The AsyncTask Framework: Key Methods

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Recognize the capabilities provided by the Android AsyncTask framework
- Know which methods are provided by AsyncTask class
Categories of Methods in AsyncTask
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods

AsyncTask

```java
public abstract class AsyncTask
extends Object

java.lang.Object
  ↓ android.os.AsyncTask&lt;Params, Progress, Result&gt;
```

AsyncTask enables proper and easy use of the UI thread. This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

AsyncTask is designed to be a helper class around Thread and Handler and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.) If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the java.util.concurrent package such as Executor, ThreadPoolExecutor and FutureTask.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.

See developer.android.com/reference/android/os/AsyncTask.html
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

  ```java
  public class AsyncTask<Params, Progress, Result> {
      public void execute(Params... params) {
          // Execute task with specified parameters
      }
      public void executeOnExecutor(Executor exec, Params... params) {
          // Execute task with specified parameters on specified Executor
      }
      public static void execute(Runnable runnable) {
          // Convenience version of execute(Object) for use with a simple Runnable object
      }
      public boolean cancel(boolean mayInterruptIfRunning) {
          // Attempts to cancel execution of this task
      }
      public boolean isCancelled() {
          // True if task was cancelled before completing
      }
  }
  ```
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

Run each async task one-at-a-time (serially) in a background thread within a process

AsyncTask<Params, Progress, Result> execute(Params... params)
  - Execute task with specified parameters

AsyncTask<Params, Progress, Result> executeOnExecutor(Executor exec, Params... params)
  - Execute task with specified parameters on specified Executor

static void execute(Runnable runnable)
  - Convenience version of execute(Object) for use with a simple Runnable object

boolean cancel
  (boolean mayInterruptIfRunning)
  - Attempts to cancel execution of this task

boolean isCancelled()
  - True if task was cancelled before completing
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

AsyncTask<Params, Progress, Result>
execute(Params... params)
- Execute task with specified parameters

AsyncTask<Params, Progress, Result>
executeOnExecutor(Executor exec,
Params... params)
- Execute task with specified parameters on specified Executor

static void execute(Runnable runnable)
- Convenience version of execute(Object) for use with a simple Runnable object

boolean cancel
  (boolean mayInterruptIfRunning)
- Attempts to cancel execution of this task

boolean isCancelled()
- True if task was cancelled before completing

Runs multiple async tasks concurrently in a pool of threads within a process
The AsyncTask class has two types of methods:

- Public methods
  - Typically invoked by clients

```java
AsyncTask<Params, Progress, Result> execute(Params... params)
  • Execute task with specified parameters

AsyncTask<Params, Progress, Result> executeOnExecutor(Executor exec, 
  Params... params)
  • Execute task with specified parameters on specified Executor

static void execute(Runnable runnable)
  • Convenience version of execute(Object) for use with a simple Runnable object

boolean cancel
  (boolean mayInterruptIfRunning)
  • Attempts to cancel execution of this task

boolean isCancelled()
  • True if task was cancelled before completing
```

A simple front-end to the underlying executor
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

  ```java
  class AsyncTask<Params, Progress, Result>
  {
    public static void execute(Runnable runnable)
    {
      // Convenience version of execute(Object)
      // for use with a simple Runnable object
    }

    public boolean cancel()
    {
      // Attempts to cancel execution of this task
    }

    public boolean isCancelled()
    {
      // True if task was cancelled before completing
    }
  }
  ```

- Requires cooperation by the async task, i.e., it’s voluntary

See earlier lessons on “Managing the Thread Lifecycle”
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected methods
    - Overridden by subclasses

```java
void onPreExecute()
    • Runs on UI thread before doInBackground()

abstract Result doInBackground(Params... params)
    • Override this method to perform a computation in a background thread

void onProgressUpdate(Progress... values)
    • Runs on UI thread after publishProgress() called

void onPostExecute(Result result)
    • Runs on UI thread after doInBackground()

void onCancelled(Result result)
    • Runs on UI thread after cancel() is invoked & doInBackground() has finished

...
The AsyncTask framework applies the Template Method pattern to call these methods at different points of time & in different thread contexts.

See [en.wikipedia.org/wiki/Template_method_pattern](en.wikipedia.org/wiki/Template_method_pattern)
The AsyncTask class has two types of methods:

- Public methods
- Protected methods
  - Overridden by subclasses

**Called in UI thread after `execute()` called, i.e., prior to other processing**

