

Android Beginners Workshop

at the

MOBILE MONDAY

m 2 d 2

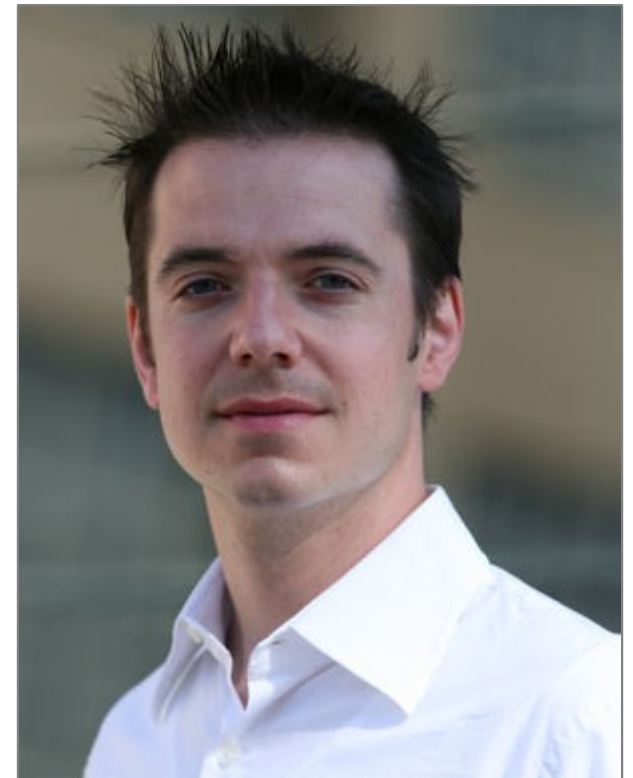
DEVELOPER DAY

February, 23th 2010

Sven Woltmann, AndroidPIT

Sven Woltmann

- Studied Computer Science at the TU Ilmenau, 1994-1999
- 12 year Java experience, 25 years total programming experience
- Founded several start-ups together with Fabien Röhliger
- Developed several products, which are still in use today
- Joined AndroidPIT in August 2009
- Developed a new CMS for the AndroidPIT website
- Developed the AndroidPIT App



E-mail: sven.woltmann@androidpit.de

AndroidPIT (www.androidpit.de)

- Daily test reports (> 400)
- Daily news
- Community (> 8.000 members)
- Forum (> 3,000 topics)
- Wiki
- Market apps database (> 30,000)
- English version in 03/2010



The screenshot shows the AndroidPIT website homepage. The header features the AndroidPIT logo, a tagline 'Fill up your mobile', and a navigation bar with links: Home, Tests, Blog & News, Forum, Wiki, Alle Apps, and Community. A user is logged in as 'Sven Woltmann' with options to 'Abmelden' (Logout) or 'Admin'. There are also buttons for 'Blog-Eintrag schreiben' (Write blog entry) and 'Testbericht schreiben' (Write test report).

The main content area is titled 'Die neuesten Android-Testberichte' (The latest Android test reports). It features a featured article for 'ACast - News Reader und Podcast Viewer' by Fabien Röhlinger, dated 15.02.2010. The article includes a QR code and a 'QR-Codes ?' link. Below the article, there are links for 'Testbericht lesen »' and '15 Kommentare'.

On the right side, there is a search bar with the text 'Suchbegriff' and a 'Suchen' button. Below the search bar, there is a checkbox for 'Nur nach Apps suchen'. The 'Blog & News' section is also visible, with a link to 'Alle Blog-Einträge' and a list of recent blog posts, including 'Neue HTC Geräte: Legend und Desire' and '3D Homescreen für Android - WOW!'.

AndroidPIT App

- All test reports
- All news
- All forum posts
- More to come...

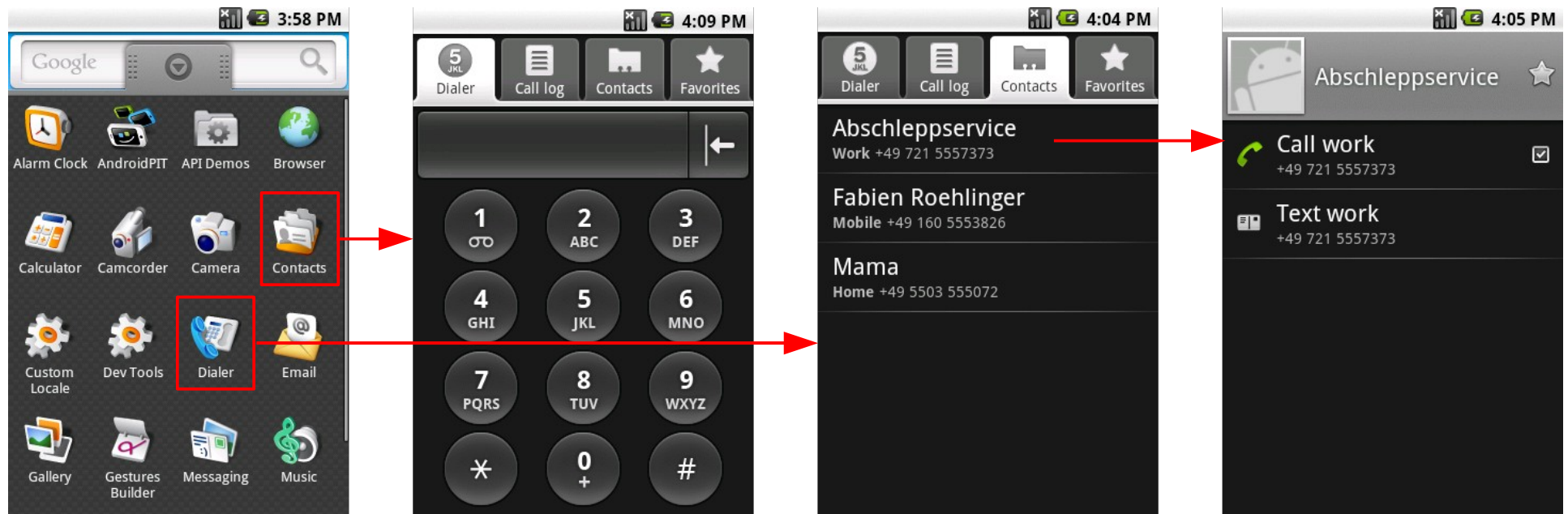


Contents of the Workshop

1. Android app basics
2. Development environment requirements
3. Setting up the development environment **(not part of the live presentation)**
4. Developing and testing a „Hello World“ app
5. Questions

1. Android App Basics

- Android apps are written in Java
- Almost all Java classes available plus encryption, http, json, xml processing libraries
- No main() function – instead: loosely coupled components, one or more of them defined as entry point(s)
- Most important component: „Activity“ – corresponds to a visible window on the screen



1. Android App Basics – Activity, View, Event, Intent

Activity

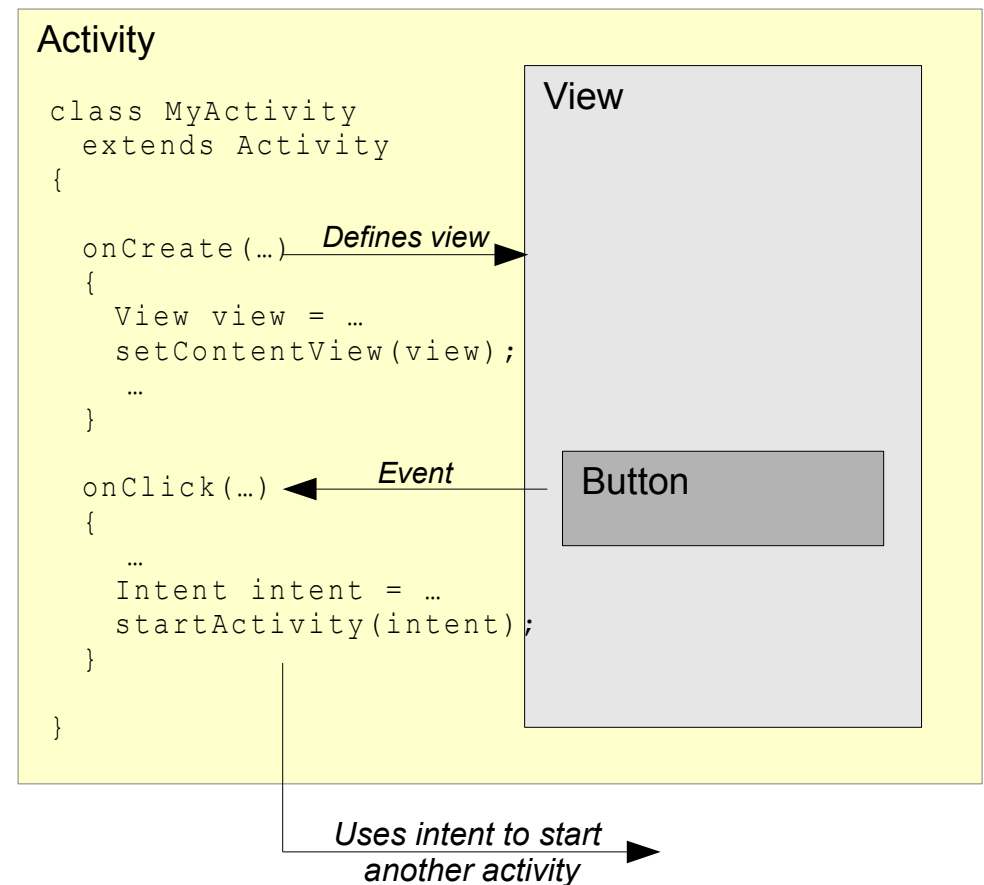
- Defines a „View“ to be displayed in its window
- Handles events (e.g. click on a button)
- Uses „Intents“ to start other activities

View

- A „View“ is the visible part of the activity
- Defined in an XML layout file (or in code)

Intent

- Starts another activity („opens a new window“)
- Can pass data to the started activity
- Can even start activities from another app
- More in the workshop „Developing Android Intents“

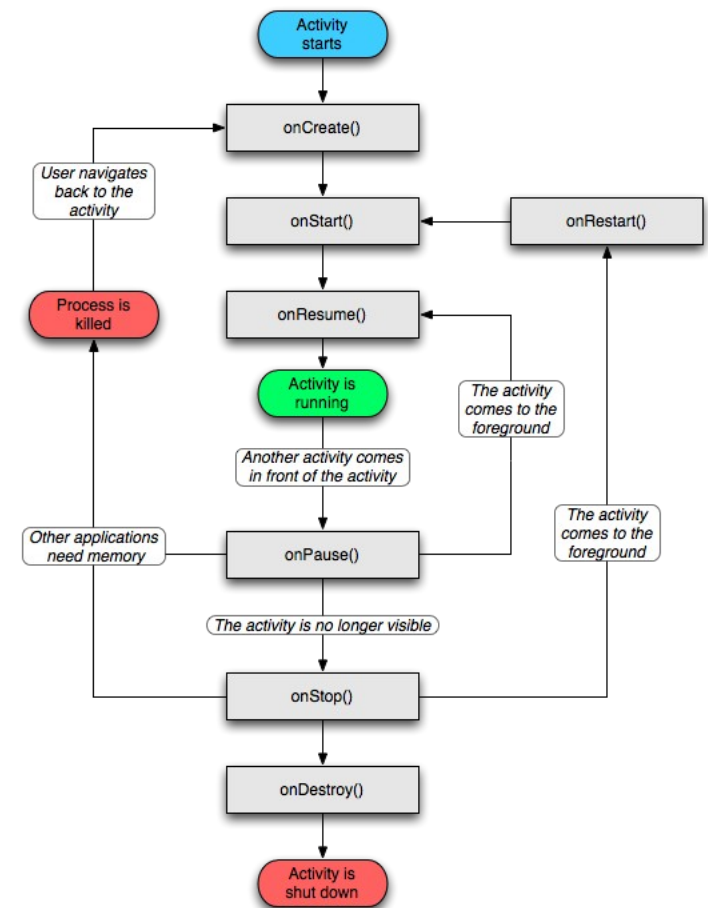


1. Android App Basics – Activity Lifecycle

As resources are limited on a mobile phone, the operating system must be able to destroy an activity, which is not active (meaning: in the foreground of the screen), at any time.

