViewPager adapter's page titles.
- The TabLayout object, tabs, is a View in the activity_main.xml layout file with the id "tabs".

Definition at line 71 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate().

7.7.3.14 TAG

```
String com.example.choturemote.MainActivity.TAG = "MainActivity"  [static], [private]
```

Debugging tool

Definition at line 109 of file MainActivity.java.

7.7.3.15 TERMINATOR

```
com.example.choturemote.MainActivity.TERMINATOR  [static], [package]
```

Command instruction terminator.

Used to indicate the end of a String instruction.

Definition at line 102 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate(), and com.example.choturemote.MainActivity.↩ writeToBluetooth().

7.7.3.16 TESTING

```
final boolean com.example.choturemote.MainActivity.TESTING = false  [private]
```

Debugging tool.

Toggle to FALSE when using it with the two HC-05's on a breadboard. Check their MAC Addresses before using.

Definition at line 41 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate().

7.7.3.17 viewPager

```
ViewPager com.example.choturemote.MainActivity.viewPager  [private]
```

Layout manager allowing left/right swipes to get access to more options.

- ViewPager is a layout manager that allows the user to flip left and right through pages of data.
- You supply an implementation of a PagerAdapter to generate the pages that the view shows. In this case it was supplied a SectionsPagerAdapter object sectionsPagerAdapter. SectionsPagerAdapter extends FragmentPagerAdapter which is an implementation of the PagerAdapter class.
- More information `at the ViewPager entry` and `PagerAdapter entry` in the Android Developer's documentation.
- The ViewPager object, viewPager, is a View in the activity_main.xml layout file with the id "view_pager".

Definition at line 81 of file MainActivity.java.

Referenced by com.example.choturemote.MainActivity.onCreate().

The documentation for this class was generated from the following file:

- MainActivity.java

7.8 com.example.choturemote.MyBluetoothService Class Reference

Handles everything Bluetooth.

Classes

- class ConnectedThread

  Thread that manages a Bluetooth connection.
- class ConnectThread

  Client thread that initiates a Bluetooth connection.

Public Member Functions

- MyBluetoothService (Context context)

  Constructor.
- synchronized void cancelThreads ()

  Cancels any running threads.
- void startClient (BluetoothDevice device, UUID uuid)
- void write (byte[ ] out)

  Writes to the "face device".
- abstract void showToast (int resourceID)

  Displays Toasts.

Static Public Member Functions

- static boolean connectedThreadRunning ()

  returns boolean status of mConnectedThread

Public Attributes

- ConnectedThread mConnectedThread

  Custom class ConnectedThread type object.

Static Public Attributes

- static boolean mConnectedThreadRunning = false

  boolean status of mConnectedThread

Package Attributes

- ProgressDialog mProgressDialog

  A progress dialog.

Static Package Attributes

- static Context mContext

  Reference to the Activity where the MyBluetoothService object instance is created.

Private Member Functions

- void connected (BluetoothSocket mmSocket, BluetoothDevice mmDevice)

Private Attributes

- final BluetoothAdapter bluetoothAdapter

  A BluetoothAdapter object.
- ConnectThread mConnectThread

  Custom class ConnectThread type object.
- BluetoothDevice mmDevice

  Represents a remote Bluetooth device.

Static Private Attributes

- static final String TAG = "MY_APP_DEBUG_TAG"

7.8.1 Detailed Description

Handles everything Bluetooth.

- Starts up and maintains Bluetooth connections between devices
- Sends and handles reception of messages from connected devices

Definition at line 37 of file MyBluetoothService.java.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 MyBluetoothService()

```
com.example.choturemote.MyBluetoothService.MyBluetoothService (
            Context context )
```

Constructor.

Parameters

context	Context of the Activity that creates an object of the MyBluetoothService class

The constructor used to create an object of the MyBluetoothService class

Definition at line 91 of file MyBluetoothService.java.

```
91                                                  {
92         mContext = context;
93         bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
94         cancelThreads();
95     }
```

References com.example.choturemote.MyBluetoothService.bluetoothAdapter, com.example.choturemote.My↩BluetoothService.cancelThreads(), and com.example.choturemote.MyBluetoothService.mContext.

7.8.3 Member Function Documentation

7.8.3.1 cancelThreads()

```
synchronized void com.example.choturemote.MyBluetoothService.cancelThreads ( )
```

Cancels any running threads.

