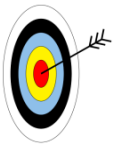
(2) Der Startbildschirm



Ihr werdet auf dem zweiten Arbeitsblatt den **Startbildschirm** für euer *„Schiffe versenken“*-Spiel erstellen. Mit dem integrierten **Drag-‘n‘-Drop Editor** könnt ihr dem Starbildschirm eurer App sein Aussehen verpassen, indem ihr: Einen Willkommens-Text, **Buttons** und ein **Feld für die Namenseingabe** hinzufügt und bearbeitet, die Elemente sinnvoll **anordnet** und ihre Eigenschaften ändert. Dabei…

- …erhaltet ihr einen **Überblick** über die Herangehensweise bei der Erstellung von Android Projekten (ERST Design, DANN Funktionalitäten),
- …lernt ihr die spezielle **Android-Ordnerstruktur** kennen und
- …manipuliert ihr Eigenschaften und Funktionen der Elemente mithilfe von **XML-Code**.

Layout vs. Funktionen

In Android-Programmierprojekten gibt es eine **typische Zweiteilung** zwischen Design bzw. Layout der App und den Funktionen der App. Anders als in „normalen“ Programmieraufgaben kommt es bei einer App natürlich besonders auf das Aussehen an: das Thema, die Farben, Anordnungen von Buttons und Text, dynamische Elemente, … spielen eine zentrale Rolle. Daher wird der reine Layout-Entwurf immer zuerst bearbeitet, während in einem zweiten Schritt die einzelnen Teile des Layouts mit ihren Funktionen („Was passiert, wenn ein Button geklickt wird?!“) implementiert werden.

Da unser Fokus in diesem Projekt auf der **GUI-Programmierung** liegt, startet auch ihr mit dem **Design des Startbildschirms** – zuerst auf rein graphischer Ebene mit dem Drag-‘n‘-Drop Editor und dann immer losgelöster davon auf textueller Ebene mit Hilfe von XML-Code.

Ändern des angezeigten Textes

Als ihr euer Projekt kompiliert habt, konntet ihr sehen, dass es zwar schon ausgeführt wird, aber nur der Standardtext *„Hallo Infosphere!“* angezeigt wird – ziemlich langweilig also. Zuerst werdet ihr also den Text des Startbildschirms ändern und anpassen.

Schaut euch einmal die Ordnerstruktur in eurem Projekt an!

Ihr wisst sicher: Texte werden in Java als **Strings** gespeichert. Was wahrscheinlich neu für euch ist: Strings zählen in Android-Projekten zu den sogenannten **Ressources** und werden unabhängig vom restlichen Quellcode im **values-Ordner** gespeichert (**s. Abbildung rechts**).

Um dabei nicht den Überblick zu verlieren, findet ihr auf der folgenden Seite erst einmal ein paar Infos zur **Ordnerstruktur** in Android-Projekten und zu den Teilen, die für euch dabei wichtig sind!

