Developing Android REST Client Applications

Virgil Dobjanschi
5/20/2010
Developing Android REST Client Applications

• View live notes and ask questions about this session on Google Wave:
REST Client Applications
REST Client Applications

• REST: A broadly adopted architecture style
REST Client Applications

• REST: A broadly adopted architecture style
• A large number of REST APIs are available
REST Client Applications

• REST: A broadly adopted architecture style
• A large number of REST APIs are available
• Why develop them if mobile friendly web sites already exist?
Incorrect Implementation of REST Methods

“I haven’t failed, I’ve found 10,000 ways that don’t work.”
- Thomas Alva Edison
The Incorrect Implementation of REST Methods

1. Get, create, update, delete

Worker thread

2. GET/POST/PUT/DELETE

3. Process data

4. Save data structure

Processor

CursorAdapter

Activity

Memory Storage
What’s wrong with this approach?
What’s wrong with this approach?

• The operating system may shut down the process
What’s wrong with this approach?

• The operating system may shut down the process
• Data is not persistently stored
Implementing REST Methods

“There’s a way to do it better ... find it.”
- Thomas Alva Edison
REST Method Implementation Patterns
REST Method Implementation Patterns

• Introducing three design patterns to handle REST methods
  – Use a Service API
  – Use the ContentProvider API
  – Use the ContentProvider API and a SyncAdapter
Implementing REST Methods
Option A: Use a Service API
Option A: Use a Service API

1. initiate(param)
2. startService(Intent)
3. start(param)
4. insert/update
5. start(param)
6. GET/POST/PUT/DELETE
7. REST method complete callback
8. insert/update
8’. ContentObserver notification
9. Operation complete callback
10. Binder callback
11. Callback to registered activities

Content Provider

CursorAdapter

Activity

Service Helper

Service

Processor

REST Method
The REST Method
The REST Method

• An entity which:
  – Prepares the HTTP URL & HTTP request body
  – Executes the HTTP transaction
  – Processes the HTTP response
The REST Method

• An entity which:
  – Prepares the HTTP URL & HTTP request body
  – Executes the HTTP transaction
  – Processes the HTTP response

• Select the optimal content type for responses
  – Binary, JSON, XML
  – New in Froyo: JSON parser (same org.json API)
The REST Method

• An entity which:
  – Prepares the HTTP URL & HTTP request body
  – Executes the HTTP transaction
  – Processes the HTTP response
• Select the optimal content type for responses
  – Binary, JSON, XML
  – New in Froyo: JSON parser (same org.json API)
• Enable the gzip content encoding when possible
The REST Method

• An entity which:
  – Prepares the HTTP URL & HTTP request body
  – Executes the HTTP transaction
  – Processes the HTTP response
• Select the optimal content type for responses
  – Binary, JSON, XML
  – New in Froyo: JSON parser (same org.json API)
• Enable the gzip content encoding when possible
• Run the REST method in a worker thread
The REST Method

• An entity which:
  – Prepares the HTTP URL & HTTP request body
  – Executes the HTTP transaction
  – Processes the HTTP response

• Select the optimal content type for responses
  – Binary, JSON, XML
  – New in Froyo: JSON parser (same org.json API)

• Enable the gzip content encoding when possible
• Run the REST method in a worker thread
• Use the Apache HTTP client
Option A: Use a Service API

1. `initiate(param)`
2. `startService(Intent)`
3. `start(param)`
4. `insert/update`
5. `start(param)`
6. `GET/POST/PUT/DELETE`
7. `REST method complete callback`
8. `insert/update`
8'. `ContentObserver notification`
9. `Operation complete callback`
10. `Binder callback`
11. `Callback to registered activities`
The Processor (POST & PUT)

**POST**
- 4.insert
  - (set STATE_POSTING)
- 8.update
  - (clear STATE_POSTING)

**PUT**
- 4.update
  - (set STATE_UPDATING)
- 8.update
  - (clear STATE_UPDATING)
The Processor (DELETE & GET)

**DELETE**

4. update
(set STATE_DELETING)

8. delete

**Processor**

**REST Method**

**Content Provider**

**GET**

8. insert new resources

**Processor**

**REST Method**

**Content Provider**
Option A: Use a Service API

1. initiate(param)
2. startService(Intent)
3. start(param)
4. insert/update
5. start(param)
6. GET/POST/PUT/DELETE
7. REST method complete callback
8. insert/update
9. Operation complete callback
10. Binder callback
11. Callback to registered activities

8’. ContentObserver notification
8”. requery
The Service
The Service

• The role of the service
The Service

• The role of the service

• Forward path: receives the Intent sent by the Service Helper and starts the corresponding REST Method
The Service

• The role of the service
• Forward path: receives the Intent sent by the Service Helper and starts the corresponding REST Method
• Return path: handles the Processor callback and invokes the Service Helper binder callback
The Service