Cancels any threads attempting to create a BluetoothSocket or any

Definition at line 101 of file MyBluetoothService.java.

```
101                                             {
102
103         Log.d(TAG, "start");
104
105         // Cancel any thread attempting to make a connection //
106         if (mConnectThread != null) {
107             mConnectThread.cancel();
108             mConnectThread = null;
109         }
110         if (mConnectedThread != null) {
111             mConnectedThread.cancel();
112             mConnectedThread = null;
113             mConnectedThreadRunning = false;
114         }
115     }
```

References com.example.choturemote.MyBluetoothService.ConnectThread.cancel(), com.example.choturemote.↩MyBluetoothService.ConnectedThread.cancel(), com.example.choturemote.MyBluetoothService.mConnected↩Thread, com.example.choturemote.MyBluetoothService.mConnectedThreadRunning, com.example.choturemote.↩MyBluetoothService.mConnectThread, and com.example.choturemote.MyBluetoothService.TAG.

Referenced by com.example.choturemote.MyBluetoothService.MyBluetoothService().

7.8.3.2 connected()

```
void com.example.choturemote.MyBluetoothService.connected (
            BluetoothSocket mmSocket,
            BluetoothDevice mmDevice )  [private]
```

Called to start a Thread to manages the Bluetooth connection

Parameters

mmSocket	RFCOMM Bluetooth Socket object
mmDevice	BluetoothDevice object

Definition at line 232 of file MyBluetoothService.java.

```
232                                                                         {
233         Log.d(TAG, "connected: Starting.");
234
235         // Start the thread to manage the connection and perform transmissions //
236         mConnectedThread = new ConnectedThread(mmSocket);
237
238         // Starts Thread //
239         mConnectedThread.start();
240     }
```

References com.example.choturemote.MyBluetoothService.mConnectedThread, and com.example.choturemote.↩ MyBluetoothService.TAG.

Referenced by com.example.choturemote.MyBluetoothService.ConnectThread.run().

7.8.3.3 connectedThreadRunning()

```
static boolean com.example.choturemote.MyBluetoothService.connectedThreadRunning ( )  [static]
```

returns boolean status of mConnectedThread

Returns

boolean status of mConnectedThread

For use in other Activities that create an object of type MyBluetoothService. Will prevent crashes if attempts to write to Bluetooth are made when the connected does not exist.

Definition at line 373 of file MyBluetoothService.java.

```
373                                                 {
374         return mConnectedThreadRunning;
375     }
```

References com.example.choturemote.MyBluetoothService.mConnectedThreadRunning.

Referenced by com.example.choturemote.MainActivity.writeToBluetooth().

7.8.3.4 showToast()

```
abstract void com.example.choturemote.MyBluetoothService.showToast (
            int resourceID )  [abstract]
```

Displays Toasts.

Parameters

resourceID	The resource ID of the String to be displayed in a Toast

Toasts only run on the main thread. This allows displaying of a Toast from another thread. By being an abstract method, it can be defined in the MainActivity which runs on the main thread. A method there, showToastMethod(), can then display the Toast.

Referenced by com.example.choturemote.MyBluetoothService.ConnectThread.run().

7.8.3.5 startClient()

```
void com.example.choturemote.MyBluetoothService.startClient (
            BluetoothDevice device,
            UUID uuid )
```

Called to start a Bluetooth connection

Parameters

device	Represents a remote Bluetooth device. A BluetoothDevice lets you create a connection with the respective device or query information about it, such as the name, address, class, and bonding state.
uuid	Universally Unique Identifier. UUIDs are not tied to particular devices. They identify software services. You just need both sides to use the same one.

Definition at line 123 of file MyBluetoothService.java.

```
123                                                         {
124         // progress dialog appears //
125         mProgressDialog = ProgressDialog.show(mContext, "Connecting Bluetooth"
126                 , "Please Wait...", true);
127
128         mConnectThread = new ConnectThread(device, uuid);
129
130         // Starts Thread //
131         mConnectThread.start();
132     }
```

References com.example.choturemote.MyBluetoothService.mConnectThread, com.example.choturemote.My↩BluetoothService.mContext, and com.example.choturemote.MyBluetoothService.mProgressDialog.

Referenced by com.example.choturemote.MainActivity.beginConnection().

7.8.3.6 write()

```
void com.example.choturemote.MyBluetoothService.write (
            byte[] out )
```

Writes to the "face device".

Parameters

out	Array of bytes

- Called from the main activity
- Hands over the byte Array the Thread managing communication with the Bluetooth device

Definition at line 363 of file MyBluetoothService.java.

```
363                                     {
364         //perform the write
365         mConnectedThread.write(out);
366     }
```

References com.example.choturemote.MyBluetoothService.mConnectedThread, and com.example.choturemote.↩ MyBluetoothService.ConnectedThread.write().

Referenced by com.example.choturemote.MainActivity.writeToBluetooth().

7.8.4 Member Data Documentation

7.8.4.1 bluetoothAdapter

```
final BluetoothAdapter com.example.choturemote.MyBluetoothService.bluetoothAdapter  [private]
```

A BluetoothAdapter object.

