Implicit Intents
&
System Resources

Prof. Dr.-Ing. V.Brovkov

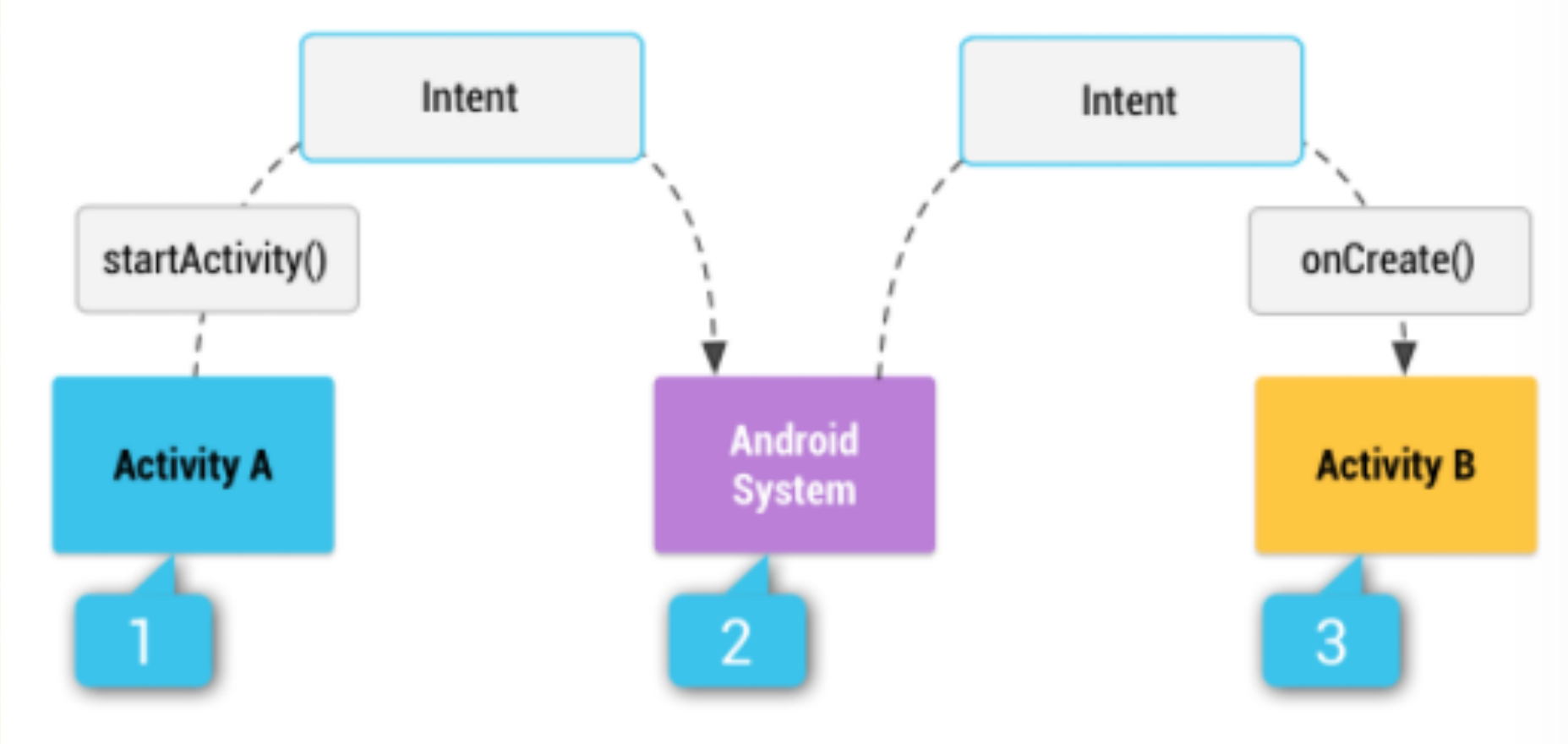
Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- There are two types of intents:
  - **Explicit intents** specify the component to start by name (the fully-qualified class name). You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start.
    - For example, start a new activity in response to a user action or start a service to download a file in the background.
  - **Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.
    - For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

Building an Intent

http://developer.android.com/guide/components/intents-filters.html



Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- The primary information contained in an Intent is the following:
  - **Component name** (the name of the component to start); makes intent **explicit**
  - **Action** (a string that specifies the generic action to perform) plus **Data**; makes intent **implicit**
    - The action largely determines how the rest of the intent is structured—particularly what is contained in the data and extras.
    - You can specify your own actions for use by intents within your app (or for use by other apps to invoke components in your app), but you should usually use action constants defined by the Intent class or other framework classes.