Therefore, every Activity has a lifecycle which is handled by the Android operating system.

To avoid losing data or the state of an activity, you must override lifecycle methods to save your data when your activity is paused (not in the foreground anymore but still visible) or stopped (not visible anymore).



2. Development Environment Requirements

1. Eclipse
2. Android SDK Starter Package from <http://developer.android.com/sdk/index.html>
3. Use SDK Starter Package (run as Administrator) to install:
 - SDKs for all platforms your app shall be running on (e.g. all from 1.5 to 2.1)
 - USB Driver Package for USB debugging (debug apps directly on your phone)
4. ADT (Android Development Tools) Eclipse Plugin
(Install from within Eclipse; location: <https://dl-ssl.google.com/android/eclipse/>)

What is the ADT good for?

- Wizard to create a new Android project
- Code editor for Android XML files
- Automatic build process (creates APK „Android Package“ file)
- Debugging of apps in the emulator or on a real phone
- Take screenshots from a real phone
- Export signed APKs to be uploaded into Android Market

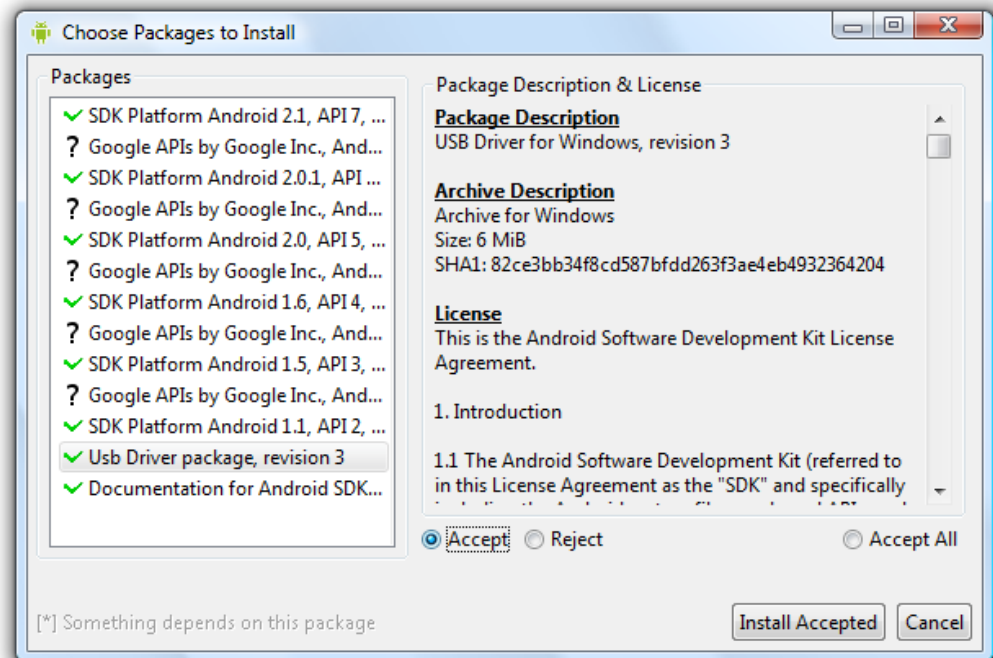
3. Setting up the Development Environment (1/3)

1. Eclipse is expected to be installed.
2. Download and install the SDK Starter Package appropriate to your operating system from <http://developer.android.com/sdk/index.html>

3. Run SDK Starter Package
(as Administrator on Vista / Windows 7).

Install...

- all SDK platforms
(for this tutorial, 1.6 is enough),
- the USB driver package
(not required for this tutorial),
- and the documentation
(not required for this tutorial).



3. Setting up the Development Environment (2/3)

4. Install the ADT (Android Development Tools) plugin.

The following instructions are for Eclipse 3.5 (Galileo) and Eclipse 3.6 (Helios).

Instructions for Eclipse 3.4 (Ganymede) can be found at <http://developer.android.com/sdk/eclipse-adt.html>.

4.1. Select „Help“ → „Install New Software“.

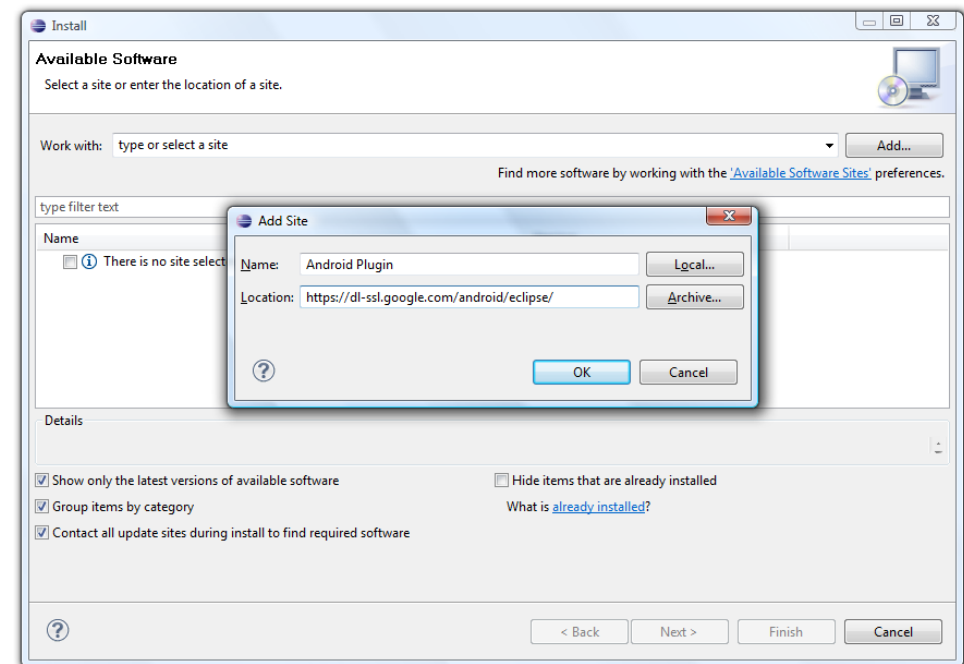
4.2. In the „Available Software“ dialog, click the „Add“ button...