```java
void onPreExecute()
  - Runs on UI thread before `doInBackground()`

abstract Result doInBackground(Params... params)
  - Override this method to perform a computation in a background thread

void onProgressUpdate(Progress... values)
  - Runs on UI thread after `publishProgress()` called

void onPostExecute(Result result)
  - Runs on UI thread after `doInBackground()`

void onCancelled(Result result)
  - Runs on UI thread after `cancel()` is invoked & `doInBackground()` has finished
```

...
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected methods
    - Overridden by subclasses

```
void onPreExecute()
  - Runs on UI thread before doInBackground()

abstract Result doInBackground(Params... params)
  - Override this method to perform a computation in a background thread

void onProgressUpdate(Progress... values)
  - Runs on UI thread after publishProgress() called

void onPostExecute(Result result)
  - Runs on UI thread after doInBackground() called

void onCancelled(Result result)
  - Runs on UI thread after cancel() is invoked & doInBackground() has finished
```

Runs in a background thread to perform the computation
Categories of Methods in the AsyncTask Class

• The AsyncTask class has two types of methods
  • Public methods
  • Protected methods
    • Overridden by subclasses

void onPreExecute()
  • Runs on UI thread before doInBackground()

abstract Result doInBackground(
    Params... params)
  • Override this method to perform a computation in a background thread

void onProgressUpdate(Progress...
    values)
  • Runs on UI thread after publishProgress() called

void onPostExecute(Result result)
  • Runs on UI thread after doInBackground() has finished

void onCancelled(Result result)
  • Runs on UI thread after cancel() is invoked & doInBackground() has finished

Called in UI thread to convey incremental results sent from a background thread
The AsyncTask class has two types of methods:

- **Public methods**
- **Protected methods** (overridden by subclasses)

### Public Methods

- **void onPreExecute()**
  - Runs on UI thread before doInBackground() is called.

- **abstract Result doInBackground(Params... params)**
  - Override this method to perform a computation in a background thread.

- **void onProgressUpdate(Progress... values)**
  - Runs on UI thread after publishProgress() called.

- **void onPostExecute(Result result)**
  - Runs on UI thread after doInBackground() is called.

- **void onCancelled(Result result)**
  - Runs on UI thread after cancel() is invoked & doInBackground() has finished.

---

**Called in UI thread after all background processing is finished successfully**
The AsyncTask class has two types of methods:

- Public methods
- Protected methods
  - Overridden by subclasses

```java
void onPreExecute()
  • Runs on UI thread before doInBackground()

abstract Result doInBackground(Params... params)
  • Override this method to perform a computation in a background thread

void onProgressUpdate(Progress... values)
  • Runs on UI thread after publishProgress() called

void onPostExecute(Result result)
  • Runs on UI thread after doInBackground() has finished

void onCancelled(Result result)
  • Runs on UI thread after cancel() is invoked & doInBackground() has finished
```

**Called in UI thread after background processing has been cancelled**
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected methods
    - Overridden by subclasses
  - Final methods

void publishProgress(Progress... values)

- Invoked from doInBackground() to publish updates on UI thread while the background computation is still running

... Each call to this method triggers execution of onProgressUpdate() in the UI thread
Overriding Hook Methods in the AsyncTask Class
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- executeOnExecutor()
- execute()
- cancel()
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

**ImageDownloadTask**
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- executeOnExecutor()
- execute()
- cancel()
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

**ImageDownloadTask**
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

*The doInBackground() method must be overridden*
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

AsyncTask

- `executeOnExecutor()`
- `execute()`
- `cancel()`
- `onPreExecute()`
- `doInBackground()`
- `onProgressUpdate()`
- `onPostExecute()`
- `onCancelled()`

ImageDownloadTask

- `onPreExecute()`
- `doInBackground()`
- `onProgressUpdate()`
- `onPostExecute()`
- `onCancelled()`

Can only be called once per async task object by code in the UI thread
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- executeOnExecutor()
- execute()
- cancel()
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

**ImageDownloadTask**
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

*Implemented as a variant of the Template Method pattern*

See [en.wikipedia.org/wiki/Template_method_pattern](http://en.wikipedia.org/wiki/Template_method_pattern)
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

AsyncTask

executeOnExecutor()
execute()
cancel()
onPreExecute()
doInBackground()
onProgressUpdate()
onPostExecute()
onCancelled()
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- executeOnExecutor()
- execute()
- cancel()
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

**ImageDownloadTask**
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

*Invoked by framework in the UI thread to perform initialization actions*
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- executeOnExecutor()
- execute()
- cancel()
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

**ImageDownloadTask**
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

Invoked by framework in a to perform long duration operations at the “background” thread priority

See www.androiddesignpatterns.com/2014/01/thread-scheduling-in-android.html
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- executeOnExecutor()
- execute()
- cancel()
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

**ImageDownloadTask**
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

*Invoked by framework in UI thread when background thread calls publishProgress()*
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- `executeOnExecutor()`
- `execute()`
- `cancel()`
- `onPreExecute()`
- `doInBackground()`
- `onProgressUpdate()`
- `onPostExecute()`
- `onCancelled()`

**ImageDownloadTask**
- `onPreExecute()`
- `doInBackground()`
- `onProgressUpdate()`
- `onPostExecute()`
- `onCancelled()`

*Invoked by framework in UI thread when `doInBackground()` returns its result*
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- `executeOnExecutor()`
- `execute()`
- `cancel()`
- `onPreExecute()`
- `doInBackground()`
- `onProgressUpdate()`
- `onPostExecute()`
- `onCancelled()`

**Called by application to attempt to stop the execution of the task**

**ImageDownloadTask**
- `onPreExecute()`
- `doInBackground()`
- `onProgressUpdate()`
- `onPostExecute()`
- `onCancelled()`

See earlier lessons on “Managing the Thread Lifecycle”
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden

**AsyncTask**
- executeOnExecutor()
- execute()
- cancel()
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

**ImageDownloadTask**
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

Invoked by framework in the UI thread after cancel() is called & doInBackground() is finished

If onCancelled() is called then onPostExecute() is *not* called & vice versa
AsyncTask must be extended & one or more of its hook methods overridden

Should periodically call isCancelled() to check to see if it’s been cancelled

Similar to using the Java interrupt() method to voluntarily shutdown threads
AsyncTask is also parameterized with three types used by its hook methods:

- **Params** – Type used in background work
- **Progress** – Type used when indicating progress
- **Result** – Type of result

### Overriding Hook Methods in the AsyncTask Class

- doInBackground()
- onPostExecute()
- onCancelled()
- onCancelled()
- onPreExecute()
- doInBackground()

### ImageDownloadTask

- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()
Overriding Hook Methods in the AsyncTask Class

- AsyncTask is also parameterized with three types used by its hook methods
  - **Params** – Type used in background work
  - **Progress** – Type used when indicating progress
  - **Result** – Type of result

```java
AsyncTask
executeOnExecutor()
execute()
cancel()
onPreExecute()
doInBackground()
onProgressUpdate()
onPostExecute()
onCancelled()
```

```java
ImageDownloadTask
onPreExecute()
doInBackground()
onProgressUpdate()
onPostExecute()
onCancelled()
```
Overriding Hook Methods in the AsyncTask Class

AsyncTask is also parameterized with three types used by its hook methods:

- **Params** – Type used in background work
- **Progress** – Type used when indicating progress
- **Result** – Type of result

```java
AsyncTask
executeOnExecutor()
execute()
cancel()
onPostExecute()
doInBackground()
onPostExecute()
onCancelled()
```

```java
ImageDownloadTask
onPreExecute()
doInBackground()
onProgressUpdate()
onPostExecute()
onCancelled()
```
Overriding Hook Methods in the AsyncTask Class

AsyncTask is also parameterized with three types used by its hook methods:

- **Params** – Type used in background work
- **Progress** – Type used when indicating progress
- **Result** – Type of result

AsyncTask

- executeOnExecutor()
- execute()
- cancel()
- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()

ImageDownloadTask

- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()
- onCancelled()
Overriding Hook Methods in the AsyncTask Class

Apps must customize the AsyncTask class to meet their concurrency needs.

