DOWNLOAD PDF ANDROID DESIGN PATTERNS AND BEST PRACTICE 2016

Chapter 1 : Android-er: Android Design Patterns and Best Practice

Android-er For Android development, from beginner to beginner.

Design Patterns; How this book works; What we will build; The scenario; The problem; The solution; Targeting platform versions; The support library; The factory pattern; UML diagrams; Running and testing an app; Connecting to a real device; Connecting to a virtual device; Monitoring devices; The abstract factory pattern; Working with more than one factory; Summary; Chapter 2: Creational Patterns; Applying themes; Customizing color and text. Using palettesCustomizing text; Adding image resources; Managing screen densities; Using designated resources; Creating a card view; Understanding card view properties; Applying CardView metrics; Updating the factory pattern; Applying a builder pattern; Summary; Chapter 3: Layout Patterns; Linear layouts; Weight and gravity. Relative layoutsThe percent support library; Screen rotation; Large screen layouts; Width qualifiers; Layout aliases; The strategy pattern; Summary; Chapter 5: Structural Patterns; Generating lists; List item layouts; Material font sizes; Connecting data; Translating string resources; Adapters and layout managers; The adapter pattern; The bridge pattern; The facade pattern; The criteria pattern; Summary; Chapter 6: Activating Patterns; Collapsing toolbars; Applying a data factory pattern; Positioning item layouts; Using the factory with the RecyclerView; Adding dividers. Configuring the floating action buttonThe dialog builder; Custom dialogs; Adding swipe and dismiss actions; Constructing layout builders; Summary; Chapter 7: Combining Patterns; Outlining specifications; The prototype pattern; Setting up a prototype; Applying the prototype; The decorator design pattern; Setting up a decorator; Applying the decorator; Extending the decorator; A sandwich builder pattern; Applying the pattern; Connecting to a UI; Selecting patterns; Adding a decorator; Attaching the pattern; Connecting the pattern to the UI; Summary; Chapter 8: The composite patternAdding a builder; A Layout composer; Adding components; Creating composites; Create composite layouts; Formatting layouts at runtime; Storage options; Creating static files; Creating and editing application files; Storing user preferences; The activity life cycle; Applying preferences; Adding a unique identifier; Summary; Chapter 9: Observing Patterns; The Observer pattern; Creating the pattern; Adding a notification; Utility observers and observables; Notifications; Setting an intent; Customizing and configuring notifications; Visibility and priority; Services; Summary. Annotation Create reliable, robust, and efficient Android apps with industry-standard design patternsAbout This Book Create efficient object interaction patterns for faster and more efficient Android development Get into efficient and fast app development and start making money from your android apps Implement industry-standard design patterns and best practices to reduce your app development time drasticallyWho This Book Is ForThis book is intended for Android developers who have some basic android development experience. Basic Java programming knowledge is a must to get the most out of this book. What You Will Learn Build a simple app and run it on real and emulated devices Explore the WYSIWYG and XML approaches to material design provided within Android Studio Detect user activities by using touch screen listeners, gesture detection, and reading sensors Apply transitions and shared elements to employ elegant animations and efficiently use the minimal screen space of mobile devices Develop apps that automatically apply the best layouts for different devices by using designated directories Socialize in the digital word by connecting your app to social media Make your apps available to the largest possible audience with the AppCompat support libraryIn DetailAre you an Android developer with some experience under your belt? Are you wondering how the experts create efficient and good-looking apps? Then your wait will end with this book! We will teach you about different Android development patterns that will enable you to write clean code and make your app stand out from the crowd. The book starts by introducing the Android development environment and exploring the support libraries. You will gradually explore the different design and layout patterns and get to know the best practices of how to use them together. Moving on, you will add user-detecting classes and APIs such as gesture detection, touch screen listeners, and sensors to your app. You will also learn to adapt your app to run on tablets and other

