Создать и запустить диалог можно несколькими способами, но всех их объединяет одно: все изменения видимой части пользовательского интерфейса после создания Activity должны происходить в специальном потоке. Многие об этом забывают и потом удивляются, почему не происходит видимых изменений. Простейший вариант это вызов следующей конструкции:  
  
  
  
ваша Активити.run OnUiThread(new Runnable() {  
  
@Override  
public void run() {  
// Изменение видимой части  
}  
});  
  
  
  
или для View:  
  
  
  
вашеВию.post(new Runnable() {  
  
@Override  
public void run() {  
// Изменение видимой части  
}  
});  
  
  
  
Ваши команды на изменение ставятся в очередь и в определенный момент вызываются Activity. Преимущество есть, хотя на первый взгляд не очевидно. Программа перестает тормозить при каждом изменении интерфейса, все перерисовки проходят как бы фоново, программа не подвисает при исполнении каких то вычислительных задач связанных одновременно с рассчетами/загрузкой и с отображение хода процесса. С другой стороны это немного усложняет код.  
  
  
  
Некоторые готовые функции уже запускаются в этом потоке. К таким функциям и относятся вызовы диалогов.  
  
Не используйте создание диалогов напрямую. Для этого есть специальный механизм:  
  
  
  
1. Назначьте каждому диалогу свой номер, например **private static final int DIALOG\_KEY = 1**.  
  
2. Создайте диалоги и назначьте их свойства в перезаписываемом методе **onCreateDialog(int id)**.  
  
3. Вызовите диалог на экран при помощи **showDialog(DIALOG\_KEY)** по его номеру.  
  
4. Удалить ставшим ненужным диалог при помощи **dismissDialog(DIALOG\_KEY)**.

**Вариант 1. Вызов диалога в Activity и прекращение работы диалог при помощи handler.**  
  
  
  
  
В приведенном примере вызывается обычный ListView, ему назначается адаптер. При выборе элемента списка появляется диалог загрузки и после окончания загрузки диалог удаляется. Загрузка производится в отдельном потоке для того чтобы ваша activity не зависала на время исполнения загрузки! После окончания загрузки вызывается перехватчик handler для того, чтобы отреагировать и удалить ставшим ненужным диалог загрузки.  
  
  
  
public class AlphabetView extends Activity {  
  
private static final int DIALOG\_LOAD\_KEY = 1;  
private Activity context;  
private Alphabet alphabet;  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
setContentView(R.layout.alphabet);  
context = this;  
alphabet = Main.getAlphabet();  
ListView lv = (ListView) findViewById(R.id.ListViewMain);  
lv.setAdapter(new ArrayAdapter<String>(this, R.layout.item, alphabet.getNames()));  
  
// Назначаем обработчик нажатий на элементе списка  
  
lv.setOnItemClickListener(new OnItemClickListener() {  
  
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
final int i = position;  
showDialog(DIALOG\_LOAD\_KEY); // Показываем диалог  
// Запускаем в отдельном потоке загрузку ваших данных  
new Thread(new Runnable() {  
  
public void run() {  
Main.loadData(i); // Вызов вашей функции загрузки  
handler.sendEmptyMessage(0); // посылаем уведомление об окончании загрузки  
}  
}).start();  
}  
});  
}  
  
// Это ваш обработчик удаления диалога и к примеру запуск новой Activity  
  
private Handler handler = new Handler() {  
@Override  
public void handleMessage(Message msg) {  
dismissDialog(DIALOG\_LOAD\_KEY); // удаляем диалог  
Intent intent = new Intent(context, AuthorsView.class);  
startActivity(intent);  
}  
};  
  
@Override // Здесь вы создаете все диалоги  
protected Dialog onCreateDialog(int id) {  
switch (id) {  
case DIALOG\_LOAD\_KEY: {  
ProgressDialog dialog = new ProgressDialog(this);  
dialog.setMessage("Загрузка, подождите пожалуйста...");  
dialog.setCancelable(true);  
return dialog;  
}  
}  
return super.onCreateDialog(id);  
}  
  
}  
  
  
  
  
**Вариант 2. Вызов диалога в Activity и прекращение работы диалог без handler.**  
  
  
  
  
В данном варианте мы не используем handler. Вместо него мы после создания диалога запускаем в отдельном потоке загрузку, а после окончания загрузки удаляем диалог при помощи вышеописанной **runOnUiThread**.  
  