A BluetoothAdapter lets you perform fundamental Bluetooth tasks, such as initiate device discovery, query a list of bonded (paired) devices, instantiate a BluetoothDevice using a known MAC address, and create a Bluetooth↩ ServerSocket to listen for connection requests from other devices, and start a scan for Bluetooth LE devices.

Definition at line 49 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.MyBluetoothService(), and com.example.↩ choturemote.MyBluetoothService.ConnectThread.run().

7.8.4.2 mConnectedThread

```
com.example.choturemote.MyBluetoothService.mConnectedThread
```

Custom class ConnectedThread type object.

Extends `Thread`

Definition at line 66 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.cancelThreads(), com.example.choturemote.My↩ BluetoothService.connected(), and com.example.choturemote.MyBluetoothService.write().

7.8.4.3 mConnectedThreadRunning

```
boolean com.example.choturemote.MyBluetoothService.mConnectedThreadRunning = false  [static]
```

boolean status of mConnectedThread

For use in other Activities that create an object of type MyBluetoothService. Will prevent crashes if attempts to write to Bluetooth are made when the connected does not exist.

Definition at line 72 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.ConnectedThread.cancel(), com.example.↩choturemote.MyBluetoothService.cancelThreads(), com.example.choturemote.MyBluetoothService.Connected↩Thread.ConnectedThread(), and com.example.choturemote.MyBluetoothService.connectedThreadRunning().

7.8.4.4 mConnectThread

```
com.example.choturemote.MyBluetoothService.mConnectThread  [private]
```

Custom class ConnectThread type object.

Extends `Thread`

Definition at line 65 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.cancelThreads(), and com.example.choturemote.↩MyBluetoothService.startClient().

7.8.4.5 mContext

```
Context com.example.choturemote.MyBluetoothService.mContext  [static], [package]
```

Reference to the Activity where the MyBluetoothService object instance is created.

Services, such as this one, require a Context from the Activity that creates an object of its type to hook it to that Activity and provide it access to the application specific resources.

Definition at line 43 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.MyBluetoothService(), and com.example.↩choturemote.MyBluetoothService.startClient().

7.8.4.6 mmDevice

```
BluetoothDevice com.example.choturemote.MyBluetoothService.mmDevice  [private]
```

Represents a remote Bluetooth device.

A BluetoothDevice lets you create a connection with the respective device or query information about it, such as the name, address, class, and bonding state.

Definition at line 78 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.ConnectThread.ConnectThread(), and com.↩ example.choturemote.MyBluetoothService.ConnectThread.run().

7.8.4.7 mProgressDialog

```
ProgressDialog com.example.choturemote.MyBluetoothService.mProgressDialog  [package]
```

A progress dialog.

A dialog showing a progress indicator and an optional text message or view. Only a text message or a view can be used at the same time.

Definition at line 55 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.ConnectedThread.ConnectedThread(), com.↩ example.choturemote.MyBluetoothService.ConnectThread.run(), and com.example.choturemote.MyBluetooth↩ Service.startClient().

7.8.4.8 TAG

```
final String com.example.choturemote.MyBluetoothService.TAG = "MY_APP_DEBUG_TAG"  [static],
[private]
```

Debugging tool

Definition at line 84 of file MyBluetoothService.java.

Referenced by com.example.choturemote.MyBluetoothService.ConnectThread.cancel(), com.example.↩ choturemote.MyBluetoothService.cancelThreads(), com.example.choturemote.MyBluetoothService.connected(), com.example.choturemote.MyBluetoothService.ConnectedThread.ConnectedThread(), com.example.choturemote.↩ MyBluetoothService.ConnectThread.ConnectThread(), com.example.choturemote.MyBluetoothService.Connect↩ Thread.run(), com.example.choturemote.MyBluetoothService.ConnectedThread.run(), and com.example.↩ choturemote.MyBluetoothService.ConnectedThread.write().

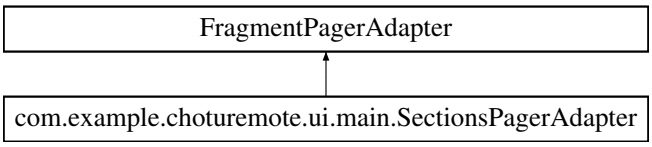
The documentation for this class was generated from the following file:

- MyBluetoothService.java

7.9 com.example.choturemote.ui.main.SectionsPagerAdapter Class Reference

Handles the tabbed "Whatsapp" look.

Inheritance diagram for com.example.choturemote.ui.main.SectionsPagerAdapter:



Public Member Functions

- SectionsPagerAdapter (Context context, FragmentManager fm)

  Constructor for SectionsPagerAdapter.
- Fragment getItem (int position)

  Return the Fragment associated with a specified position.
- CharSequence getPageTitle (int position)

  Called by ViewPager to obtain title for page.
- int getCount ()

  Total pages.

Private Attributes

- final Context mContext

  Holder of Activity context reference.