devices and platforms, including Android Wear, auto, and TV. Finally, you will see how to connect your app to social media and explore deployment patterns as well as the best publishing and monetizing practices. The book will start by introducing the Android development environment and exploring the support libraries. You will gradually explore the different Design and layout patterns and learn the best practices on how to use them together. You will then develop an application that will help you grasp Activities, Services and Broadcasts and their roles in Android development. Moving on, you will add user detecting classes and APIs such as at gesture detection, touch screen listeners and sensors to our app. You will also learn to adapt your app to run on tablets and other devices and platforms, including Android Wear, Auto, and TV. Finally, you will learn to connect your app to social media and explore deployment patterns and best publishing and monetizing practices. Style and approachThis book takes a step-by-step approach. The steps are explained using real-world practical examples. Each chapter uses case studies where we show you how using design patterns will help in your development process. Reviews Add a review and share your thoughts with other readers. Add a review and share your thoughts with other readers.

Chapter 2 : Architecting AndroidThe clean way? | Fernando Cejas

Android Design Patterns and Best Practice and millions of other books are available for Amazon Kindle. Learn more Enter your mobile number or email address below and we'll send you a link to download the free Kindle App.

I have questions though: Would you implement the presenter as its own public class, or as an inner class of the Activity? Or a fragment also inner class? Do you mean that transfer classes shall be used as instead of the actual model classes in the Activity view? Yes, I use them as views within the MVP pattern. I explained this quite poorly, the sentence "forward the classes needed" is misleading. What I mean, is the presenter sits between the view and the model, it reads the model and then updates the view. I will update my answer to be a bit more clear: How I use MVC: Use Activities purely for user IO, and use a local service for all of your processing. When the service wants to show something - broadcast it to your activities! So, I released a full blown video course about Android applications architecture. This answer was updated in order to remain relevant as of November It looks like you are seeking for architectural patterns rather than design patterns. Design patterns aim at describing a general "trick" that programmer might implement for handling a particular set of recurring software tasks. In OOP, when there is a need for an object to notify a set of other objects about some events, the observer design pattern can be employed. Architectural patterns, on the other hand, do not address particular software tasks - they aim to provide templates for software organization based on the use cases of the software component in question. It sounds a bit complicated, but I hope an example will clarify: If some application will be used to fetch data from a remote server and present it to the user in a structured manner, then MVC might be a good candidate for consideration. In fact, even today most applications that I see still do not follow OOP best practices and do not show a clear logical organization of code. But today the situation is different - Google recently released the Data Binding library , which is fully integrated with Android Studio, and, even, rolled out a set of architecture blueprints for Android applications. Due to the fact that a search for a best architectural pattern for Android has officially been started, I think we are about to see several more ideas come to light. At this point, it is really impossible to predict which pattern or patterns will become industry standards in the future - we will need to wait and see I guess it is matter of a year or two. However, there is one prediction I can make with a high degree of confidence: Usage of the Data Binding library will not become an industry standard. Once long-term effects of this library will surface - it will be abandoned. In my applications I use my own implementation of an MVC architecture. It is simple, clean, readable and testable, and does not require any additional libraries. This MVC is not just cosmetically different from others - it is based on a theory that Activities in Android are not UI Elements , which has tremendous implications on code organization.

Chapter 3 : Android UI Design Patterns & Best Practices â€“ part 1

Note: Citations are based on reference standards. However, formatting rules can vary widely between applications and fields of interest or study. The specific requirements or preferences of your reviewing publisher, classroom teacher, institution or organization should be applied.