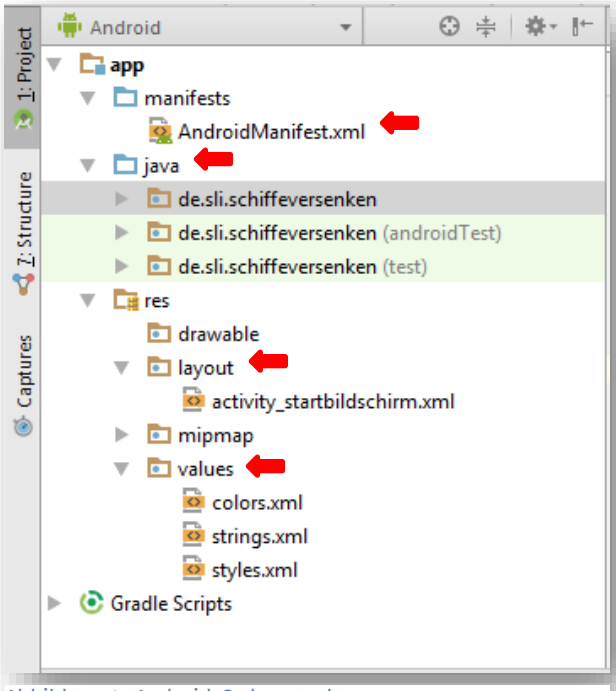


Abbildung 1: Android-Ordnerstruktur

CC BY SA

(2) Der Startbildschirm

Die Android-Ordnerstruktur

Im Gegensatz zu euch bekannten Java-Projekten generiert Android Studio bei der Erstellung eines Android-Projekts eine ganze **Hierarchie von Ordnern und Unterordnern**, die vor allem Android-spezifische Informationen über die Version, einzubindende Bibliotheken usw. enthalten - für euch sind diese Ordner aber erst einmal nicht wichtig. Wir konzentrieren uns jetzt auf drei Ordner, die ihr auch während des *„Schiffe versenken"*-Projekts nutzen werdet (mit roten Pfeilen in der **Abbildung 1 auf S. 1** markiert):

- **java/**
  Enthält die **Java-Quellcodedateien** des Projekts. Hier findet ihr auch eine vordefinierte Java-Datei mit dem Namen der **Activity**, die ihr beim Anlegen des Projekts erstellt habt. Sie wird ausgeführt, wenn ihr auf das App-Icon auf eurem Smartphone klickt.
- **res/**
  Dies ist der wichtige **Ressources-Ordner**. Er enthält viele Unterordner für die Dinge, die die App benötigt, aber aus dem „normalen" Quellcode ausgelagert werden: **Bilder, Strings und Layouts**. Dadurch kann der Compiler den Code besser für die Verwendung auf Smartphones optimieren. Außerdem können so z.B. verschiedene Layouts – eines für Smartphones und eines für Tablets – in ein und demselben Projekt realisiert werden. Schauen wir uns diese Unterordner noch genauer an:
  - **drawable/**
    Diese Ordner beinhalten **graphische Elemente**, wie z.B. Bilder, in verschiedenen Auflösungen und für verschiedene Bildschirme.
  - **layout/**
    Hier findet ihr **xml-Files**, die das Aussehen der einzelnen Benutzerbildschirme festlegen.
  - **values/**
    Der Ordner values bündelt verschiedene xml-Dateien, in denen **grundlegende Ressources** wie Strings oder Farbdefinitionen gespeichert werden.
- **Gradle Scripts/**
  In diesem Ordner findet sich alles, was zum Kompilieren der App benötigt wird. Hier solltet ihr tunlichst nichts ändern.

Neben diesen Ordnern ist das sogenannte **Android-Manifest** (*AndroidManifest.xml*) noch von Bedeutung. Das Manifest wird noch einmal wichtig werden, wenn ihr versuchen wollt, z.B. durch einen Button-Klick eine andere Bildschirmansicht aufzurufen. Dazu gibt es also später wieder mehr Infos...
Zuerst wollen wir uns aber wieder der ursprünglichen Aufgabe widmen und den Text auf der Startseite ändern!

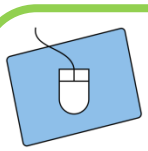
Welches ist also jetzt wohl die Datei, die euch die Bildschirmanordnung ermöglicht?!

CC BY SA

(2) Der Startbildschirm

Funktionalitäten eines Startbildschirms

Jetzt werden erste Änderung am *„Schiffe versenken“*-Projekt umgesetzt und ein kleines, lauffähiges Programm entworfen! Nun sollen natürlich mehrere Elemente den Bildschirm bevölkern und die App soll wachsen.



Wie soll euer fertiger Startbildschirm einmal aussehen?

Überlegt euch folgendes:

1. Wie wollt ihr die Elemente auf dem Bildschirm **anordnen**?
    - Ein Label für einen Begrüßungstext
    - Zwei Knöpfe zum Starten und Beenden des Spiels
    - Ein Texteingabefeld zur Namenseingabe
2. Welche **Farben** soll der Bildschirmhintergrund haben, welche die Buttons und Texte?

