Introdução ao Framework JPDroid

Conhecendo os recursos do framework JPDroid

Rafael Centenaro
rafael_centenaro@hotmail.com
Bacharel em Sistemas de Informação pela Faculdade Uniguaçu– FAESI de São Miguel do Iguaçu – PR. Atua como desenvolvedor de sistemas na Sysmo. Aluno de Especialização em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis.

Introdução ao framework JPDroid:

O Jpdroid (Java Persistence For Android) é um framework ORM para Android que propõe facilitar o desenvolvimento de aplicações que necessitem persistir objetos no banco de dados SQLite. (http://www.sqlite.org/)

Em que situação o tema é útil:

Este artigo é útil para programadores que desejam trabalhar com persistência de dados no Android utilizando um framework.

O Jpdroid é um framework de mapeamento objeto relacional que oferece um mecanismo de persistência simples e transparente para o desenvolvedor. Também disponibiliza um conjunto de anotações para o mapeamento dos objetos e recursos para importação e exportação de dados.

A utilização do *framework* Jpdroid pode colaborar com o desenvolvedor dando uma maior produtividade e qualidade, visto que, as funcionalidades oferecidas dispensam a utilização de comandos DML (Data Manipulation Language).

Para proteger os direitos autorais e garantir que a distribuição continue livre o JPDroid é distribuído sob licença GPL v3. Esta licença exige que ao distribuir cópias, as mesmas liberdades sejam mantidas. Deve-se garantir também a distribuição dos fontes ou como adquiri-los. Os termos desta mesma licença devem acompanhar o código fonte, para que todos conheçam seus direitos. De uma forma resumida a licença concede permissão legal para copiar, distribuir e/ou modificar.

Repositório JPDroid

https://github.com/RafaelCentenaro/jpdroid

Anotações

As anotações presentes no *framework* permitem realizar o mapeamento objeto relacional de forma semelhante a implementações existentes da especificação JPA (*Java Persistence API*), porém foram desenvolvidas de forma a atender as necessidades do framework.

A seguir será apresentada uma lista com as anotações presentes no *framework* e uma breve descrição:

a) @Entity: Esta é a primeira anotação a ser definida, pois ela é responsável por identificar o mapeamento da tabela na aplicação.

b) @PrimaryKey: Toda classe mapeada como tabela deve possuir obrigatoriamente um atributo como sendo chave-primária, e esta anotação é responsável por identificar o atributo como chave-primária.

c) @ForeignKey: Quando existir uma associação entre classes que caracterize uma chave estrangeira no banco o atributo deve receber a anotação @Foreignkey, esta anotação será responsável por realizar este mapeamento.

d) @Column: Todo atributo da classe mapeada que corresponda a uma coluna no banco de dados deverá ser identificada por esta anotação.

e) @RelationClass: Identifica a associação entre classes, a presença desta configuração permite ao *framework* realizar operações em cascata, como recuperação, inclusão, exclusão e atualização de registros.

f) @ViewColumn: Havendo a necessidade de apresentar algum dado referente a outra tabela relacionada, o *framework* permite apresenta-lo desde que exista associação correspondente a este atributo identificado pela anotação @ForeignKey.

g) @Ignorable: Identifica atributo para não ser exportado para arquivos.

h) @Dto: Identifica classe como objeto de transferência de dados. Quando necessário converter a classe do modelo de dados em um objeto de transferência, este novo objeto deve ser anotado como DTO (*Data Transfer Object*), esta anotação será utilizada pelo método convert() da classe JpdroidDtoConverter.

i) @DtoField: Identifica atributo da classe Dto como coluna correspondente ao modelo mapeado.

j) @DefaultOrder: Define ordenação padrão dos objetos pelo atributo anotado, esta anotação será utilizada ao recuperar objetos através do método retrieve().

Tipos de Dados

O banco de dados SQLite possui poucos tipos de dados (NULL, INTEGER, REAL, TEXT e BLOB). Para estender a utilização de outros tipos foi realizado um mapeamento dos tipos da linguagem Java para os tipos do banco SQLite, de acordo com a tabela a seguir o *framework* realiza as conversões necessárias de forma transparente para o desenvolvedor:

Tipos de Dados JAVA x SQLite

Java	SQLite
String	TEXT
Boolean	TEXT
Date	TEXT
Calendar	TEXT
Double	REAL
Float	REAL
Integer	INTEGER
Long	INTEGER
Short	INTEGER
Byte[ ]	BLOB
Bitmap	BLOB

Parametrização

Com as entidades mapeadas em algum ponto da inicialização do aplicativo o *framework* de persistência deve ser parametrizado e inicializado. Aconselha-se fazer isso nas *Activities* que fazem uso do banco de dados, preferencialmente no seu método onCreate().

```
1. Jpdroid dataBase = Jpdroid.getInstance();
2. dataBase.setContext(this);
3. dataBase.addEntity(Endereco.class);
4. dataBase.addEntity(Pessoa.class);
5. dataBase.addEntity(Contato.class);
6. dataBase.open();
```

Persistência

A persistência de objetos ocorre através do método persist(), este método é responsável por atualizar e inserir registros.

```
1. Jpdroid dataBase = Jpdroid.getInstance();
2. Pessoa pessoa = new Pessoa();
3. pessoa.setNome(etNome.getText().toString());
4. pessoa.setFoto(loadBitmapFromView(ivFoto));
5. pessoa.setContato(contato);
6. pessoa.setEndereco(endereco);
7. dataBase.persist(pessoa);
```

Consulta Nativa

A recuperação de dados pode ocorrer por meio de consultas SQL, esta funcionalidade está disponível através dos métodos createQuery(), query() ou rawQuery(), onde o retorno será um Cursor.

```
1. Jpdroid dataBase = Jpdroid.getInstance();
2. Cursor retorno1 = dataBase.rawQuery("SELECT * FROM CIDADE”,null);
3. Cursor retorno2 = dataBase.createQuery(Cidade.class,”nome = ‘Barracao’”);
4. Cursor retorno3 = dataBase.query("CIDADE", new String[] { "CIDADE._ID", "CIDADE.NOME" },
"CIDADE._ID=?",new String[] { String.valueOf(1) }, null, null, null, null);
```

