Lesson 3:

Navigation With Coordinates

Introduction

This lesson shows the navigation with MapTrip Interface based on coordinates.

Alternately, interface calls and callbacks are used to perform a slightly more complex action.

During this lesson you will

- ensure that MapTrip is not currently performing an active navigation
- delete (possible) existing destinations from a route
- add coordinates to a list of destinations
- start a navigation or (if no GPS is available) start a navigation simulation
- react to the callback that the destination is reached

Prerequisites

You should finish the first lesson successfull to understand the basic mechanisms of MTI.

This lesson also shows how to switch between MapTrip and your own app, as already shown in lesson 2.

Nevertheless you can examine this lesson without considering the lessons before because the basics like initializing the API and ensuring MapTrip is running are already implemented in lesson 1.

The project has to be cloned to your computer. For testing the implementation, MapTrip should be installed at your device or device emulation.

Initiate a Navigation with MapTrip via MTI

Navigation from one point to another (or some other) points requires different process steps:

- setting a starting point
- setting one or more destination points
- calculating the route between the points

Using MapTrip or MTI the points are geo-coordinates, consisting of latitude and longitude.

Start with Project

If not done before you should clone the tutorial project from github.

Cloning the MTI Tutorial

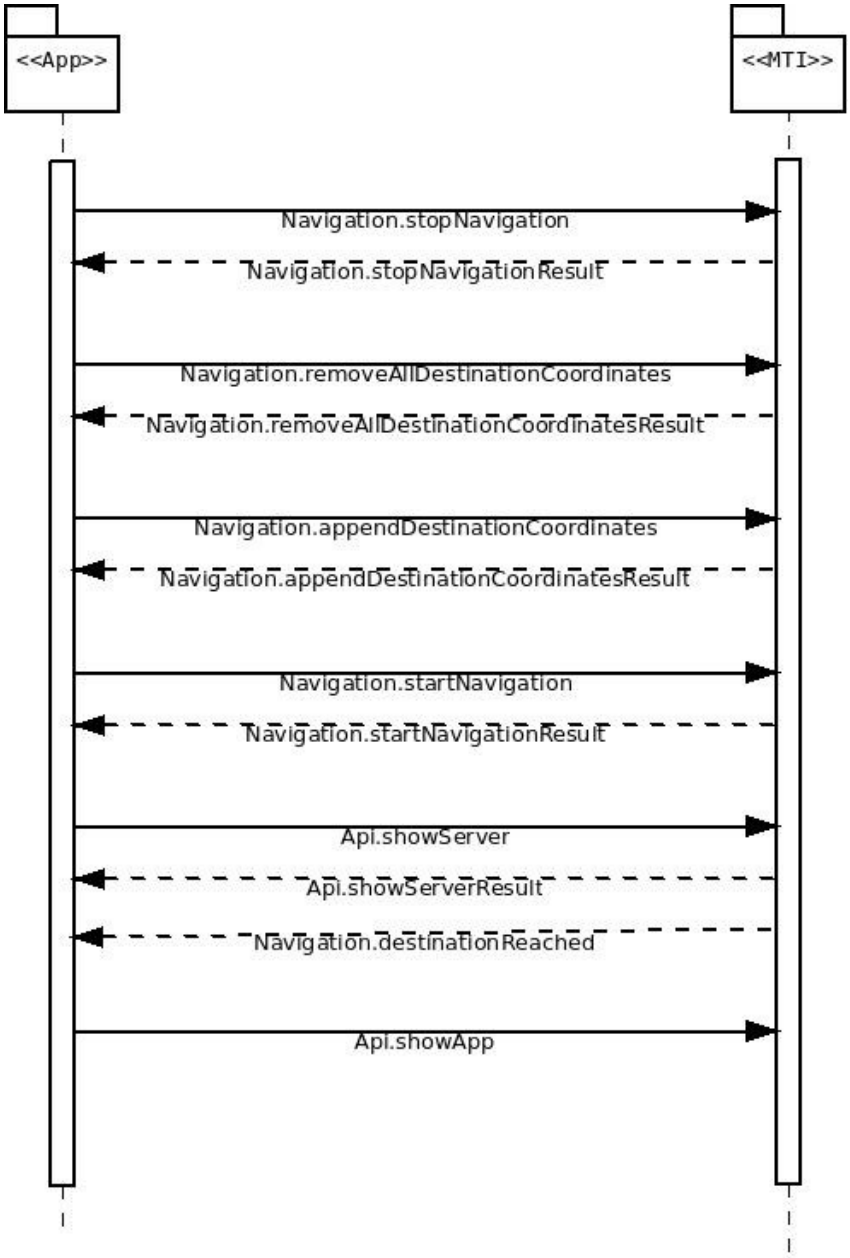
- SSH: git@github.com:MapTrip-Navigation/MTI-Tutorial.git
- HTTPS: https://github.com/MapTrip-Navigation/MTI-Tutorial.git
- GitHub CLI: gh repo clone MapTrip-Navigation/MTI-Tutorial

Open lesson 3 with Android Studio

In Android Studio Click at File menu and select the module ***lesson2_navigateWithCoordinates***.

