DOWNLOAD PDF CLONING INTERNET APPLICATIONS WITH RUBY

Chapter 1 : Cloning Internet Applications with Ruby - Technology and Engineering eBooks

*Cloning Internet Applications with Ruby [Chang Sau Sheong] on blog.quintoapp.com *FREE* shipping on qualifying offers. This is a hands-on book with plenty of well-explained code. Each chapter has a standalone project in which a complete web application with specific features of a social networking site is emphasized.*

This has been true since the time of Newton and even before and it is certainly true now. Much of what we know and learn of programming, we learnt from the pioneering programmers before us and what we leave behind to future generations of programmers is our hard-earned experience and precious knowledge. This book is all about being the scaffolding upon which the next generation of programmers stands when they build the next Sistine Chapel of software. There are many ways that we can build this scaffolding but one of the best ways is simply to copy from what works. Many programming books attempt to teach with code samples that the readers can reproduce and try it out themselves. This book goes beyond code samples. We explore how these applications are coded and also the rationale behind the way they are designed. The aim is to guide the programmer through the steps of building clones of the various popular Internet applications. What This Book Covers Chapter 1, Cloning Internet Applications gives a brief description of the purpose of the book, the target readers of the book, and a list of the four popular Internet applications we will be cloning in the subsequent chapters. The bulk of this chapter gives a brief rundown on the various technologies we will be using to build those clones. The clone in this chapter emulates one of the hottest and most popular Internet web applications nowâ€”Twitter. It describes the basic principles of a microblogging application and explains how to recreate a feature-complete Twitter clone. Chapter 4, Photo -sharing â€” Cloning Flickr. Flickr is one of the most popular and enduring photo-sharing applications on the Internet. This chapter describes how the reader can re-create a feature complete photo-sharing application the simplest way possible, following the interface and style in Flickr. The final two chapters describe the various aspects of Internet social networking services, focusing on one of the most popular out there nowâ€”Facebook. These two chapters also describe the minimal features of a social networking service and show the reader how to implement these features in a complete step-by-step guide. The first part is described in this chapter, which sets the groundwork for the clone and proceeds to describe the data model used in the clone. The final chapter is part two in describing how to create a Facebook clone. This chapter follows on the previous chapter and describes the application flow of the Facebook clone we started earlier. Social Networking Services â€” Cloning Facebook 1 One of the most dominant Internet services today is the social networking service. According to a report by the Nielsen Company, in January , the amount of time an average person spent on Facebook is more than seven hours per month, which amounts to more than 14 minutes per day. If you lump together the time spent on Google, Yahoo! By March , Facebook accounted for more than seven percent of all Internet traffic in the United States, surpassing visits to Google. Social networking services have risen in the past few years to be more than just a passing fad, to be an important communications tool as well as a part of daily life. We will be building our last and most complex clone based on Facebook, the most popular social networking service as of date. The clone we will build here will be described over this and the next chapter. In this chapter we will cover basic information about social networking services, main features of the clone that we will build, as well as the description of the data model we will be using for the clone. All about social networking services A social networking service is an Internet service that models social relationships among people. Essentially it consists of a user profile, his or her social links, and a variety of additional services. Most social networking services are web-based and provide various ways for users to interact over the Internet, including sharing content and communications. Early social networking websites started in the form of generalized online communities such as The WELL , theglobe. These early communities focused on communications through chat rooms, and sharing personal information and topics via simple publishing tools. Other communities took the approach of simply having people link to each other via e-mail addresses. These sites in cluded Classmates , focusing on ties with former schoolmates, and SixDegrees , focusing on indirect ties. The basic features of the first online social networking services include user profiles, adding friends to a friends list, and sending private messages.