Recuperação de Objetos

Para recuperação de objetos o *framework* possui o método retrieve(), este método permite recuperar objetos em cascata, por exemplo a entidade “Pessoa”, possui as entidades “Contato” e “Endereco” como classes relacionadas, ao invocar o método passando o valor “true” para o parâmetro “fillRelationClass” estes objetos serão carregados automaticamente. Também é possível informar uma restrição para recuperar objetos.

```
5. Jpdroid dataBase = Jpdroid.getInstance();
6. List<Pessoa> listaPessoa = dataBase.retrieve(Pessoa.class,true);
7. List<Pessoa> listaRestrita  = dataBase.retrieve(Pessoa.class,”_id = 1”, true);
```

Exclusão de Objetos

A exclusão de objetos ocorre através dos métodos delete() e deleteAll(). Uma exclusão pode ser seletiva, basta passar os parâmetros de restrição.

```
1. Jpdroid dataBase = Jpdroid.getInstance();
2. dataBase.delete(Pessoa.class, "_id = 1");
3. dataBase.deleteAll(Pessoa.class,);
```

Importação de Dados

Ao iniciar o aplicativo pela primeira vez o banco de dados pode necessitar de uma carga inicial de dados, como por exemplo, cidades e estados. O Jpdroid possui um método chamado importSqlScript(), este método é capaz de importar e executar scripts SQL, estes arquivos podem estar armazenados na pasta Assets do projeto Android ou no cartão de memória do dispositivo.

```
1. if(dataBase.isCreate()){
2.    dataBase.importSqlScript(ScriptPath.Assets, "import.sql");
3. }
```

Exportação de Dados

A exportação de dados em arquivos pode ocorrer em três formatos disponíveis: Xml, Json e Csv.

```
1. JpdroidCsvFile.export(dataBase.retrieve(Pessoa.class, true), " PessoaExport.csv");
2. JpdroidXmlFile.export(dataBase.retrieve(Pessoa.class, true), " PessoaExport.xml");
3. JpdroidJsonFile.export(dataBase.retrieve(Pessoa.class, true)," PessoaExport.json");
```

Conversão de Objetos

A simples conversão de objetos sem a escrita de arquivos pode ser feita através da classe JpdroidConverter.

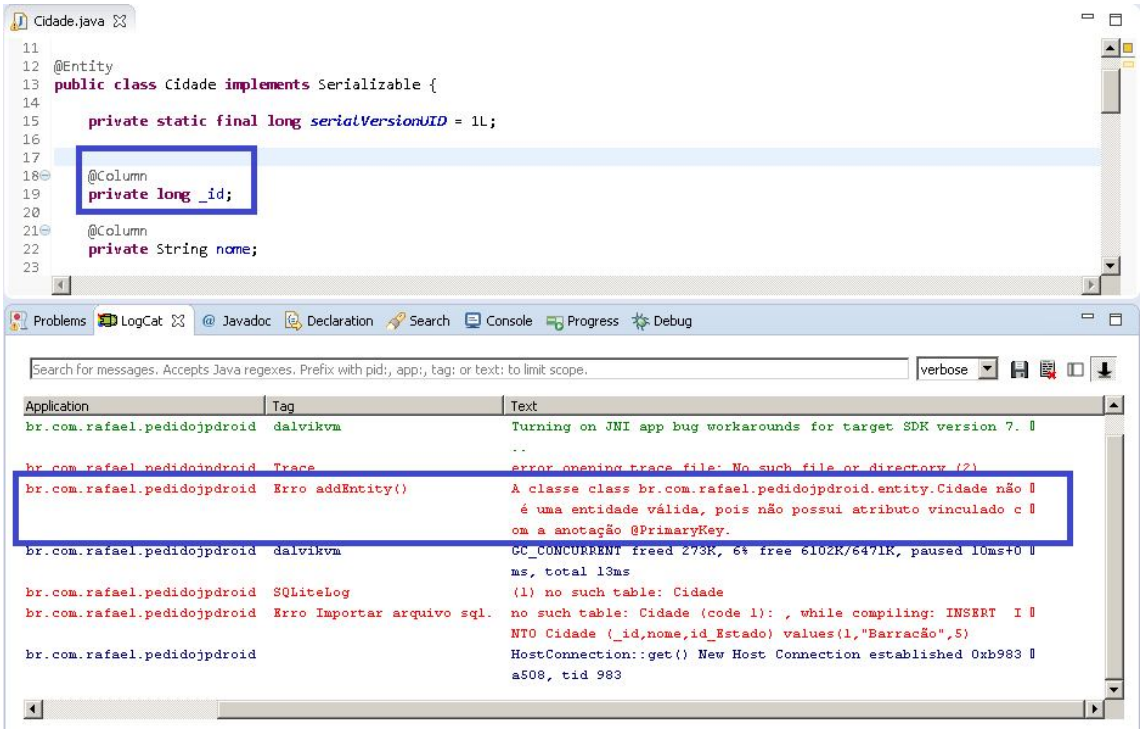
```
1. JSONArray json = JpdroidConverter.toJson(dataBase.retrieve(Pessoa.class));
2. String xml = JpdroidConverter.toXml(dataBase.retrieve(Pessoa.class));
3. String csv = JpdroidConverter.toCsv(dataBase.retrieve(Pessoa.class));
```

Padrões

É importante resaltar que o framework possui algumas regras, como por exemplo, toda chave, seja ela primária ou estrangeira deve ser do tipo primitivo long.

Toda entidade deve possuir obrigatoriamente um atributo "_id" do tipo long e anotado com @Column e @PrimaryKey.

Quando ocorrer algum erro de configuração a classe JpdroidEntityValidation irá disparar exceções que poderão ser consultadas no "Log Cat", conforme mostrado na figura abaixo:



Para exemplificar a validação realizada pelo *framework*, foi removida a anotação @PrimaryKey da entidade Cidade, ao executar aplicativo uma exceção foi lançada, esta pode ser consultada no "Log Cat" do Eclipse, conforme destacado na figura acima.

Criando um projeto Android com JPDroid

Pré-Requisito

Possuir ambiente configurado para desenvolvimento Android com Eclipse.