4.3. In the „Add Site“ dialog, enter a name (e.g. „Android Plugin“) and the following URL as location:

<https://dl-ssl.google.com/android/eclipse/>

(try „http“, if „https“ does not work).

Click the „OK“ button.



3. Setting up the Development Environment (3/3)

4.4. Back in the „Available Software“ dialog, select the checkbox next to „Developer Tools“ and click „Next“.

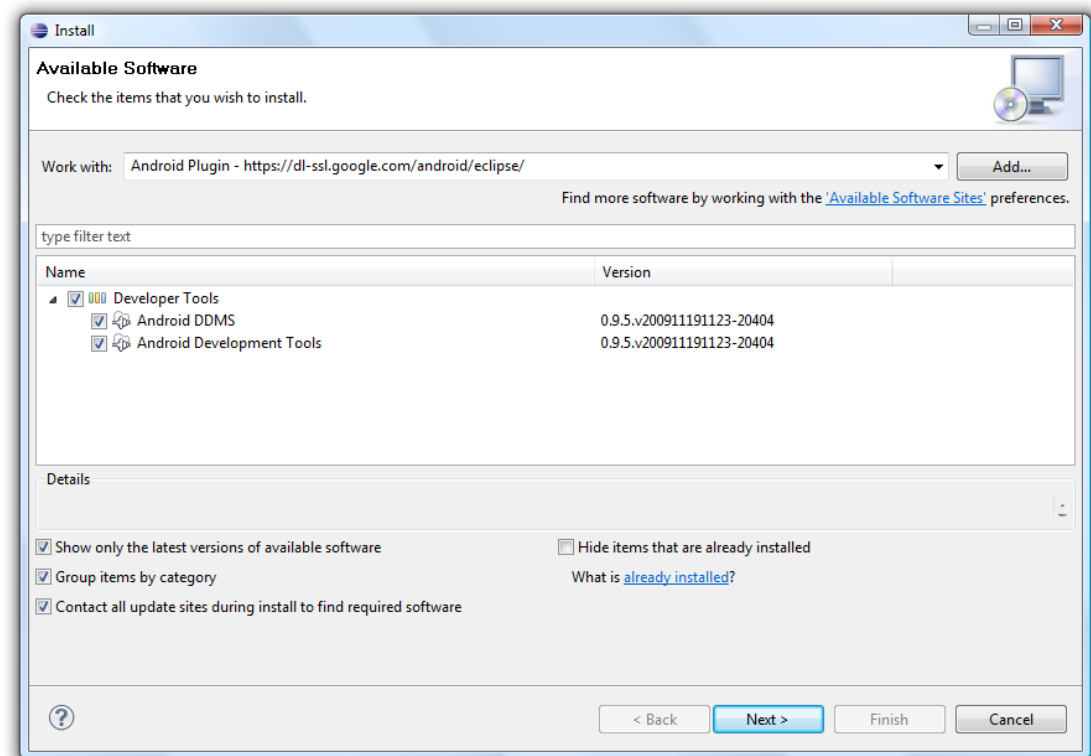
4.5. In the „Install Details“ dialog, click „Next“ to accept the license Agreement, then click „Finish“.

4.6. Restart Eclipse.

4.7. Select „Window“ → „Preferences“.

4.8. In the „Preferences“ dialog, select „Android“.

4.9. Enter the SDK folder (e.g. „C:\Programme\android-sdk-windows“) and click „OK“.



4. Developing and Testing a Hello World App (1/22)

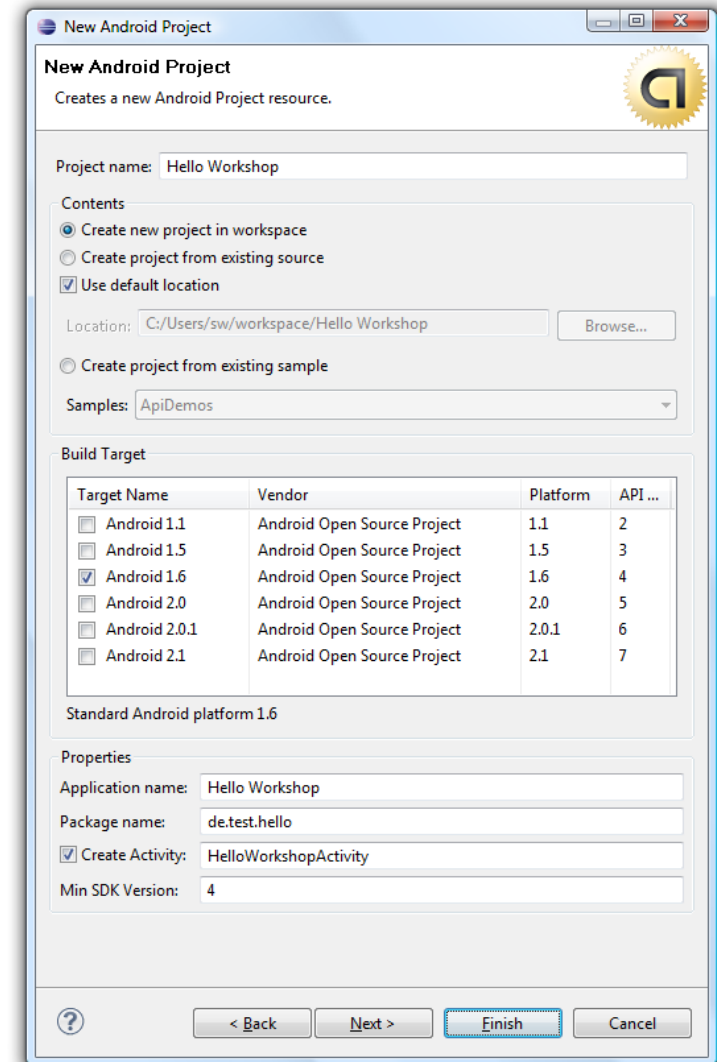
Use the ADT's „New Project“ wizard to create a new project:

Select „File“ → „New“ → „Project...“; in the „New Project“ dialog select „Android Project“ from the „Android“ folder.