The Proliferation of UI Patterns One of the side effects of responsive design has meant that a lot of sites look similar. The rise of WordPress sites and the booming theme market also have a hand in it. In other words, a checkout will still be a checkout and should function as such. Same with a login model. UI patterns must guide users through a smooth experience. Silenza via awwwards Account registration: Photo credit Typeform Long scroll: Placing all your important elements above the fold is now a well-known myth. Furthermore, almost everyone is accustomed to long scrolls thanks to mobile devices. The technique works especially well for sites that want to lure users through storytelling, and you can still mimic a multi-page site by breaking the scroll into clear sections. Pioneered by Pinterest, cards are everywhere on the web because they present information in bite-sized chunks perfect for scanning. Each card represents one unified concept. Animations can be thought of in terms of two groups: These are used as a primary interaction tool have more impact on users and include effects like parallax scrolling and pop-up notifications. Loading animations These are used to entertain users and delight users during an otherwise tedious experience. Loading animations tend to be popular for flat design , minimalism , portfolios and one-page sites. Keep them simple and avoid adding sound. Navigation and menus nonscrolling Hidden navigation menus have become increasingly popular, especially as they can be used to save screen space. As you can see in the example below created in the collaborative prototyping tool UXPin , these use animations to reveal a menu when clicking on a specific button and prevent a jarring transition like a navigation drawer hidden behind a hamburger icon. Hover animations Hover effects give a more intuitive feel to a site as users mouse over content. Humaan Galleries and slideshows Galleries and slideshows are an effective way to showcase multiple images without overburdening the users. These are great for photography sites, product showcases, and portfolios. Motion can also help with visual hierarchy. This can help to add interest and intrigue to forms, CTAs and menu items. Bugaboo via awwwards Scrolling Smooth scrolling relies on animation and gives further control to the user, who can determine the pace of how the animation unfolds. The key is to work on individual sections or create a gentle movement of an entire image. Microinteractions Microinteractions happen all around us, from turning off the alarm on your mobile phone to liking that cat picture on Facebook. Each one done without a second thought. By turning off the alarm on your mobile phone, you engaged with a user interface in a single moment. And more and more of these are baked into the apps and devices we use. Micro-interactions tend to do, or help you do, several different things: Communicate a status or bit of feedback See the result of an action Help the user manipulate something Micro-interactions are a vital part of any app. Consider each detail with care, and make each interaction feel human. That is make text conversational and not robotic. Micro-interactions are an important part of almost every digital design project. Each of these interaction types lead users to a path of more human-centered design. This concept of making devices more human-like in their moments is a key to adoption and usability. It uses shadow effects and the concepts of movement and depth in order to create designs that appear more realistic to the user. With its minimalistic look, Material Design has a lot in common with another growing trend â€” flat design. Material Design, however, makes use of depth and shadow, which allows for more depth than pure flat design. Google however announced Material Design Lite in July, which is more suited to websites. Responsive Design Responsive web design has become incredibly popular in recent years thanks to the rise of mobile internet usage. But responsive web design does come with some issues if not carried out properly, the most important being performance. This still downloads the image to the device and adds unnecessary weight to a page. Use responsive images which are defined using a percentage. Use conditional loading for JavaScript as many of the JavaScript components used on a desktop

site will not be used on smaller devices. Pay particular attention to third-party scripts such as those used for social sharing as these often impact negatively and reduce performance. Performance is important not only to UX, but also to Google in the wake of the Mobile Friendly update which released in April Responsive web design is also highly compatible with minimalism, thanks to the necessity to keep page weight down. The Guardian Responsive web design is becoming less of a trend and more of a best practice. And designers have come up with clever ways to get around any speed issues. These bring more depth to flat designs. Popular UI frameworks and templates have prompted many to begin using more vibrant colors in their designs. Simple typefaces help to ensure that text remains legible and readable in flat design. These allow for functionality without distracting from the UX and are often represented as outlined, clickable links that change when the user hovers over them. Looks to cut down on the number of elements in order to create a fresh, uncluttered UI. Trends are nothing more than additional tools in your designer toolbox. Always pick the right ones for the job. For more advice on the 10 most important web design trends, check out the free e-book Web Design Trends and Related Articles 10 Collect.

Chapter 4 : 7 tips to create awesome mobile app designs

You will gradually explore the different Design and layout patterns and learn the best practices on how to use them together. You will then develop an application that will help you grasp Activities, Services and Broadcasts and their roles in Android development.

