Objective

• Working in Background – AsyncTask

• Cloud Computing – Windows Azure

• Two Presentation
  – Exploring Urban Characteristics Using Movement History of Mass Mobile Microbloggers
    • Presenter: Robert Lewis
  – SocialCircuits: The Art of Using Mobile Phones for Modeling Personal Interactions
    • Presenter: Wayne Stilwell
Stickyness – onStartCommand(intent, flags, startId)

• START_STICKY
  – System will try to re-create the Service and call onStartCommand(). If no pending start commands, it will be called with a null intent object.

• START_NOT_STICKY
  – If no new start intents to deliver to Service, system will don't recreate until a future explicit call to Context.startService(Intent).

• START_REDELIVER_INTENT
  – System will schedule to restart Service passing last delivered Intent in onStartCommand(). This continues until the service calls stopSelf() with the start ID provided to onStartCommand()
Stickyness – flags parameter

• You can use the parameter passed to startService to determine if the service is a system-based restart
  – Null
    • Initial call
  – START_FLAG_REDELIVERY
    • The Intent is a re-delivery of a previously delivered intent. Service returned START_REDELIVER_INTENT but had been killed before calling stopSelf() for that Intent.
  – START_FLAG_RETRY
    • Service restarted after an abnormal termination when service was set to START_STICKY
Determining start condition

```java
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    if ((flags & START_FLAG_RETRY) == 0) {
        // TODO If it's a restart, do something.
    }
    else {
        // TODO Alternative background process.
    }
    return Service.START_STICKY;
}
```
AsyncTask
Background threads

• To make app responsive, move all time-consuming operations off main app thread to child thread. Very important!

• Two options:
  – AsyncTask
  – Write own Threads
Creating AsyncTask

```java
new DownloadFilesTask().execute(url1, url2, url3, url4);

private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    @Override
    protected Long doInBackground(URL... urls) { //Background thread. Do not interact with UI
        int myProgress = 0; long result=0;
        // [... Perform background processing task, update myProgress ...]
        PublishProgress(myProgress)
        // [... Continue performing background processing task ...]

        // Return the value to be passed to onPostExecute
        return result;
    }

    @Override
    protected void onProgressUpdate(Integer... progress) { //Post interim updates to UI thread; access UI
        // [... Update progress bar, Notification, or other UI element ...]
    }

    @Override
    protected void onPostExecute(Long result) { //Run when doInBackground completed; access UI
        // [... Report results via UI update, Dialog, or notification ...]
        showDialog("Downloaded " + result + " bytes");
    }
}
```
Regular Thread

// This method is called on the main GUI thread.
private void mainProcessing() {
    // This moves the time consuming operation to a child thread.
    Thread thread = new Thread(doBackgroundThreadProcessing);
    thread.start();
}

// Runnable that executes the background processing method.
private Runnable doBackgroundThreadProcessing = new Runnable() {
    public void run() {
        [ ... Time consuming operations ... ]
    }
};
Important: power management

• Just because you have code in a BroadcastReceiver or Service doesn’t mean it will run if the phone goes into a low-power state

• Common problem: create a Broadcast receiver. Create a thread from within it to run code....

• All works fine when phone on and plugged into computer during development

• Fails under normal use because phone shuts down quickly in power management state

• Need to use a WakeLock!
WakeLock

• If you **start a Service** or **broadcast an Intent** within the **onReceive()** handler of a BroadcastReceiver, it is possible that the WakeLock it holds will be released before your Service has started! To ensure the Service is executed; you will need to put a separate WakeLock policy in place.
WakeLock

• Control the power state on device (somewhat)

• Used to
  – Keep the CPU running
  – Prevent screen dimming or going off
  – Prevent backlight from turning on

• Only use when necessary and release as quickly as possible
Creating a WakeLock

```
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
WakeLock wakeLock = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
    "MyWakeLock");
wakeLock.acquire();
[ ... Do things requiring the CPU stay active ... ]
wakeLock.release();
```

PARTIAL_WAKE_LOCK keeps the CPU running without the screen on

For additional Flag Values:
Cloud Computing

• Control the power state on device (somewhat)
Cloud Computing

- SaaS: Software as a Service
- PaaS: Platform as a Service
- IaaS: Infrastructure as a Service

Top Cloud Computing Providers:
- Microsoft
- Amazon Web Services
- Google
- IBM
- Salesforce
Windows Azure

- Windows Azure is a foundation for running applications and storing data in the cloud

- Customers use it to run applications and store data on Internet-accessible machines owned by Microsoft.
Azure Components
Storage

HTTP/HTTPS, OData

Blobs

Tables

Queues

Applications and Data

Connect

Compute

Storage

CDN

Fabric Controller
Fabric Controller
Content Delivery Network
Connect
Example 1: Scalable Web Application
Example 2: Parallel Processing Application
Example 3: Using Cloud Storage
Hawaii/Azure Cloud Services

- **Relay Service**
  - Provides a relay point in the cloud that mobile applications can use to communicate

- **Endpoint**

  ```csharp
  //Create endpoint.
  Endpoint endpoint = new Endpoint("DeviceId", "RelayTestClient", TimeSpan.MaxValue);
  ```

- **Functions:**
  - Join or Leave an existing group
  - Send or receive messages to members
  - Deregister the endpoint
Hawaii/Azure Cloud Services

• OCR in the Cloud

• takes a photographic image that contains some text and returns the text

```csharp
OcrClient service = new OcrClient("ocr.hawaii-services.net", clientId);

service.OcrCompleted += OnOcrCompleted;
service.RecognizeImageAsync(photoBits);

...

private void OnOcrCompleted(object sender, OcrCompletedEventArgs e)
{
    ...
}
```
• Speech to Text
  • Takes a spoken phrase and returns text.

```
SpeechRecognitionClient service = new SpeechRecognitionClient("stt.hawaiiservices.net", clientId)

service.SpeechGrammarsReceived += this.OnSpeechGrammarsReceived;
service.GetSpeechGrammarsAsync();

service.SpeechRecognitionCompleted += OnSpeechRecognitionCompleted;
service.RecognizeSpeechAsync(audioBuffer);
...
private void OnSpeechGrammarsReceived(object sender,
    SpeechGrammarsReceivedEventArgs e)
{
    ...
}
```
Hawaii/Azure Cloud Services

• **Rendezvous Service**
  - Mapping service from well-known human-readable names to endpoints in the Hawaii Relay Service.

• **User Identification**
  - Using Windows Live ID to identifying visitors to web sites.

• **Mapping**
  - Using Virtual Earth to provide maps for given latitude and longitude coordinates.

• **Computation**

• **Storage**
Resources

• Azure: http://www.microsoft.com/windowsazure/learn/get-started/

• Hawaii: http://research.microsoft.com/en-us/um/redmond/projects/hawaii/students/

• Supported Schools: http://research.microsoft.com/en-us/um/redmond/projects/hawaii/instructors/

Questions?
To Do

• Assignment #4: Vision and Scope
  • Think out-of-box
  • Outrageous, but feasible, idea will have high grades
  • Interesting domains for ideas:
    - Increase driving safety
    - Traffic monitoring
    - Enhance education experience
    - Monitor/support personal health
    - Monitor/save energy consumption
    - Support smart environments