  
  
public class AlphabetView extends Activity {  
  
private static final int DIALOG\_LOAD\_KEY = 1;  
private Activity context;  
private Alphabet alphabet;  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
setContentView(R.layout.alphabet);  
context = this;  
alphabet = Main.getAlphabet();  
ListView lv = (ListView) findViewById(R.id.ListViewMain);  
lv.setAdapter(new ArrayAdapter<String>(this, R.layout.item, alphabet.getNames()));  
  
// Назначаем обработчик нажатий на элементе списка  
  
lv.setOnItemClickListener(new OnItemClickListener() {  
  
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
final int i = position;  
showDialog(DIALOG\_LOAD\_KEY); // Показываем диалог  
}  
});  
}  
  
@Override // Здесь вы создаете все диалоги  
protected Dialog onCreateDialog(int id) {  
switch (id) {  
case DIALOG\_LOAD\_KEY: {  
final ProgressDialog dialog = new ProgressDialog(this);  
dialog.setMessage("Загрузка, подождите пожалуйста...");  
dialog.setCancelable(true);  
new Thread(new Runnable() {  
  
@Override  
public void run() {  
  
Main.loadData(); // Вызов вашей функции загрузки  
// Удаление диалога после загрузки  
runOnUiThread(new Runnable() {  
  
@Override  
public void run() {  
dismissDialog(DIALOG\_LOAD\_DATA);  
}  
  
});  
  
}  
  
}).start();  
return progressDialog;  
}  
}  
return super.onCreateDialog(id);  
}  
  
}  
  
  
  
Возможны также комбинации этих двух вариантов. К примеру в первом варианте возможно вместо hadler также использовать приведенный во втором варианте код:  
  
  
  
Main.loadData(); // Вызов вашей функции загрузки  
// Удаление диалога после загрузки  
runOnUiThread(new Runnable() {  
  
@Override  
public void run() {  
dismissDialog(DIALOG\_LOAD\_DATA);  
}  
  
});  
  
  
  
  
  
  
**Вариант 3. Вызов диалога в Activity при помощи AsyncTask.**  
  
  
  
  
Иногда удобнее использовать асинхронную задачу AsyncTask. Подробно это описано на[developer.android.com/resources/articles/painless-threading.html](http://developer.android.com/resources/articles/painless-threading.html).  
  
  
  
К примеру вы по нажатию вызываете новую асинхронную задачу для загрузки вашего файла:  
  
  
  
public void onClick(View v) {  
new DownloadImageTask().execute("http://example.com/image.png");  
}  
  
  
  
DownloadImageTask расширяет класс AsyncTask в котором есть все методы для нормального отображения диалога. Для инициализации диалога используем функцию onCreateDialog приведенную в примере 1.  
  
  
  
private class DownloadImageTask extends AsyncTask<string, void,="" bitmap=""> {  
  
protected Bitmap doInBackground(String... urls) {  
return loadImageFromNetwork(urls[0]); // ваша функция загрузки  
}  
  
protected void onPreExecute() {  
showDialog(DIALOG\_LOAD\_KEY); // Показываем диалог  
}  
  
protected void onProgressUpdate() {  
// Здесь мы можем обрабатывать ход прогресса загрузки.  
}   
  
protected void onPostExecute(Bitmap result) {  
dismissDialog(DIALOG\_LOAD\_DATA); // удаляем диалог  
}  
  