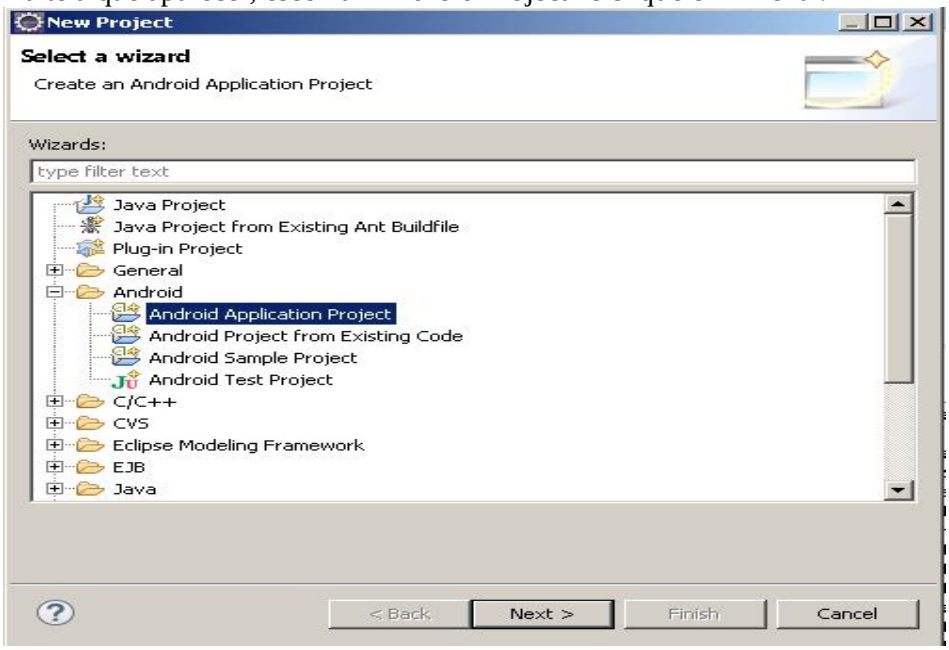
Eclipse - http://www.eclipse.org/downloads/
ADT - http://developer.android.com/tools/sdk/eclipse-adt.html
SDK - http://developer.android.com/sdk/index.html

Criando Projeto

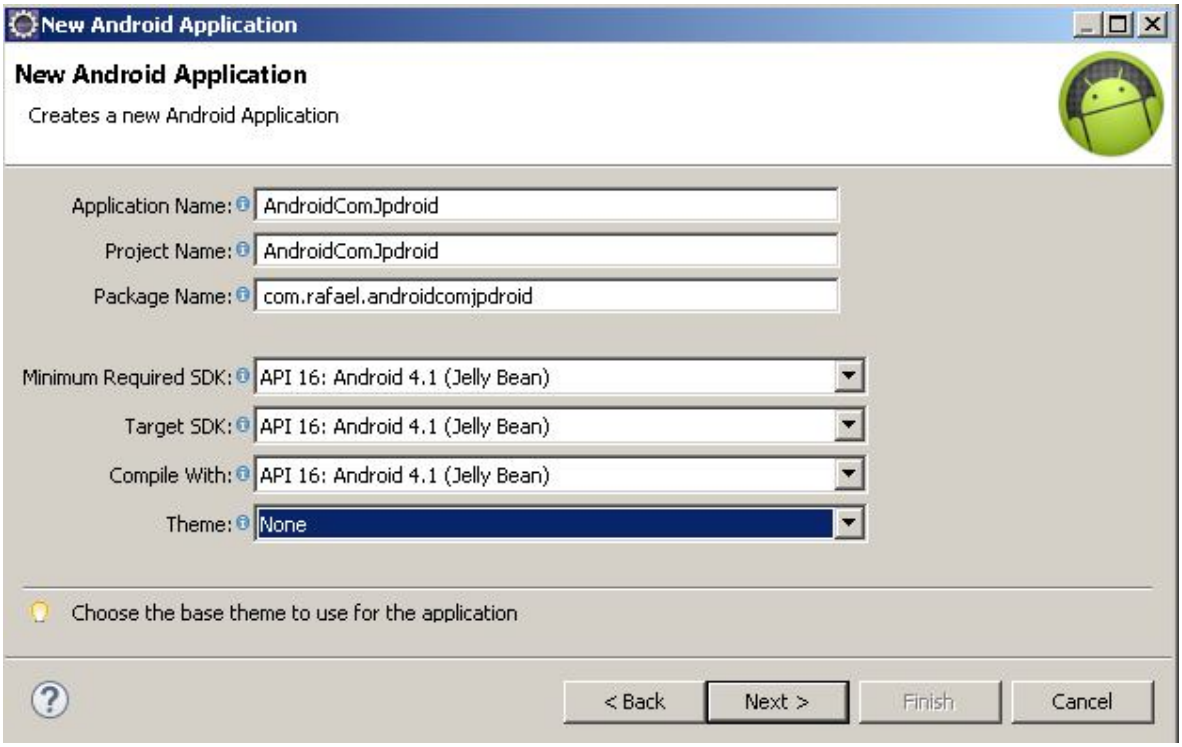
Abra o Eclipse, e crie um novo projeto Android.

File>New>Project

Na tela que aparecer, escolha “Android Project” e clique em “Next”.



Após isso, irá aparecer a tela com as configurações de seu projeto android.



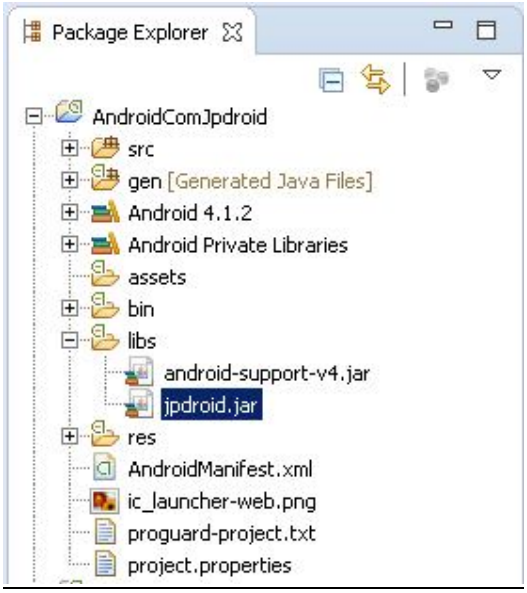
Configure de acordo com sua preferencia ou mantenha o valor padrao das propriedades até o final da configuração.