• The role of the service

• Forward path: receives the Intent sent by the Service Helper and starts the corresponding REST Method

• Return path: handles the Processor callback and invokes the Service Helper binder callback

• It can implement a queue of downloads
Option A: Use a Service API

1. initiate(param)

Service Helper

2. startService(Intent)

Service

3. start(param)

Processor

4. insert/update

5. start(param)

REST Method

6. GET/POST/PUT/DELETE

7. REST method complete callback

8. requery

8’. ContentObserver notification

8” requery

9. Operation complete callback

Content Provider

10. Binder callback

11. Callback to registered activities
The Service Helper
The Service Helper

• Singleton which exposes a simple asynchronous API to be used by the user interface
The Service Helper

• Singleton which exposes a simple asynchronous API to be used by the user interface

• Prepare and send the Service request
  – Check if the method is already pending
  – Create the request Intent
  – Add the operation type and a unique request id
  – Add the method specific parameters
  – Add the binder callback
  – Call startService(Intent)
  – Return the request id
The Service Helper

• Singleton which exposes a simple asynchronous API to be used by the user interface
• Prepare and send the Service request
  – Check if the method is already pending
  – Create the request Intent
  – Add the operation type and a unique request id
  – Add the method specific parameters
  – Add the binder callback
  – Call startService(Intent)
  – Return the request id
• Handle the callback from the service
  – Dispatch callbacks to the user interface listeners
Option A: Use a Service API

1. initiate(param)
2. startService(Intent)
3. start(param)
4. insert/update
5. start(param)
6. GET/POST/PUT/DELETE
7. REST method complete callback
8. requery
8'. ContentObserver notification
9. Operation complete callback
10. Binder callback
11. Callback to registered activities

Activity

Service Helper

Service

Processor

REST Method

CursorAdapter

Content Provider

Footer: 20
Handling the REST Method in an Activity

Activity & CursorAdapter
Handling the REST Method in an Activity

• Add an operation listener in onResume and remove it in onPause
Handling the REST Method in an Activity

• Add an operation listener in onResume and remove it in onPause
• Consider these cases:
  – The Activity is still active when the request completes
  – The Activity is paused then resumed and then the request completes
  – The Activity is paused when the request completes and then Activity is resumed
Handling the REST Method in an Activity

• Add an operation listener in onResume and remove it in onPause

• Consider these cases:
  – The Activity is still active when the request completes
  – The Activity is paused then resumed and then the request completes
  – The Activity is paused when the request completes and then Activity is resumed

• The CursorAdapter handles the ContentProvider notification by implementing a ContentObserver
Option A: Use a Service API

1. initiate(param)
2. startService(Intent)
3. start(param)
4. insert/update
5. start(param)
6. GET/POST/PUT/DELETE
7. REST method complete callback
8. insert/update
9. Operation complete callback
10. Binder callback
11. Callback to registered activities

8'. ContentObserver notification
8'’. requery

Activity
Service Helper
Service
Processor
REST Method

CursorAdapter
Content Provider
Implementing REST Methods
Option B: Use the ContentProvider API
Option B: Use the ContentProvider API

1. query, insert, update, delete
2. start(Intent)
3. startService(Intent)
4. start(param)
5. GET/POST/PUT/DELETE
6. Process data
7. insert/update

7’. ContentObserver notification
7”. query

Activity & CursorAdapter

Content Provider

Service Helper

Service

REST Method

Processor
A Simple Pattern for the REST of us

Option C: Use a ContentProvider API and a SyncAdapter

“To have a great idea, have a lot of them.”
-Thomas Alva Edison
A Simple Pattern Using the ContentProvider API

Use a sync adapter to initiate all your REST methods

1. Get items that need to be synced
2. start(param)
3. GET/POST/PUT/DELETE
4. Process data
5. insert/update/delete
6. ContentObserver notification
6’. requery

Sync Adapter

Activity & CursorAdapter

Content Provider

REST Method

Processor
Conclusions

“The value of an idea lies in the using of it.”
- Thomas Alva Edison
Conclusions
Conclusions

• Do not implement REST methods inside Activities
Conclusions

• Do not implement REST methods inside Activities
• Start long running operations from a Service
Conclusions

• Do not implement REST methods inside Activities
• Start long running operations from a Service
• Persist early & persist often
Conclusions

- Do not implement REST methods inside Activities
- Start long running operations from a Service
- Persist early & persist often
- Minimize the network usage
Conclusions

• Do not implement REST methods inside Activities
• Start long running operations from a Service
• Persist early & persist often
• Minimize the network usage
• Use a sync adapter to execute background operations which are not time critical
  – New in Froyo: Android Cloud to Device Messaging
Developing Android REST Client Applications

• View live notes and ask questions about this session on Google Wave: