Files Management

Introduction

- The android platform has a files system based on the linux operating system.
- The android platform allows us to access files that were prepackaged with our application and files that our application creates on the handset itself.

Accessing Simple Files

- We can easily create new files stored on the android files system.
- Calling `openFileOutput()` returns a `FileOutputStream` object we can use as if we were developing a standard Java application for our desktop.

```
...
FileOutputStream fos = openFileOutput(
                              “myfile.txt”,
                              Context.MODE_PRIVATE);
...
```

Accessing Simple Files

- We can easily access files already stored in the android files system.
- Calling the `openFileInput` method returns a `FileInputStream` we can use as if we were developing a standard Java application for our desktop.

```
...
FileInputStream fis = openFileInput(“myfile.txt”);
...
```

Accessing Simple Files

- The `openFileInput()` and `openFileOutput()` methods can not accept a filepath (e.g. `path/myfile.txt`). These two methods accept simple file names only.
- Calling the `flush()` method on our FileOutputStream object ensures the data was indeed written to the file.

Accessing Simple Files

```
package com.abelski.samples;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;

public class SimpleFileAccessActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        String txt = null;
        TextView tv = new TextView(this);
        FileOutputStream fos = null;
        FileInputStream fis = null;
```

Accessing Simple Files

```
try
{
    //write to file
    fos = openFileOutput("myfile.txt",Context.MODE_PRIVATE);
    fos.write("hello".getBytes());
    fos.flush();
    fis = openFileInput("myfile.txt");
    byte[] vec = new byte[100];
    fis.read(vec);
    txt = new String(vec);
}
catch(Exception e)
{
    txt = e.getMessage();
}
```

Accessing Simple Files

```
        finally
        {
            if(fos!=null)
            {
                try {fos.close();}
                catch(IOException e) { txt+=e.getMessage(); }
            }
            if(fis!=null)
            {
                try {fis.close();}
                catch(IOException e) {txt+=e.getMessage(); }
            }
            tv.setText(txt);
            setContentView(tv);
        }
    }
}
```

Accessing Simple Files

Accessing Simple Files

- Overriding the `onPause()` and `onResume()` methods we can save the state of our application to a simple file and get it back when the application resumes.

Accessing Simple Files

```
public class SimpleNoteActivity extends Activity
{
    private final static String FILENAME = "mynote.txt";
    private EditText editor;
    private Button bt;

    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        editor=(EditText)findViewById(R.id.text_editor);
        bt=(Button)findViewById(R.id.exit_button);
        bt.setOnClickListener(new Button.OnClickListener()
        {
            public void onClick(View v)
            {
                finish();
            }
        }
    );
}
```

Accessing Simple Files

```
public void onResume()
{
    super.onResume();
    InputStream is = null;
    InputStreamReader isr = null;
    BufferedReader br = null;
    try
    {
        is = openFileInput(FILENAME);
        Log.i("note","input stream was created");
        if (is != null)
        {
            isr = new InputStreamReader(is);
            Log.i("note","input stream reader was created");
            br = new BufferedReader(isr);
            Log.i("note","buffered reader was created");
            StringBuffer sb = new StringBuffer();
            String str = br.readLine();
            Log.i("note","'"+str+"' was read");
            while (str != null)
            {
                sb.append(str + "\n");
                Log.i("note","'"+str+"' was read");
                str = br.readLine();
            }
```

Accessing Simple Files

```
            editor.setText(sb.toString());
        }
    }
    catch (java.io.FileNotFoundException e)
    {
        // most likely the file still was not created
    }
    catch (Throwable t)
    {
        Log.e("note",t.toString());
    }
    finally
    {
        if(is!=null)
        {
            try{is.close();} catch(Exception e)
            {Log.e("note",e.toString());}
        }
        if(isr!=null)
        {
            try{isr.close();} catch(Exception e)
            {Log.e("note",e.toString());}
        }
```