Static Private Attributes

- static final int[ ] TAB_TITLES = new int[ ]{R.string.tab_text_1, R.string.tab_text_2, R.string.tab_text_3}

  Integer references to String resources.

7.9.1 Detailed Description

Handles the tabbed "Whatsapp" look.

Class that extends FragmentPagerAdapter. Implementation of PagerAdapter that represents each page as a Fragment that is persistently kept in the fragment manager as long as the user can return to the page.

Definition at line 22 of file SectionsPagerAdapter.java.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 SectionsPagerAdapter()

```
com.example.choturemote.ui.main.SectionsPagerAdapter.SectionsPagerAdapter (
            Context context,
            FragmentManager fm )
```

Constructor for SectionsPagerAdapter.

Used to create an instance of SectionsPagerAdapter from an Activity. The super() method initializes the parent class FragmentPagerAdapter by calling its default constructor. The FragmentManager object passed to this constructor is passed along to the constructor of the base class.

Parameters

context	Reference to the Activity where the SectionsPagerAdapter object instance is created
fm	FragmentManager passed from the Activity where the SectionsPagerAdapter object instance is created. ??

Definition at line 43 of file SectionsPagerAdapter.java.

```
43                                                                  {
44         super(fm);
45         mContext = context;
46     }
```

References com.example.choturemote.ui.main.SectionsPagerAdapter.mContext.

7.9.3 Member Function Documentation

7.9.3.1 getCount()

```
int com.example.choturemote.ui.main.SectionsPagerAdapter.getCount ( )
```

Total pages.

Returns the number of views available

Returns

Number of views available

Definition at line 87 of file SectionsPagerAdapter.java.

```
87                          {
88         return 3;
89     }
```

7.9.3.2 getItem()

```
Fragment com.example.choturemote.ui.main.SectionsPagerAdapter.getItem (
            int position )
```

Return the Fragment associated with a specified position.

- getItem is called to instantiate the fragment for the given page
- Here it returns a Fragment of a certain class type

Parameters

position	Integer for position

Returns

Fragment associated with position

Definition at line 57 of file SectionsPagerAdapter.java.

```
57                                          {
58
59         switch(position){
60             case 0:
61                 return new ExpressionsFragment();
62             case 1:
63                 return new LocomotionFragment();
64             default: // case 2: //
65                 return new SpeakingFragment();
66         }
67     }
```

7.9.3.3 getPageTitle()

```
CharSequence com.example.choturemote.ui.main.SectionsPagerAdapter.getPageTitle (
            int position )
```

Called by ViewPager to obtain title for page.

This method is called by the ViewPager to obtain a title string to describe the specified page. This method may return null indicating no title for this page. The default implementation returns null.

Parameters

position	The position of the title requested

Returns

`@Nullable` denotes that a value can be null.

Definition at line 77 of file SectionsPagerAdapter.java.

```
77                                              {
78         return mContext.getResources().getString(TAB_TITLES[position]);
79     }
```

References com.example.choturemote.ui.main.SectionsPagerAdapter.mContext, and com.example.choturemote.↩ ui.main.SectionsPagerAdapter.TAB_TITLES.

7.9.4 Member Data Documentation

7.9.4.1 mContext

```
final Context com.example.choturemote.ui.main.SectionsPagerAdapter.mContext  [private]
```

Holder of Activity context reference.

Context refers to the activity where the adapter is created. It is helps you with accessing system services and resources in case you need them. mContext will store the reference to the Activity where the SectionsPagerAdapter object instance is created.

Definition at line 35 of file SectionsPagerAdapter.java.

Referenced by com.example.choturemote.ui.main.SectionsPagerAdapter.getPageTitle(), and com.example.↩ choturemote.ui.main.SectionsPagerAdapter.SectionsPagerAdapter().

7.9.4.2 TAB_TITLES

```
final int [] com.example.choturemote.ui.main.SectionsPagerAdapter.TAB_TITLES = new int[]{R.↩
string.tab_text_1, R.string.tab_text_2, R.string.tab_text_3}  [static], [private]
```

Integer references to String resources.

Integer Array consisting of references to String resources. `@StringRes` identifies the integers as String references.

Definition at line 29 of file SectionsPagerAdapter.java.

Referenced by com.example.choturemote.ui.main.SectionsPagerAdapter.getPageTitle().

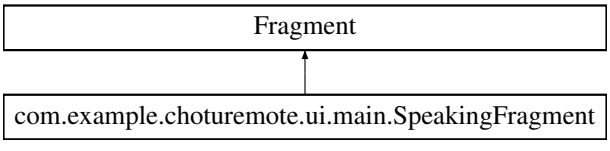
The documentation for this class was generated from the following file:

- SectionsPagerAdapter.java

7.10 com.example.choturemote.ui.main.SpeakingFragment Class Reference

Speaking Fragment.

