Kartendarstellungen mit Google Maps

- Package `com.google.android.maps`, ist nicht Bestandteil von Android!
- Voraussetzungen für die Nutzung:
  - Verfügbarkeit auf dem Gerät
  - installierter API-Key
  - Eintrag der Maps-Bibliothek im Manifest innerhalb des `application`-Elements:
    ```
    <uses-library android:name="com.google.android.maps" />
    ```
- Berechtigung für den Internet Zugriff:

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

Wichtige Klassen von Google Maps

- `MapView`: View zur Darstellung eines Kartenausschnitts
- `MapActivity`: zwingend erforderliche spezialisierte Activity-Klasse für MapViews, sorgt im Hintergrund für die Verwaltung des Kartenmaterials
- `MapController`: Zur Steuerung des Kartenausschnitts, z.B. Zooming
- `Overlay`: allgemeine Transparente Schicht zum Zeichnen
- `MyLocationOverlay`: spezialisierte Overlay-Klasse um die aktuelle Position und Bewegungsrichtung anzuzeigen
- `ItemizedOverlay` und `OverlayItem`: Overlays für die Markierung von Positionen

MapActivity

- Zur Anzeige von Karten (`MapView`) muss eine Activity definiert werden, die von `MapActivity` abgeleitet ist.
- Innerhalb der Activity definiert man typischerweise Variablen für die `MapView` und einen `MapController`.
- zu überschreibende abstrakte Methode `isRouteDisplayed()`: gibt an, ob Routeninformation angezeigt wird

```
public class MyMapActivity extends MapActivity {

    private MapView       mapView;
    private MapController mapController;

    @Override
    public void onCreate(Bundle savedINstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.map_layout);
        mapView = (MapView) findViewById(R.id.map_view);
    }

    @Override
    protected boolean isRouteDisplayed() {
        // IMPORTANT: This method must return true if your Activity
        // is displaying driving directions. Otherwise return false.
        return false;
    }
```

MapView

- abgeleitet von `android.view.ViewGroup`, Zugriff z.B. mittels `findViewById()`
- erfordert zwingend eine `MapActivity`
- stellt die bekannten Anzeigemodi von Google Maps bereit: Kartendarstellung, Satellitenansicht mit und ohne Straßenverläufen, Street View, Verkehrsinformation
- verwaltet eine Liste von Overlays
- ermöglicht Umrechnungen zwischen Geokoordinaten und Bildschirmkoordinaten

Kartendarstellungen

Default: Straßenkarte

Methoden in `com.google.android.maps.MapView`:

- An- und Ausschalten der verfügbaren Anzeigemodi:
  ```
  public void setSatellite(boolean on)
  public void setTraffic(boolean on)
  public void setStreetView(boolean on)
  ```

- Überprüfen, ob ein bestimmter Anzeigemodus aktiv ist:
  ```
  public boolean isSatellite()
  public boolean isTraffic()
  public boolean isStreetView()
  ```

- aktuelle bzw. maximale Zoom-Stufe ermitteln:

```
int maxZoom = mapView.getMaxZoomLevel();
int currentZoom = mapView.getZoomLevel();
```

- Geo-Koordinaten des Punktes im Zentrum der MapView emitteln:

```
GeoPoint center = mapView.getCenter();
```

- Standard Zoom-Controller anzeigen:

```
mapView.setBuiltInZoomControls(true);
```

MapController

`com.google.android.maps.MapController`: ermöglicht programmgesteuert das Zoomen und Verschieben des Kartenausschnitts

Wichtige `MapView`-Methoden:

- Zugriff auf den `MapController` einer `MapView`:
  `public MapController getController()`

- Ermittlung der höchsten Zoom-Stufe für den aktuellen Kartenausschnitt:
  `public int getMaxZoomLevel()`

  Berücksichtigt den Anzeige-Modus!

