Бесплатная электронная книга

УЧУСЬ android-asynctask

Free unaffiliated eBook created from
Stack Overflow contributors.

#android-asynctask

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: android-asynctask

It is an unofficial and free android-asynctask ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official android-asynctask.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с андроид-асинтезной

замечания

В этом разделе представлен обзор того, что такое андроид-асинтеза, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в андроид-асинтете, и ссылаться на связанные темы. Поскольку Documentation for android-asynctask является новым, вам может потребоваться создать начальные версии этих связанных тем.

Examples

AsyncTask от концепции к реализации

концепция

AsyncTask - это класс, который позволяет выполнять операции в фоновом режиме, при этом результаты публикуются в потоке пользовательского интерфейса. Основная цель состоит в том, чтобы устранить весь код шаблона для запуска / запуска потока, исключив обработчики и все, что необходимо для управления потоками. Кроме того, цель AsyncTask - иметь кратковременные операции над фоновым потоком (всего несколько секунд), а не длительные операции. Поэтому важно, чтобы AsyncTask не путался с общей инфраструктурой потоков. Если нужно выполнять длительные операции, рекомендуется использовать параллельный пакет.

Общие Соображения

AsyncTask определяется тремя родовыми типами: Params, Progress и Results. С момента его выполнения он проходит 4 шага (методы). Сначала это ***onPreExecute*** , где кто-то может определить диалог загрузки или какое-то сообщение UI, которое может уведомить пользователя о запуске выполнения. Затем ***doInBackground*** - это метод, который выполняется асинхронно в другом потоке, чем поток Ui. Третий метод - ***onProgressUpdate,*** который также может работать в потоке пользовательского интерфейса, который может уведомить пользователя о статусе. Последний метод, называемый ***onPostExecute,*** в основном используется для публикации результатов.

Ниже приведен пример использования AsyncTask, возвращающего строку.
Пример 1

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (FloatingActionButton) findViewById(R.id.btn);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                executeAsyncTaskOperation();
            }
        });
    }


      private void executeAsyncTaskOperation() {
        new CustomAsyncTask(this).execute();
    }

    private static class CustomAsyncTask extends AsyncTask<Void, Void, String> {

        private Context context;
        private ProgressDialog progressDialog;

        public CustomAsyncTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            progressDialog = ProgressDialog.show(context, "Please wait...", "Loading data from
web");
        }

        @Override
        protected String doInBackground(Void... params) {
            String object = null;
            try {
                Log.d(CustomAsyncTask.class.getCanonicalName(), "doInBackground");
                Thread.sleep(500);
                //bject = "new object";
            } catch (Exception exc) {
                Log.e(CustomAsyncTask.class.getCanonicalName(), "exception");
                object = null;
            }
            return object;
        }

        @Override
        protected void onPostExecute(String s) {
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
            }
            if (s != null) {
                Toast.makeText(context, "finished successfully!", Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(context, "finished unsuccessfully!", Toast.LENGTH_LONG).show();

            }
        }
```

```
    }
}
```

Пример 2.

Здесь AsyncTask немного отличается, метод execute получает список данных для анализа в фоновом режиме. Результат возврата зависит от этой проверки.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();

                executeAsyncTaskOperation();
            }
        });
    }


    private void executeAsyncTaskOperation() {
        Boolean[] bools = new Boolean[10];
        for (int k = 0; k < 10; k++) {
            if (k % 2 == 0) {
                bools[k] = true;
            } else {
                bools[k] = false;
            }
        }
        new CustomAsyncTask(this).execute(bools);
    }

    private static class CustomAsyncTask extends AsyncTask<Boolean, Void, Integer> {

        private Context context;
        private ProgressDialog progressDialog;

        public CustomAsyncTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            progressDialog = ProgressDialog.show(context, "Please wait...", "Loading data from
web");
        }

        @Override
        protected Integer doInBackground(Boolean... params) {
```

```
            int count = 0;
            try {
                Thread.sleep(1000);
                Log.d(CustomAsyncTask.class.getCanonicalName(), "doInBackground");
                for (Boolean param : params) {
                    if (param) {
                        count++;
                    }
                }
            } catch (Exception exc) {
                Log.e(CustomAsyncTask.class.getCanonicalName(), "exception");
                count = 0;
            }
            return count;
        }

        @Override
        protected void onPostExecute(Integer s) {
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
            }
            if (s != null && s > 0) {
                Toast.makeText(context, "finished loading: " + s + " tasks",
Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(context, "finished unsuccessfully!", Toast.LENGTH_LONG).show();

            }
        }
    }
}
```