Inheritance diagram for com.example.choturemote.ui.main.SpeakingFragment:



Public Member Functions

- SpeakingFragment ()

  Constructor.
- View onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)

  Called to have the fragment instantiate its user interface view.

Package Attributes

- int defaultPercentage

  Pitch is displayed as a percentage.

Static Private Attributes

- static float PITCH_FACTOR = 0.025f

  Fixed value.
- static float DEFAULT_PITCH = 1.0f

  Default Pitch.
- static float myPitch

  Float to hold current pitch.

7.10.1 Detailed Description

Speaking Fragment.

- A Fragment is a piece of an application's user interface or behavior that can be placed in an Activity.
- This one deals with the Speaking aspect

Definition at line 37 of file SpeakingFragment.java.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 SpeakingFragment()

`com.example.choturemote.ui.main.SpeakingFragment.SpeakingFragment ( )`

Constructor.

Required empty public constructor

Definition at line 66 of file SpeakingFragment.java.

```
66                                  {
67     }
```

7.10.3 Member Function Documentation

7.10.3.1 onCreateView()

```
View com.example.choturemote.ui.main.SpeakingFragment.onCreateView (
            LayoutInflater inflater,
            ViewGroup container,
            Bundle savedInstanceState )
```

Called to have the fragment instantiate its user interface view.

- This fragment has a View inflated from the speaking_view.xml file
- This View has a GridView with the resourceID "grid"; an EditText; a few buttons; a TextView for percentage display; and a seekBar
- This GridView is associated with an Adapter which is an ExpressionsWordAdapter object
- The ExpressionsWordAdapter object returns custom itemView object to populate the GridView
- The EditView works with the speakNowButton to send custom text to be spoken
- The SeekBar works with two buttons to handle pitch settings

Parameters

inflater	The LayoutInflater object that can be used to inflate any views in the fragment
container	This is the parent view that the fragment's UI should be attached to
savedInstanceState	If non-null, this fragment is being re-constructed from a previous saved state as given here

Returns

The View for the fragment's UI, or null.