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- When you create an implicit intent, the Android system finds the appropriate component to start by comparing the contents of the intent to the intent filters declared in the manifest file of other apps on the device.
- If the intent matches an intent filter, the system starts that component and delivers it the Intent object.
- If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use.

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- Intent Actions:
  - ACTION_VIEW
  - ACTION_SEND
  - ACTION_EDIT
  - ...
- If you define your own actions, be sure to include your app's package name as a prefix:

```
static final String ACTION_TIMETRAVEL = "com.example.action.TIMETRAVEL";
```

- Intent Data:
  - The URI (a Uri object) that references the data to be acted on and/or the MIME type of that data. The type of data supplied is generally dictated by the intent's action.

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- When creating an intent, it's often important to specify the type of data (its MIME type) in addition to its URI.
  - For example, an activity that's able to display images probably won't be able to play an audio file, even though the URI formats could be similar.
  - Specifying the MIME type of your data helps the Android system find the best component to receive your intent.
  - To set only the data URI, call **setData().**
  - To set only the MIME type, call setType().
  - You can set both explicitly with setDataAndType().

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- **Category**
  - A string containing additional information about the kind of component that should handle the intent.
  - Any number of category descriptions can be placed in an intent, but most intents do not require a category.
- Some common categories:
  - CATEGORY_BROWSABLE
  - CATEGORY_LAUNCHER
  - CATEGORY_DEFAULT
  - CATEGORY_ALTERNATIVE
- You can specify a category with **addCategory()**.

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- An intent can carry additional information that does not affect how it is resolved to an app component.
- An intent can supply:
  - **Extras:** Key-value pairs that carry additional information required to accomplish the requested action.
    - Just as some actions use particular kinds of data URIs, some actions also use particular extras.
    - You can add extra data with various **putExtra()**
    - You can also create a **Bundle** object with all the extra data, then insert the Bundle in the Intent with **putExtras()**.
    - For example, when creating an intent to send an email with ACTION_SEND, you can specify the "to" recipient with the EXTRA_EMAIL key, and specify the "subject" with the EXTRA_SUBJECT key.

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- The Intent class specifies many EXTRA_* constants for standardized data types.
  - If you need to declare your own extra keys (for intents that your app receives), be sure to include your app's package name as a prefix. For example:

```
static final String EXTRA_GIGAWATTS = "com.example.EXTRA_GIGAWATTS";
```

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- An intent can supply **Flags**:
  - The flags may instruct the Android system how to launch an activity, for example:
    - which task the activity should belong to
    - how to treat it after it's launched (for example, whether it belongs in the list of recent activities)
  - For more information, see the **setFlags()** method.

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- Example implizit intent:

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

Building an Intent

http://developer.android.com/guide/components/intents-filters.html

- Example implizit intent
  - To show the chooser, create an Intent using createChooser() and pass it to startActivity(). For example:

```
Intent sendIntent = new Intent(Intent.ACTION_SEND);
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
String title = getResources().getString(R.string.chooser_title);
// Create intent to show the chooser dialog
Intent chooser = Intent.createChooser(sendIntent, title);

// Verify the original intent will resolve to at least one activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```

Building an Intent

http://developer.android.com/reference/android/content/Intent.html

- The primary pieces of information in an intent are:
  - **action** -- The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.
  - **data** -- The data to operate on, such as a person record in the contacts database, expressed as a Uri.
- Some examples of action/data pairs

- `ACTION_VIEW` ***content://contacts/people/1*** -- Display information about the person whose identifier is "1".
- `ACTION_DIAL` ***content://contacts/people/1*** -- Display the phone dialer with the person filled in.

Building an Intent

http://developer.android.com/reference/android/content/Intent.html

- Some examples of action/data pairs

- `ACTION_VIEW` ***tel:123*** -- Display the phone dialer with the given number filled in. Note how the VIEW action does what what is considered the most reasonable thing for a particular URI.

- `ACTION_DIAL` ***tel:123*** -- Display the phone dialer with the given number filled in. To place a phone call directly, use the `ACTION_CALL` action
- `ACTION_EDIT` ***content://contacts/people/1*** -- Edit information about the person whose identifier is "1".

- `ACTION_VIEW` ***content://contacts/people/*** -- Display a list of people, which the user can browse through. This example is a typical top-level entry into the Contacts application, showing you the list of people. Selecting a particular person to view would result in a new intent { `ACTION_VIEW` ***content://contacts/N*** } being used to start an activity to display that person.

Building an Intent

http://developer.android.com/reference/android/content/Intent.html

- Here are some examples of other operations you can specify as intents using these additional parameters:

- `ACTION_MAIN` **with category** `CATEGORY_HOME` -- Launch the home screen.

- `ACTION_GET_CONTENT` **with MIME type** `vnd.android.cursor.item/phone` -- Display the list of people's phone numbers, allowing the user to browse through them and pick one and return it to the parent activity.

- `ACTION_GET_CONTENT` **with MIME type */* and category** `CATEGORY_OPENABLE` -- Display all pickers for data that can be opened with `ContentResolver.openInputStream()`, allowing the user to pick one of them and then some data inside of it and returning the resulting URI to the caller. This can be used, for example, in an e-mail application to allow the user to pick some data to include as an attachment.

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- **Caution**: If there are no apps on the device that can receive the implicit intent, your app will ***crash*** when it calls startActivity().
- To first verify that an app exists to receive the intent, call resolveActivity() on your Intent object.
  - If the result is non-null, there is at least one app that can handle the intent and it's safe to call startActivity().
  - If the result is null, you should not use the intent and, if possible, you should disable the feature that invokes the intent.

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Create an alarm