HowTo: Performing navigation step-by-step

The graphic shows the communication process between your app and MTI:



Navigation.stopNavigation() is the initial call to the Navigation module. After that the callbacks from MTI are used like triggers, each followed by a further function call.

After the successfull navigation start, MapTrip is brought to the foreground.

With navigation finish, the app comes back to foreground.

Extend the app with the next button

Add the next button in file file ui/home/HomeFragment.java:

```
/*
 * Within this method we will extend the button list for available functions lession by
lession
 */
private ArrayList getLessions () {
    int lessonIndex = 0;
    // Placeholder for later implemented lessons
    lessonArrayList.add(new Lesson1_Initialize(lessonIndex++,"Start MapTrip And Init
MTI", this));
    lessonArrayList.add(new Lesson2_ShowApps(lessonIndex++,"Show MapTrip 10 seconds",
this));
    lessonArrayList.add(new Lesson3_NavigateWithCoordinates(lessonIndex++, "Route in
Cologne", this));

    return lessonArrayList;
}
```

Usage and implementing the MTI module Navigation

For the first two lessons we only needed the module Api. Our new little project requires the usage of the module Navigation. The callback interfaces must be implemented.

Open the java file **MtiListener.java** and implement the NavigationListener**:**

```
public class MtiListener implements ApiListener, NavigationListener {
...
```

Let Android Studio complement the interface methods.

Later our app has to react to some callbacks. Do you remember how we ‚provided' callbacks to the lessons?

Similar to a pattern, we added a few lines of code to each callback to be passed on to one of the lessons. Let's have a look to the callback infoMsg():

```
@Override
public void infoMsg(Info info, int i) {
    // prevent recursive calls if subclass didn't implement this method
    if (!this.getClass().equals(MtiListener.class)) {
        return;
    }

    for (Lesson lesson : registeredLessons.values()) {
        lesson.infoMsg(info, i);
    }
}
```

Complete the following methods according to this principle

- stopNavigationResult()
- removeAllDestinationCoordinatesResult()
- appendDestinationCoordinateResult()
- startNavigationResult()
- startSimulationResult()
- destinationReached()

The class Lesson.java

In lesson 2 the callback *showServerResult()* is overwritten. To avoid that this function brings the app to front, the boolean navigationActivated helps as a switch.

```
public abstract class Lesson extends MtiListener {
    private int functionId;
    private String buttonCaption;
    protected Fragment fragment;

    protected static boolean statusInitialized = false;
    protected static boolean listenerRegistered = false;
    protected static boolean navigationActivated = false;
```

The class Lesson1_Initialize.java

Due to usage the MTI module Navigation and its callbacks the listener must be registered also.

```
private void registerListener() {
    if (!listenerRegistered) {
        MTIHelper.initialize(fragment.getContext());
        Api.registerListener(getMtiListener());
        Navigation.registerListener(getMtiListener());
        listenerRegistered = true;
    }
}
```

The class Lesson2_ShowApps.java

Considering the boolean navigationActivated (see above).

```
@Override
public void doSomething() {
    // At first check if MTI is ready to use
    if (!statusInitialized) {
```

```
        return;
    }

    // to differ between the behaviour between the second and the third button
    navigationActivated = false;
    setMapTripToFront();
}
```

The class *Lesson3_NavigateWithCoordinates.java* snippet by snippet

Click to the new button performs doSomething(). This is the initial entry point.

```
@Override
public void doSomething() {
    // At first check if MTI is ready to use
    if (!statusInitialized) {
        return;
    }

    beginNewNavigation();
}
```

If MapTrip executes any active navigation, beginNewNavigation() calls Navigation.stopNavigation() to end the navigation. MTI answers with stopNavigationResult().

```
// 1. initial step: be sure that no routing is active
private void beginNewNavigation() {
    stopNavigationRequestId = Navigation.stopNavigation();
}
```

After stopping the (possible) navigation the destination list should be cleared. If not with the next navigation start, still existing destinations will be considered for routing calculation. Please note that in this example not all potential fired ApiErrors are checked. The ApiError.NO_ROUTE is thrown if there was no active navigation.

```
// 1a. possible routing stopped
@Override
public void stopNavigationResult(int requestId, ApiError apiError) {
    // relates the returned requestId our stopNavigationId?
    if (stopNavigationRequestId != requestId) {
        return;
    }

    // check error
    if (apiError.equals(ApiError.OK) || apiError.equals(ApiError.NO_ROUTE) ||
apiError.equals(ApiError.NO_DESTINATION)) {
```

```
        removeAllDestinationCoordinates();
    }
}
```

