```
/*
 * Copyright (c) 2018. Property of Dennis Kwabena Bilson. No unauthorized duplication of this material should
 be made without prior permission from the developer
 */

package io.pergasus.ui

import `in`.uncod.android.bypass.Bypass
import android.annotation.SuppressLint
import android.app.Activity
import android.content.res.Resources
import android.net.Uri
import android.os.Bundle
import android.support.customtabs.CustomTabsIntent
import android.support.v4.content.ContextCompat
import android.support.v4.view.PagerAdapter
import android.support.v4.view.ViewPager
import android.support.v7.widget.RecyclerView
import android.text.Layout
import android.text.SpannableString
import android.text.Spanned
import android.text.TextUtils
import android.text.style.AlignmentSpan
import android.transition.TransitionInflater
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import com.bumptech.glide.load.engine.DiskCacheStrategy
import com.bumptech.glide.load.resource.drawable.DrawableTransitionOptions.withCrossFade
import com.bumptech.glide.request.RequestOptions
import com.bumptech.glide.request.target.Target
import io.pergasus.R
import io.pergasus.api.PhoenixClient
import io.pergasus.ui.widget.CircularImageView
import io.pergasus.ui.widget.ElasticDragDismissFrameLayout
import io.pergasus.ui.widget.FourThreeImageView
import io.pergasus.ui.widget.InkPageIndicator
import io.pergasus.util.HtmlUtils
import io.pergasus.util.bindView
import io.pergasus.util.customtabs.CustomTabActivityHelper
import io.pergasus.util.glide.GlideApp
import org.jetbrains.annotations.Nullable
import java.security.InvalidParameterException


/** About the application */
class AboutActivity : Activity() {

    private val draggableFrame: ElasticDragDismissFrameLayout by bindView(R.id.draggable_frame)
    private val pager: ViewPager by bindView(R.id.pager)
    private val pageIndicator: InkPageIndicator by bindView(R.id.indicator)

    private lateinit var client: PhoenixClient

-------------------------------------------------------------------------------

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_about)

        client = PhoenixClient(this@AboutActivity)

        pager.adapter = AboutPagerAdapter()
        pager.pageMargin = resources.getDimensionPixelSize(R.dimen.spacing_normal)
        pageIndicator.setViewPager(pager)
```

```
    draggableFrame.addListener(object : ElasticDragDismissFrameLayout.ElasticDragDismissCallback() {
        override fun onDragDismissed() {
            // if we drag dismiss downward then the default reversal of the enter
            // transition would slide content upward which looks weird. So reverse it.
            if (draggableFrame.translationY > 0) {
                window.returnTransition = TransitionInflater.from(this@AboutActivity)
                        .inflateTransition(R.transition.about_return_downward)
            }
            finishAfterTransition()
        }
    })

}

internal inner class AboutPagerAdapter : PagerAdapter() {
    private val PAGES: Int = 4
    private var layoutInflater: LayoutInflater = LayoutInflater.from(this@AboutActivity)
    private val markdown: Bypass = Bypass(this@AboutActivity, Bypass.Options())
    private val resources: Resources = this@AboutActivity.resources

    private var aboutPhoenix: View? = null
    @Nullable
    private var phoenixDescription: TextView? = null
    private var aboutIcon: View? = null
    @Nullable
    private var iconDescription: TextView? = null
    @Nullable
    var icon: FourThreeImageView? = null
    private var aboutLibs: View? = null
    private var libsList: RecyclerView? = null
    private var aboutDeveloper: View? = null
    @Nullable
    var profile: CircularImageView? = null
    @Nullable
    var follow: Button? = null


    override fun isViewFromObject(view: View, `object`: Any): Boolean {
        return view == `object`
    }

    override fun instantiateItem(container: ViewGroup, position: Int): Any {
        val layout = getPage(position, container)
        container.addView(layout)
        return layout
    }

    private fun getPage(position: Int, container: ViewGroup): View {
        return when (position) {
            0 -> {
                if (aboutPhoenix == null) {
                    aboutPhoenix = layoutInflater.inflate(R.layout.about_app, container, false)
                    phoenixDescription = aboutPhoenix?.findViewById(R.id.about_description)
                    // fun with spans & markdown
                    val about0: CharSequence = markdown.markdownToSpannable(
                            resources.getString(R.string.about_app_0), phoenixDescription, null)
                    val about1 = SpannableString(
                            resources.getString(R.string.about_app_1))
                    about1.setSpan(AlignmentSpan.Standard(Layout.Alignment.ALIGN_CENTER),
                            0, about1.length, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
                    val about2 = SpannableString(markdown.markdownToSpannable
                    (resources.getString(R.string.about_app_2),
                            phoenixDescription, null))
                    about2.setSpan(AlignmentSpan.Standard(Layout.Alignment.ALIGN_CENTER),
                            0, about2.length, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
                    val desc: CharSequence = TextUtils.concat(about0, "\n\n", about1, "\n", about2, "\n\n")
                    HtmlUtils.setTextWithNiceLinks(phoenixDescription, desc)
```

```
            }
            this.aboutPhoenix!!