Nutzt die **Vorlage**, um eurer Fantasie freien Lauf zu lassen! Zeichnet einen Entwurf und ordnet die Elemente so an, wie ihr es möchtet. Überlegt euch dabei, was sinnvoll ist und wie das Farbschema aussehen soll. Macht euch ggf. Notizen, auf die ihr während der Entwicklung zurückgreifen könnt.

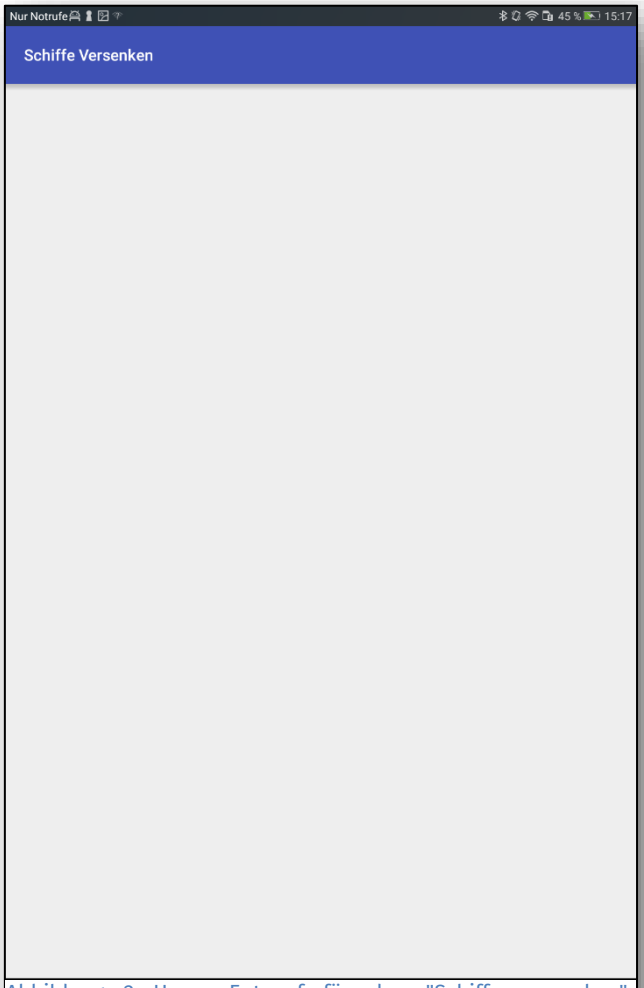


Abbildung 2 Unser Entwurf für den "Schiffe versenken"-Startbildschirm

CC BY SA

(2) Der Startbildschirm

Einen Button hinzufügen und gestalten

Jetzt werdet ihr die **Ansicht** der Seite des Startbildschirms ändern und einen neuen Button hinzufügen, der später das *„Schiffe versenken“*-Spiel starten soll.

Öffnet nun aus dem **layout-Ordner** (in /res) die Datei *activity_startbildschirm.xml*; das ist die **Haupt-Activity**, die ihr beim Erstellen des Projektes habt erzeugen lassen (s. Abb. 3)!
[**Hinweis:** *Auch hier sollte sich die graphische Ansicht mit dem Drag-‘n‘-Drop Editor öffnen. Falls nicht, erledigt das ein Klick auf den Reiter Design.*]