Com o projeto criado, para iniciar o desenvolvimento utilizando o *framework* Jpdroid é necessário baixar a biblioteca "jpdroid.jar" e adiciona-la ao "*Build Path"* do projeto.

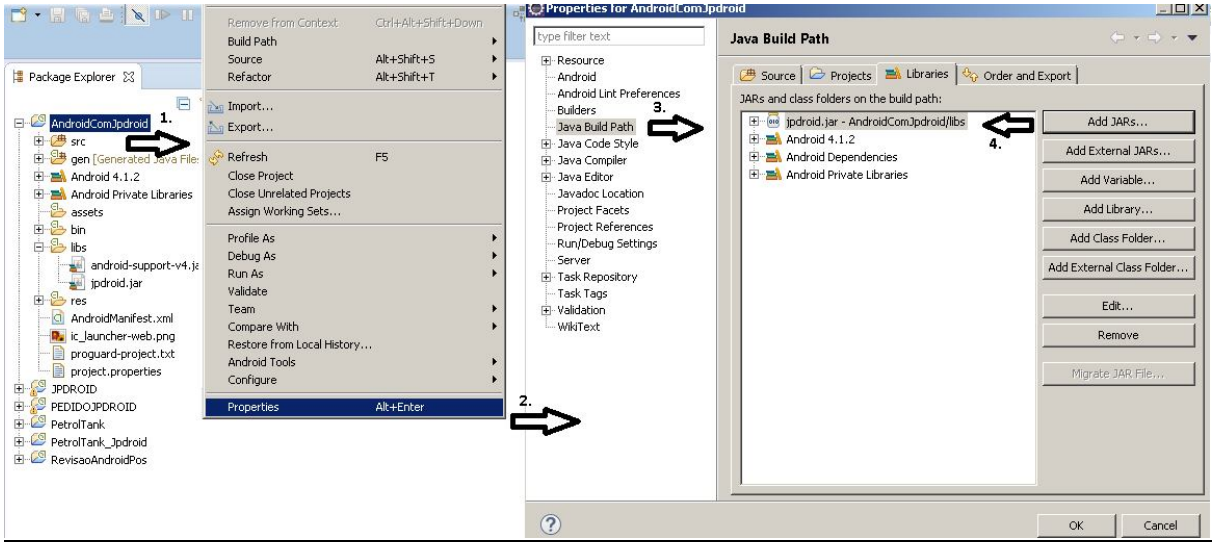
Associando biblioteca ao Projeto

Para vincular a biblioteca ao projeto, acesse o repositório do framework (https://github.com/RafaelCentenaro/jpdroid), navegue até o diretório "JPDROID_Jar" localize a biblioteca "jpdroid.jar"e efetue o download.

Para adicionar a biblioteca ao “Build Path”, primeiro é necessário colar o jar na pasta libs, conforme figura abaixo:



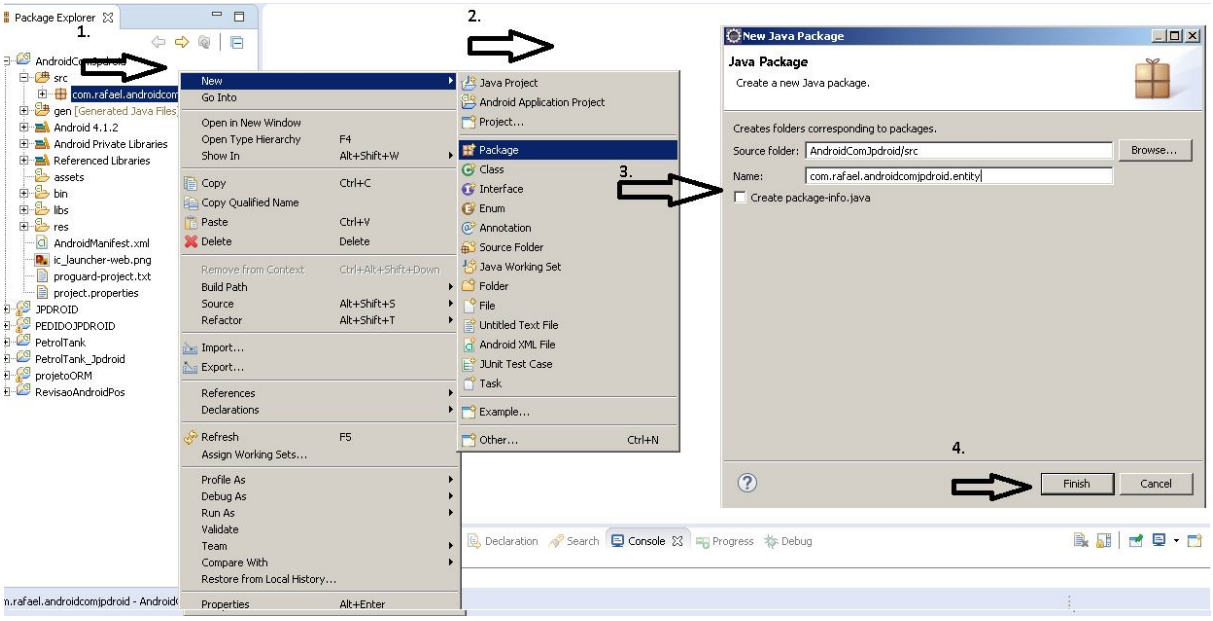
Após colar a biblioteca na pasta libs, a mesma deverá ser vinculada como uma biblioteca do projeto, conforme mostra a figura:



1.Botão Direito Mouse Sobre o Projeto
2.Clique na opção “Properties”
3.Localize a guia “Java Build Path”
4.Na aba “Libraries” clique no botão “Add JARs” localize o projeto, abra a pasta “libs” e selecione a biblioteca “jpdroid.jar”.