}

Работа с прогресс диалогами

Создать и запустить диалог можно несколькими способами, но всех их объединяет одно: все изменения видимой части пользовательского интерфейса после создания Activity должны происходить в специальном потоке. Многие об этом забывают и потом удивляются, почему не происходит видимых изменений. Простейший вариант это вызов следующей конструкции:

вашаАктивити.runOnUiThread(**new** Runnable() {

**@Override**

**public** **void** **run**() {

*// Изменение видимой части*

}

});

или для View:

вашеВию.post(**new** Runnable() {

**@Override**

**public** **void** **run**() {

*// Изменение видимой части*

}

});

Ваши команды на изменение ставятся в очередь и в определенный момент вызываются Activity. Преимущество есть, хотя на первый взгляд не очевидно. Программа перестает тормозить при каждом изменении интерфейса, все перерисовки проходят как бы фоново, программа не подвисает при исполнении каких то вычислительных задач связанных одновременно с рассчетами/загрузкой и с отображение хода процесса. С другой стороны это немного усложняет код.  
  
Некоторые готовые функции уже запускаются в этом потоке. К таким функциям и относятся вызовы диалогов.  
Не используйте создание диалогов напрямую. Для этого есть специальный механизм:  
  
1. Назначьте каждому диалогу свой номер, например **private static final int DIALOG\_KEY = 1**.  
2. Создайте диалоги и назначьте их свойства в перезаписываемом методе **onCreateDialog(int id)**.  
3. Вызовите диалог на экран при помощи **showDialog(DIALOG\_KEY)** по его номеру.  
4. Удалить ставшим ненужным диалог при помощи **dismissDialog(DIALOG\_KEY)**.

Вариант 1. Вызов диалога в Activity и прекращение работы диалог при помощи handler.

В приведенном примере вызывается обычный ListView, ему назначается адаптер. При выборе элемента списка появляется диалог загрузки и после окончания загрузки диалог удаляется. Загрузка производится в отдельном потоке для того чтобы ваша activity не зависала на время исполнения загрузки! После окончания загрузки вызывается перехватчик handler для того, чтобы отреагировать и удалить ставшим ненужным диалог загрузки.

**public** **class** **AlphabetView** **extends** **Activity** {

**private** **static** **final** **int** DIALOG\_LOAD\_KEY = 1;

**private** Activity context;

**private** Alphabet alphabet;

**@Override**

**public** **void** **onCreate**(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.alphabet);

context = **this**;

alphabet = Main.getAlphabet();

ListView lv = (ListView) findViewById(R.id.ListViewMain);

lv.setAdapter(**new** ArrayAdapter<String>(**this**, R.layout.item, alphabet.getNames()));

*// Назначаем обработчик нажатий на элементе списка*

lv.setOnItemClickListener(**new** OnItemClickListener() {

**public** **void** **onItemClick**(AdapterView<?> parent, View view, **int** position, **long** id) {

**final** **int** i = position;

showDialog(DIALOG\_LOAD\_KEY); *// Показываем диалог*

*// Запускаем в отдельном потоке загрузку ваших данных*

**new** Thread(**new** Runnable() {

**public** **void** **run**() {

Main.loadData(i); *// Вызов вашей функции загрузки*

handler.sendEmptyMessage(0); *// посылаем уведомление об окончании загрузки*

}

}).start();

}

});

}

*// Это ваш обработчик удаления диалога и к примеру запуск новой Activity*

**private** Handler handler = **new** Handler() {

**@Override**

**public** **void** **handleMessage**(Message msg) {

dismissDialog(DIALOG\_LOAD\_KEY); *// удаляем диалог*

Intent intent = **new** Intent(context, AuthorsView.class);

startActivity(intent);

}

};

**@Override** *// Здесь вы создаете все диалоги*

**protected** Dialog **onCreateDialog**(**int** id) {

**switch** (id) {

**case** DIALOG\_LOAD\_KEY: {

ProgressDialog dialog = **new** ProgressDialog(**this**);

dialog.setMessage("Загрузка, подождите пожалуйста...");

dialog.setCancelable(**true**);

**return** dialog;

}

}

**return** **super**.onCreateDialog(id);

}

}

Вариант 2. Вызов диалога в Activity и прекращение работы диалог без handler.