Definition at line 84 of file SpeakingFragment.java.

```
85                                                 {
86
87         // Inflating speaking_view.xml (modified grid_view.xml) //
88         View rootView = inflater.inflate(R.layout.speaking_view, container, false);
89
90         // Entering custom text for speaking option //
91         final EditText editText = rootView.findViewById(R.id.editText);
92         ImageButton speakNowButton = rootView.findViewById(R.id.speak_now);
93
94         // A SeekBar for changing pitch//
95         final SeekBar pitchBar = rootView.findViewById(R.id.pitch_bar);
96         final TextView pitchPercentage = rootView.findViewById(R.id.pitch_percentage);
97
98         // Resetting and saving pitch settings //
99         ImageButton resetPitch = rootView.findViewById(R.id.reset_pitch);
100         ImageButton setDefaultPitch = rootView.findViewById(R.id.set_default_pitch);
101
102         // A list of Word objects //
103         final ArrayList<Word> words = new ArrayList<Word>();
104         words.add(new Word(R.string.HI, R.drawable.speech_bubble));
105         words.add(new Word(R.string.HELLO, R.drawable.speech_bubble));
106         words.add(new Word(R.string.HOW_ARE_YOU, R.drawable.speech_bubble));
107         words.add(new Word(R.string.WHATS_YOUR_NAME, R.drawable.speech_bubble));
108         words.add(new Word(R.string.MY_NAME_IS, R.drawable.speech_bubble));
109         words.add(new Word(R.string.IM_YOUR_FRIEND, R.drawable.speech_bubble));
110         words.add(new Word(R.string.THANK_YOU,R.drawable.speech_bubble));
```

```
111         words.add(new Word(R.string.YOU_ARE_WELCOME,R.drawable.speech_bubble));
112         words.add(new Word(R.string.GOOD_JOB,R.drawable.speech_bubble));
113         words.add(new Word(R.string.VERY_GOOD,R.drawable.speech_bubble));
114         words.add(new Word(R.string.WOW,R.drawable.speech_bubble));
115         words.add(new Word(R.string.BYE, R.drawable.speech_bubble));
116         words.add(new Word(R.string.FINISH,R.drawable.speech_bubble));
117         words.add(new Word(R.string.YES, R.drawable.speech_bubble));
118         words.add(new Word(R.string.NO, R.drawable.speech_bubble));
119 
120         // Command String variables and calculations for expressions //
121         myPitch = DEFAULT_PITCH;
122         defaultPercentage = (int) (DEFAULT_PITCH/PITCH_FACTOR);
123         pitchBar.setProgress(defaultPercentage);
124         pitchPercentage.setText(getString(R.string.PITCH_SLIDER_INFO) + ": " + defaultPercentage + "%");
125         final String SPEAK_OUT_CLASSIFIER = getString(R.string.SPEAK_OUT_CLASSIFIER);
126 
127         // An ExpressionsWordAdapter whose data source is a list of Words //
128         // The adapter knows how to create custom itemViews for each item in the list //
129         ExpressionsWordAdapter adapter = new ExpressionsWordAdapter(getActivity(), words,
    R.color.category_expressions);
130 
131         // Find the GridView view hierarchy of the Activity with the ID "grid" //
132         GridView gridView = rootView.findViewById(R.id.grid);
133 
134         // Number of columns in the gridView //
135         gridView.setNumColumns(1);
136 
137         // Make gridView use the SpeakingWordAdapter object we created above, so that the gridView will
    display items for each Word in the list.
138         gridView.setAdapter(adapter);
139 
140         // Set a click listener to send command String when the list item is clicked //
141         gridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
142             @Override
143             public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {
144 
145                 Word word = words.get(position);
146                 MainActivity.writeToBluetooth(SPEAK_OUT_CLASSIFIER,
    getString(word.getStringResourceId()),  String.valueOf(myPitch));
147             }
148         });
149 
150         // Set a click listener to send command String when the speakNowButton is clicked //
151         speakNowButton.setOnClickListener(new View.OnClickListener() {
152             @Override
153             public void onClick(View v) {
154 
155                 MainActivity.writeToBluetooth(SPEAK_OUT_CLASSIFIER, editText.getText().toString(),
    String.valueOf(myPitch));
156                 //editText.getText().clear();
157             }
158         });
159 
160         // Set a click listener to reset pitch //
161         resetPitch.setOnClickListener(new View.OnClickListener() {
162             @Override
163             public void onClick(View view) {
164                 pitchBar.setProgress(defaultPercentage);
165                 Toast.makeText(getContext(), "Pitch reset",Toast.LENGTH_SHORT).show();
166             }
167         });
168 
169         // Set a long click listener to reset pitch to default if updated //
170         resetPitch.setOnLongClickListener(new View.OnLongClickListener() {
171             @Override
172             public boolean onLongClick(View view) {
173                 defaultPercentage = (int) (DEFAULT_PITCH/PITCH_FACTOR);
174                 pitchBar.setProgress(defaultPercentage);
175                 Toast.makeText(getContext(), "Default pitch set to "+ defaultPercentage +
    "%",Toast.LENGTH_SHORT).show();
176                 return true;
177             }
178         });
179 
180         // Set a long click listener to update default pitch //
181         setDefaultPitch.setOnLongClickListener(new View.OnLongClickListener() {
182             @Override
183             public boolean onLongClick(View view) {
184                 defaultPercentage = pitchBar.getProgress();
185                 Toast.makeText(getContext(), "Default pitch set to "+ defaultPercentage + "%",
    Toast.LENGTH_SHORT).show();
186                 return true;
187             }
188         });
189 
190         // Set a listener to update when seekBar updated by user//
191         pitchBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
```

```
192
193             @Override
194             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
195                 if (progress == 0) progress = 1;
196                 myPitch = progress * PITCH_FACTOR;
197                 pitchPercentage.setText(getString(R.string.PITCH_SLIDER_INFO) + ": " + progress + "%");
198             }
199
200             @Override
201             public void onStartTrackingTouch(SeekBar seekBar) {
202
203             }
204
205             @Override
206             public void onStopTrackingTouch(SeekBar seekBar) {
207
208             }
209         });
210
211         return rootView;
212     }
```

References com.example.choturemote.ui.main.SpeakingFragment.DEFAULT_PITCH, com.example.choturemote.↩ ui.main.SpeakingFragment.defaultPercentage, com.example.choturemote.ui.main.Word.getStringResourceId(), com.example.choturemote.ui.main.SpeakingFragment.myPitch, com.example.choturemote.ui.main.Speaking↩ Fragment.PITCH_FACTOR, and com.example.choturemote.MainActivity.writeToBluetooth().

7.10.4 Member Data Documentation

7.10.4.1 DEFAULT_PITCH

```
com.example.choturemote.ui.main.SpeakingFragment.DEFAULT_PITCH = 1.0f  [static], [private]
```

Default Pitch.

Default Pitch as a float

Definition at line 58 of file SpeakingFragment.java.

Referenced by com.example.choturemote.ui.main.SpeakingFragment.onCreateView().

7.10.4.2 defaultPercentage

```
com.example.choturemote.ui.main.SpeakingFragment.defaultPercentage  [package]
```

Pitch is displayed as a percentage.

DEFAULT_PITCH/PITCH_FACTOR is how the default pitch percentage is calculated

Definition at line 60 of file SpeakingFragment.java.

Referenced by com.example.choturemote.ui.main.SpeakingFragment.onCreateView().