```
public void createAlarm(String message, int hour, int minutes) {
    Intent intent = new Intent(AlarmClock.ACTION_SET_ALARM)
            .putExtra(AlarmClock.EXTRA_MESSAGE, message)
            .putExtra(AlarmClock.EXTRA_HOUR, hour)
            .putExtra(AlarmClock.EXTRA_MINUTES, minutes);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SET_ALARM" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Create an alarm

`EXTRA_RINGTONE`

A `content:` URI specifying a ringtone to use with the alarm, or `VALUE_RINGTONE_SILENT` for no ringtone.

To use the default ringtone, do not specify this extra.

`EXTRA_VIBRATE`

A boolean specifying whether to vibrate for this alarm.

`EXTRA_SKIP_UI`

A boolean specifying whether the responding app should skip its UI when setting the alarm. If true, the app should bypass any confirmation UI and simply set the specified alarm.

In order to invoke the `ACTION_SET_ALARM` intent, your app must have the `SET_ALARM` permission:

```
<uses-permission android:name="com.android.alarm.permission.SET_ALARM" />
```

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Create a timer

```
public void startTimer(String message, int seconds) {
    Intent intent = new Intent(AlarmClock.ACTION_SET_TIMER)
            .putExtra(AlarmClock.EXTRA_MESSAGE, message)
            .putExtra(AlarmClock.EXTRA_LENGTH, seconds)
            .putExtra(AlarmClock.EXTRA_SKIP_UI, true);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

```
<uses-permission android:name="com.android.alarm.permission.SET_ALARM" />
```

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SET_TIMER" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Show all alarms

Action

`ACTION_SHOW_ALARMS`

Data URI

None

MIME Type

None

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SHOW_ALARMS" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Add a calendar event

Action

`ACTION_INSERT`

Data URI

`Events.CONTENT_URI`

MIME Type

`"vnd.android.cursor.dir/event"`

Extras

Extras

`EXTRA_EVENT_ALL_DAY`

A boolean specifying whether this is an all-day event.

`EXTRA_EVENT_BEGIN_TIME`

The start time of the event (milliseconds since epoch).

`EXTRA_EVENT_END_TIME`

The end time of the event (milliseconds since epoch).

`TITLE`

The event title.

`DESCRIPTION`

The event description.

`EVENT_LOCATION`

The event location.

`EXTRA_EMAIL`

A comma-separated list of email addresses that specify the invitees.

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Add a calendar event

```
public void addEvent(String title, String location, Calendar begin, Calendar end) {
    Intent intent = new Intent(Intent.ACTION_INSERT)
            .setData(Events.CONTENT_URI)
            .putExtra(Events.TITLE, title)
            .putExtra(Events.EVENT_LOCATION, location)
            .putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, begin)
            .putExtra(CalendarContract.EXTRA_EVENT_END_TIME, end);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.INSERT" />
        <data android:mimeType="vnd.android.cursor.dir/event" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Capture a picture or video and return it

To open a camera app and receive the resulting photo or video, use the `ACTION_IMAGE_CAPTURE` or `ACTION_VIDEO_CAPTURE` action. Also specify the URI location where you'd like the camera to save the photo or video, in the `EXTRA_OUTPUT` extra.

Action

`ACTION_IMAGE_CAPTURE` or
`ACTION_VIDEO_CAPTURE`

Data URI Scheme

None

MIME Type

None

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Capture a picture or video and return it

Extras

`EXTRA_OUTPUT`

The URI location where the camera app should save the photo or video file (as a `Uri` object).

When the camera app successfully returns focus to your activity (your app receives the `onActivityResult()` callback), you can access the photo or video at the URI you specified with the `EXTRA_OUTPUT` value.