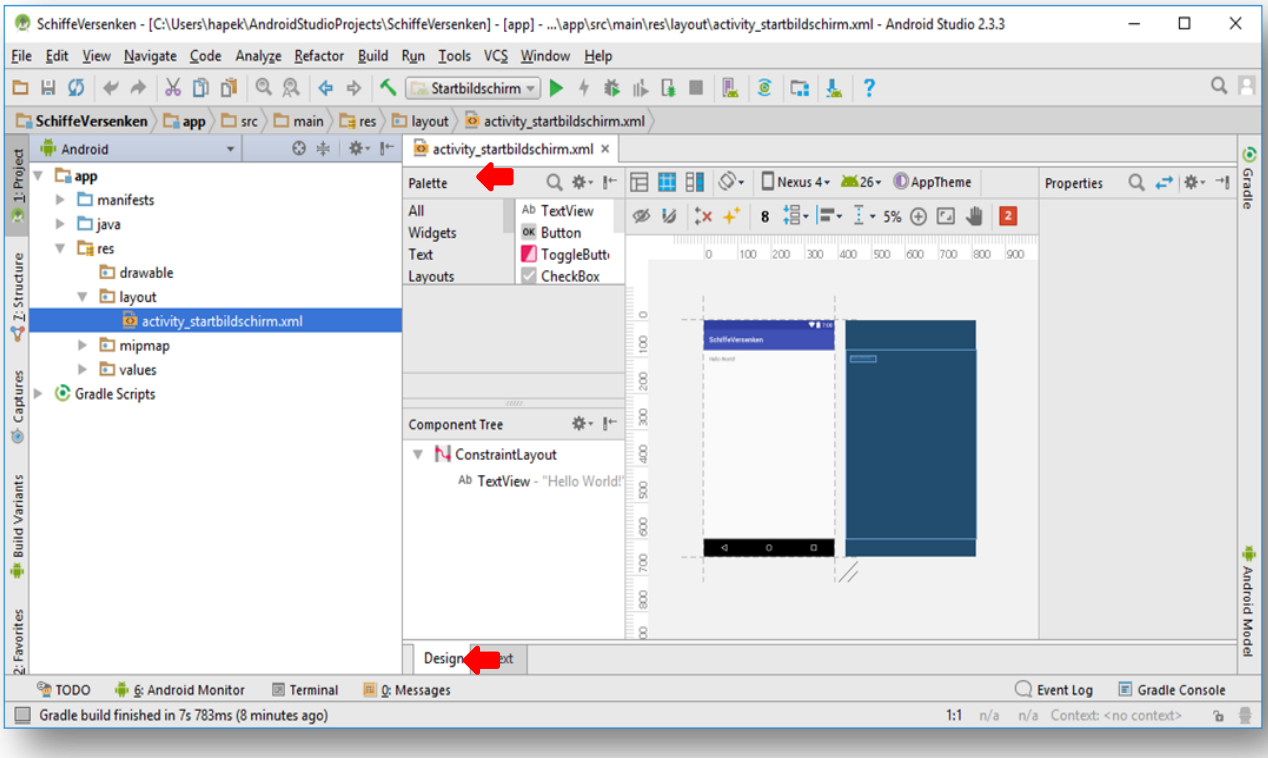


Abbildung 3: Drag and Drop Editor

Der Drag-‘n‘-Drop Editor beinhaltet erstmal zwei wichtige Fenster für eure nächsten Schritte:

- ***Vorschaufenster***: der große **Arbeitsbereich** in der Mitte, wo ihr das Design ändert und es immer direkt überprüfen könnt.
- ***Palette***: bietet einzelne **Pop-Up Menus**, wobei hier **Form Widgets** und **Text Fields** (sogenannte View-Objekte wie z.B. Buttons oder Textfelder) und **Layouts** (Hilfen/Raster zum Anordnen von Elementen) die für euch Interessanten sind.
- ***Component Tree:*** gibt euch eine Übersicht über bereits eingefügte **Views** und deren hierarchische Anordnung.

Sucht einen **Button** in der Palette und zieht ihn auf den Startbildschirm

CC BY SA

(2) Der Startbildschirm

So weit, so gut… Der Button steht, sieht aber noch ziemlich langweilig aus und sagt dem Benutzer ja auch nicht wirklich, dass er dazu gedacht ist, das Spiel zu starten. Ein entsprechender Text fehlt!

1. Öffnet nun als erstes die XML-Datei, in der die String-Resources gespeichert werden: es öffnet sich eine Liste, die alle **Strings** eurer App enthält. Wie ihr seht, ist bereits ein **String** vorhanden: app_name. Über den Link **„Open editor“** könnt ihr eine **graphische Ansicht** öffnen.
   [**Hinweis:** *Falls ihr euch schon wieder fragt, was denn überhaupt dieses XML ist, geduldet euch noch ein wenig – ihr werdet die sogenannte Auszeichnungssprache XML kennen lernen, wenn wir im nächsten Schritt Resources manipulieren wollen!*]

2. Klickt auf das **„+“** im XML-Editor und Erstellt einen **String für euren Button**. Gebt ihm einen sinnvollen Namen und Inhalt.