Unfortunately, SixDegrees was too advanced for its time and eventually closed in Interestingly the most popular social networking service in Korea, CyWorld, was started around this time in The original intention for CyWorld was to develop an online dating service similar to Match and provide an open public meeting place for users to meet online. In , CyWorld launched the minihompy service, a feature that allows each user to create a virtual homepage. This was highly successful as celebrities and politicians took to this platform to reach out to their fans and audience. Between and , a few social networking services became highly popular. Friendster, started by Jon Abraham in to compete with Match. However due to platform and scalability issues, its popularity plummeted as newer social networking services were launched. MySpace , launched in , was started as a Friendster alternative and became popular with independent rock bands from Los Angeles as promoters used the platform to advertise VIP passes for popular clubs. Subsequently, MySpac e facilitated a two-way conversation between bands and their fans, and music became the growth engine of MySpace. MySpace also introduced the concept of allowing users to personalize their pages and to generate unique layouts and backgrounds. Eventually MySpace became the most dominant social networking service in U. Mixiis the largest online social networking service in Japan with a total of 20 million users to date and over ninety percent of users being Japanese. Launched in February by founder KenjiKasahara, the focus of Mixiis to enable users to meet new people who share common interests. This feature is only found in niche and private social networks such as http: This invitation-based model holds the user responsible for who they invite, and thus reduces unwanted behavior within the network, refl ecting Japanese culture itself. Social networking began to emerge as a part of business Internet strategy at around when Yahoo! It was around this time as well that the first mainland Chinese social networks started. The three most notable examples in chronological order are On the other hand, Xiaoneihas a user interface that follows Facebook, though it also offers the user flexibility to change the look and feel, similar to MySpace. Kaixin, the latest social networking platform in China with the fastest growing number of users, started in and the platform and user interface are remarkably similar to Facebook. It was also around this time that more niche social networking services focusing on specific demographics sprang up, with the most successful example being LinkedIn, which focused on business professionals. At the same time media content sharing sites began slowly incorporated social networking service features and became social networking services themselves. As mentioned earlier, as of early social networking services are the dominant service and purpose for many users on the Internet, with Internet traffic in US surpassing the previous giant of the Internet. Facebook Facebook is the most dominant social networking service till date, with million active users, 5 billion pieces of content shared each week, and more than million active users concurrently accessing Facebook through their mobile devices. It is also the most widespread, with 70 percent of its users from outside of US, its home market. Mark Zuckerberg and some of his Harvard college roommates launched Facebook in February Initially intended as an online directory for college students the initial membership was limited to Harvard College students it was later expanded to include other colleges, then high schools, and finally anyone around the world who is 13 years old and above. Facebook features are typically that of many social networks that were created around that time. Over time, Facebook included features to form virtual groups, to blog, to start events, chat with instant messaging, and even send virtual gifts to friends. Facebook launched Facebook Platform in May , providing a framework for software developers to create applications that interact with Facebook. It soon became wildly popular, and within a year , developers have registered for the platform, and built 33, applications. As of writing date there are more than , active applications in Facebook, developed by more than 1 million developers and there are more than applications with more than 1 million monthly active users! In this chapter we will be cloning Facebook and creating an application called Colony, which has the basic but essential features of Facebook. Main features Online social networking services are complex applications with a large number of features. However, these features can be roughly grouped into a few common categories: User Content-sharing Developer User features are features that relate directly to and with the user. For example, the ability to create and share their own profiles, and the ability to share status and activities are user features. Community features are features that connect users with each other. An example of this is the friends list feature, which shows the number of friends a user has connected with in the social network. Content sharing

features are quite easy to understand. These are features that allow a user to share his self-created content with other users, for example photo sharing or blogging. Social bookmarking features are those features that allow users to share content they have discovered with other users, such as sharing links and tagging items with labels. Finally, developer features are features that allow external developers to access the services and data in the social networks. While the social networking services out in the market often try to differentiate themselves from each other in order to gain an edge over their competition, in this chapter we will be building a stereotypical online social networking service. We will be choosing only a few of the more common features in each category, except for developer features, which for practical reasons will not be implemented here. User User features are features that relate directly to users: Users can post brief status updates to all users. Users can add or remove friends by inviting them to link up. Friendship in both ways need to be approved by the recipient of the invitation. Community Community features connect users with each other: Users can post to a wall belonging to a user, group, or event. A wall is a place where any user can post on and can be viewed by all users. Users can send private messages to other users. Users can create events that represent an actual event in the real world. Events pull together users, content, and provide basic event management capabilities, such as RSVP. Users can form and join groups. Groups represent a grouping of like-minded people and pulls together users and content. Users can comment on various types of shared and created content including photos, pages, statuses, and activities. Comments are textual only. Users can indicate that they like most types of shared and created content including photos, pages, statuses, and activities. Content sharing Content sharing features allow users to share content, either self-generated or discovered, with other users: Users can create albums and upload photos to them Users can create standalone pages belonging to them or attached pages belonging to events and groups You might notice that some of the features in the previous chapters are similar to those here.

Chapter 2 : PDF Cloning Internet Applications With Ruby Free Download | Download PDF Journalist Esdel

Cloning Internet Applications with Ruby Make clones of some of the best applications on the Web using the dynamic and object-oriented features of Ruby Preview Online.