        }
        1 -> {
            if (aboutIcon == null) {
                aboutIcon = layoutInflater.inflate(R.layout.about_icon, container, false)
                iconDescription = aboutIcon?.findViewById(R.id.icon_description)
                icon = aboutIcon?.findViewById(R.id.icon)
                val icon0 = resources.getString(R.string.about_icon_0)
                val icon2 = resources.getString(R.string.about_accra_mall)
                val icon1 = markdown.markdownToSpannable(resources
                        .getString(R.string.about_icon_1), iconDescription, null)
                val iconDesc = TextUtils.concat(icon2, "\n\n", icon0, "\n", icon1)
                HtmlUtils.setTextWithNiceLinks(iconDescription, iconDesc)
            }
            this.aboutIcon!!
        }
        2 -> {
            if (aboutLibs == null) {
                aboutLibs = layoutInflater.inflate(R.layout.about_libs, container, false)
                libsList = aboutLibs?.findViewById(R.id.libs_list)
                libsList?.adapter = LibraryAdapter(this@AboutActivity)
            }
            this.aboutLibs!!
        }
        3 -> {
            if (aboutDeveloper == null) {
                aboutDeveloper = layoutInflater.inflate(R.layout.about_developer, container, false)
                profile = aboutDeveloper?.findViewById(R.id.developer_profile)
                follow = aboutDeveloper?.findViewById(R.id.follow)

                //Follow developer on github
                follow?.setOnClickListener({
                    CustomTabActivityHelper.openCustomTab(
                            this@AboutActivity,
                            CustomTabsIntent.Builder()
                                    .setToolbarColor(ContextCompat.getColor(this@AboutActivity,
                                            R.color.background_super_dark))
                                    .addDefaultShareMenuItem()
                                    .build(), Uri.parse("https://github.com/Quabynah"))
                })

                //Load developer profile image from resource
                GlideApp.with(profile!!.context)
                        .load(getString(R.string.quabynah_url))
                        .circleCrop()
                        .override(Target.SIZE_ORIGINAL, Target.SIZE_ORIGINAL)
                        .diskCacheStrategy(DiskCacheStrategy.AUTOMATIC)
                        .placeholder(R.drawable.avatar_placeholder)
                        .error(R.drawable.avatar_placeholder)
                        .fallback(R.drawable.avatar_placeholder)
                        .into(profile!!)
            }
            this.aboutDeveloper!!