- View für die Zoom-Steuerung anzeigen:
  `public void setBuiltInZoomControls(boolean on)`

  Ersetzt `getZoomControls()`, deprecated

`MapController`-Methoden zur Zoomsteuerung:

- Einstellen einer Zoom-Stufe:
  `public int setZoom(int zoomLevel)`

- Um eine Zoom-Stufe vergrößern:
  `public boolean zoomIn()`:
  `public boolean zoomInFixing(int xPixel, int yPixel)`

  Die zweite Methode fixiert das angegebene Pixel auf dem Schirm.

- Um eine Zoom-Stufe verkleinern:
  ```
  public boolean zoomOut()
  public boolean zoomOutFixing(int xPixel, int yPixel)
  ```

- Zooming auf ein Rechteck anpassen:
  ```
  public void zoomToSpan(int latSpanE6, int lonSpanE6)
  ```
  Größe des Ausschnitts wird durch `latSpanE6` und `lonSpanE6` in Mikrograd angegeben.

`MapController`-Methoden für Positionierung und Scrolling:

- Mittelpunkt der Karte ausrichten:
  ```
  public void setCenter(GeoPoint point)
  ```
  Die Klasse `com.google.android.maps.GeoPoint` kapselt eine unveränderliche geographische Position, Angaben sind in Mikrograd

- Animierter Übergang zu einer geographischen Position:

```
public void animateTo(GeoPoint point)
public void animateTo(GeoPoint point, android.os.Message message)
public void animateTo(GeoPoint point, java.lang.Runnable runnable)
public void stopAnimation(boolean jumpToFinish)
```

- Kartenausschnitt scrollen, pixelorientiert, keine Animation:

```
public void scrollBy(int x, int y)
```

In die Karte zeichnen mit Overlays

- abstrakte Klasse `com.google.android.maps.Overlay`
- Eigene Overlay-Klasse von `Overlay` ableiten und Methode `draw()` überschreiben
- `public void draw(android.graphics.Canvas canvas, MapView mapView, boolean shadow)`

  Auf `canvas` können wir wie gewöhnlich zeichnen, `mapView` ist die dem Overlay zugeordnete MapView

  Die Methode wird einmal mit `shadow=true` aufgerufen, um den Layer für die Schattierungen zu zeichnen, und dann ein weiteres mal mit `shadow=false`.

- Üblicherweise vor dem eigentlichen Zeichnen `draw`-Methode der Oberklasse aufrufen.

Overlay mit MapView verbinden:

- `MapView` verfügt über eine Methode zum Zugriff auf die Liste der Overlays:
  `public final java.util.List<Overlay> getOverlays()`

- Anschließend kann man mit der Methode `add()` der Liste ein Overlay hinzufügen:

```
MapView mapView = findViewById(R.id.mapView);
Overlay overlay = new MeinOverlay();
mapView.getOverlays().add(overlay);
mapView.postInvalidate();
```

Geo- zu Bildschirmkoordinaten

Problem beim Zeichnen: Welche Bildschirmkoordinaten entsprechen den Geokoordinaten, die ich zeichnerisch markieren möchte?

Hierzu Umwandlungsmechanismen in Schnittstelle `com.google.android.maps.Projection`:

- `android.graphics.Point toPixels(GeoPoint in, android.graphics.Point out)`

  Wandelt die Geokoordinaten von `in` in Bildschirmkoordinaten. Die Bildschirmkoordinaten werden in dem Objekt `out` abgelegt, falls `out==null` wird eine neue `Point`-Instanz erzeugt.

- `GeoPoint fromPixels(int x, int y)`

  Zu den Bildschirmkoordinaten `x` und `y` werden die Geokoordinaten berechnet.

Wie erhält man eine `Projection`-Instanz:

- `MapView`-Methode:
  `public Projection getProjection()`

- Eine `Projection`-Instanz sollte nur für ein `draw()` verwendet werden, da die Umwandlung vom Kartenausschnitt abhängt.

- Üblicherweise Zugriff auf die `Projection`-Instanz innerhalb der `draw()`-Methode von `Overlay`, siehe Parameter.