By Darren Jones Rails or Sinatra: The Best of Both Worlds? The Ruby world has been blessed with more than its fair share of frameworks for developing web applications over the years. Recently, though, two of them have emerged as the clear leaders in the field. Most readers of this site will have heard of Ruby on Rails. It was created by in by David Heinemeier Hansson and provides a MVC framework that helps to boost productivity whilst keeping developers happy. Sinatra was created by Blake Mizerany in and is a domain-specific language that wraps a lightweight HTTP request and response layer on top of Rack. The stats at RubyGems. Rails has 7 million downloads and Sinatra has 1. In fact I have found that Sinatra, along with a few gems, is usually all I need when building an app. This led me to wonder whether Sinatra is suitable for tackling any project. When is is best to use Sinatra and when is Rails the right choice? To answer my question, I asked for the thoughts of some prominent Ruby developers. Konstantin Haase is the current maintainer of Sinatra and feels that they both cater for different types of application: While Rails is a framework focused on writing model driven web applications, Sinatra is a library for dealing with HTTP from the server side. If you need full integration and as much boilerplate as possible, Rails is the way to go. David Heinemeier Hansson also believes that there was room for both of them, but feels that is is size of your app that should influence which one to use: Sinatra is great for the micro-style, Rails is not. As long as you stay micro, Sinatra will beat Rails. If you go beyond micro, Rails will beat Sinatra. In fact, most people tend to agree that Rails is better suited to big projects whereas Sinatra is a better fit for small micro apps or apis. David goes on to point out a general rule of thumb for choosing whether to use Rails or Sinatra: Essentially if your entire controller structure can fit in a page or two of code, running it in a single file is great. Rick Olson of Github confirms that Sinatra does indeed make a great choice for the smaller stuff: David Heinemeier Hansson explains why Rails is such a good choice when it comes to building big web applications: Konstantin Haase shares his enthusiasm for Rails and its philosophy: Rails has been pushing the way we think about web applications. Rails is solving issues for you Sinatra never can, since it can make more assumptions about your application. This can be ultimately worth it, especially when it comes to creating a big project. Not everyone agrees that Rails is the only choice for a big project, however. Despite this, Konstantin Haase highlights a potential problem with using Sinatra to create big applications: Writing a more complex Sinatra application or rather, an application using, amongst other things, Sinatra requires the developer to spend way more time thinking about architecture and tooling. This is both an advantage a disadvantage. Some developers would certainly see this as an advantage as they prefer the tight control over what goes into their application and understanding of how it all fits together. Geoffrey Grosenbach thinks that this is a trade off that is definitely worth it: A huge benefit of Sinatra is its stability. You trade the convenience of a few Rails generators for your own design decisions. Writing in Sinatra can give developers and businesses API stability because the framework rarely changes. You own your code and you decide when it should change. He goes on to say that Sinatra is the perfect candidate when it comes to building an application: Sinatra fulfills all I need for a base platform and anything else I need I can either use gems or is already provided by cloud platforms and services like Heroku and its eco-system of add-ons. Anything else left over I can write it myself. In fact, this idea could be taken a step further by using Sinatra to roll your own bespoke framework that fits your needs perfectly. Start with Sinatra and then add in the gems that provide the features you need in order to create your own bespoke framework. In some ways, this is what the Padrino project has done. Breaking everything into tiny little pieces and having developers assemble everything on their own is the antithesis of what Rails is all about and how it won the framework game. Tens of thousands of man hours have gone into getting us to this point. But while having more control over what goes into your app might be a nice idea, Konstantin Haase warns that this could take up a lot of your time: The biggest disadvantage with Sinatra not solving the issue for you is, well, Sinatra not solving the issue for you. You actually have to deal with that issue. You might end up wasting too much time

