Content 2

05 February 2024 06:06

Add a scrollable list (android.com)

2. Create a list item data class

NEW PACKAGE (MODEL)-.> DATA CLASS -> NEWS

```
packagecom.example.demo2.model

importandroidx.annotation.DrawableRes
importandroidx.annotation.StringRes

dataclassnews(
@StringResvalstringResourceId:Int,
@DrawableResvalimageResourceId:Int
)
```

NEW PACKAGE (DATA)-.> CLASS -> NEWSOURCE

```
packagecom.example.demo2.data

importcom.example.demo2.R
importcom.example.demo2.model.news

classnewsource(){
funloadnews():List<news>{
return listOf<news>(
news(R.string.news1,R.drawable.image1),
news(R.string.news2,R.drawable.image2),
news(R.string.news3,R.drawable.image3),
news(R.string.news4,R.drawable.image4),
news(R.string.news5,R.drawable.image5),
news(R.string.news6,R.drawable.image6))

}
}
```

MAINACTIVITY

```
packagecom.example.demo2

importandroidx.compose.foundation.layout.Column
importandroidx.compose.material3.Card
importcom.example.demo2.model.news


importandroid.os.Bundle
importandroidx.activity.ComponentActivity
importandroidx.activity.compose.setContent
importandroidx.compose.foundation.layout.fillMaxSize
importandroidx.compose.material3.MaterialTheme
importandroidx.compose.material3.Surface
importandroidx.compose.material3.Text
importandroidx.compose.runtime.Composable
importandroidx.compose.ui.Modifier
importandroidx.compose.ui.tooling.preview.Preview
importcom.example.demo2.ui.theme.Demo2Theme

importandroidx.compose.foundation.Image
importandroidx.compose.foundation.layout.fillMaxWidth
importandroidx.compose.foundation.layout.height
importandroidx.compose.foundation.layout.padding
importandroidx.compose.foundation.lazy.LazyColumn
importandroidx.compose.foundation.lazy.items
importandroidx.compose.ui.layout.ContentScale
importandroidx.compose.ui.platform.LocalContext
importandroidx.compose.ui.res.painterResource
```

1. `@StringRes val stringResourceId: Int`: This annotation (`@StringRes`) indicates that the integer value represents a string resource ID. This is typically used to refer to string resources defined in XML files within the `res/values/` directory of an Android project.

2. `@DrawableRes val imageResourceId: Int`: Similarly, this annotation (`@DrawableRes`) indicates that the integer value represents a drawable resource ID. Drawable resources typically include images or other visual assets that are also defined within the `res/drawable/` directory of an Android project.

By using these annotations, you're ensuring that the `stringResourceId` and `imageResourceId` properties of the `News` class are only assigned values that correspond to valid string and drawable resources in your Android project. This helps prevent runtime errors related to resource type mismatches.

1. `import com.example.demo2.R`: This imports the R class generated by the Android build system, which contains references to all resources in your project, including strings and drawables.

2. `import com.example.demo2.model.News`: This imports the `News` data class you defined earlier.

3. `class NewsSource()`: This is the definition of the `NewsSource` class.

4. `fun loadNews(): List<News>`: This function is responsible for loading news items and returns a list of `News` objects.

5. `return listOf<News>(...)`: This line returns a list of `News` objects. Each news item is created using the `news` constructor defined in the `News` data class. The constructor takes a string resource ID (`R.string.newsX`) and a drawable resource ID (`R.drawable.imageX`). These IDs are provided for each news item, presumably from your resources directory.

6. The list contains news items with titles and images, each corresponding to a different resource ID.

Your code appears to be an Android application written in Kotlin using Jetpack Compose for building the UI. Here's a breakdown of the key components:

1. **`MainActivity`**: This is the entry point of your application. It sets the content view to `Demo2Theme` and inside it, it surfaces the `NewsApp`.

2. **`NewsApp` Composable**: This is the main composable function that represents your news application. It displays a surface with a background color from the theme and contains the `NewsList`.

3. **`NewsList` Composable**: This function takes a list of `news` items and displays them using a `LazyColumn`. Each item in the list is represented by a `NewsCard`.

```
importandroidx.compose.foundation.lazy.items
importandroidx.compose.ui.layout.ContentScale
importandroidx.compose.ui.platform.LocalContext
importandroidx.compose.ui.res.painterResource
importandroidx.compose.ui.res.stringResource
importandroidx.compose.ui.unit.dp
importcom.example.demo2.data.newsource


classMainActivity:ComponentActivity(){
overridefunonCreate(savedInstanceState:Bundle?){
super.onCreate(savedInstanceState)
setContent{
Demo2Theme{
//Asurfacecontainerusingthe'background'colorfromthetheme
Surface(
modifier=Modifier.fillMaxSize(),
color=MaterialTheme.colorScheme.background
){
NewsApp()
}
}
}
}
}

@Composable
funNewsApp(){
Newslist(
newslist=newsource().loadnews(),
)
}

@Composable
funNewslist(newslist:List<news>,modifier:Modifier=Modifier){
LazyColumn(modifier=modifier){
items(newslist)
{
demo2->
NewsCard(demo2=demo2,
modifier=Modifier.padding(8.dp))
}
}
}

@Composable
funNewsCard(demo2:news,modifier:Modifier=Modifier){
Card(modifier=modifier){
Column{
Image(
painter=painterResource(demo2.imageResourceId),
contentDescription=stringResource(demo2.stringResourceId),
modifier=Modifier
.fillMaxWidth()
.height(194.dp),
contentScale=ContentScale.Crop
)
Text(
text=LocalContext.current.getString(demo2.stringResourceId),
modifier=Modifier.padding(16.dp),
style=MaterialTheme.typography.headlineSmall
)
}
}
}


@Preview(showBackground=true)
@Composable
privatefunNewsCardPreview(){
NewsCard(news(R.string.news1,R.drawable.image1))
}
```

RUN CODE

3. **`NewsList` Composable**: This function takes a list of `news` items and displays them using a `LazyColumn`. Each item in the list is represented by a `NewsCard`.

4. **`NewsCard` Composable**: This composable represents a single news item displayed within a `Card`. It contains an `Image` representing the news image and a `Text` displaying the news title.

5. **`newsource` Class**: This class seems to be responsible for providing news data. It loads news items and returns them as a list.

6. **`@Preview` Composables**: These are used for previewing the UI layout in Android Studio's preview pane.

Overall, your code sets up a simple news application UI using Jetpack Compose, where news items are displayed in a list with each item represented by a card containing an image and a title. The data for the news items is loaded from a `newsource`.