Unfortunately, not all designers place a high priority on designing for excellent navigation, which does a disservice to clients, and visitors. To create a very credible reputation as a designerâ€”and to be an in-demand designerâ€”you have to master one of the most basic aspects of web design. Sticky navigation Sticky navigation is an essential component of navigation since it empowers your users to instantly access the menu and easily find what they want on your site. Sticky navigation stays locked in place as users scroll down a page. Remember that, on the Web, speed is everything loading pages, finding desired content, etc. Alternative Press features a good example of how to incorporate sticky navigation into your pages. Note how all the most important sections are easily accessible and identifiable, by effective color contrast, in the navigation bar, no matter how far you scroll down the page. Hyperlinks are probably the most identifiable example of hypertext. Overall, this adds to the navigability of the site you build. I recommend making hypertext blatant by two, simple, design features: Similarly, when you make the color of hyperlinks different than the text and background of your pages, you again, make it stand out on the page, so users can easily identify and click on it. Marketing site The Daily Egg uses both underlining and different colors for its hypertext, making it very easy for users to identify links. These are essentially huge, drop-down panels that open up additional layers of navigation to help your users find specifically what they want more quickly. Such menus improve the user experience by including benefits like: All told, these features make it a lot easier to use the navigation menu. Notice how the flyout menu displays one of the most sought-after pieces of content of the siteâ€”the recipesâ€in an easy-to-see format that lets users immediately choose where they want to go instead of scrolling or pagination for more options. Pattern and language affordances A pattern affordance is using a conventional symbol that the majority of site visitors will understand immediately, such as a house icon to represent the homepage of your site. Affordances communicate to users how they can interact with the elements in your design. Simply test the page speed of pages you design. There are a number of reliable tools to help you determine page load times:

Chapter 5 : Android Design Patterns and Best Practice. (eBook,) [blog.quintoapp.com]

Ebook Description. Create reliable, robust, and efficient Android apps with industry-standard design patternsAbout This Book Create efficient object interaction patterns for faster and more efficient Android development* Get into efficient and fast app development and start making money from your android apps* Implement industry-standard design patterns and best practices to reduce your app.*

Independent of any external agency. It is not a must to use only 4 circles as you can see in the picture , because they are only schematic but you should take into consideration the Dependency Rule: Here is some vocabulary that is relevant for getting familiar and understanding this approach in a better way: These are the business objects of the application. These use cases orchestrate the flow of data to and from the entities. Are also called Interactors. This set of adapters convert data from the format most convenient for the use cases and entities. Presenters and Controllers belong here. This is where all the details go: UI, tools, frameworks, etc. For a better and more extensive explanation, refer to this article or this video. Our Scenario I will start with a simple scenario to get things going: Here is a quick video: Android Architecture The purpose is the separation of concerns by keeping the business rules not knowing anything at all about the outside world, thus, they can can be tested without any dependency to any external element. To achieve this, my proposal is about breaking up the project into 3 different layers, in which each one has its own purpose and works separately from the others. It is worth mentioning that each layer uses its own data model so this independence can be reached you will see in code that a data mapper is needed in order to accomplish data transformation, a price to be paid if you do not want to cross the use of your models over the entire application. Here is an schema so you can see how it looks like: I did not use any external library except gson for parsing json data and junit, mockito, robolectric and espresso for testing. The reason is that it makes the example clearer. Anyway do not hesitate to add ORMs for storing disk data or any dependency injection framework or whatever tool or library you are familiar with, that could make your life easier. Presentation Layer Is here, where the logic related with views and animations happens. I will not get into details on it, but here fragments and activities are only views, there is no logic inside them other than UI logic, and this is where all the rendering stuff takes place. Presenters in this layer are composed with interactors use cases that perform the job in a new thread outside the main android UI thread, and come back using a callback with the data that will be rendered in the view. Domain Layer Business rules here: Regarding the android project, you will see all the interactors use cases implementations here as well. This layer is a pure java module without any android dependencies. All the external components use interfaces when connecting to the business objects. Data Layer All data needed for the application comes from this layer through a UserRepository implementation the interface is in the domain layer that uses a Repository Pattern with a strategy that, through a factory, picks different data sources depending on certain conditions. For instance, when getting a user by id, the disk cache data source will be selected if the user already exists in cache, otherwise the cloud will be queried to retrieve the data and later save it to the disk cache. The idea behind all this is that the origin of the data is transparent to the client, which does not care if the data is coming from memory, disk or the cloud, the only truth is that the information will arrive and will be gotten. In terms of code I have implemented a very simple and primitive disk cache using the file system and android preferences, it was for learning purpose. Error Handling This is always a topic for discussion and could be great if you share your solutions here. My strategy was to use callbacks, thus, if something happens in the data repository for example, the callback has 2 methods onResponse and onError. This approach brings some difficulties because there is a chains of callbacks one after the other until the error goes to the presentation layer to be rendered. Code readability could be a bit compromised. Testing Regarding testing, I opted for several solutions depending on the layer: JUnit plus mockito for unit tests was used here. Robolectric since this layer has android dependencies plus junit plus mockito for integration and unit tests. Show me the code I know that you may be wondering where is the code, right? Well here is the github link where you will find

