Android vehicle tracking source code

I'm not robot

reCAPTCHA

Continue

Almost everyone who drives has forgotten where they are parked. Places such as stadiums, shopping malls, festivals and other similar places have parking spaces that are miles long from end to end. Wouldn't it be nice to just know where your car is? As it turned out, there are applications for this. Unfortunately, there are no apps that will track the stolen car. You will need to buy equipment for this. However, we can help you find where you parked your car! Here are the best car tracking apps for Android.HurdlrPrice: Free/$5.99 per month/$79.99 per year. It tracks your daily activities as a function of your work. You can easily track how far you go, how much money you spend on gas, and other statistics like this. You can use these numbers in tax time to get a proper deductible. It has integration with Square, Uber, PayPal and tons of other companies. It also has a few additional tools for self-employed people. It's not what you traditionally think in this space. However, it is actually very useful for people in this area of work and it should easily save more on taxes than it costs per year. You can check out the app just below or the official website here. Google MapsPrice: FreeGoogle Maps can do a lot of things. This is what makes it one of the best car tracking apps. An update in 2017 has given you the ability to keep where your car was on the map. You then use Maps to find your way back to your car. It has many other features as well. You know most of them. This will give you step-by-step directions, show different transit information, and you can even re-route your commute to avoid obstacles like traffic or construction. This is one of a kind applications. Those who use this app should just keep using it to track where your car is. SpotAngelsPrice: FreeSpotAngels is probably the best of car tracking apps. This is quick and easy to save where you parked for an easy recall later. In addition, the app can show you no parking zones, so you can avoid parking tickets. A parking space is available everywhere. However, some parking tickets are only available in a few cities. It has expanded significantly over the past year, but the choice is still fairly limited. Fortunately, people can report places where they are likely to get tickets. This information will be available to other people who use the app. It is a hidden gem for people who live in big cities. App for car manufacturersPrice: make a bigger push in the mobile space. Many car companies now have apps that connect to their cars. Ford, Chevy, Lexus, and many others are doing this. Apps aren't all that good. However, every now and then they can be really useful. We recommend you try your luck with one of these apps if your car is new enough to support it. At worst, there are other on the list that will get the job done for sure. These applications will probably improve over time to do better. Dedicated car tracking equipmentPrice: Free applications / Equipment costs varyThe safest and most stable option for tracking cars is dedicated vehicle tracking equipment. The most popular example is LoJack. You install a module on your car and then use the app to see where your car is, track it if it gets stolen, and even receive alerts when your alarm goes off. There are other options. This gets you a lot further than something like Google Maps. The downside is the price. LoJack SureDrive costs about $700 to install in your car according to its frequently asked questions. It's kind of like fitness trackers, though. The hardware still does it better than just software and you get extra security features as well. If we missed any great car tracking apps, tell us about them in the comments! You can also click here to check out our latest Android app and game lists! App widgets can be seen as a small window or controller for an Android app that can be built into another app (such as a home screen). They can be very useful, allowing users to view or manage an app without actually running it. For example, skipping tracks with a music player widget or viewing weather information. The great thing about widgets is that they can be updated automatically (after a period of time) or in response to user actions. In this developer tutorial, we're going to create a simple Android widget that updates automatically every 30 minutes, or in response to the user pressing the update button on the widget. Our widget generates and displays a random number on each update (whether automatic or user interaction). It takes four steps to create a widget: Designing a widget layout. At least you'll