7.10.4.3 myPitch

`com.example.choturemote.ui.main.SpeakingFragment.myPitch [static], [private]`

Float to hold current pitch.

Holds the current pitch so tp display in the slider

Definition at line 59 of file SpeakingFragment.java.

Referenced by com.example.choturemote.ui.main.SpeakingFragment.onCreateView().

7.10.4.4 PITCH_FACTOR

`com.example.choturemote.ui.main.SpeakingFragment.PITCH_FACTOR = 0.025f [static], [private]`

Fixed value.

Dividing the actual pitch value

Definition at line 57 of file SpeakingFragment.java.

Referenced by com.example.choturemote.ui.main.SpeakingFragment.onCreateView().

The documentation for this class was generated from the following file:

- SpeakingFragment.java

7.11 com.example.choturemote.ui.main.Word Class Reference

Represents a collection of resource IDs.

Public Member Functions

- Word (int stringResourceId)
  Constructor.
- Word (int stringResourceId, int imageResourceId)
  Constructor.
- int getStringResourceId ()
- int getImageResourceId ()
- boolean hasImage ()

Private Attributes

- int mStringResourceId
- int mImageResourceId = NO_IMAGE_PROVIDED

Static Private Attributes

- static final int NO_IMAGE_PROVIDED = -1

7.11.1 Detailed Description

Represents a collection of resource IDs.

It contains resource IDs for Strings and optional Images

Definition at line 29 of file Word.java.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Word() [1/2]

```
com.example.choturemote.ui.main.Word.Word (
            int stringResourceId )
```

Constructor.

A constructor that only takes in a String resourceID. Not used. Kept for later use.

Parameters

stringResourceId	is the string resource ID for the text associated with a Word

Definition at line 45 of file Word.java.

```
45                                      {
46         mStringResourceId = stringResourceId;
47     }
```

References com.example.choturemote.ui.main.Word.mStringResourceId.

7.11.2.2 Word() [2/2]

```
com.example.choturemote.ui.main.Word.Word (
            int stringResourceId,
            int imageResourceId )
```

Constructor.

A constructor that only takes in a String resourceID and an Image resourceId

Parameters

stringResourceId	is the string resource ID for the text associated with a Word
image↩ ResourceId	is the drawable resource ID for the image associated with the Word

Definition at line 56 of file Word.java.

```
56                                                      {
57         mStringResourceId = stringResourceId;
58         mImageResourceId = imageResourceId;
59     }
```

References com.example.choturemote.ui.main.Word.mImageResourceId, and com.example.choturemote.ui.↩ main.Word.mStringResourceId.

7.11.3 Member Function Documentation

7.11.3.1 getImageResourceId()

```
int com.example.choturemote.ui.main.Word.getImageResourceId ( )
```

Get the image resource ID for the image associated with the Word

Returns

integer Resource ID

Definition at line 74 of file Word.java.

```
74                                      {
75         return mImageResourceId;
76     }
```

References com.example.choturemote.ui.main.Word.mImageResourceId.

Referenced by com.example.choturemote.ui.main.ExpressionsWordAdapter.getView(), and com.example.↩ choturemote.ui.main.LocomotionWordAdapter.getView().

7.11.3.2 getStringResourceId()

```
int com.example.choturemote.ui.main.Word.getStringResourceId ( )
```

Get the string resource ID for the text associated with the Word

Returns

integer Resource ID

Definition at line 65 of file Word.java.

```
65                                      {
66         return mStringResourceId;
67     }
```

References com.example.choturemote.ui.main.Word.mStringResourceId.

Referenced by com.example.choturemote.ui.main.ExpressionsWordAdapter.getView(), com.example.choturemote.↩ ui.main.ExpressionsFragment.onCreateView(), and com.example.choturemote.ui.main.SpeakingFragment.on↩ CreateView().

7.11.3.3 hasImage()

```
boolean com.example.choturemote.ui.main.Word.hasImage ( )
```

Returns whether or not there is an image for this word.

Returns

boolean TRUE or FALSE

Definition at line 82 of file Word.java.

```
82                                {
83         return mImageResourceId != NO_IMAGE_PROVIDED;
84     }
```

References com.example.choturemote.ui.main.Word.mImageResourceId, and com.example.choturemote.ui.↩ main.Word.NO_IMAGE_PROVIDED.

Referenced by com.example.choturemote.ui.main.ExpressionsWordAdapter.getView(), and com.example.↩ choturemote.ui.main.LocomotionWordAdapter.getView().

7.11.4 Member Data Documentation

7.11.4.1 mImageResourceId

```
int com.example.choturemote.ui.main.Word.mImageResourceId = NO_IMAGE_PROVIDED  [private]
```

Optional image resource ID for the image associated with a Word

Definition at line 35 of file Word.java.