Fill out the „New Android Project“ dialog as follows:

- Project name: „Hello Workshop“
- Use default location or define your own location
- Build target: Android 1.6 (we'll make this 1.5-compatible later)
- Application name: „Hello Workshop“
- Package name: „de.test.hello“
- Create Activity: „HelloWorkshopActivity“
- Min SDK Version: „4“

Click on the „Finish“ button.



4. Developing and Testing a Hello World App (2/22)

The „New Project“ Wizard created the following files and folders:

<ul style="list-style-type: none"> Hello Workshop <ul style="list-style-type: none"> src <ul style="list-style-type: none"> de.test.hello <ul style="list-style-type: none"> HelloWorkshopActivity.java gen [Generated Java Files] <ul style="list-style-type: none"> de.test.hello <ul style="list-style-type: none"> R.java Android 1.6 assets res <ul style="list-style-type: none"> drawable-hdpi <ul style="list-style-type: none"> icon.png drawable-ldpi <ul style="list-style-type: none"> icon.png drawable-mdpi <ul style="list-style-type: none"> icon.png layout <ul style="list-style-type: none"> main.xml values <ul style="list-style-type: none"> strings.xml AndroidManifest.xml default.properties 	<ul style="list-style-type: none"> Main Activity class Resource ID definitions (auto-created; never modify this file) Folder for binary files (everything not handled by Android) Folder for resource files (images, layouts, strings, ...) Logo in high resolution (72 by 72 pixels – 240 dpi) Logo in low resolution (36 by 36 pixels – 120 dpi) Logo in medium resolution (48 by 48 pixels – 160 dpi) Layout definition for main activity's view String definitions „Manifest file“ defining things like app name, logo and main activity Project properties
--	--

4. Developing and Testing a Hello World App (3/22)

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.test.hello"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloWorkshopActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="4" />
</manifest>
```

Base package name

Numeric version code, increase with new version

Version name displayed in Market

Reference to icon in drawable-xxx folders

Reference to app_name in strings.xml

Main activity's class name

Main activity's label (at the top of the screen)

Intent filter to define that this activity can be launched from the launch menu

4. Developing and Testing a Hello World App (4/22)

For phones with high density screens (e.g. Motorola Milestone) to use the hdpi resources, we must add the following entry to the AndroidManifest.xml file:

```
<supports-screens android:anyDensity="true" />
```

Otherwise, those phones will use the mdpi resources and scale them up.

For phones with large (Archos 5) or small screens (HTC Tattoo) to use the full screen or to run the app at all, we must further add the following definitions to the entry created before:

```
<supports-screens android:anyDensity="true"  
    android:largeScreens="true"  
    android:smallScreens="true" />
```

Otherwise, phones with large screens will display the app only in a small area of the screen with a black border around it, and phones with small screens won't start the app at all.

4. Developing and Testing a Hello World App (5/22)

values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloWorldActivity!</string>
    <string name="app_name">Hello Workshop</string>
</resources>
```

Defines strings, referenced by the manifest and layout file (can also be referenced by code):

`android:label="@string/app_name"` is equivalent to
`android:label="Hello Workshop"`

Of course only as long as we don't add *strings.xml* files in other languages.
If you plan to localize your app, you should always put all strings into resource files.

4. Developing and Testing a Hello World App (6/22)

layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://[...]"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
</LinearLayout>
```

Defines a (simple) layout called „main“ (just a name, no meaning), which is later referenced by the activity to create its view.

A screenshot of the Hello World app interface. It shows a black background with a single line of white text at the top: "Hello World, HelloWorldActivity!". A red box highlights the text, and a green box highlights the text itself. A red arrow points from the `android:text="@string/hello"` line in the XML code to the text in the screenshot. A green arrow points from the `android:layout_height="wrap_content"` line in the XML code to the text in the screenshot.

Hello World, HelloWorldActivity!

4. Developing and Testing a Hello World App (7/22)

R.java

Automatically created – defines integer constants for each resource.
Never modify this file – no need to even look at it.

```
package de.test.hello;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

4. Developing and Testing a Hello World App (8/22)

HelloWorkshop2Activity.java

```
package de.test.hello;

import android.app.Activity;
import android.os.Bundle;

public class HelloWorkshopActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

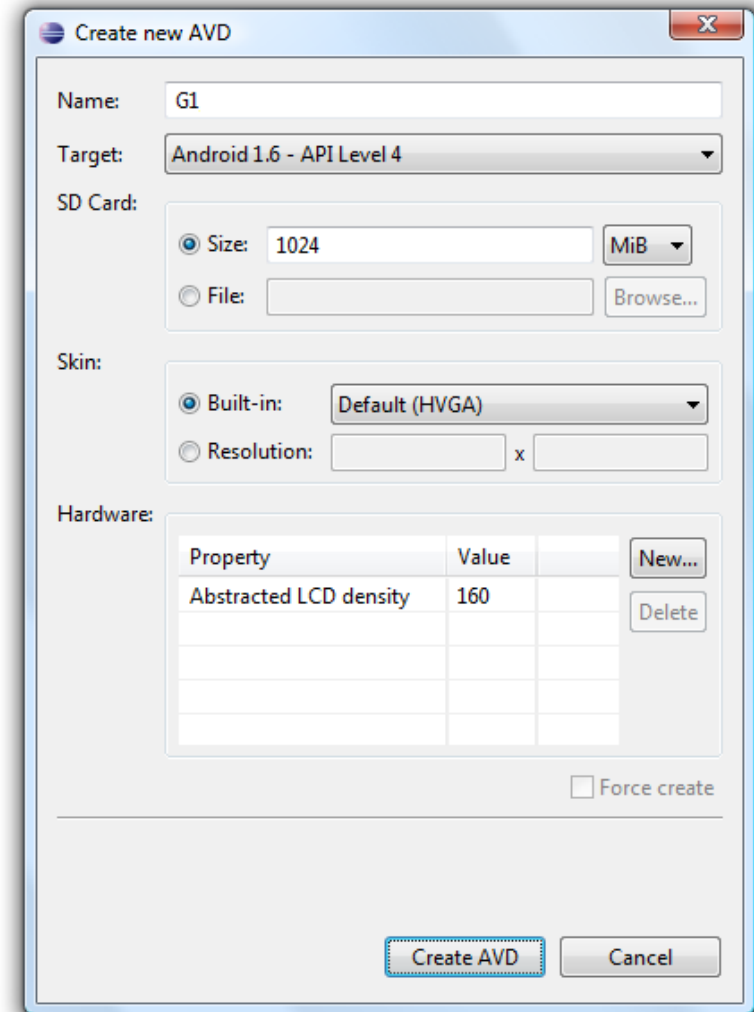
Constant from R class referencing the layout defined in main.xml

Sets the specified view as view for this activity.

4. Developing and Testing a Hello World App (9/22)

Create an „Android Virtual Device“ (Emulator)

- Click on the „Android SDK and AVD manager“ icon in the Eclipse icon bar.
 - Click on „New“
 - In the „Create new AVD“ dialog, enter:
 - Name: „G1“
 - Target: „Android 1.6 - API Level 4“
 - SD Card Size: „1024“ MiB
 - Skin: Built-in: „Default (HVGA)“
- then click on „Create AVD“ (might take a while now)



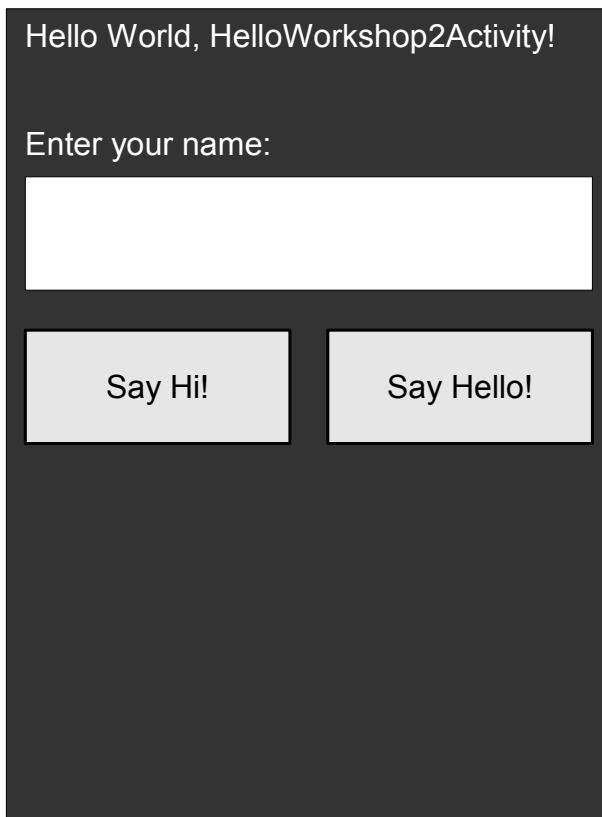
4. Developing and Testing a Hello World App (10/22)

Run the App for the First Time

- Select „Run“ → „Run“ (or press Ctrl + F11).
- In the „Run As“ dialog, select „Android Application“ and click on „OK“.
- Eclipse will now automatically start the „G1“ AVD we just created (if you have multiple matching AVDs, a selection dialog will let you chose one).
- After about a minute we should see our app running in the emulator!



4. Developing and Testing a Hello World App (11/22)

A screenshot of an Android application interface. At the top, it says "Hello World, HelloWorldActivity!". Below this is a text input field with the placeholder "Enter your name:". Under the input field are two buttons: "Say Hi!" and "Say Hello!".

Hello World, HelloWorldActivity!

Enter your name:

Say Hi! Say Hello!

This is what we want to do today:

- Create a view that looks like the one at the left
- Let the user enter his name in the input field
- When the user clicks one of the buttons and no name was entered, display an alert dialog with an error message.
- When the user clicks a button with his name entered
 - the user shall be greeted with a popup message („toast“)
 - the greeting at the top of the screen shall change

4. Developing and Testing a Hello World App (12/22)

Add Components to the Layout

Add the „Enter your name“ label below the first TextView component:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="@string/enter_your_name"
/>
```

Eclipse will now show an error message:

„No resource found that matches the given name (at 'text' with value '@string/enter_your_name').

Add the missing resource to the strings.xml file:

```
<string name="enter_your_name">Enter your name:</string>
```

The error message from main.xml will now disappear.

4. Developing and Testing a Hello World App (13/22)

Add the input field below the „Enter your name“ component:

```
<EditText  
    android:id="@+id/name_field"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
/>
```

The first line assigns the id „name_field“ to this component.

Below the name field we want to buttons beside each other.

Therefore we need a nested LinearLayout with horizontal orientation, and inside it the two buttons.

4. Developing and Testing a Hello World App (14/22)

Nested LinearLayout with two Buttons:

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >
    <Button
        android:id="@+id/hi_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/hi_button"
        />
    <Button
        android:id="@+id/hello_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/hello_button"
        />
</LinearLayout>
```

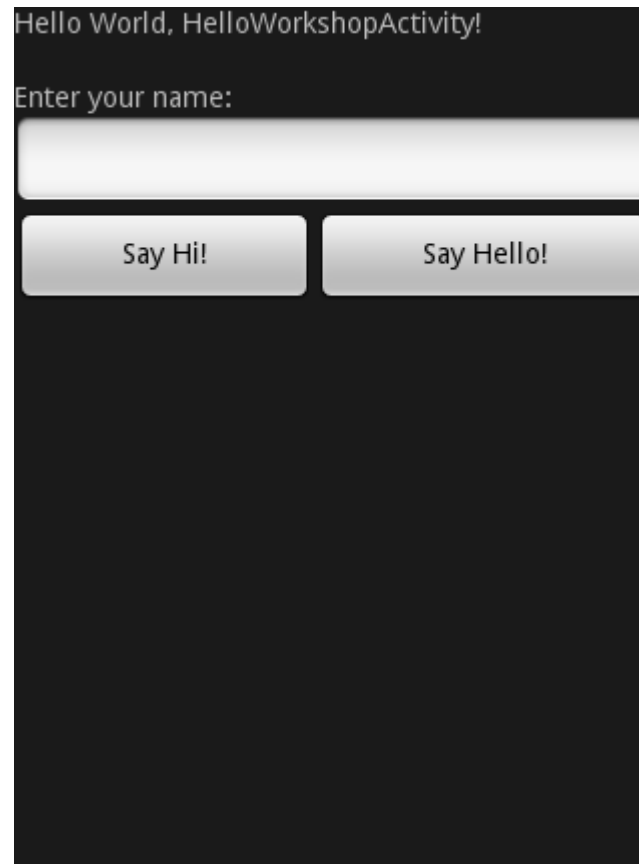
Add the button texts to the strings.xml file:

```
<string name="hi_button">Say Hi!</string>
<string name="hello_button">Say Hello!</string>
```


4. Developing and Testing a Hello World App (15/22)

Let's have a look at the layout!

In the main.xml tab in Eclipse, click on the „Layout“ tab at the bottom, and you will see the following:



4. Developing and Testing a Hello World App (16/22)

Now we need to extend our Activity. Open `HelloWorkshopActivity.java` and add the following field definitions at the top of the class:

```
private Button hiButton;  
private Button helloButton;
```

At the end of the `onCreate` method, insert the following code:

```
hiButton = (Button)findViewById(R.id.hi_button);  
hiButton.setOnClickListener(this);  
  
helloButton = (Button)findViewById(R.id.hello_button);  
helloButton.setOnClickListener(this);
```

We will now see an error message as `HelloWorkshopActivity` does not implement `OnClickListener` yet. Click on the error message and select „Let 'HelloWorkshopActivity' implement 'OnClickListener'“.

We'll see the next error message because `HelloWorkshopActivity` does not implement the methods defined in `OnClickListener`. Click on the error message and select „Add unimplemented methods“.

4. Developing and Testing a Hello World App (17/22)

When the user clicks on one of the buttons, we first need to check if he entered his name. Insert the following code into the „onClick“ method Eclipse has just created for us:

```
EditText nameField = (EditText) findViewById(R.id.name_field);
String name = nameField.getText().toString();
if (name.length() == 0) {
    new AlertDialog.Builder(this)
        .setMessage(R.string.error_name_missing)
        .setNeutralButton(R.string.error_ok, null)
        .show();
    return;
}
```

Add the two string resources we just used and two others we use in the next step to the strings.xml file:

```
<string name="error_name_missing">Please enter your name.</string>
<string name="error_ok">OK</string>
<string name="hi_greeting">Hi %s!</string>
<string name="hello_greeting">Hello %s!</string>
```

4. Developing and Testing a Hello World App (18/22)

We've made sure the user entered a name. Now display the greeting depending on the button:

```
if (v == hiButton || v == helloButton)
{
    int resourceId = v == hiButton ? R.string.hi_greeting
        : R.string.hello_greeting;

    String greeting = getResources().getString(resourceId, name);
    Toast.makeText(this, greeting, Toast.LENGTH_LONG).show();
}
```

Finally, we want the greeting at the top of the screen to be changed. To change the text of that component, we first must assign an ID to it. So edit the main.xml file and insert the id into the first TextView element:

```
<TextView
    android:id="@+id/greeting_field"
    ... />
```

4. Developing and Testing a Hello World App (19/22)

The method we've just created can now be extended:

```
if (v == hiButton || v == helloButton)
{
    int resourceId = v == hiButton ? R.string.hi_greeting
        : R.string.hello_greeting;

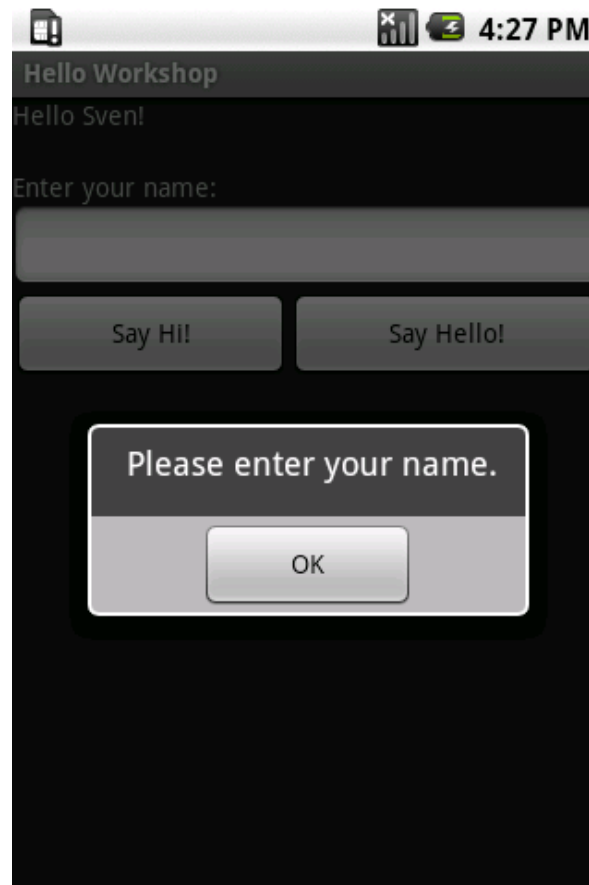
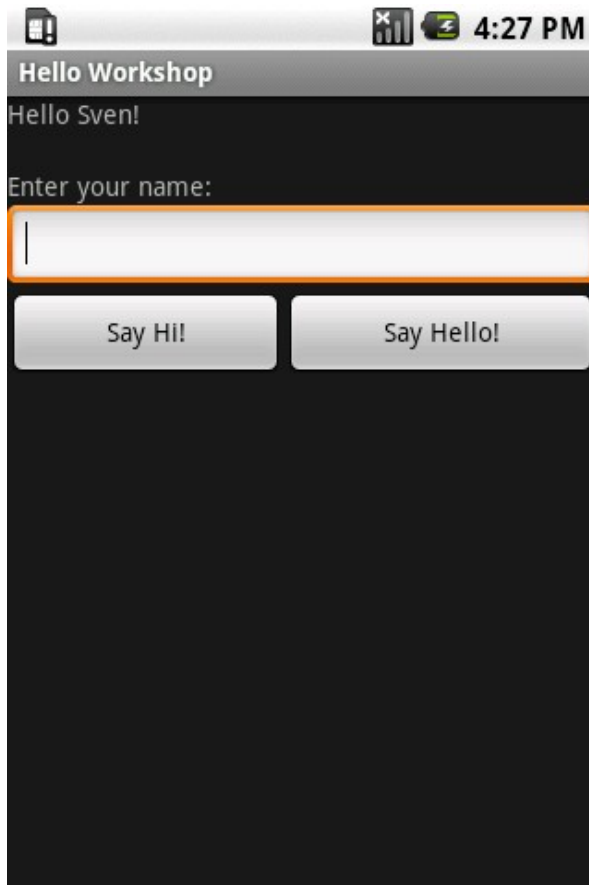
    String greeting = getResources().getString(resourceId, name);
    Toast.makeText(this, greeting, Toast.LENGTH_LONG).show();

    TextView greetingField = (TextView) findViewById(R.id.greeting_field);
    greetingField.setText(greeting);
}
```

And now? We're ready - run the project!

You'll find the complete sources in the appendix.

4. Developing and Testing a Hello World App (20/22)



4. Developing and Testing a Hello World App (21/22)

Making the project compatible for Android 1.5 (e.g. HTC Hero)

In Android 1.6, different screen resolutions and densities were introduced. Before Android 1.5, the folders drawable-hdpi, drawable-ldpi, drawable-mdpi were not known.

- Rename folder „drawable-hdpi“ to „drawable-hdpi-v4“, so Android 1.5 and lower will ignore it
- Rename folder „drawable-ldpi“ to „drawable-ldpi-v4“, so Android 1.5 and lower will ignore it
- Rename folder „drawable-mdpi“ to „drawable“, so Android 1.5 will see it

In AndroidManifest.xml change the line

```
<uses-sdk android:minSdkVersion="4" />
```

to

```
<uses-sdk android:minSdkVersion="3" targetSdkVersion="4" />
```

4. Developing and Testing a Hello World App (22/22)

Where to go from here?

- Go to the other workshops
- Create different layouts for different screen orientations
- Add a second Activity and start that from the existing Activity (for example by clicking on a new button)
- Play around with other layouts
(<http://developer.android.com/resources/tutorials/views/index.html>)
- Check out all the other tutorials on <http://developer.android.com>

5. Questions



Thank You!

Links:

- <http://www.androidpit.de/>
- <http://www.androidpit.de/de/android/forum>
- <http://www.androidpit.de/de/android/market/apps/app/de.androidpit.app/AndroidPIT>

Official page with tutorials and examples:

- <http://developer.android.com/>

This presentation:

- <http://www.androidpit.de/files/androidpit-beginners-workshop-2010.pdf>



Appendix – Source Codes

HelloWorkshopActivity.java:

```
package de.test.hello;

import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class HelloWorkshopActivity extends Activity
    implements OnClickListener {

    private Button hiButton;
    private Button helloButton;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        hiButton = (Button) findViewById(R.id.hi_button);
        hiButton.setOnClickListener(this);

        helloButton = (Button) findViewById(R.id.hello_button);
        helloButton.setOnClickListener(this);
    }
}
```

```
@Override
public void onClick(View v) {
    EditText nameField = (EditText) findViewById(R.id.name_field);

    String name = nameField.getText().toString();

    if (name.length() == 0) {
        new AlertDialog.Builder(this).setMessage(
            R.string.error_name_missing).setNeutralButton(
                R.string.error_ok
                null).show();
        return;
    }

    if (v == hiButton || v == helloButton) {
        int resourceId = v == hiButton ? R.string.hi_greeting
            : R.string.hello_greeting;

        String greeting = getResources().getString(resourceId, name);
        Toast.makeText(this, greeting, Toast.LENGTH_LONG).show();

        TextView greetingField = (TextView) findViewById(R.id.greeting_field);
        greetingField.setText(greeting);
    }
}
```

Appendix – Source Codes

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/greeting_field"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="@string/enter_your_name"
        />
    <EditText
        android:id="@+id/name_field"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >
    <Button
        android:id="@+id/hi_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hi_button"
        android:layout_weight="1"
        />
    <Button
        android:id="@+id/hello_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_button"
        android:layout_weight="1"
        />
</LinearLayout>
</LinearLayout>
```

Appendix – Source Codes

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloWorldWorkshopActivity</string>
    <string name="app_name">Hello Workshop</string>
    <string name="enter_your_name">Enter your name</string>

    <string name="hi_button">Say Hi</string>
    <string name="hello_button">Say Hello</string>

    <string name="error_name_missing">Please enter your name</string>
    <string name="error_ok">OK</string>

    <string name="hi_greeting">Hi %s</string>
    <string name="hello_greeting">Hello %s</string>
</resources>
```