> **Note:** When you use `ACTION_IMAGE_CAPTURE` to capture a photo, the camera may also return a downscaled copy (a thumbnail) of the photo in the result `Intent`, saved as a `Bitmap` in an extra field named `"data"`.

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Capture a picture or video and return it

```
static final int REQUEST_IMAGE_CAPTURE = 1;
static final Uri mLocationForPhotos;

public void capturePhoto(String targetFilename) {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    intent.putExtra(MediaStore.EXTRA_OUTPUT,
            Uri.withAppendedPath(mLocationForPhotos, targetFilename));
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, REQUEST_IMAGE_CAPTURE);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bitmap thumbnail = data.getParcelable("data");
        // Do other work with full size photo saved in mLocationForPhotos
        ...
    }
}
```

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Compose an email with optional attachments

To compose an email, use one of the below actions based on whether you'll include attachments, and include email details such as the recipient and subject using the extra keys listed below.

Action

`ACTION_SENDTO` (for no attachment) or
`ACTION_SEND` (for one attachment) or
`ACTION_SEND_MULTIPLE` (for multiple attachments)

Data URI Scheme

None

MIME Type

`"text/plain"`

`"*/*"`

Building an Intent

http://developer.android.com/guide/components/intents-common.html

Extras

`Intent.EXTRA_EMAIL`

A string array of all "To" recipient email addresses.

`Intent.EXTRA_CC`

A string array of all "CC" recipient email addresses.

`Intent.EXTRA_BCC`

A string array of all "BCC" recipient email addresses.

`Intent.EXTRA_SUBJECT`

A string with the email subject.

`Intent.EXTRA_TEXT`

A string with the body of the email.

`Intent.EXTRA_STREAM`

A `Uri` pointing to the attachment. If using the `ACTION_SEND_MULTIPLE` action, this should instead be an `ArrayList` containing multiple `Uri` objects.

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Compose an email with optional attachments

Example intent:

```
public void composeEmail(String[] addresses, String subject, Uri attachment) {
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setType("*/*");
    intent.putExtra(Intent.EXTRA_EMAIL, addresses);
    intent.putExtra(Intent.EXTRA_SUBJECT, subject);
    intent.putExtra(Intent.EXTRA_STREAM, attachment);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Compose an email with optional attachments

If you want to ensure that your intent is handled only by an email app (and not other text messaging or social apps), then use the `ACTION_SENDTO` action and include the `"mailto:"` data scheme. For example:

```
public void composeEmail(String[] addresses, String subject) {
    Intent intent = new Intent(Intent.ACTION_SENDTO);
    intent.setData(Uri.parse("mailto:")); // only email apps should handle this
    intent.putExtra(Intent.EXTRA_EMAIL, addresses);
    intent.putExtra(Intent.EXTRA_SUBJECT, subject);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

Building an Intent

http://developer.android.com/guide/components/intents-common.html

- Compose an email with optional attachments

Example intent filter:

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <data android:type="*/*" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.SENDTO" />
        <data android:scheme="mailto" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

URI. System Ressourcen Access

URI. System Ressourcen Access 1

uri.getScheme(): http
uri.getSchemeSpecificPart(): //developer.android.com/reference/android/net/Uri.html
uri.getAuthority(): developer.android.com
uri.getHost(): developer.android.com
uri.getPath(): /reference/android/net/Uri.html
uri.getLastPathSegment(): Uri.html

```
Uri uri = Uri.parse("ftp:// bob@google.com:80/data/files");
```

```
Uri uri = Uri.parse("http://developer.android.com/reference/android/net/Uri.html");
```

uri.getScheme(): ftp
uri.getSchemeSpecificPart(): // bob@google.com:80/data/files
uri.getAuthority(): bob@google.com:80
uri.getHost(): google.com
uri.getPort(): 80
uri.getPath(): /data/files
uri.getLastPathSegment(): files
uri.getUserInfo(): bob

URI. System Ressourcen Access 2

Номер телефона

```
Uri uri = Uri.parse("tel:12345");
```

uri.getScheme(): tel
uri.getSchemeSpecificPart():12345

Координаты

```
Uri uri = Uri.parse("geo:55.754283,37.62002");
```

uri.getScheme(): geo
uri.getSchemeSpecificPart(): 55.754283,37.62002

URI. System Ressourcen Access 3

Контакт из адресной книги

```
Uri uri = Uri.parse("content://contacts/people/1");
```