В данном варианте мы не используем handler. Вместо него мы после создания диалога запускаем в отдельном потоке загрузку, а после окончания загрузки удаляем диалог при помощи вышеописанной **runOnUiThread**.

**public** **class** **AlphabetView** **extends** **Activity** {

**private** **static** **final** **int** DIALOG\_LOAD\_KEY = 1;

**private** Activity context;

**private** Alphabet alphabet;

**@Override**

**public** **void** **onCreate**(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.alphabet);

context = **this**;

alphabet = Main.getAlphabet();

ListView lv = (ListView) findViewById(R.id.ListViewMain);

lv.setAdapter(**new** ArrayAdapter<String>(**this**, R.layout.item, alphabet.getNames()));

*// Назначаем обработчик нажатий на элементе списка*

lv.setOnItemClickListener(**new** OnItemClickListener() {

**public** **void** **onItemClick**(AdapterView<?> parent, View view, **int** position, **long** id) {

**final** **int** i = position;

showDialog(DIALOG\_LOAD\_KEY); *// Показываем диалог*

}

});

}

**@Override** *// Здесь вы создаете все диалоги*

**protected** Dialog **onCreateDialog**(**int** id) {

**switch** (id) {

**case** DIALOG\_LOAD\_KEY: {

**final** ProgressDialog dialog = **new** ProgressDialog(**this**);

dialog.setMessage("Загрузка, подождите пожалуйста...");

dialog.setCancelable(**true**);

**new** Thread(**new** Runnable() {

**@Override**

**public** **void** **run**() {

Main.loadData(); *// Вызов вашей функции загрузки*

*// Удаление диалога после загрузки*

runOnUiThread(**new** Runnable() {

**@Override**

**public** **void** **run**() {

dismissDialog(DIALOG\_LOAD\_DATA);

}

});

}

}).start();

**return** progressDialog;

}

}

**return** **super**.onCreateDialog(id);

}

}

Возможны также комбинации этих двух вариантов. К примеру в первом варианте возможно вместо hadler также использовать приведенный во втором варианте код:

Main.loadData(); *// Вызов вашей функции загрузки*

*// Удаление диалога после загрузки*

runOnUiThread(**new** Runnable() {

**@Override**

**public** **void** **run**() {

dismissDialog(DIALOG\_LOAD\_DATA);

}

});

Вариант 3. Вызов диалога в Activity при помощи AsyncTask.

Иногда удобнее использовать асинхронную задачу AsyncTask. Подробно это описано на[developer.android.com/resources/articles/painless-threading.html](http://developer.android.com/resources/articles/painless-threading.html).  
  
К примеру вы по нажатию вызываете новую асинхронную задачу для загрузки вашего файла:

**public** **void** **onClick**(View v) {

**new** DownloadImageTask().execute("http://example.com/image.png");

}

DownloadImageTask расширяет класс AsyncTask в котором есть все методы для нормального отображения диалога. Для инициализации диалога используем функцию onCreateDialog приведенную в примере 1.  
  
**private** **class** **DownloadImageTask** **extends** **AsyncTask**<**string**, **void**,="" bitmap=""> {

**protected** Bitmap **doInBackground**(String... urls) {

**return** loadImageFromNetwork(urls[0]); *// ваша функция загрузки*

}

**protected** **void** **onPreExecute**() {

showDialog(DIALOG\_LOAD\_KEY); *// Показываем диалог*

}

**protected** **void** **onProgressUpdate**() {

*// Здесь мы можем обрабатывать ход прогресса загрузки.*

}

**protected** **void** **onPostExecute**(Bitmap result) {

dismissDialog(DIALOG\_LOAD\_DATA); *// удаляем диалог*

}

}