[**Hinweis: Default Value** *enthält den Text, der nachher auch angezeigt werden soll.* ***Name*** *ist ein einzigartiger Bezeichner für den String, damit er im Java-Code richtig referenziert werden kann. Überlegt euch daher eine konsistente Benennung der Strings eines Typs, z.B. button_ als Beginn von Strings für Buttons!*]

3. Speichert die Datei!

In der XML-Datei sollte jetzt euer String stehen. Noch hat sich nichts am Aussehen eurer App geändert. Wir müssen dem Knopf noch den Text zuweisen.

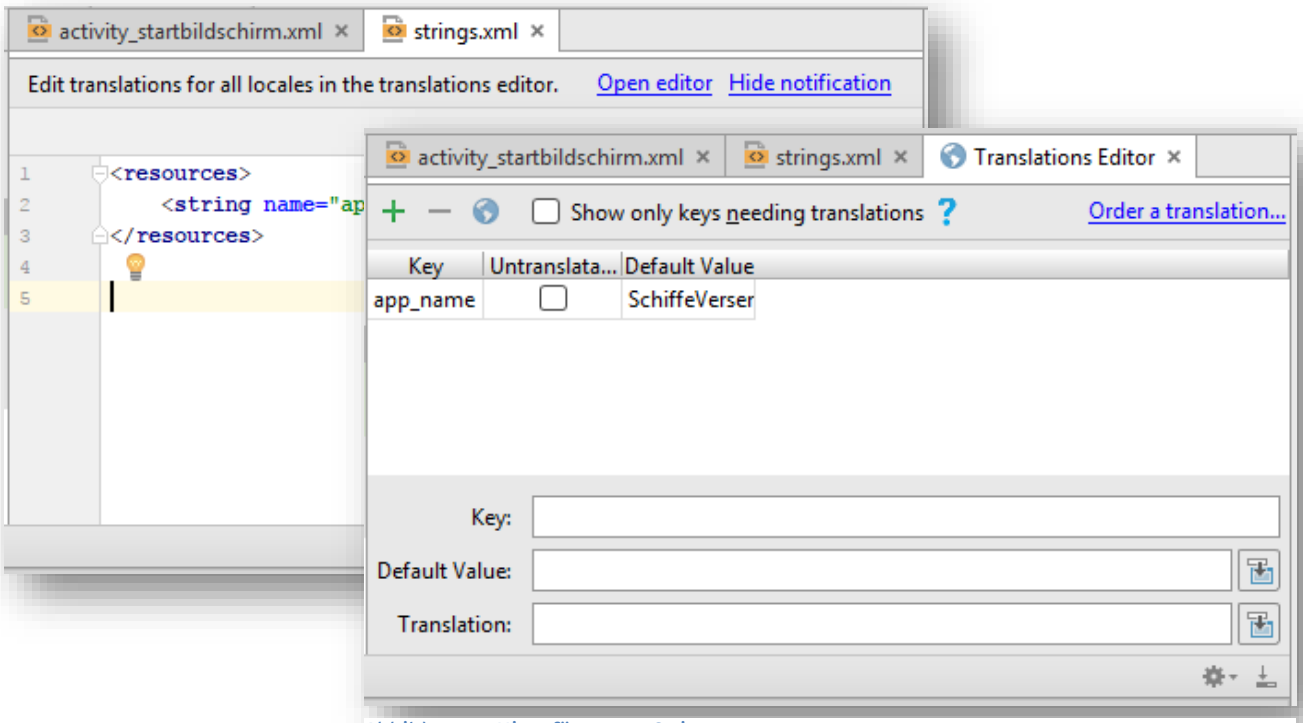


Abbildung 4: Hinzufügen von Strings

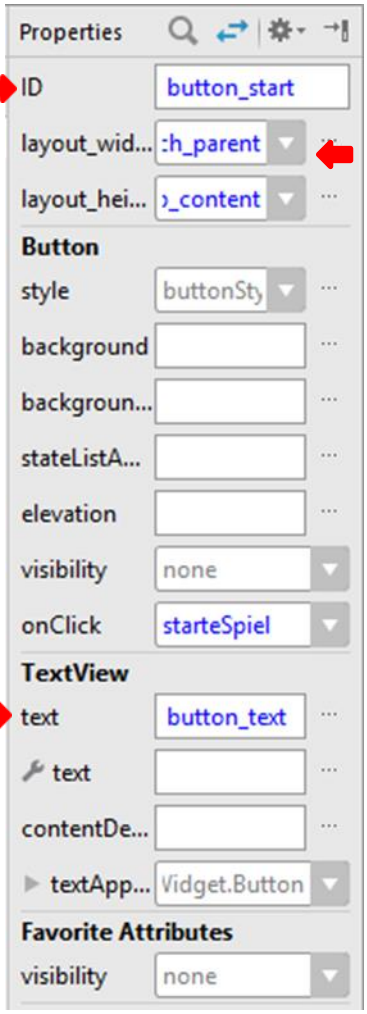
CC BY SA

(2) Der Startbildschirm

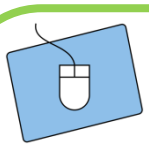
Jetzt wird eurem Button die String-Ressource zugewiesen.

1. Wechselt zurück in den graphischen Editor der XML-Datei eures **Startbildschirms** im Ordner **layout**.
2. Klickt auf den Button und seht euch das Fenster **Properties** auf der rechten Seite an. Dort sind alle **Eigenschaften des Buttons** aufgelistet, die ihr ändern könnt.
3. Überlegt euch jetzt eine **aussagekräftige ID** für den Button!
4. Um die String-Ressource jetzt auf dem Button sichtbar werden zu lassen, müsst ihr zum Schluss noch eine Eintragung in den Properties vornehmen:

Sucht die Zeile mit der Eigenschaft „Text“ und **löscht den bisherigen Eintrag**. Der String aus den Resources wird jetzt durch den Befehl @string/<name> aufgerufen. **@string** sagt dem Programm, dass ein Wert aus der Datei *strings.xml* verwendet werden soll; **<name>** benennt den String und muss von euch durch den Bezeichner eures Strings ersetzt werden!