need one mock-up file that describes the widget layout. However, you can also provide additional layout files for the widget before it receives any data. Widget on the lock screen (Android 4.0 and above). The widget is on the lock screen before it gets any data (Android 4.0 and above). Expand AppWidgetProvider. This class provides methods called during the widget lifecycle. Provide appWidgetProviderInfo metadata. Basic widget information such as minimum width and height, upgrade frequency and more. Add a widget to the app's manifesto. The design of the widget, the first thing we do is design our Layout. When laying out a widget application similar to a statement of activity and/or snippet, there is a very important factor to note. App widget layouts are based on RemoteViews layouts. This means that not all View subclasses can be used in the widget. In fact, only supported classes are FrameLayout, LinearLayout, RelativeLayout, GridLayout, AnalogClock, button, chronometer, imageButton, ImageView, ProgressBar, TextView, ViewFlipper, ListView, GridView, GridView, и АдаптерВьюФлиппер. Подклассы и потомки этих даже не поддерживаются. С этим в виду, мы разрабатываем наш виджет макет, названный simple_widget.xml<LinearLayout xmlns:android= android:orientation=horizontal android:layout_width=match_parent android:layout_height=match_parent android:padding=@dimen/widget_margin android:background=#55000000> <ImageView android:id=@+id/imageView android:layout_width=0dp android:layout_height=match_parent android:layout_weight=1 android:src=@drawable/aa></ImageView> <TextView android:id=@+id/textView android:layout_width=0dp android:layout_height=match_parent android:layout_weight=1 android:gravity=center android:text=000 android:textsize=@dimen/abc_text_size_large_material android:textstyle=bold></TextView> <Button android:id=@+id/actionButton android:layout_width=0dp android :layout_height=match_parent android:layout_weight=1 android:text=Refresh></Button> </LinearLayout> Примечание андроид: обивка в фрагменте кода выше. С Android 4.0 виджеты приложений автоматически получают обивку между рамкой виджета и границами виджетов. Устройства Pre-4.0, однако, не обеспечивают автоматическую обивку для виджетов. Чтобы построить виджет, который имеет поля для более ранних версий, но без дополнительных полей для 4.0 и выше, создать два измерения ресурсов resивалues/dimens.xml и res/values-v14/dimens.xml, чтобы обеспечить различные значения для марки виджета, и установите свою цельSdkVersion до 14.res/values/dimens.xml<resources> <dimen name=widget_margin>8dp</dimen> </resources> res/values-v14/dimens.xml<resources> <dimen name=widget_margin>0dp</dimen> </resources> Расширение AppWidgetProviderNow расширить AppWidgetProvider, создав класс SimpleWid. AppWidgetProvider имеет методы, которые называются, когда виджет приложения обновляется, удаляется, включен и отключен среди других. Для нашей реализации, мы только переопределить onUpdate (), потому что это метод, называемый всякий раз, когда виджет добавляется в host.public класса SimpleWidgetProvider расширяет AppWidgetProvider - @Override публичная пустота onUpdate (Контекст контекст, AppWidgetManager приложениеWidgetManager, int' аррWidgetIds) < count; i++) { int widgetId = appWidgetIds[i]; String number = String.format(%03d, (new Random().nextInt(900) + 100)); RemoteViews remoteViews = new RemoteViews(context.getPackageName(), R.layout.simple_widget); remoteViews.setTextView(R.id.textView, number); Intent intent = new Intent(context, SimpleWidgetProvider.class); intent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE); intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds); PendingIntent pendingIntent = PendingIntent.getBroadcast(context, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT); pendingIntent); count;= i++)= {= int= widgetid=appWidgetIds[i]; string= number=String.format(%03d, (new= random().nextint(900)= += 100));= remoteviews= remoteviews=new remoteviews(context.getpackagename(),= r.layout.simple_widget);= remoteviews.settextviewtext(r.id.textview,= number);= intent= intent=new intent(context,= simplewidgetprovider.class);= intent.setaction(appwidgetmanager.action_appwidget_update);= intent.putextra(appwidgetmanager.extra_appwidget_ids,= appwidgetids);= pendingintent= pendingintent=PendingIntent.getBroadcast(context, 0,= intent,= pendingintent.flag_update_current);= remoteviews.setonclickpendingintent(r.id.actionbutton,= pendingintent);=></ count;= i++) { int widgetId = appWidgetIds[i]; String number = String.format(%03d, (new Random().nextInt(900) + 100)); RemoteViews remoteViews = new