```java
class DownloadTask extends AsyncTask<Uri, Integer, Long> {
    protected void onPreExecute()
        { startDialog("Downloading file"); } 

    protected Long doInBackground(Uri... urls)
        { /* Download url & publish process */ } 

    protected void onProgressUpdate(Integer... progress)
        { setProgressPercent(progress[0]); } 

    protected void onPostExecute(Long res)
        { stopDialog("Got " + res + " bytes"); } 

    new DownloadTask().execute(downloadURL); 
}
```

Variant of developer.android.com/reference/android/os/AsyncTask.html
Overriding Hook Methods in the AsyncTask Class

- Apps must customize the AsyncTask class to meet their concurrency needs.

```java
class DownloadTask extends AsyncTask<Uri, Integer, Long> {
    protected void onPreExecute()
    { startDialog("Downloading file"); }

    protected Long doInBackground(Uri... urls)
    { /* Download url & publish process */ }

    protected void onProgressUpdate(Integer... progress)
    { setProgressPercent(progress[0]); }

    protected void onPostExecute(Long res)
    { stopDialog("Got " + res + " bytes"); }
}

new DownloadTask().execute(downloadURL);
```
Overriding Hook Methods in the AsyncTask Class

Apps must customize the AsyncTask class to meet their concurrency needs.

```java
class DownloadTask extends AsyncTask<Uri, Integer, Long> {
    protected void onPreExecute()
    { startDialog("Downloading file"); }

    protected Long doInBackground(Uri... urls)
    { /* Download url & publish process */ } 

    protected void onProgressUpdate(Integer... progress)
    { setProgressPercent(progress[0]); } 

    protected void onPostExecute(Long res)
    { stopDialog("Got " + res + " bytes"); }
}

new DownloadTask().execute(downloadURL);
```

This template method initiates async processing.
Overriding Hook Methods in the AsyncTask Class

- Apps must customize the AsyncTask class to meet their concurrency needs

```java
class DownloadTask extends 
    AsyncTask<Uri, Integer, Long> {

    protected void onPreExecute()
    { startDialog("Downloading file"); } 

    protected Long doInBackground
        (Uri... urls)
    { /* Download url & publish process */ } 

    protected void onProgressUpdate
        (Integer... progress)
    { setProgressPercent(progress[0]); } 

    protected void onPostExecute(Long res)
    { stopDialog("Got " + res + " bytes"); } 
}

new DownloadTask().execute(downloadURL);
```
Overriding Hook Methods in the AsyncTask Class

- Apps must customize the AsyncTask class to meet their concurrency needs.

```java
class DownloadTask extends AsyncTask<Ur, Integer, Long> {

    protected void onPreExecute()
    { startDialog("Downloading file"); }

    protected Long doInBackground(Ur... urls)
    { /* Download url & publish process */ }

    protected void onProgressUpdate(Integer... progress)
    { setProgressPercent(progress[0]); }

    protected void onPostExecute(Long res)
    { stopDialog("Got " + res + " bytes"); }
}

new DownloadTask().execute(downloadURL);
```
Apps must customize the AsyncTask class to meet their concurrency needs.

```java
class DownloadTask extends AsyncTask<Uri, Integer, Long> {

    protected void onPreExecute() {
        startDialog("Downloading file");
    }

    protected Long doInBackground(Uri... urls) {
        /* Download url & publish process */
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long res) {
        stopDialog("Got " + res + " bytes");
    }

    new DownloadTask().execute(downloadURL);
}
```
Overriding Hook Methods in the AsyncTask Class

- Apps must customize the AsyncTask class to meet their concurrency needs.

```java
class DownloadTask extends 
        AsyncTask<Uri, Integer, Long> {

    protected void onPreExecute()
        { startDialog("Downloading file"); } 

    protected Long doInBackground
        (Uri... urls)
        { /* Download url & publish process */ } 

    protected void onProgressUpdate
        (Integer... progress)
        { setProgressPercent(progress[0]); } 

    protected void onPostExecute(Long res)
        { stopDialog("Got " + res + " bytes"); } 
}

new DownloadTask().execute(downloadURL);
```
End of the AsyncTask Framework: Key Methods