dealing with it. And this is time that the developers simply may not have at their disposal if they are working to a tight budget. David Heinemeier Hansson is concerned that using Sinatra involves wasting so much time trying to reinvent the wheel: I pity the person who would try to build all of Basecamp, GitHub, Shopify, or any other major app in Sinatra alone. Rails is involved and large because it contains solutions to most of the problems that most people building apps of such scale will encounter. Trying to recreate all these solutions by hand is the anti-simplicity. Ryan Bates, presenter of Railcasts feels that an advantage of using Rails is the amount of time that is saved when starting a project from scratch: The default Rails app provides a lot that I need which requires extra setup in Sinatra. This can lead to faster development in Rails. This sentiment is echoed by Rick Olson who believes that Rails made it easy to get GitHub off the ground: I think Rails was a major asset to the founders [of GitHub] in the first year. They were able to take advantage of some of the higher level Rails features and iterate quickly. The problem is that these conventions can sometimes feel like a straight jacket at times as Rick Olson points out: WDNNSP means that Sinatra is beyond flexible, and it contains no preconceptions of how you organize your domain or business logic. It has no implicit folder structure, no default Database Abstractor, and no restrictions about how or where you use it. Using Sinatra can be liberating, as you are not restrained by size, structure or work flow. You can build an API, web application or pack your code up as a gem. So is there anything that Sinatra brings to the table that he would consider adding to Rails? We considered adding a single-file mode to Rails but rejected it because why would we want to optimize for something Sinatra already does so well? Rails 3 tries to mitigate this as much as possible by decoupling some of the different features, which can lead to a lighter and more confgurable exerience, as Chad Fowler points out: Rails 3 introduced a lot of configuration options that allow you to tailor the experience to suit what you want to do. Despite the ability to cut out the cruft, Konstantin Haase warns that it is often just too tempting to keep it all in there, which leads to bloat: In my experience, Rails apps often and up as one monolithic application. However, you could reach the same architecture while carefully constructing Rails applications, it is just very tempting not to. There is nothing practical gained by physically removing a piece of code in Rails that you do not use. Nobody is forcing you to use all features. Rails is becoming lighter and Sinatra has shown it can do the heavy lifting which means that there is an increasing amount of overlap between the two. Chad Fowler believes that, in actual fact, both Sinatra and Rails could be used for a variety of projects: I think in truth each could be used for both ends of the spectrum. Ryan Bates summarizes what type of person would be more likely to choose which framework: In the end I think it depends on your preference, whether you like to start lighter and add as needed, or start heavier and remove something to make lighter if needed. Geoffrey Grosenbach proposes that there are fundamentally two different types of developer: Sinatra provides the former, Rails provides the latter. If anything recent developments have made the 2 frameworks complimentary. Sinatra has let me modularise my app in a way I have not been able to do before so easily. In fact Chad Fowler feels that the whole concept of choosing one over the other is meaningless: Geoffrey Grosenbach feels that this will give apps a more modular feel: Many people are embedding Sinatra apps inside Rails apps. This mimics the design of the Django framework, where an app consists of many smaller apps that handle specific parts of the app and are often reusable between apps. David Heinemeier Hansson also feels this was a nice way of doing things: You can even have micro Sinatra apps inside your Rails structure. GitHub has that for a few things. Rick Olson ellaborates on how GitHub frequently use the pair in tandem: Rather than trying to force Rails to your will for a specific feature, you can route the request to a Sinatra or Rack endpoint and do exactly what you need. Everybody agrees that there is a place for both of these frameworks in the Ruby web ecosystem. In fact, they work incredibly well together and in many ways complement each other to a certain degree. A lot of people seem to think that the two frameworks work best together and fill different needs in general. Where there is some overlap, they appeal to different types of developers. They both do what they do so well and are exceptional examples of good practice â€” so much so that they have been copied to death in other languages.

Chapter 3 : Ruby Phrasebook - pdf - Free IT eBooks Download

DOWNLOAD PDF CLONING INTERNET APPLICATIONS WITH RUBY

Cloning Internet Applications with Ruby. We stand on the shoulders of giants. This has been true since the time of Newton (and even before) and it is certainly true now.

Chapter 4 : Cloning Internet Applications with Ruby - ePub - Chang Sau Sheong - Achat ebook | fnac

Cloning Internet Applications with Ruby has 8 ratings and 1 review. This is a hands-on book with plenty of well-explained code. Each chapter has a standa.

Chapter 5 : Chang Sau Sheong (Author of Cloning Internet Applications with Ruby)

Cloning Internet Applications with Ruby, eBook de. Editorial: Packt Publishing. Â¡DescÃ¡rgate ya la versiÃ³n de eBook!

Chapter 6 : Cloning Internet Applications with Ruby

Book Description. Make clones of some of the best applications on the Web using the dynamic and object-oriented features of Ruby. Build your own custom social networking, URL shortening, and photo sharing websites using Ruby.

Chapter 7 : Cloning Internet Applications with Ruby | PACKT Books

Find helpful customer reviews and review ratings for Cloning Internet Applications with Ruby at blog.quintoapp.com Read honest and unbiased product reviews from our users.

Chapter 8 : Cloning Internet Applications with Ruby - Sinopsis y Precio | FNAC

Download cloning internet applications with ruby ebook free in PDF and EPUB Format. cloning internet applications with ruby also available in docx and mobi. Read cloning internet applications with ruby online, read in mobile or Kindle.

Chapter 9 : Rails or Sinatra: The Best of Both Worlds? â€” SitePoint

Read "Cloning Internet Applications with Ruby" by Chang Sau Sheong with Rakuten Kobo. This is a hands-on book with plenty of well-explained code. Each chapter has a standalone project in which a complete we.