MyLocationOverlay

- spezialisiertes Overlay zur Anzeige der eigenen Postion und Bewegungsrichtung auf einer Karte (`MapView`)
- implementiert u.a. `SensorListener` und `LocationListener`
- Erzeugen und der Liste der Overlays einer MapView hinzufügen:

```
List<Overlay> overlays = mapView.getOverlays();
MyLocationOverlay myLocationOverlay = new MyLocationOverlay(context, m
overlays.add(myLocationOverlay);
```

- Positionsmarkierung und Markierung für Bewegungsrichtung einschalten:

```
myLocationOverlay.enableCompass();
myLocationOverlay.enableMyLocation();
```

ItemizedOverlay

- `ItemizedOverlay` ist eine (vor-)spezialisierte Overlay-Klasse für Markierungen von Positionen mit Grafiken und Text.
- Ein ItemizedOverlay verwaltet `OverlayItem`s. Instanzen dieser Klasse kapseln ein `Drawable` für die Anzeige, einen `GeoPoint` sowie Texte für die Anzeige.
- Die Klasse ist abstrakt mit den abstrakten Methoden `createItem()` und `size()`, die überschrieben werden müssen.
- `abstract int size()`

  Anzahl der Items in dem Overlay.

- `protected abstract Item createItem(int i)`

  Liefert das `i`-te Item des Overlays.

- Falls eine Änderung am ItemizedOverlay vorgenommen wurde, muss anschließend die Methode `populate()` aufgerufen werden.

Für die interne Verwaltung der `OverlayItem`-Instanzen ist man selbst zuständig. Entsprechende Logik ist in der Unterklasse zu `ItemizedOverlay` zu definieren.

OverlayItem

Die wichtigsten Methoden:

- `OverlayItem(GeoPoint point, String title, String snippet)`

  Konstruktor

- `void setMarker(Drawable marker)`

  Sets the marker to be used when drawing this item on the map.

- `Drawable getMarker()`
  `GeoPoint getPoint()`
  `String getTitle()`
  `String getSnippet()`

Beispiel zu ItemizedOverlay

```
public class MyItemizedOverlay extends ItemizedOverlay<OverlayItem>  {

    private ArrayList<OverlayItem> items;

    public MyItemizedOverlay(Drawable defaultMarker) {
        super(boundCenterBottom(defaultMarker));
        items = new ArrayList<OverlayItem>();
        populate();
    }

    @Override
    protected OverlayItem createItem(int index) {
        return items.get(index);
    }
```

```
@Override
public int size() {
    return items.size();
}

public void addItem(GeoPoint location, String markerText, String snippet) {
    items.add(new OverlayItem(location, markerText, snippet));
    populate();
}

public void removeItem(int index) {
    items.remove(index);
    populate();
}
```

API Schlüssel

- Für die Nutzung von Google Maps wird ein Maps API-Key benötigt.
- Hierzu wird das Zertifikat benötigt, mit dem man die eigene Anwendung signiert.
- Während der Entwicklung: Debug-Zertifikat
- Wo liegt das Debug-Zertifikat? siehe Eclipse→Preferences→Android→Build

- Mit `keytool` erzeugen wir einen MD5-Fingerabdruck zu dem Zertifikat.

```
> keytool -list -alias androiddebugkey -keystore pfad
    -storepass android -keypass android
```

- Beispielausgabe:

```
androiddebugkey, 19.12.2008, keyEntry,
Zertifikatsfingerabdruck (MD5): BA:D5:65:BE:71:E5:86:6C:47:F1:9F:8B:B4:6A:06:B8
```

- Mit dem Formular unter
  `http://code.google.com/android/maps-api-signup.html`
  und dem MD5-Fingerabdruck kann der Maps API-Key erzeugt werden.

- Der Maps API-Key muss im MapView-Element der zugehörigen Layout-Definition eingetragen werden:

```
<LinearLayout ...
...
<com.google.android.maps.MapView
   android:id="@+id/map_view"
   ...
   android:apiKey="0Z4LcIRpu4It9t8q_891FlbbweXMDNDOr5up2KA"/>
...
</LinearLayout>
```