After the list of destinations is cleared the new destinations will be added for our route.

```
// 2. clear (possible) destinations from destination list
private void removeAllDestinationCoordinates() {
    removeAllDestinationCoordinatesRequestId =
Navigation.removeAllDestinationCoordinates();
}


// 2a. destination list cleared – append new coordinates
@Override
public void removeAllDestinationCoordinatesResult(int requestId, ApiError apiError) {
    // relates the returned requestId our requestId?
    if (removeAllDestinationCoordinatesRequestId != requestId) {
        return;
    }

    // check error
    if (apiError.equals(ApiError.OK)) {
        appendCoordinates();
    }
}
```

The next method adds the destinations to the internal list. Every *appendDestinationCoordination()* call will be responded by a callback. Since it is unlikely that an error will occur, we only react to the last corresponding callback. The coordinates are near by the Cologne Cathedral.

```
// 3. no active route and destination list is empty - set destination to list
private void appendCoordinates() {
    // if GPS doesn't work we will start a simulation
    // for simulation two coordinates are required

    // first coordinate near the Cologne Cathedral
    Navigation.appendDestinationCoordinate(50.942666455132745, 6.957345467113134);
    // second coordinate in some hundred meters distance
    appendCoordinateRequestId =
Navigation.appendDestinationCoordinate(50.94158521260109, 6.953404794243524);
}


// 3a. new coordinate was added to destination list
@Override
public void appendDestinationCoordinateResult(int requestId, int listIndex, ApiError
apiError) {
    if (appendCoordinateRequestId != requestId) {
        return;
    }

    // while adding coordinates we set the requestId with the last call of
appendDestinationCoordinates()
```

```
    // so all appends before are ignored here and only with the last callback navigation
will be started

    // check error
    if (apiError.equals(ApiError.OK) || apiError.equals(ApiError.NO_ROUTE)) {
        startNavigation();
    }
}
```

In case no GPS signal is available, the method startSimulation() is implemented as an alternative to startNavigation(). This decision will be done in startNavigationResult().

```
// 4. destination list updated - start routing
private void startNavigation() {
    startNavigationRequestId = Navigation.startNavigation();
}

// 5. navigation could not be started - perhaps no GPS, so we try simulation
private void startSimulation() {
    startSimulationRequestId = Navigation.startSimulation();
}

// 4a. navigation successfull started - show MapTrip
@Override
public void startNavigationResult(int requestId, ApiError apiError) {
    if (requestId != startNavigationRequestId) {
        return;
    }

    switch (apiError) {
        case OK:
            // bring MapTrip to front
            Api.showServer();
            navigationActivated = true;
            break;

        case ROUTING:
            // perhaps the routing couldn't be started because of missed GPS signal
            // in production apps the check must be more detailed
            // in this tutorial we simply try to start a simulation
            startSimulation();
            break;

        default:
            break;
    }
}

// 5a. simulation successfully started - show MapTrip
@Override
public void startSimulationResult(int requestId, ApiError apiError) {
    if (requestId != startSimulationRequestId) {
        return;
    }

    switch (apiError) {
```

```
        case OK:
            // bring MapTrip to front
            Api.showServer();
            navigationActivated = true;
            break;

        default:
            // some error handling here
            break;
    }
}
```

With the last methods the process will be finished and MapTrip brought to front. At the end of the navigation again the app comes back to front.

```
// 6. Navigation  or Simulation started - bring MapTrip to front
private void showMapTrip() {
    Api.showServer();
}

// 6a. error handling if MapTrip couldn't be set to front
@Override
public void showServerResult(int requestId, ApiError apiError) {
    // some error handling here if server couldn't be shown
}

// when destination reached bring tutorial app to front
@Override
public void destinationReached(int requestId) {
    setTutorialAppToFront();
}

// this method is a copy of Lesson2_ShowApps.java
private void setTutorialAppToFront() {
    String className = fragment.getActivity().getClass().getCanonicalName();
    String packageName = fragment.getActivity().getPackageName();

    // MTI call
    Api.showApp(packageName, className);
}
```

Don’t forget to click at first at the button START MAPTRIP AND INIT MTI ensuring that MapTrip is running and the MTI lib is initialized.

Abstract

At the end of this lesson your app can control MapTrip for navigation purposes.

The code fragments mentioned in this lesson are also stored in the folder content/lesson3_navigateWithCoordinates. This are:

- Lesson3_NavigateWithCoordinates.java (controls MapTrip as navigation app)
- HomeFragment.java (extended with a button for this lesson)
- Lesson.java (with one more boolean member)
- Lesson1_Initialize.java (registering the Navigation module)
- Lesson2_ShowApps (using a boolean avoiding showing the app)
- MtiListener.java (adding Navigation callbacks)

The complete github project is hosted here: https://github.com/MapTrip-Navigation/MTI-Tutorial