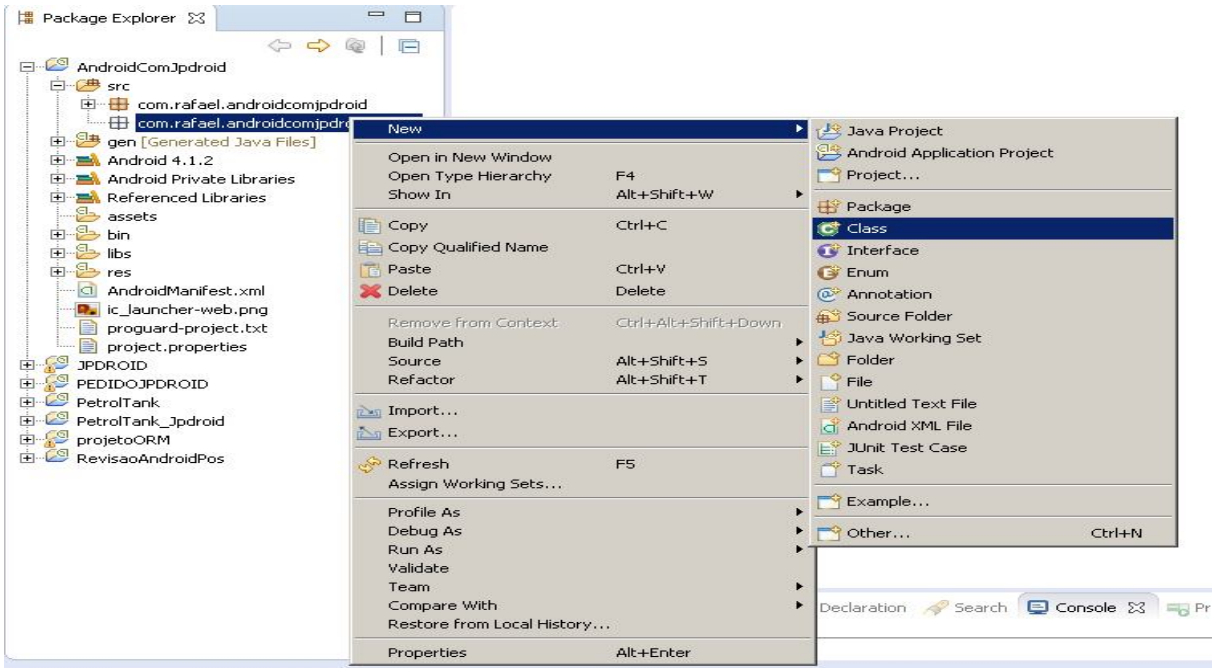
Pacotes

Para organizar nosso código fonte vamos criar um pacote para armazenar as classes mapeadas:



1. Clique com botão direito sobre o pacote principal.
2. Escolha a opção New
3. Clique na opção Package
4. Na tela que surgir, no campo “Name” informe o nome do pacote como “entity” e pressione o botão “Finish”.

Criar Classes



No pacote com.rafael.androidcomjpdroid.entity crie três classes: Pessoa, Contato e TipoContato.

Estas classes deverão ser iguais ao modelo a seguir:

Tipo Contato:

```
@Entity
public class TipoContato implements Serializable{

	private static final long serialVersionUID = 1L;

	@PrimaryKey
	@Column
	private long _id;

	@Column
	private String descricao;

	public long get_id() {
		return _id;
	}

	public void set_id(long _id) {
		this._id = _id;
	}

	public String getDescricao() {
		return descricao;
	}

	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}
}
```

Contato:

```
@Entity
public class Contato implements Serializable {

	private static final long serialVersionUID = 1L;

	@PrimaryKey
	@Column
	private long _id;

	@ForeignKey(joinEntity=TipoContato.class,joinPrimaryKey="_id")
	@Column
	private long idTipoContato;

	@Ignorable
	@ViewColumn(entity=TipoContato.class,foreignKey="idTipoContato", atributo =
"descricao")
	private String nomeTipoContato;

	@ForeignKey(joinEntity=Pessoa.class,joinPrimaryKey="_id",deleteCascade=true)
	@Column
	private long idPessoa;
```

```
	@RelationClass(relationType=RelationType.OneToMany, joinColumn="idTipoContato")
	private TipoContato tipoContato;

	@Column
	private String contato;

	public long get_id() {
		return _id;
	}

	public void set_id(long _id) {
		this._id = _id;
	}

	public long getIdTipoContato() {
		return idTipoContato;
	}

	public void setIdTipoContato(long idTipoContato) {
		this.idTipoContato = idTipoContato;
	}

	public TipoContato getTipoContato() {
		return tipoContato;
	}

	public void setTipoContato(TipoContato tipoContato) {
		this.tipoContato = tipoContato;
	}

	public String getContato() {
		return contato;
	}

	public void setContato(String contato) {
		this.contato = contato;
	}

	public String getNomeTipoContato() {
		return nomeTipoContato;
	}

	public void setNomeTipoContato(String nomeTipoContato) {
		this.nomeTipoContato = nomeTipoContato;
	}
}
```

Pessoa:

```
@Entity
public class Pessoa {

	@PrimaryKey
	@Column
	private long _id;

	@Column
	private String nome;

	@RelationClass(relationType=RelationType.ManyToOne,joinColumn="idPessoa")
```

```
	private List<Contato> contatos;

	public long get_id() {
		return _id;
	}
	public void set_id(long _id) {
		this._id = _id;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public List<Contato> getContatos() {
		return contatos;
	}
	public void setContatos(List<Contato> contatos) {
		this.contatos = contatos;
	}
}
```

Parametrização:

Na classe MainActivity.class devemos criar uma variável do tipo Jpdroid.

```
public class MainActivity extends Activity {

	Jpdroid database;

	.

	.

	.
}
```

Esta variável irá criar um ponto global de acesso a instancia do framework JPDroid. No método onCreate() adicione as classes mapeadas ao controle do framework.