Referenced by com.example.choturemote.ui.main.Word.getImageResourceId(), com.example.choturemote.ui.↩ main.Word.hasImage(), and com.example.choturemote.ui.main.Word.Word().

7.11.4.2 mStringResourceId

```
int com.example.choturemote.ui.main.Word.mStringResourceId  [private]
```

String resource ID for the text associated with a Word

Definition at line 32 of file Word.java.

Referenced by com.example.choturemote.ui.main.Word.getStringResourceId(), and com.example.choturemote.↩ ui.main.Word.Word().

7.11.4.3 NO_IMAGE_PROVIDED

```
final int com.example.choturemote.ui.main.Word.NO_IMAGE_PROVIDED = -1  [static], [private]
```

Constant value that represents no image was provided for this word

Definition at line 38 of file Word.java.

Referenced by com.example.choturemote.ui.main.Word.hasImage().

The documentation for this class was generated from the following file:

- Word.java

Chapter 8

File Documentation

8.1 ExpressionsFragment.java File Reference

Extends Fragment.

Classes

- class com.example.choturemote.ui.main.ExpressionsFragment

  Expressions Fragment.

Packages

- package com.example.choturemote.ui.main

8.1.1 Detailed Description

Extends Fragment.

- A Fragment is a piece of an application's user interface or behavior that can be placed in an Activity.
- This one deals with the Expressions aspect

8.2 ExpressionsWordAdapter.java File Reference

An ArrayAdapter Implementation.

Classes

- class com.example.choturemote.ui.main.ExpressionsWordAdapter

  An ArrayAdapter Implementation.

Packages

- package com.example.choturemote.ui.main

8.2.1 Detailed Description

An ArrayAdapter Implementation.

Contains an ArrayAdapter that can provide the layout for each list item based on a data source, which is a list of Word class objects.

8.3 LocomotionFragment.java File Reference

Extends Fragment.

Classes

- class com.example.choturemote.ui.main.LocomotionFragment

  Locomotion Fragment.

Packages

- package com.example.choturemote.ui.main

8.3.1 Detailed Description

Extends Fragment.

- A Fragment is a piece of an application's user interface or behavior that can be placed in an Activity.
- This one deals with the Locomotionaspect

8.4 LocomotionWordAdapter.java File Reference

An ArrayAdapter Implementation.

Classes

- class com.example.choturemote.ui.main.LocomotionWordAdapter

  An ArrayAdapter Implementation.

Packages

- package com.example.choturemote.ui.main

8.4.1 Detailed Description

An ArrayAdapter Implementation.

Contains an ArrayAdapter that can provide the layout for each list item based on a data source, which is a list of Word class objects.

8.5 MainActivity.java File Reference

The main activity.

Classes

- class com.example.choturemote.MainActivity
  This is where everything important happens.

Packages

- package com.example.choturemote

8.5.1 Detailed Description

The main activity.

The first screen to appear when the user launches the app.

8.6 MyBluetoothService.java File Reference

Handles everything Bluetooth.

Classes

- class com.example.choturemote.MyBluetoothService
  Handles everything Bluetooth.
- class com.example.choturemote.MyBluetoothService.ConnectThread
  Client thread that initiates a Bluetooth connection.
- class com.example.choturemote.MyBluetoothService.ConnectedThread
  Thread that manages a Bluetooth connection.

Packages

- package com.example.choturemote

8.6.1 Detailed Description

Handles everything Bluetooth.

- Starts up and maintains Bluetooth connections between devices
- Sends and handles reception of messages from connected devices

8.7 README.md File Reference

8.8 SectionsPagerAdapter.java File Reference

Handles the tabbed "Whatsapp" look.

Classes

- class com.example.choturemote.ui.main.SectionsPagerAdapter

  Handles the tabbed "Whatsapp" look.

Packages

- package com.example.choturemote.ui.main

8.8.1 Detailed Description

Handles the tabbed "Whatsapp" look.

Class that extends FragmentPagerAdapter. Implementation of PagerAdapter that represents each page as a Fragment that is persistently kept in the fragment manager as long as the user can return to the page.

8.9 SpeakingFragment.java File Reference

Extends Fragment.

Classes

- class com.example.choturemote.ui.main.SpeakingFragment

  Speaking Fragment.

Packages

- package com.example.choturemote.ui.main

8.9.1 Detailed Description

Extends Fragment.

- A Fragment is a piece of an application's user interface or behavior that can be placed in an Activity.
- This one deals with the Speaking aspect

8.10 Word.java File Reference

Represents a collection of resource IDs.

Classes

- class com.example.choturemote.ui.main.Word
  Represents a collection of resource IDs.

Packages

- package com.example.choturemote.ui.main

8.10.1 Detailed Description

Represents a collection of resource IDs.

It can contain resource IDs for Strings, Images, audio files, among other things

It contains resource IDs for Strings and optional Images

Index