uri.getScheme(): content

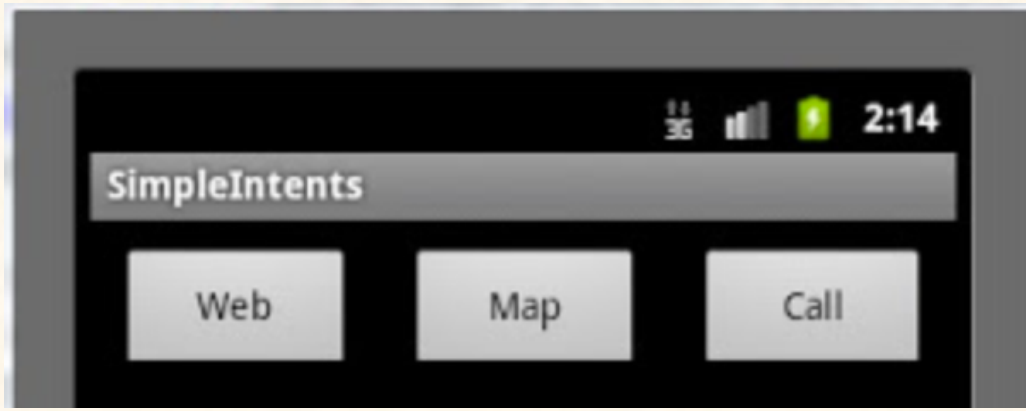
uri.getSchemeSpecificPart(): //contacts/people/1