Sehr gut. Jetzt wird direkt ersichtlich, wofür der Button gebraucht wird! Als nächstes wollen wir uns mit Aussehen und Position des Buttons befassen.



1. Seht euch jetzt unter den **Properties** die beiden **Attribute** *layout_width* (Breite) und *layout_height* (Höhe), die zentral für die meisten Views im Android-Projekt sind. *Könnt ihr euch denken, was die Voreinstellung* **wrap_content** *bewirkt?*
2. Ändert die Einstellungen für Breite und Höhe und experimentiert mit den Möglichkeiten – **wrap_content** und **match_parent** – herum!
   [**Hinweis:** *Es werden keine konkreten Pixelanzahlen für die Werte eingegeben, damit das Layout auf unterschiedliche Bildschirmgrößen übertragbar bleibt.*]
3. Legt jetzt einen **weiteren Button** an – für das Spielende.
4. Außerdem benötigen wir jetzt aus der Rubrik **Text** ein Namens-Eingabefeld! Zieht auch dieses Element herüber und ordnet es entsprechend eures Entwurfs an.

[**Wichtig:** *Texte werden immer in der Datei* **strings.xml** *gespeichert.]*

Standardmäßig wird ein vertikales **Linear-Layout** verwendet. Dieses ordnet Elemente untereinander an. Android bietet aber noch viele andere Layout-Möglichkeiten, z.B. ein **RelativeLayout**.

Es ist wieder an der Zeit, zu testen!

Führt euer Programm aus und beobachtet, welche Änderungen sich beim Button ergeben. Kontrolliert, ob die Buttons und der Text richtig angezeigt werden und in der App stimmig sind – der Drag-‘n‘-Drop Editor gibt nicht immer richtig wieder, was auch später auf dem Bildschirm zu sehen ist! Ändert ggf. noch eure Einstellungen entsprechend ab!