what I have done. About the folder structure, something to mention, is that the different layers are represented using modules: It is an android module that represents the presentation layer. A java module without android dependencies. An android module from where all the data is retrieved. Tests for the data layer. Due to some limitations when using Robolectric I had to use it in a separate java module. I hope you have found this article useful and, as always, any feedback is more than welcome.

DOWNLOAD PDF ANDROID DESIGN PATTERNS AND BEST PRACTICE 2016

Chapter 6 : Which design patterns are used on Android? - Stack Overflow

Get this from a library! Android design patterns and best practice: create reliable, robust, and efficient Android apps with industry-standard design patterns. [Kyle Mew].

Building flows around content gives you a much more accurate assessment of the total number of pages required for your app. From there, you could continue iterating the sketches on paper and cut them out for a paper prototype, or move to a digital prototyping tool like UXPin. The outline helps you quickly explore different page flows. The sketches bring those flows to life with more detail around layout and structure. Finally, a quick prototype helps you test those ideas with users. Enhance Usability With Familiar Mobile Patterns Mobile design revolves around many device-specific nuances , such as thumb placement , and orientation and posture. Use common UI patterns as a baseline for usability, then layer on your own creativity. As showcased in the free e-book Mobile UI Design Patterns , there are two categories of interaction design patterns you must master for good mobile design. Touch devices are defined by gestures. Touch, swipe, double-tap, pinch and zoom are becoming second nature to users. Motion keeps users grounded in the UI while adding context. The former is deletion; the latter is available for use later. When animations are combined with gestures, they add another depth to the experience. Mobile interaction patterns help dictate the layout of common interface elements. For example, navigation buttons at the bottom of the screen are easier for users to tap with a thumb than buttons at the top. The app is very clean, with nice, big buttons that state their purpose. Some tips for text: Phrase labels positively to make users feel in control. Important words must appear first. Wording must be consistent and styled uniformly across each screen. Check out the UI libraries capptivate and Use Your Interface for excellent interface examples that incorporate all the gestural, animated, and textual nuances of mobile design. Specifically, allow enough space for users to tap with a fingertip. Here are a few tips to keep in mind when designing buttons and other touch targets: We hold our phones in different ways. According to Josh Clark, there are three ways to hold a phone: And there are also different ways to hold a tablet, but users mostly hold tablets on the side. Our fingers are fat. But they are currently about pixels wide , which is bigger than what most guidelines recommend for touch targets. For instance, Apple recommends a target around 44 pixels square. Kennedy points out in his excellent Medium article , shadows help our brains interpret a user interface â€” which is probably why they were so common during the full-skeuomorphic fad. When thinking about buttons, toggles and other visual cues, you must consider the use of shadows. You can use shadows and gradients to create 3D buttons and input forms, where the effect makes the element appear inset or outset.

Chapter 7 : Mew K. - Android Design Patterns and Best Practice [, PDF/EPUB/MOBI, ENG] :: blog.quintoa

These patterns act as a guide, creating a clear path from problem to solution, and although applying a design pattern does not guarantee best practice in itself, it will hugely assist the process and make the discovery of design flaws far easier.

Chapter 8 : 6 Web Design Trends You Must Know for &

Android Design Patterns is a website for developers who wish to better understand the Android application framework. The tutorials here emphasize proper code design and project maintainability.

Chapter 9 : Android Design Patterns and Best Practice - WOW! eBook

Design your app to work offline, and it'll work beautifully all the time. And, as promised, they released their sample code

(blog.quintoapp.com), so that you can see it for yourself. Adam and Yigit illustrated these best practices by walking the audience through some common scenarios to avoid, by offering clear and actionable advice, and.