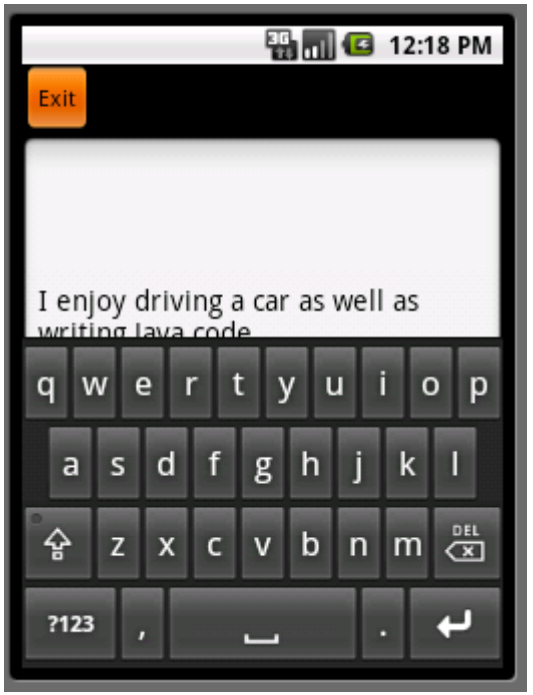
Accessing Simple Files

```
        if(br!=null)
        {
            try{br.close();} catch(Exception e)
            Log.e("note",e.toString());}
        }
    }
}

public void onPause()
{
    super.onPause();
    OutputStream os = null;
    OutputStreamWriter osr = null;
    BufferedWriter bw = null;
    try
    {
        os = openFileOutput(FILENAME,MODE_PRIVATE);
        Log.i("note","output stream was created");
        OutputStreamWriter osw = new OutputStreamWriter(os);
        Log.i("note","output stream writer was");
        bw = new BufferedWriter(osw);
        Log.i("note","buffered writer was created");
        bw.write(editor.getText().toString());
        bw.flush();
    }
```

Accessing Simple Files

```
        catch (Throwable t)
        {
            Log.e("note",t.toString());
        }
        finally
        {
            if(os!=null)
            {
                try{os.close();} catch(Exception e){}
            }
            if(osr!=null)
            {
                try{osr.close();} catch(Exception e){}
            }
            if(bw!=null)
            {
                try{bw.close();} catch(Exception e){}
            }
        }
    }
}
```

Accessing Simple Files



Accessing Raw Resources

- We can easily access raw files part of our application. The raw files should be placed within the `res/raw` folder. Files we place in that folder are not compiled.

```
...
Resources resources = this.getResources();
InputStream is = resources.openRawResource(R.raw.ilove);
...
```

Accessing Raw Resources

```
package com.abelski.samples;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import android.app.Activity;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TextView;

public class RawFilesActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        String txt = null;
        Resources resources = null;
        InputStream is = null;
        InputStreamReader isr = null;
        StringBuilder sb = new StringBuilder();
```

Accessing Raw Resources

```
try
{
    resources = this.getResources();
    is = resources.openRawResource(R.raw.ilove);
    isr = new InputStreamReader(is);
    char[] vec = new char[100];
    while(isr.read(vec)!=-1)
    {
        sb.append(new String(vec));
    }
}
catch(Exception e)
{
    txt = e.getMessage();
}
```

Accessing Raw Resources

```
        finally
        {
            if(is!=null)
            {
                try {is.close();}
                catch(IOException e){txt = e.getMessage();}
            }
            if(isr!=null)
            {
                try {isr.close();}
                catch(IOException e){txt = e.getMessage();}
            }
            tv.setText(sb.toString());
            this.setContentView(tv);
        }
    }
}
```

Accessing Raw Resources

Accessing Raw Resources

- Accessing a raw file we can get an `InputStream` only. There are no means for modifying the file.
- Placing files within the raw directory might be useful for initialization purposes.
- Using the `XML` format we can easily initialize our application with structured data the `XML` document holds.

Accessing Raw Resources

```
public class XMLBasedListActivity extends ListActivity
{
    TextView selectedColor;
    ArrayList<String> listItems = new ArrayList<String>();

    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        InputStream in = null;
        selectedColor = (TextView) findViewById(R.id.selectedcolor);
        try
        {
            in = getResources().openRawResource(R.raw.colors);
            DocumentBuilder builder =
                DocumentBuilderFactory.newInstance()
                .newDocumentBuilder();
            Document doc = builder.parse(in);
            NodeList colors = doc.getElementsByTagName("color");
```

Accessing Raw Resources

```
            for (int i = 0; i < colors.getLength(); i++)
            {
                listItems.add(colors.item(i).
                    getFirstChild().getNodeValue());
            }
        }
        catch (Throwable t)
        {
            Log.e("onCreate",t.getMessage());
        }
        finally
        {
            if(in!=null) {try{in.close();}catch(Exception e){}}
        }
        setListAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, listItems));
}
```

Accessing Raw Resources

```
    public void onListItemClick(ListView parent, View v,
        int position, long id)
    {
        selectedColor.setText(listItems.get(position).toString());
    }
}
```

Accessing Raw Resources

Accessing XML Resources

- We can easily access XML files that were stored within the `res/xml` folder. Unlike the other resources, accessing these files is done differently.
- These files are compiled into an efficient binary form when deployed.

```
...
Resources resources = this.getResources();
XmlPullParser parser = resources.getXml(R.xml.cars);
...
```

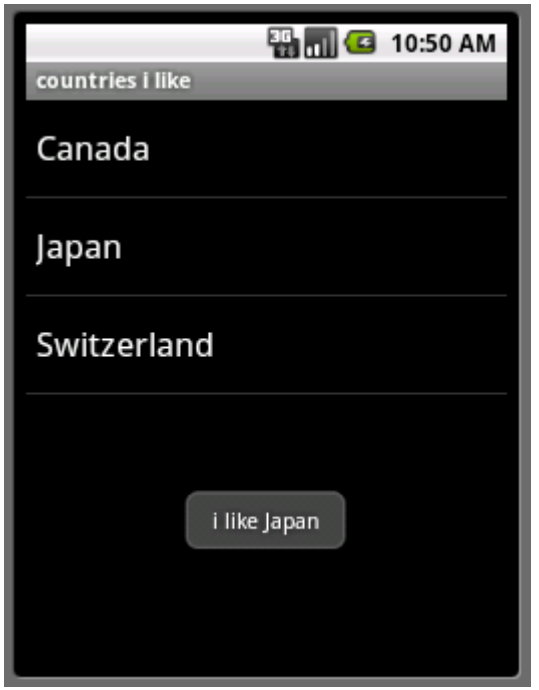
Accessing XML Resources

```
public class SimpleCountriesListActivity extends ListActivity
{
    ArrayList<String> list = new ArrayList<String>();

    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        try
        {
            XmlPullParser parser = getResources().getXml(R.xml.countries);
            int eventType = parser.getEventType();
            while(eventType!=XmlPullParser.END_DOCUMENT)
            {
                if(eventType == XmlPullParser.TEXT)
                {
                    list.add(parser.getText());
                }
                eventType = parser.next();
            }
        }
```

Accessing XML Resources

```
        catch (Exception e)
        {
            Log.i("xmlxml",e.getMessage());
        }
        setListAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, list));
    }

    public void onListItemClick(
        ListView parent, View v, int position, long id)
    {
        String toastText = "i like "+list.get(position);
        Toast toast = Toast.makeText(this,toastText,Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

Accessing XML Resources



Accessing SD Card External Storage

- The android platform allows our code to access the available secure digital flash memory card (SD Card) external storage.

```
...
File cardDir = new File(“/sdcard/”);
FILE file = new File(cardDir,”our_file.txt”);
FileOutputStream fos = new FileOutputStream(file);
...
```

Accessing SD Card External Storage

- The external storage is accessible by all applications, whereas the default behavior of the `openFileInput()` and the `openFileOutput()` methods allows accessing the application private domain only.

Accessing SD Card External Storage

- Starting with android 1.6 there is a need in obtaining the `android.permission.WRITE_EXTERNAL_STORAGE` user permission.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Accessing SD Card External Storage

```
public class SimpleNoteActivity extends Activity
{
    private final static String FILENAME = "mynote.txt";
    private EditText editor;
    private Button bt;

    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        editor=(EditText)findViewById(R.id.text_editor);
        bt=(Button)findViewById(R.id.exit_button);
        bt.setOnClickListener(new Button.OnClickListener()
            {
                public void onClick(View v)
                {
                    finish();
                }
            }
                                                    );
    }
```

Accessing SD Card External Storage

```
public void onResume()
{
    super.onResume();
    InputStream is = null;
    InputStreamReader isr = null;
    BufferedReader br = null;
    try
    {
        File card = Environment.getExternalStorageDirectory();
        File file = new File(card,FILENAME);
        is = new FileInputStream(file);
        if (is != null)
        {
            isr = new InputStreamReader(is);
            br = new BufferedReader(isr);
            StringBuffer sb = new StringBuffer();
            String str = br.readLine();
            while (str != null)
            {
                sb.append(str + "\n");
                str = br.readLine();
            }
            editor.setText(sb.toString());
        }
    }
```

Accessing SD Card External Storage

```
catch (java.io.FileNotFoundException e)
{
    // most likely the file still wasnot created
}
catch (Throwable t)
{
    Log.e("note input",t.toString());
}
finally
{
    if(is!=null)
    {
        try{is.close();} catch(Exception e)
        {Log.e("note input",e.toString());}
    }
    if(isr!=null)
    {
        try{isr.close();} catch(Exception e)
        {Log.e("note input",e.toString());}
    }
```

Accessing SD Card External Storage

```
            if(br!=null)
            {
                try{br.close();} catch(Exception e)
                {Log.e("note input",e.toString());}
            }
        }
    }
```

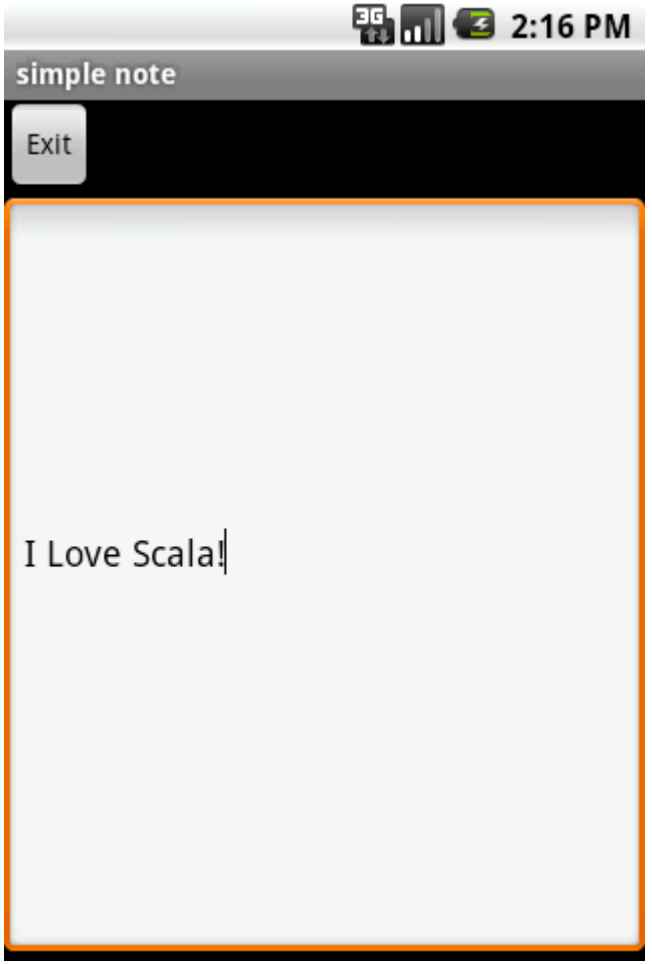
Accessing SD Card External Storage

```
public void onPause()
{
    super.onPause();
    OutputStream os = null;
    OutputStreamWriter osr = null;
    BufferedWriter bw = null;
    try
    {
        File card = Environment.getExternalStorageDirectory();
        File file = new File(card,FILENAME);
        if(!file.exists())
        {
            file.createNewFile();
        }
        os = new FileOutputStream(file);
        os = new FileOutputStream(file);
        OutputStreamWriter osw = new OutputStreamWriter(os);
        bw = new BufferedWriter(osw);
        bw.write(editor.getText().toString());
        bw.flush();
    }
```

Accessing SD Card External Storage

```
        catch (Throwable t)
        {
            Log.e("note output",t.toString());
        }
        finally
        {
            if(os!=null)
            {
                try{os.close();}
                catch(Exception e){}
            }
            if(osr!=null)
            {
                try{osr.close();}
                catch(Exception e){}
            }
            if(bw!=null)
            {
                try{bw.close();}
                catch(Exception e){}
            }
        }
    }
}
```

Accessing SD Card External Storage



Files Management

Introduction

- The android platform has a files system based on the linux operating system.
- The android platform allows us to access files that were prepackaged with our application and files that our application creates on the handset itself.

Accessing Simple Files

- We can easily create new files stored on the android files system.
- Calling `openFileOutput()` returns a `FileOutputStream` object we can use as if we were developing a standard Java application for our desktop.

```
...
FileOutputStream fos = openFileOutput(
                            "myfile.txt",
                            Context.MODE_PRIVATE);
...
```

Accessing Simple Files

- We can easily access files already stored in the android files system.
- Calling the `openFileInput` method returns a `FileInputStream` we can use as if we were developing a standard Java application for our desktop.

```
...
FileInputStream fis = openFileInput("myfile.txt");
...
```

Accessing Simple Files

- The `openFileInput()` and `openFileOutput()` methods can not accept a filepath (e.g. `path/myfile.txt`). These two methods accept simple file names only.
- Calling the `flush()` method on our FileOutputStream object ensures the data was indeed written to the file.

Accessing Simple Files

```
package com.abelski.samples;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;

public class SimpleFileAccessActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        String txt = null;
        TextView tv = new TextView(this);
        FileOutputStream fos = null;
        FileInputStream fis = null;
```

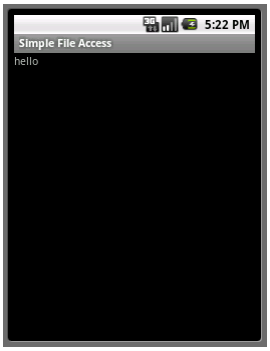
Accessing Simple Files

```
try
{
    //write to file
    fos = openFileOutput("myfile.txt",Context.MODE_PRIVATE);
    fos.write("hello".getBytes());
    fos.flush();
    fis = openFileInput("myfile.txt");
    byte[] vec = new byte[100];
    fis.read(vec);
    txt = new String(vec);
}
catch(Exception e)
{
    txt = e.getMessage();
}
```

Accessing Simple Files

```
            finally
            {
                if(fos!=null)
                {
                    try {fos.close();}
                    catch(IOException e) { txt+=e.getMessage(); }
                }
                if(fis!=null)
                {
                    try {fis.close();}
                    catch(IOException e) {txt+=e.getMessage(); }
                }
                tv.setText(txt);
                setContentView(tv);
            }
        }
    }
```

Accessing Simple Files



Accessing Simple Files

- Overriding the `onPause()` and `onResume()` methods we can save the state of our application to a simple file and get it back when the application resumes.

Accessing Simple Files

```
public class SimpleNoteActivity extends Activity
{
    private final static String FILENAME = "mynote.txt";
    private EditText editor;
    private Button bt;

    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        editor=(EditText)findViewById(R.id.text_editor);
        bt=(Button)findViewById(R.id.exit_button);
        bt.setOnClickListener(new Button.OnClickListener()
        {
            public void onClick(View v)
            {
                finish();
            }
        }
    );
}
```

Accessing Simple Files

```
public void onResume()
{
    super.onResume();
    InputStream is = null;
    InputStreamReader isr = null;
    BufferedReader br = null;
    try
    {
        is = openFileInput(FILENAME);
        Log.i("note","input stream was created");
        if (is != null)
        {
            isr = new InputStreamReader(is);
            Log.i("note","input stream reader was created");
            br = new BufferedReader(isr);
            Log.i("note","buffered reader was created");
            StringBuffer sb = new StringBuffer();
            String str = br.readLine();
            Log.i("note","'"+str+"' was read");
            while (str != null)
            {
                sb.append(str + "\n");
                Log.i("note","'"+str+"' was read");
                str = br.readLine();
            }
```

Accessing Simple Files

```
            editor.setText(sb.toString());
        }
    }
    catch (java.io.FileNotFoundException e)
    {
        // most likely the file still was not created
    }
    catch (Throwable t)
    {
        Log.e("note",t.toString());
    }
    finally
    {
        if(is!=null)
        {
            try{is.close();} catch(Exception e)
            {Log.e("note",e.toString());}
        }
        if(isr!=null)
        {
            try{isr.close();} catch(Exception e)
            {Log.e("note",e.toString());}
        }
```

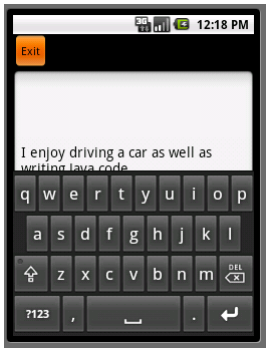
Accessing Simple Files

```
            if(br!=null)
            {
                try{br.close();} catch(Exception e)
                Log.e("note",e.toString());}
            }
        }
    }

    public void onPause()
    {
        super.onPause();
        OutputStream os = null;
        OutputStreamWriter osr = null;
        BufferedWriter bw = null;
        try
        {
            os = openFileOutput(FILENAME,MODE_PRIVATE);
            Log.i("note","output stream was created");
            OutputStreamWriter osw = new OutputStreamWriter(os);
            Log.i("note","output stream writer was");
            bw = new BufferedWriter(osw);
            Log.i("note","buffered writer was created");
            bw.write(editor.getText().toString());
            bw.flush();
        }
```

Accessing Simple Files

```
            catch (Throwable t)
            {
                Log.e("note",t.toString());
            }
            finally
            {
                if(os!=null)
                {
                    try{os.close();} catch(Exception e){}
                }
                if(osr!=null)
                {
                    try{osr.close();} catch(Exception e){}
                }
                if(bw!=null)
                {
                    try{bw.close();} catch(Exception e){}
                }
            }
        }
    }
```

Accessing Simple Files



Accessing Raw Resources

- We can easily access raw files part of our application. The raw files should be placed within the `res/raw` folder. Files we place in that folder are not compiled.

```
...
Resources resources = this.getResources();
InputStream is = resources.openRawResource(R.raw.ilove);
...
```

Accessing Raw Resources

```
package com.abelski.samples;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import android.app.Activity;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TextView;

public class RawFilesActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        String txt = null;
        Resources resources = null;
        InputStream is = null;
        InputStreamReader isr = null;
        StringBuilder sb = new StringBuilder();
```

Accessing Raw Resources

```
try
{
    resources = this.getResources();
    is = resources.openRawResource(R.raw.ilove);
    isr = new InputStreamReader(is);
    char[] vec = new char[100];
    while(isr.read(vec)!=-1)
    {
        sb.append(new String(vec));
    }
}
catch(Exception e)
{
    txt = e.getMessage();
}
```

Accessing Raw Resources

```
            finally
            {
                if(is!=null)
                {
                    try {is.close();}
                    catch(IOException e){txt = e.getMessage();}
                }
                if(isr!=null)
                {
                    try {isr.close();}
                    catch(IOException e){txt = e.getMessage();}
                }
                tv.setText(sb.toString());
                this.setContentView(tv);
            }
        }
    }
```

Accessing Raw Resources

Accessing Raw Resources

- Accessing a raw file we can get an `InputStream` only. There are no means for modifying the file.
- Placing files within the raw directory might be useful for initialization purposes.
- Using the `XML` format we can easily initialize our application with structured data the `XML` document holds.

Accessing Raw Resources

```
public class XMLBasedListActivity extends ListActivity
{
    TextView selectedColor;
    ArrayList<String> listItems = new ArrayList<String>();

    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        InputStream in = null;
        selectedColor = (TextView) findViewById(R.id.selectedcolor);
        try
        {
            in = getResources().openRawResource(R.raw.colors);
            DocumentBuilder builder =
                DocumentBuilderFactory.newInstance()
                .newDocumentBuilder();
            Document doc = builder.parse(in);
            NodeList colors = doc.getElementsByTagName("color");
```

Accessing Raw Resources

```
            for (int i = 0; i < colors.getLength(); i++)
            {
                listItems.add(colors.item(i).
                    getFirstChild().getNodeValue());
            }
        }
        catch (Throwable t)
        {
            Log.e("onCreate",t.getMessage());
        }
        finally
        {
            if(in!=null) {try{in.close();}catch(Exception e){}}
        }
        setListAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, listItems));
    }
```

Accessing Raw Resources

```
    public void onListItemClick(ListView parent, View v,
        int position, long id)
    {
        selectedColor.setText(listItems.get(position).toString());
    }
}
```

Accessing Raw Resources

Accessing XML Resources

- We can easily access XML files that were stored within the `res/xml` folder. Unlike the other resources, accessing these files is done differently.
- These files are compiled into an efficient binary form when deployed.

```
...
Resources resources = this.getResources();
XmlPullParser parser = resources.getXml(R.xml.cars);
...
```

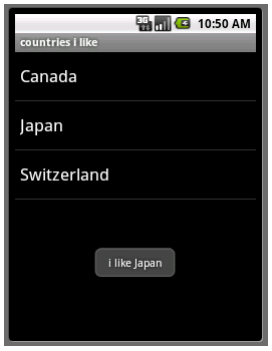
Accessing XML Resources

```
public class SimpleCountriesListActivity extends ListActivity
{
    ArrayList<String> list = new ArrayList<String>();

    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        try
        {
            XmlPullParser parser = getResources().getXml(R.xml.countries);
            int eventType = parser.getEventType();
            while(eventType!=XmlPullParser.END_DOCUMENT)
            {
                if(eventType == XmlPullParser.TEXT)
                {
                    list.add(parser.getText());
                }
                eventType = parser.next();
            }
        }
```

Accessing XML Resources

```
            catch (Exception e)
            {
                Log.i("xmlxml",e.getMessage());
            }
            setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, list));
        }

        public void onListItemClick(
            ListView parent, View v, int position, long id)
        {
            String toastText = "i like "+list.get(position);
            Toast toast = Toast.makeText(this,toastText,Toast.LENGTH_SHORT);
            toast.show();
        }
    }
```

Accessing XML Resources



Accessing SD Card External Storage

- The android platform allows our code to access the available secure digital flash memory card (SD Card) external storage.

```
...
File cardDir = new File("/sdcard/");
FILE file = new File(cardDir,"our_file.txt");
FileOutputStream fos = new FileOutputStream(file);
...
```

Accessing SD Card External Storage

- The external storage is accessible by all applications, whereas the default behavior of the `openFileInput()` and the `openFileOutput()` methods allows accessing the application private domain only.

Accessing SD Card External Storage

- Starting with android 1.6 there is a need in obtaining the `android.permission.WRITE_EXTERNAL_STORAGE` user permission.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Accessing SD Card External Storage

```
public class SimpleNoteActivity extends Activity
{
    private final static String FILENAME = "mynote.txt";
    private EditText editor;
    private Button bt;

    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        editor=(EditText)findViewById(R.id.text_editor);
        bt=(Button)findViewById(R.id.exit_button);
        bt.setOnClickListener(new Button.OnClickListener()
            {
                public void onClick(View v)
                {
                    finish();
                }
            }
                                                    );
    }
```

Accessing SD Card External Storage

```
public void onResume()
{
    super.onResume();
    InputStream is = null;
    InputStreamReader isr = null;
    BufferedReader br = null;
    try
    {
        File card = Environment.getExternalStorageDirectory();
        File file = new File(card,FILENAME);
        is = new FileInputStream(file);
        if (is != null)
        {
            isr = new InputStreamReader(is);
            br = new BufferedReader(isr);
            StringBuffer sb = new StringBuffer();
            String str = br.readLine();
            while (str != null)
            {
                sb.append(str + "\n");
                str = br.readLine();
            }
            editor.setText(sb.toString());
        }
    }
```

Accessing SD Card External Storage

```
catch (java.io.FileNotFoundException e)
{
    // most likely the file still wasnot created
}
catch (Throwable t)
{
    Log.e("note input",t.toString());
}
finally
{
    if(is!=null)
    {
        try{is.close();} catch(Exception e)
        {Log.e("note input",e.toString());}
    }
    if(isr!=null)
    {
        try{isr.close();} catch(Exception e)
        {Log.e("note input",e.toString());}
    }
```

Accessing SD Card External Storage

```
            if(br!=null)
            {
                try{br.close();} catch(Exception e)
                {Log.e("note input",e.toString());}
            }
        }
    }
```

Accessing SD Card External Storage

```
public void onPause()
{
    super.onPause();
    OutputStream os = null;
    OutputStreamWriter osr = null;
    BufferedWriter bw = null;
    try
    {
        File card = Environment.getExternalStorageDirectory();
        File file = new File(card,FILENAME);
        if(!file.exists())
        {
            file.createNewFile();
        }
        os = new FileOutputStream(file);
        os = new FileOutputStream(file);
        OutputStreamWriter osw = new OutputStreamWriter(os);
        bw = new BufferedWriter(osw);
        bw.write(editor.getText().toString());
        bw.flush();
    }
```

Accessing SD Card External Storage

```
            catch (Throwable t)
            {
                Log.e("note output",t.toString());
            }
            finally
            {
                if(os!=null)
                {
                    try{os.close();}
                    catch(Exception e){}
                }
                if(osr!=null)
                {
                    try{osr.close();}
                    catch(Exception e){}
                }
                if(bw!=null)
                {
                    try{bw.close();}
                    catch(Exception e){}
                }
            }
        }
    }
```

Accessing SD Card External Storage