uri.getAuthority(): contacts

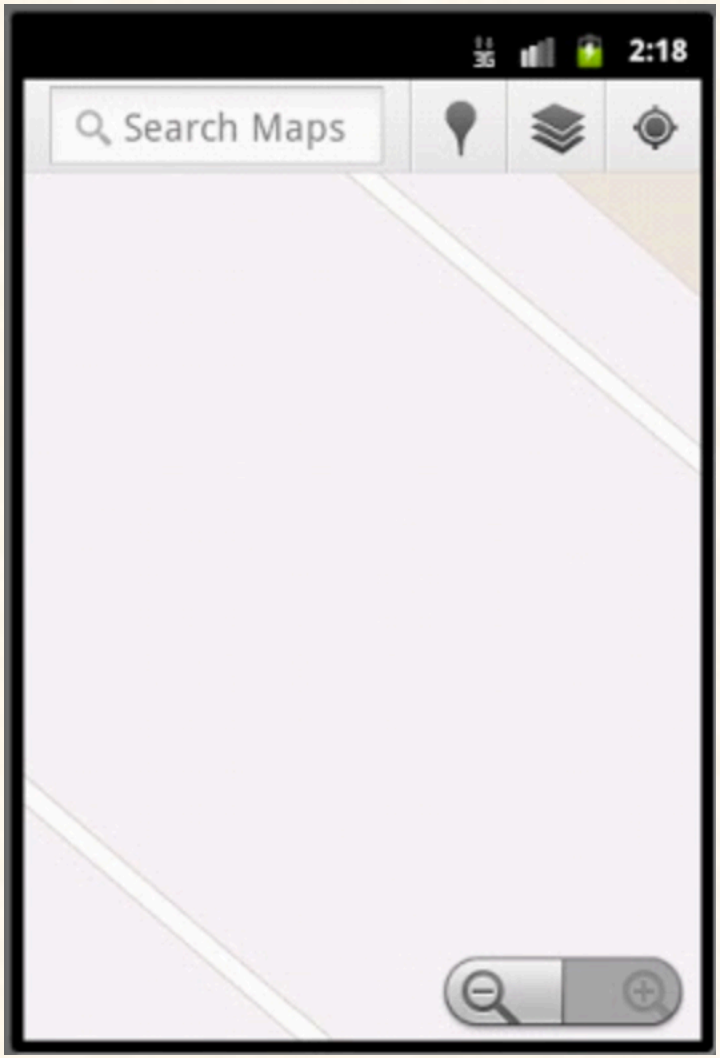
uri.getPath(): /people/1

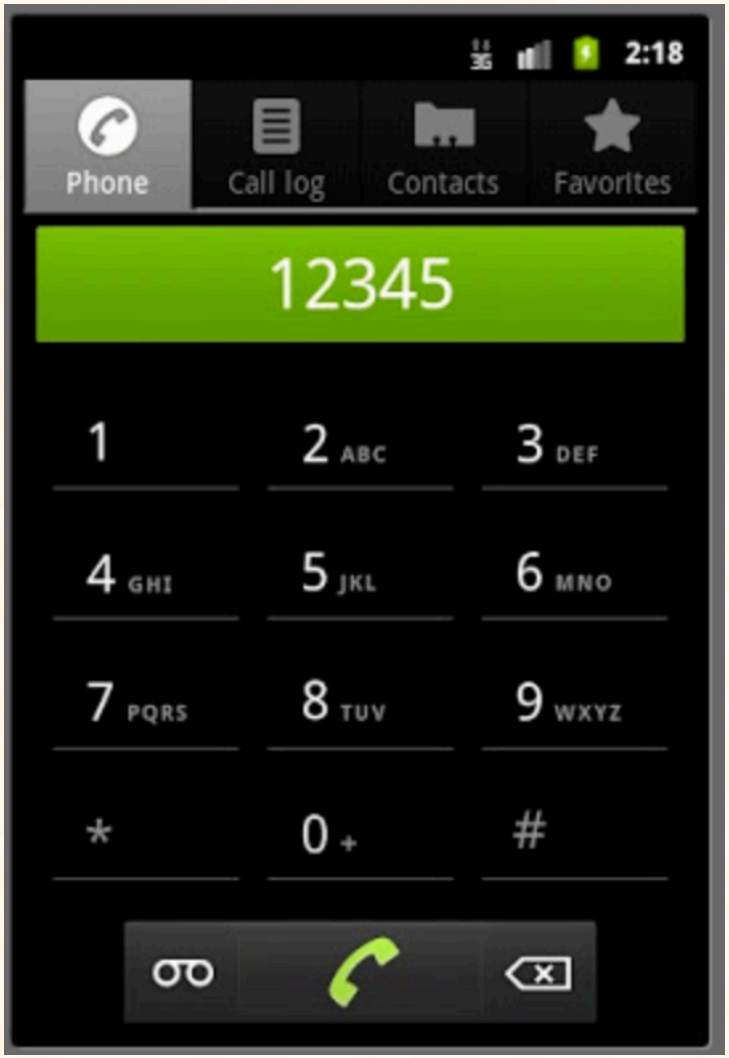
uri.getLastPathSegment(): 1

URI. System Ressourcen Access 4



URI. System Ressourcen Access 5





URI. System Ressourcen Access 6

```
@Override
public void onClick(View v) {
  Intent intent;
  switch (v.getId()) {
  case R.id.btnWeb:
    intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://developer.android.com"));
    startActivity(intent);
    break;
  case R.id.btnMap:
    intent = new Intent();
    intent.setAction(Intent.ACTION_VIEW);
    intent.setData(Uri.parse("geo:55.754283,37.62002"));
    startActivity(intent);
    break;
  case R.id.btnCall:
    intent = new Intent(Intent.ACTION_DIAL);
    intent.setData(Uri.parse("tel:12345"));
    startActivity(intent);
    break;
  }
}
```

WEB Ressource

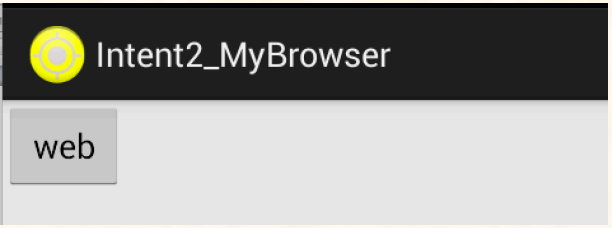
Map

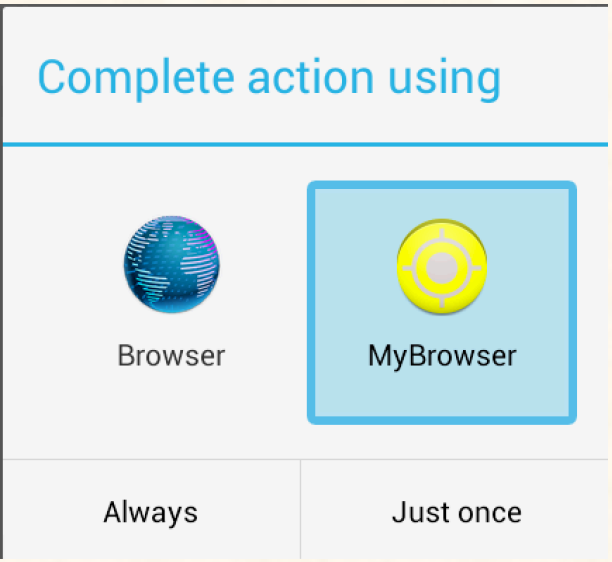
Call

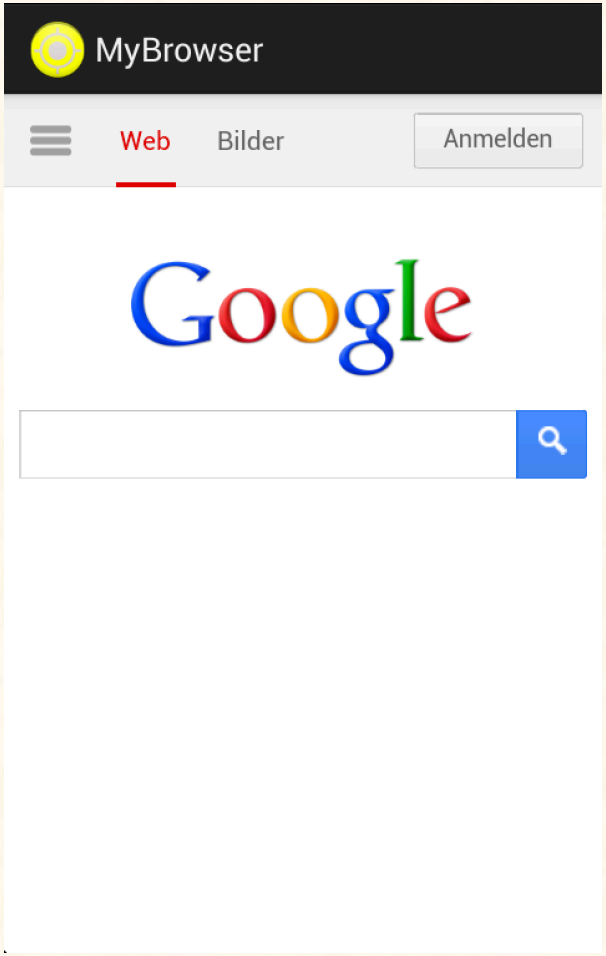
URI. Standart App & Activity 1

Projekt: Intent2_MyBrowser

URI. Standart App & Activity 1







URI. Standart App & Activity 2

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

    (findViewById(R.id.btnWeb)).setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View v) {
        startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.ya.ru")));
      }
    });
  }
}
```

URL

URI. Standart App & Activity 3

```
8  public class BrowserActivity extends Activity {
9
10   @Override
11   protected void onCreate(Bundle savedInstanceState) {
12     super.onCreate(savedInstanceState);
13     setContentView(R.layout.browser);
14     WebView webView = (WebView) findViewById(R.id.webView);
15     Uri data = getIntent().getData();
16     webView.loadUrl(data.toString());
17   }
18 }
```

URI. Standart App & Activity 4

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ua.opu.brovkov.intent2_mybrowser"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
    <uses-permission
        android:name="android.permission.INTERNET">
    </uses-permission>
```

URI. Standart App & Activity 5

```
14      <application
15          android:allowBackup="true"
16          android:icon="@drawable/ic_launcher"
17          android:label="@string/app_name"
18          android:theme="@style/AppTheme" >
19          <activity
20              android:name="ua.opu.brovkov.intent2_mybrowser.MainActivity"
21              android:label="@string/app_name" >
22              <intent-filter>
23                  <action android:name="android.intent.action.MAIN" />
24                  <category android:name="android.intent.category.LAUNCHER" />
25              </intent-filter>
26          </activity>
27          <activity
28              android:label="MyBrowser"
29              android:name="BrowserActivity">
30              <intent-filter>
31                  <action android:name="android.intent.action.VIEW"></action>
32                  <data android:scheme="http"></data>
33                  <category android:name="android.intent.category.DEFAULT"></category>
34              </intent-filter>
35          </activity>
36      </application>
37
38  </manifest>
```

URI mit Parameters 1

```
10 public class MainActivity extends Activity {
11     //Intent intent;
12     //Bundle bundle;
13   /** Called when the activity is first created. */
14   @Override
15   public void onCreate(Bundle savedInstanceState) {
16     super.onCreate(savedInstanceState);
17     setContentView(R.layout.activity_main);
18   }
19   public void onClick(View v) {
20       if (v.getId() == R.id.btnURI){
21           startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.de")));
22       }else if(v.getId() == R.id.btnURI_Params){
23           String uri_with_params = "http:///www.google.de?Param1=Value1&Param2=Value2";
24           Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri_with_params));
25           startActivity(intent);
26       }else{
27           Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http:///www.google.de"));
28           Bundle bundle=new Bundle();
29           bundle.putString("Param3", "Value3");
30           bundle.putString("Param4", "Value4");
31           intent.putExtras(bundle);
32           startActivity(intent);
33       }
34   }
35 }
```

1

2

3

URI mit Parameters 2

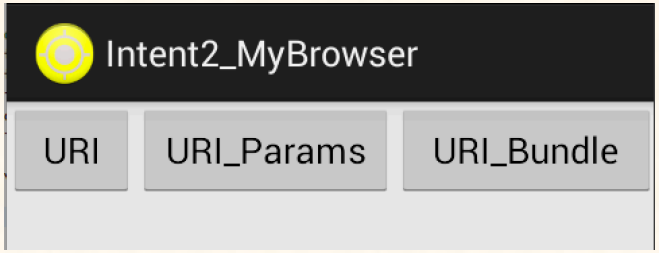
```
9  public class BrowserActivity extends Activity {
10   @Override
11   protected void onCreate(Bundle savedInstanceState) {
12     super.onCreate(savedInstanceState);
13     setContentView(R.layout.browser);
14     WebView webView = (WebView) findViewById(R.id.webView);
15     Intent intent = getIntent();
16     Uri uri = intent.getData();
17     String s1 = uri.toString();
18     String s2 = uri.getQueryParameter("Param1");
19     String s3 = uri.getQueryParameter("Param2");
20     Bundle bundle=intent.getExtras();
21     if(s2 != null){
22         this.setTitle(s2 + " " + s3);
23     }
24     if(bundle != null){
25         String s4 = bundle.getString("Param3", " ");
26         String s5 = bundle.getString("Param4", " ");
27         this.setTitle(s4 + " " + s5);
28     }
29     webView.loadUrl(s1);
30   }
31 }
```

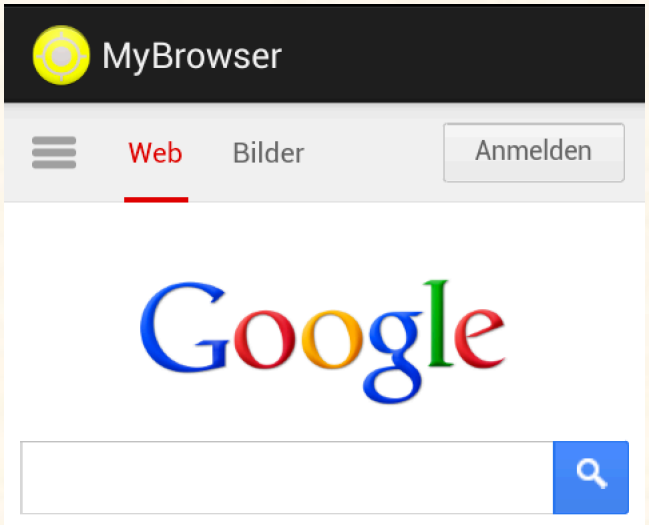
1

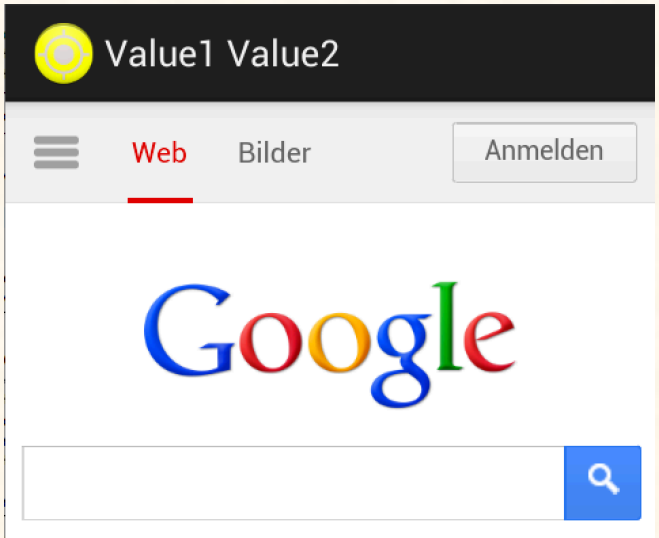
2

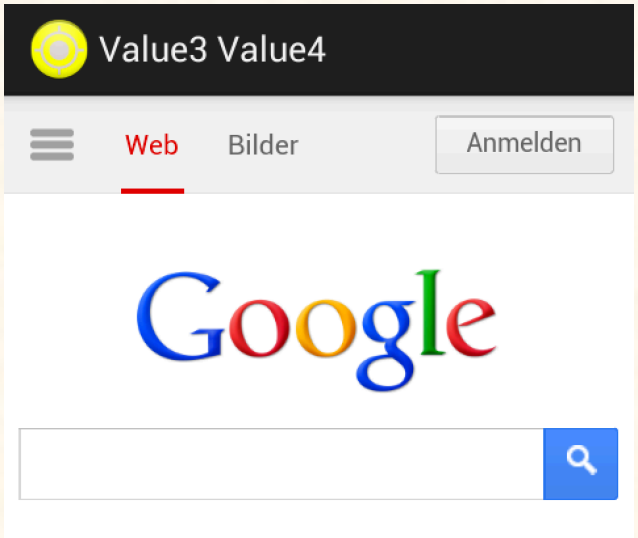
3

URI mit Parameters 3









Fragen?