Прочитайте Начало работы с андроид-асинтезной онлайн: https://riptutorial.com/ru/android-asynctask/topic/8779/начало-работы-с-андроид-асинтезной

глава 2: Отмена AsyncTask

Вступление

Отмена AsyncTask

Examples

Отмена AsyncTask

В следующем примере, если кто-то нажал кнопку «домой» во время выполнения задачи, задача отменяется. В этом конкретном отмене он должен прерываться при запуске.

```
public class MainActivity extends AppCompatActivity {

   private static AtomicBoolean inWork;
   private CustomAsyncTask asyncTask;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       inWork = new AtomicBoolean(false);
       FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
       fab.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View view) {
               Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                       .setAction("Action", null).show();

               executeAsyncTaskOperation();
           }
       });
   }


   private void executeAsyncTaskOperation() {
       Boolean[] bools = new Boolean[10];
       for (int k = 0; k < 10; k++) {
           if (k % 2 == 0) {
               bools[k] = true;
           } else {
               bools[k] = false;
           }
       }
       asyncTask = new CustomAsyncTask(this);
       asyncTask.execute(bools);
   }

   //pressing the home button while the task is running will trigger the onStop being called.
   @Override
   protected void onStop() {
```

```
        if (asyncTask.getStatus() == AsyncTask.Status.RUNNING) {
            asyncTask.cancel(true);
        }
        super.onStop();
    }

    private static class CustomAsyncTask extends AsyncTask<Boolean, Void, Integer> {

        private Context context;
        private ProgressDialog progressDialog;

        public CustomAsyncTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onCancelled() {
            inWork.set(false);
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
                Log.d(CustomAsyncTask.class.getCanonicalName(), "progressdialog is
dismissed.");
            }
        }

        @Override
        protected void onPreExecute() {
            progressDialog = ProgressDialog.show(context, "Please wait...", "Loading data from
web");
        }

        @Override
        protected Integer doInBackground(Boolean... params) {
            int count = 0;
            inWork.set(true);
            try {
                Thread.sleep(1000);
                Log.d(CustomAsyncTask.class.getCanonicalName(), "doInBackground");
                if (!isCancelled()) {
                    for (Boolean param : params) {
                        if (param) {

                            count++;
                        }
                    }
                } else {
                    Log.d(CustomAsyncTask.class.getCanonicalName(), "doInBackground is
cancelled.");
                }

            } catch (Exception exc) {
                Log.e(CustomAsyncTask.class.getCanonicalName(), "exception");
                count = 0;
            }
            return count;
        }

        @Override
        protected void onPostExecute(Integer s) {
            if (!isCancelled()) {
                inWork.set(false);
```

```
                if (progressDialog != null && progressDialog.isShowing()) {
                    progressDialog.dismiss();
                }
                if (s != null && s > 0) {
                    Toast.makeText(context, "finished loading: " + s + " tasks",
Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(context, "finished unsuccessfully!",
Toast.LENGTH_LONG).show();

                }
            } else {
                Log.d(CustomAsyncTask.class.getCanonicalName(), "onPostExecute is
cancelled.");
            }
        }
    }
}
```

Прочитайте Отмена AsyncTask онлайн: https://riptutorial.com/ru/android-asynctask/topic/8783/отмена-asynctask

кредиты

S. No	Главы	Contributors
1	Начало работы с андроид-асинтезной	Andrei T, Community, rossettistone
2	Отмена AsyncTask	Andrei T