        }
        else -> throw InvalidParameterException("View not implemented")
    }

}
```

```
override fun destroyItem(container: ViewGroup, position: Int, `object`: Any) {
    container.removeView(`object` as View)
}
```

```
override fun getCount(): Int {
    return PAGES
}
```

```
    }

    private class LibraryAdapter internal constructor(internal val host: Activity) : RecyclerView.Adapter<
RecyclerView.ViewHolder>() {

        override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {
            when (viewType) {
                VIEW_TYPE_INTRO -> return LibraryIntroHolder(LayoutInflater.from(parent.context)
                        .inflate(R.layout.about_lib_intro, parent, false))
                VIEW_TYPE_LIBRARY -> return createLibraryHolder(parent)
            }
            throw InvalidParameterException()
        }

        private fun createLibraryHolder(parent: ViewGroup): LibraryHolder {
            val holder = LibraryHolder(LayoutInflater.from(parent.context)
                    .inflate(R.layout.library, parent, false))
            val clickListener = View.OnClickListener {
                val position = holder.adapterPosition
                if (position == RecyclerView.NO_POSITION) return@OnClickListener
                CustomTabActivityHelper.openCustomTab(
                        host,
                        CustomTabsIntent.Builder()
                                .setToolbarColor(ContextCompat.getColor(host, R.color.background_super_dark))
                                .addDefaultShareMenuItem()
                                .build(), Uri.parse(libs[position - 1].link))
            }
            holder.itemView.setOnClickListener(clickListener)
            holder.link.setOnClickListener(clickListener)
            return holder
        }

        override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
            if (getItemViewType(position) == VIEW_TYPE_LIBRARY) {
                bindLibrary(holder as LibraryHolder, libs[position - 1]) // adjust for intro
            }
        }

        override fun getItemViewType(position: Int): Int {
            return if (position == 0) VIEW_TYPE_INTRO else VIEW_TYPE_LIBRARY
        }

        override fun getItemCount(): Int {
            return libs.size + 1 // + 1 for the static intro view
        }

        @SuppressLint("CheckResult")
        private fun bindLibrary(holder: LibraryHolder, lib: Library) {
            holder.name.text = lib.name
            holder.description.text = lib.description
            val request = GlideApp.with(holder.image.context)
                    .load(lib.imageUrl)
                    .apply(RequestOptions().placeholder(R.drawable.avatar_placeholder))
                    .apply(RequestOptions().diskCacheStrategy(DiskCacheStrategy.ALL))
                    .transition(withCrossFade())
            if (lib.circleCrop) {
                request.apply(RequestOptions().optionalCircleCrop())
            }
            request.into(holder.image)
        }

        companion object {

            private const val VIEW_TYPE_INTRO = 0
            private const val VIEW_TYPE_LIBRARY = 1
            internal val libs = arrayOf(
                    Library("Android support libraries",
```

```
                "The Android support libraries offer a number of features that are not built into the framework
.",
                "https://developer.android.com/topic/libraries/support-library",
                "https://developer.android.com/images/android_icon_125.png",
                false),
            Library("ButterKnife",
                "Bind Android views and callbacks to fields and methods.",
                "http://jakewharton.github.io/butterknife/",
                "https://avatars.githubusercontent.com/u/66577",
                true),
            Library("Bypass",
                "Skip the HTML, Bypass takes markdown and renders it directly.",
                "https://github.com/Uncodin/bypass",
                "https://avatars.githubusercontent.com/u/1072254",
                true),
            Library("Glide",
                "An image loading and caching library for Android focused on smooth scrolling.",
                "https://github.com/bumptech/glide",
                "https://avatars.githubusercontent.com/u/423539",
                false),
            Library("JSoup",
                "Java HTML Parser, with best of DOM, CSS, and jquery.",
                "https://github.com/jhy/jsoup/",
                "https://avatars.githubusercontent.com/u/76934",
                true),
            Library("Firebase",
                "Backend for the application",
                "https://github.com/jhy/jsoup/",
                "https://avatars.githubusercontent.com/u/76934",
                true),
            Library("Hubtel",
                "Mobile Money Provider for \'The Phoenix\'",
                "https://github.com/jhy/jsoup/",
                "https://avatars.githubusercontent.com/u/76934",
                true),
            Library("OkHttp",
                "An HTTP & HTTP/2 client for Android and Java applications.",
                "http://square.github.io/okhttp/",
                "https://avatars.githubusercontent.com/u/82592",
                false),
            Library("Retrofit",
                "A type-safe HTTP client for Android and Java.",
                "http://square.github.io/retrofit/",
                "https://avatars.githubusercontent.com/u/82592",
                false))
    }
}

internal class LibraryHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

    var image: ImageView = itemView.findViewById(R.id.library_image)
    var name: TextView = itemView.findViewById(R.id.library_name)
    var description: TextView = itemView.findViewById(R.id.library_description)
    var link: Button = itemView.findViewById(R.id.library_link)
}

internal class LibraryIntroHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

    var intro: TextView = itemView as TextView

}

/**
 * Models an open source library we want to credit
 */
private class Library internal constructor(internal val name: String, internal val description: String,
                                           internal val link: String, internal val imageUrl: String,
```

```
                internal val circleCrop: Boolean)

}
```