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

	database = Jpdroid.getInstance();
	database.setContext(this);

	database.addEntity(Pessoa.class);
	database.addEntity(TipoContato.class);
	database.addEntity(TipoContato.class);

	database.open();
}
```

Ao chamar o método open() o framework irá criar o banco de dados com as respectivas tabelas de acordo com o mapeamento realizado.

Altere o layout do activity_main.xml para LinearLayout(Horizontal) e adicione um ListView e altere o id para lvPessoas.

Na classe MainActivity.class declare a variavel do tipo ListView:

```
public class MainActivity extends Activity {

	Jpdroid database;
	ListView lvPessoa;
	.
	.
	.
}
```

No método onCreate() encontre a view criada para o list através do método findViewById (), e atribua a variavel “lvPessoa”. Logo após a parametrização do framework vamos alimentar o ListView:

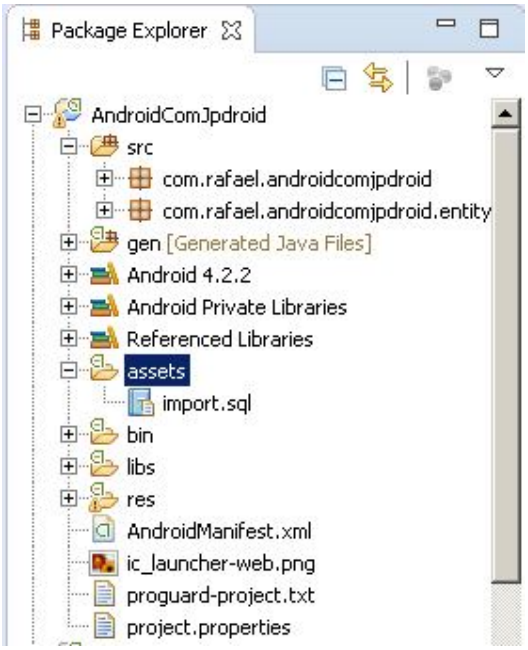
```
protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		lvPessoa = (ListView)findViewById(R.id.lvPessoas);

		database = Jpdroid.getInstance();
		database.setContext(this);
		database.addEntity(Pessoa.class);
		database.addEntity(TipoContato.class);
		database.addEntity(TipoContato.class);
		database.open();

		Cursor cursor = database.createQuery(Pessoa.class);

		SimpleCursorAdapter dataAdapter = new SimpleCursorAdapter(
				this, android.R.layout.simple_list_item_2, cursor,new
String[]{"_id","nome"},
				new int[]{android.R.id.text1,android.R.id.text2}, 0);

		lvPessoa.setAdapter(dataAdapter);


}
```

Importação de Dados

Quando o aplicativo precisar de uma carga incial de dados, recomenda-se fazer isso logo após a parametrização do framework, segue exemplo:

```
if(dataBase.isCreate()){
	dataBase.importSqlScript(ScriptPath.Assets, "import.sql");
}
```

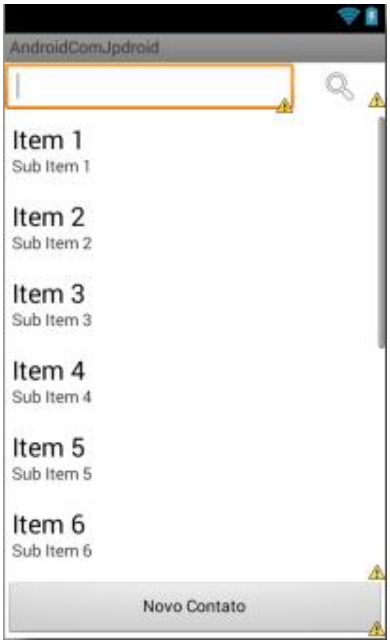
Se o aplicativo estiver sendo inicializado pela primeira vez, o script do arquivo import.sql será executado na base de dados.



Este arquivo deve ser criado na pasta assets do projeto conforme mostra a figura, o script segue abaixo:

```
INSERT  INTO TipoContato (_id,descricao) values(1,"E-mail");
INSERT  INTO TipoContato (_id,descricao) values(2,"Telefone");
INSERT  INTO TipoContato (_id,descricao) values(3,"Celular");
```

Tela Principal



Código-Fonte

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    tools:context="com.rafael.androidcomjpdroid.MainActivity" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="44dp"
        android:orientation="horizontal" >

        <EditText
            android:id="@+id/etPesquisa"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10" >

            <requestFocus />
        </EditText>

        <ImageButton
            android:id="@+id/btPesquisaPessoa "
            android:layout_width="64dp"
            android:layout_height="40dp"
            android:layout_weight="0.42"
            android:background="@android:color/white"
            android:onClick="btnPesquisaOnClick"
            android:src="@android:drawable/ic_menu_search" />

    </LinearLayout>

    <ListView
```

```
        android:id="@+id/lvPessoa"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1" >
    </ListView>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btnNovoContato"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:onClick="btnNovoContatoOnClick"
            android:text="Novo Contato" />

    </LinearLayout>

</LinearLayout>
```

Tela Cadastro Pessoa

Código-Fonte

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.rafael.androidcomjpdroid.PessoaActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/nome" />

    <EditText
        android:id="@+id/etNome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_weight="0.15"
        android:orientation="vertical" >

        <ListView
            android:id="@+id/lvContatos"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >
        </ListView>
    </LinearLayout>

    <LinearLayout
        android:id="@+id/LinearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btnAdd"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.23"
            android:onClick="btnAddOnClick"
            android:text="@string/add" />

        <Button
            android:id="@+id/btnSalvar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right|bottom"
            android:onClick="btnSalvarOnClick"
            android:text="Salvar" />

        <Button
            android:id="@+id/btnCancelar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="btnCancelarOnClick"
            android:text="Cancelar" />
    </LinearLayout>

</LinearLayout>
```

Tela Cadastro Contatos

Código-Fonte

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.rafael.androidcomjpdroid.ContatoActivity" >

    <TextView
        android:id="@+id/tvTipoContato"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/tipo" />

    <Spinner
        android:id="@+id/spTipoContato"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/tvContato"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/contato" />

    <EditText
        android:id="@+id/etContato"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10" >
```

```
        <requestFocus />
    </EditText>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btnSalvarContato"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.31"
            android:onClick="btnSalvarContatoOnClick"
            android:text="Salvar" />

        <Button
            android:id="@+id/btnCancelarContato"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="btnCancelarContatoOnClick"
            android:text="Cancelar" />
    </LinearLayout>

</LinearLayout>
```