RemoteViews(context.getPackageName(), R.layout.simple_widget); remoteViews.setTextViewText(R.id.textView, number); Intent intent = new Intent(context, SimpleWidgetProvider.class); intent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE); intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds); PendingIntent pendingIntent = PendingIntent.getBroadcast(context, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT); remoteViews.setOnClickPendingIntent(R.id.actionButton, pendingIntent); > > remoteViews); In the onUpdate method () above, we iterate through all of our widgets (in case the user has posted multiple widgets), get a RemoteViews object, update the RemoteView text view with a new random number between 100 and 999, and then specify the action that should occur when the button is tapped. We use PendingIntent to request a manual update at the touch of an update button. The action for intent is set to AppWidgetManager.ACTION_APPWIDGET_UPDATE. This is the same action sent by the system when the widget needs to be updated automatically. We also specify widgets that need to be updated (all app widgets) by callingintent.putExtra (AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds). To update only the current widget, you can callintent.putExtra (AppWidgetManager.EXTRA_APPWIDGET_ID, widgetId); Finally, we ask AppWidgetManager to update the app's widget, giving it the current widgetId and the current RemoteViews facility. Providing metadata AppWidgetProviderInfoIt is an xml file that identifies additional information, features and data related to the widget. Data such as the minimum layout sizes (width and height) if the widget should be available on the lock screen (Android 4.2 and above) as often the widget should be updated, among many others. We identify an xml file called simple_widget_info.xml and save it in a res/xml folder. zlt;appwidget-provider xmlns:android/ minwidth/120dp android:minheight/60dp android:updateperiodmillis'1800000 android:initiallay @layout/simple_widget android:resizemode'horizontal'vertical android:widgetcategory'home_screen'keyguard android:previewimage'@drawable/preview'gt;'lt;appwidget-provider'gt; Most attributes have fairly clear names. minWidth and minHeight determine the minimum width and height of the widget. ThePeriodMillis update determines the frequency of the update in milliseconds for the widget. Please note that frequent updates will have a significant impact on the user's battery. Take note of the widgetCategory attribute. This determines if your widget can be accessed on the lock screen as well as on the home screen. All widgets are available on the home screen by default, and if not specified. Android 4.2 included key options, indicating that the widget could be added to the lock screen. If the widget appears on the lock screen, you can show different data or a different layout. To determine if the widget is on the lock screen, you request widget options using AppWidgetManager getWidgetOptions (int widgetId). This method returns a package that can be requested for AppWidgetManager.OPTION_APPWIDGET_HOST_CATEGORY. It will be either WIDGET_CATEGORY_HOME_SCREEN or WIDGET_CATEGORY_KEYGUARD. The code sample below checks and displays another other For each type of host. AppWidgetManager appVidgtManager; int widgetId; Set of myOptions and appWidgetManager.getAppWidgetOptions (widgetId); Get the OPTION_APPWIDGET_HOST_CATEGORY int - myOptions.getInt (AppWidgetManager.OPTION_APPWIDGET_HOST_CATEGORY, -1); If the value WIDGET_CATEGORY_KEYGUARD; Is int baseLayout a keyguard? R.layout.keyguard_widget_layout : R.layout.widget_layout; Announce the widget in the Manifest app The smart move is to add the app widget to the app's manifesto. In the item's tags, add the following of the simpleWidgetProvider AppWidgetProviderInfo.WIDGET_CATEGORY_KEYGUARD; android.appwidget.action.APPWIDGET_UPDATE (@xml/simple_widget_info/meta-data/zgt; receiver/receiver)gt; Don't forget to change the android receiver into the implementation of AppWidgetProvider, meta-given android At this point you should be able to run the app, and place the widget on the home screen or lock screen. Clicking the update button should automatically update all widgets. Can you change your code to update only the widget that was used? How about updating just one random widget on tap? Have some fun with this, but remember that the widget that is frequently updated will leak on the battery. As usual, the full code is available to modify and reuse the contents of your hearts on github. Android Developer Newsletter vehicle tracking android app source code. google maps vehicle tracking source code android. vehicle tracking android source code