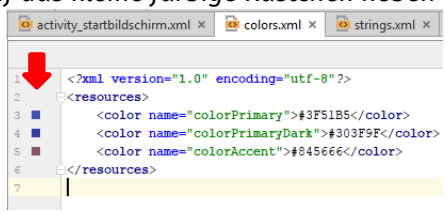
(2) Der Startbildschirm

Farben über Farben…

Da ihr sicher mit **verschiedenen Farben** arbeiten möchtet, werdet ihr nun am Beispiel der **Hintergrundfarbe** lernen, wie man diese ändern kann. Außerdem könnt ihr euch hier noch einmal an einem anderen Beispiel mit Resources vertraut machen!

Auch Farbwerte werden als Resources gespeichert. Sie werden im **ARGB-Format** angegeben. Dabei wird je für **A**lpha (die Deckkraft der Farbe) und die Farben **R**ot, **G**rün und **B**lau ein zweistelliger Wert im **Hexadezimal-System** angegeben. Dadurch wird die Farbe eindeutig definiert.

Hinweis: *Android Studio hat eine Auswahlfunktion für Farben. Ihr müsst keine Zahlen von Hand umrechnen. Ihr könnt Farben ändern und auswählen, indem ihr auf das kleine farbige Kästchen neben einer Farbe klickt.*]

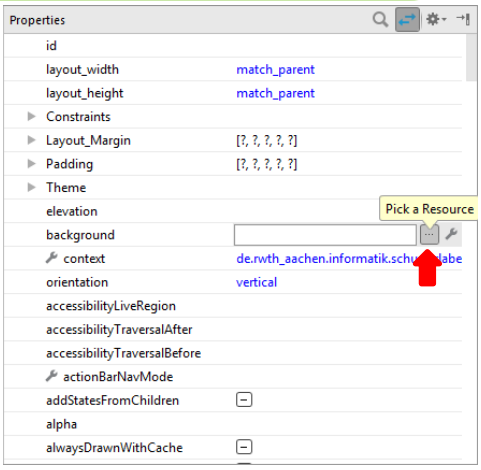


Den Alpha-Wert könnt ihr selber festlegen. Der Wert 00 bedeutet komplette Durchlässigkeit, was gleichbedeutend damit ist, dass die Farbe nicht sichtbar ist. Der höchste hexadezimale Wert FF (entspricht 255 im Dezimalsystem) stellt die Farbe deckend dar.

1. Öffnet Im **values-Ordner** die Datei colors.xml
2. Speichert in dieser Datei eine neue beliebige Farbe. Dafür gibt es diesmal keinen Editor. Ihr müsst den Eintrag für eine neue Farbe selbst eingeben. Danach erst könnt ihr die Farbe wählen.
3. Klickt nun auf das kleine Kästchen neben dem Eintrag, um eine Farbe zu wählen.
4. Wählt nun diese Farbe als **Hintergrundfarbe** für euren Startbildschirm.
   a. Wechselt zu der Datei **activity_startbildschirm.xml**.
   b. Klickt dazu auf den (weißen) Hintergrund im Vorschaufenster und seht euch die **Properties** an.
   c. Klickt ganz unten auf **„View all properties“.**
   d. In der Eigenschaft **background** müsst ihr jetzt nur noch die Farbdefinition referenzieren. Das funktioniert hier genauso wie bei String-Ressourcen.
5. Testet euer Programm und überprüft, ob sich die Farbe geändert hat!

Ihr seid gefragt!

Klasse! An dieser Stelle im Projekt seid ihr schon ein ganzes Stück vorwärtsgekommen. Jetzt ist es an der Zeit, noch einmal euer Projekt zu **testen** und zu schauen, ob ihr mit der Anordnung eurer Elemente und den Farben zufrieden seid; danach geht's auf dem nächsten AB weiter!



Quelle: , , , sowie die Screenshots aus Android Studio angefertigt vom InfoSphere-Team

CC BY SA