MainActivity.class

```
public class MainActivity extends Activity {

	private static final int PESSOA = 1;
	Jpdroid database;
	ListView lvPessoa;
	EditText etPesquisa;

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

		lvPessoa = (ListView) findViewById(R.id.lvPessoa);
		etPesquisa = (EditText) findViewById(R.id.etPesquisa);

		database = Jpdroid.getInstance();
		database.setContext(this);
		database.addEntity(Pessoa.class);
		database.addEntity(Contato.class);
		database.addEntity(TipoContato.class);
		database.open();

		if (database.isCreate()) {
			database.importSqlScript(ScriptPath.Assets, "import.sql");
		}

		adquirirPessoas("");

		registerForContextMenu(lvPessoa);
```

```
	}

	public void btnPesquisaOnClick(View v) {
		adquirirPessoas(etPesquisa.getText().toString());
	}

	private void adquirirPessoas(String filtro) {
		String where = "";
		if (filtro.trim().matches("^[0-9]*$") && filtro.trim().length() > 0) {
			where = "_id = " + filtro;
		} else {
			where = "nome like '%" + filtro + "%'";
		}

		Cursor cursor = database.createQuery(Pessoa.class, where, " _id asc ");

		SimpleCursorAdapter dataAdapter = new SimpleCursorAdapter(this,
				android.R.layout.simple_list_item_2, cursor, new String[] {
						"_id", "nome" }, new int[] { android.R.id.text1,
						android.R.id.text2 }, 0);

		lvPessoa.setAdapter(dataAdapter);

	}

	public void btnNovoContatoOnClick(View v) {
		Intent i = new Intent(this, PessoaActivity.class);
		i.putExtra("posicao", 0);
		startActivityForResult(i, PESSOA);

	}

	private void deletePessoa(final int posicao) {
		AlertDialog.Builder builder = new AlertDialog.Builder(this);
		builder.setMessage("Confirma a exclusão?");
		builder.setPositiveButton("Sim", new DialogInterface.OnClickListener() {
			@Override
			public void onClick(DialogInterface dialog, int which) {
				if (database.delete(Pessoa.class,
						(Cursor) lvPessoa.getItemAtPosition(posicao)) <=
0) {
					Toast.makeText(getBaseContext(),
							"A pessoa não pode ser excluída!",
							Toast.LENGTH_SHORT).show();
				}
				adquirirPessoas(etPesquisa.getText().toString());

				dialog.dismiss();
			}
		});
		builder.setNegativeButton("Não", new DialogInterface.OnClickListener() {
			@Override
			public void onClick(DialogInterface dialog, int which) {
				dialog.dismiss();
			}
		});

		builder.show();

	}
```

```
	public void onActivityResult(int requestCode, int resultCode, Intent data) {
		if (requestCode == PESSOA) {
			if (resultCode == Activity.RESULT_OK && data != null) {
				adquirirPessoas("");
			}
		}
	}

	@Override
	public void onCreateContextMenu(ContextMenu menu, View v,
			ContextMenuInfo menuInfo) {
		super.onCreateContextMenu(menu, v, menuInfo);
		MenuInflater inflater = getMenuInflater();
		inflater.inflate(R.menu.pessoa, menu);
	}

	@Override
	public boolean onContextItemSelected(MenuItem item) {
		AdapterContextMenuInfo info = (AdapterContextMenuInfo) item
				.getMenuInfo();
		switch (item.getItemId()) {
		case R.id.excluirPessoa:
			deletePessoa(info.position);
			break;
		case R.id.editarPessoa:
			Intent i = new Intent(this, PessoaActivity.class);
			Cursor cursor = (Cursor) lvPessoa.getItemAtPosition(info.position);
			long id = cursor.getLong(cursor.getColumnIndex("_id"));
			i.putExtra("_id", id);
			startActivity(i);
			break;
		default:
			return super.onContextItemSelected(item);
		}
		return super.onContextItemSelected(item);
	}
}
```

PessoaActivity.class

```
public class PessoaActivity extends Activity {

	private static final int ADD_CONTATO = 1;
	private Jpdroid database;
	private ListView lvContatos;
	private EditText etNome;
	private Long _id;
	private Pessoa pessoa;
	private List<Contato> contatos = new ArrayList<Contato>();

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

		database = Jpdroid.getInstance();
		lvContatos = (ListView) findViewById(R.id.lvContatos);
		etNome = (EditText) findViewById(R.id.etNome);
```

```
        lvContatos.setOnItemClickListener(evento);

        Intent i = getIntent();
        _id = i.getLongExtra("_id", 0);
        if (_id > 0) {
              pessoa = (Pessoa) database.retrieve(Pessoa.class, "_id = " + _id,
                          true).get(0);
              contatos = pessoa.getContatos();
              etNome.setText(pessoa.getNome());
              fillContato();
        } else {
              pessoa = new Pessoa();
        }
        registerForContextMenu(lvContatos);
}

OnItemClickListener evento = new OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View view, int position,
                    long id) {
              Contato con = contatos.get(position);
              if (con.getIdTipoContato() == 2 || con.getIdTipoContato() == 3) {
                    String uri = "tel:" + con.getContato().trim();
                    Intent intent = new Intent(Intent.ACTION_CALL);
                    intent.setData(Uri.parse(uri));
                    startActivity(intent);
              } else if (con.getIdTipoContato() == 1) {
                    Intent email = new Intent(Intent.ACTION_SEND);
                    email.putExtra(Intent.EXTRA_EMAIL,
                                new String[] { con.getContato() });
                    email.putExtra(Intent.EXTRA_SUBJECT, "Assunto");
                    email.putExtra(Intent.EXTRA_TEXT,
                                "Escreva sua mensagem aqui...");
                    email.setType("message/rfc822");
                    startActivity(Intent.createChooser(email, "Enviar com :"));
              }
        }
};

public void btnAddOnClick(View v) {
        Intent i = new Intent(this, ContatoActivity.class);
        i.putExtra("posicao", 0);
        startActivityForResult(i, ADD_CONTATO);

}

private void fillContato() {

        MatrixCursor matrixCursor = JpdroidConverter.toMatrixCursor(contatos,
                    false);

        SimpleCursorAdapter dataAdapter = new SimpleCursorAdapter(this,
                    android.R.layout.simple_list_item_2, matrixCursor,
                    new String[] { "nomeTipoContato", "contato" }, new int[] {
                                android.R.id.text1, android.R.id.text2 }, 0);

        lvContatos.setAdapter(dataAdapter);
}

public void btnCancelarOnClick(View v) {
        Intent it = new Intent();
        setResult(RESULT_CANCELED, it);
```

```
			finish();
		}

		private void deleteContato(int position) {
			Contato del = contatos.get(position);
			if (del.get_id() > 0) {
				database.delete(del);
			}
			contatos.remove(position);
			fillContato();

		}

		public void btnSalvarOnClick(View v) {

			try {

				if (etNome.getText() == null
						|| etNome.getText().toString().trim().length() == 0) {
					Toast.makeText(this, "Nome não informado!",
Toast.LENGTH_SHORT)
							.show();
					etNome.requestFocus();
					return;
				}
				if (contatos.isEmpty()) {
					Toast.makeText(this, "Favor Cadastrar pelo menos um
contato!",
							Toast.LENGTH_SHORT).show();
					return;
				}

				pessoa.setNome(etNome.getText().toString());
				pessoa.setContatos(contatos);

				database.persist(pessoa);

				Intent it = new Intent();
				it.putExtra("_id", _id);
				setResult(RESULT_OK, it);
				finish();

			} catch (JpdroidException e) {

				e.printStackTrace();
			}
		}

		public void onActivityResult(int requestCode, int resultCode, Intent data) {
			if (requestCode == ADD_CONTATO) {
				if (resultCode == Activity.RESULT_OK && data != null) {
					Contato novo = (Contato) data.getExtras().getSerializable(
							"contato");
					if (novo.get_id() == 0) {
						contatos.add(novo);
					} else {
						contatos.set(data.getIntExtra("posicao", 0), novo);
					}
					fillContato();
				}
			}
```

```
	}

	@Override
	public void onCreateContextMenu(ContextMenu menu, View v,
			ContextMenuInfo menuInfo) {
		super.onCreateContextMenu(menu, v, menuInfo);
		MenuInflater inflater = getMenuInflater();
		inflater.inflate(R.menu.contato, menu);
	}

	@Override
	public boolean onContextItemSelected(MenuItem item) {
		AdapterContextMenuInfo info = (AdapterContextMenuInfo) item
				.getMenuInfo();
		switch (item.getItemId()) {
		case R.id.excluirContato:
			deleteContato(info.position);
			break;
		case R.id.editarContato:
			Intent it = new Intent(this, ContatoActivity.class);

			Bundle bundle = new Bundle();
			bundle.putSerializable("contato",
					(Serializable) contatos.get(info.position));

			it.putExtras(bundle);
			it.putExtra("posicao", info.position);

			startActivityForResult(it, ADD_CONTATO);

			break;
		default:
			return super.onContextItemSelected(item);
		}
		return super.onContextItemSelected(item);
	}

}
```

ContatoActivity.class

```
public class ContatoActivity extends Activity {

	Jpdroid database;
	private Contato contato = null;
	private EditText etContato;
	private static Spinner spTipo;
	private int posicao = 0;

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

		database = Jpdroid.getInstance();

		etContato = (EditText) findViewById(R.id.etContato);
		spTipo = (Spinner) findViewById(R.id.spTipoContato);
		spTipo.setOnItemSelectedListener(evento);
```

```
		adquirirTipoContato();

		Intent it = getIntent();
		Serializable param = it.getExtras().getSerializable("contato");
		if (param != null) {
			posicao = it.getIntExtra("posicao", 0);
			Contato novo = (Contato) param;
			contato = novo;
			etContato.setText(contato.getContato());

	spTipo.setSelection((Long.valueOf(contato.getIdTipoContato()).intValue()-1));

		} else {
			contato = new Contato();
		}
	}

	private void adquirirTipoContato() {
		Cursor matrixCursor = database.createQuery(TipoContato.class);

		SimpleCursorAdapter dataAdapter = new SimpleCursorAdapter(
				this, android.R.layout.simple_list_item_2, matrixCursor,new
String[]{"_id","descricao"},
				new int[]{android.R.id.text1,android.R.id.text2}, 0);

		spTipo.setAdapter(dataAdapter);

	}

	public void btnSalvarContatoOnClick(final View v) {

		contato.setContato(etContato.getText().toString());
		Cursor selectItem = (Cursor) spTipo.getSelectedItem();
		contato.setIdTipoContato(selectItem.getLong(0));
		contato.setNomeTipoContato(selectItem.getString(1));
		Intent it = new Intent();

		Bundle bundle = new Bundle();
		bundle.putSerializable("contato", (Serializable) contato);

		it.putExtras(bundle);
		it.putExtra("posicao", posicao);

		setResult(RESULT_OK, it);
		finish();
	}

	public void btnCancelarContatoOnClick(View v){
		Intent it = new Intent();
		setResult(RESULT_CANCELED, it);
		finish();
	}
	AdapterView.OnItemSelectedListener evento = new
AdapterView.OnItemSelectedListener() {
		public void onItemSelected(AdapterView<?> adapterView, View view, int i,
long l) {
			if (spTipo.getSelectedItem().equals("Celular")) {
				etContato.setInputType(InputType.TYPE_CLASS_PHONE);
			} else if (spTipo.getSelectedItem().equals("E-mail")) {

	etContato.setInputType(InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS);
```

```
                }
            }

            public void onNothingSelected(AdapterView<?> adapterView) {
                return;
            }
        };
}
```

Conclusão

Este artigo apresentou conceitos básicos sobre o desenvolvimento de aplicações Android com o framework JPDroid, estes conceitos poderão ser utilizados para desenvolver diversos tipos de aplicativos que necessitem de um mecanismo de persistência, como por exemplo, sistemas para pesquisa de mercado, coletores de dados, vendas porta á porta, jogos, entre outros.

O projeto “AndroidComJpdroid” esta disponível no repositório:
https://github.com/RafaelCentenaro/jpdroid/tree/master/AndroidComJpdroid