2-2014

Gentle Introduction Resource

James Anderson Laird Evans
Graduate Center, City University of New York

How does access to this work benefit you? Let us know!
Follow this and additional works at: https://academicworks.cuny.edu/gc_etds
Part of the Communication Technology and New Media Commons, Curriculum and Instruction Commons, and the Instructional Media Design Commons

Recommended Citation
https://academicworks.cuny.edu/gc_etds/140

This Thesis is brought to you by CUNY Academic Works. It has been accepted for inclusion in All Dissertations, Theses, and Capstone Projects by an authorized administrator of CUNY Academic Works. For more information, please contact deposit@gc.cuny.edu.
GENTLE INTRODUCTION RESOURCE – DIGITAL HUMANITIES WEB APPLICATION

by

JAMES ANDERSON LAIRD EVANS

A Master’s thesis submitted to the Graduate Faculty in Liberal Studies in partial fulfillment of the requirements for the degree of Master of Arts, The Graduate Center, The City University of New York

2014
This manuscript has been read and accepted by the Graduate Faculty in Liberal Studies in satisfaction of the thesis requirement for the degree of Master of Arts

Professor Matthew K. Gold _________________________________

_________________________ _________________________________________
Date Thesis Advisor

Professor Matthew K. Gold _________________________________

_________________________ _________________________________________
Date Executive Officer

THE CITY UNIVERSITY OF NEW YORK
Abstract

Gentle Introduction Resource

by

James Anderson Laird Evans

Thesis Advisor: Matthew K. Gold

The Gentle Introduction Resource (GIR) is an open-access web application that offers digital humanities researchers a malleable platform with which they can crowdsource technical material. Such material often poses challenges to the traditional humanities researcher when trying to implement digital layers to his or her research. The Gentle Introduction Resource (GIR) is an open-access web application designed to allow a single classroom or single institution to crowdsource technical information/instruction filtering out the influence of rhetorical analysis. It is written in the Ruby language on the Rails web framework. Though a GIR instance can be invoked to crowdsource primarily web-based information on any dictated theme, though metadata on physical media can be utilized as well. Initializing the GIR creates a private microblog network made up of an administrative super user, general users, and a shared datastream which can then be used to source relevant material. The GIR is also reliant on programmed constraints decided on by the administrator. Already included constraints include word count (which can be altered), set pagination, and the ability to examine output only from followed users. Topics focused on during a GIR instance can be customized and reconsidered by users during live/in-person sessions, addressing problems with filtration
of big data from an exponentially growing number of users inherent in larger, open social networks like twitter. The code running the application is released under an Open Source MIT license that allows for complete code alteration with proper attribution, giving researchers freedom to experiment with the code, to add personalization, and to customize its functionality.
# TABLE OF CONTENTS

AN INTRODUCTION TO THE GENTLE INTRODUCTION RESOURCE ........................................... 1

A BLANK CANVAS OR LEARNING BY DOING ................................................................. 12

EVALUATING INITIALIZATION OF THE GENTLE INTRODUCTION RESOURCE .............. 20

MOVING FORWARD ........................................................................................................... 21

BRIEF ASIDE ON LONG-TERM IMPACT ........................................................................ 22

REFORMING DIGITAL PEDAGOGY (CONCLUSION) ......................................................... 22

APPENDICES .................................................................................................................... 24

  Appendix 1: Illustrations ............................................................................................... 24

  Appendix 2: Code Examples ......................................................................................... 25

  Appendix 3: Application Screenshots ......................................................................... 40

BIBLIOGRAPHY ................................................................................................................ 48
GENTLE INTRODUCTION RESOURCE - DIGITAL HUMANITIES WEB APPLICATION

An Introduction to The Gentle Introduction Resource

In his article “What is Digital Humanities and What’s It Doing in English Departments?” Professor Matthew Kirschenbaum cites Wikipedia for what he calls a “working definition” of Digital Humanities (DH) that he claims “serves as well as any I’ve seen”:\(^1\)

> The digital humanities, also known as humanities computing, is a field of study, research, teaching, and invention concerned with the intersection of computing and the disciplines of the humanities. It is methodological by nature and interdisciplinary in scope. It involves investigation, analysis, synthesis, and presentation of information in electronic form. It studies how these media affect the disciplines in which they are used, and what these disciplines have to contribute to our knowledge of computing.\(^2\)

Such a “working definition” may serve to enlighten one that simply wants to know what DH is, but for a student wanting to engage in DH research, many of the stated goals and functions of the discipline can lead students and researchers down vastly differing paths. How is one to investigate, analyze, synthesize, and present in electronic form? The technical skills needed in the emerging field of DH are difficult to pin down; the exploration of humanities content using digital tools can be successfully implemented with both raw, multifaceted computer code, and prepackaged user-based software.

---


\(^2\) Ibid.
In surveying major voices from the DH field, one can see that approaches to technical skills are far from homogenous, even among the most noted practitioners. Some DH scholars raise digital expertise--specifically the use of coding languages--to as high a station as the humanities content such expertise is being used to explore and research. At the 2011 MLA conference Professor Stephan Ramsay made this remark: "Do you have to know how to code? I’m a tenured professor of digital humanities and I say ‘yes.’ So if you come to my program, you’re going to have to learn to do that eventually."³ This perspective aroused massive debate among conference attendees and the DH community at large. In Matthew K. Gold's introduction to Debates in the Digital Humanities, Gold suggests that such debates illustrate the emergence of the DH field from a "small circle of like-minded scholars" to a "heterogeneous set of practitioners who sometimes ask more disruptive questions."⁴ A scholar like Ramsay might dissect the formation of the Fibonacci sequence in its classic mathematical notation alongside its manifestation as algorithmic syntax in the Lisp computer language, as he does in his most recent book Reading Machines.⁵ Others approach subjects like computer language with far less vigor and enthusiasm. They tend to focus on pragmatic goals in relation to digital engagement, leaving expectations of technical expertise separated or removed from humanities based scholarship.

What I found was that regardless of the path I might choose to take in making DH research work for my own academic goals, it was up to me to discover what core

---
⁵ Stephen Ramsay, Reading Machines (University of Illinois Press, 2011), 64-65
competencies were necessary in relation to my own path. I appreciated the freedom that came with this realization, but finding instructional material amidst the bulk of rhetorical texts assigned in the classroom was difficult. Professors did an admirable job pointing out techniques used by DH authors, but anxious students tended to gloss over practical computational discussions in favor of larger discussions about the nature/significance of DH. It was clear that newcomers to DH, unfamiliar with code, were quick to appreciate the larger philosophical questions raised by DH, but were often at a loss when it came to the subject of technical expertise. These observations informed my first steps toward a final thesis project: The Gentle Introduction Resource. The Gentle Introduction Resource or GIR came into being as a response to my inability as a developing digital humanist to find a collection of coherent introductions to contemporary digital/computational techniques that might have utility in my research.

I have addressed why The GIR was conceived, but this isn’t very telling. What is a “gentle introduction,” and how might it alleviate the aforementioned issues as they relate to DH? A Gentle Introduction is an introduction to difficult material simplified to a point where researchers new to a given subject could clearly understand core conceptual features. While there is no dictionary-sanctioned definition to be found for "Gentle Introductions," I have found examples of its use as I’ve described it in academic writing as far back as 1963. The moniker tends to be applied to subject matter so dense that even a general introduction would prove far too difficult to comprehend.

---

without some prerequisite understanding. In my original thesis proposal,\(^7\) I pointed to gentle introductions I had found useful in my research, such as University of Pittsburgh Professor David J. Birnbaum’s approach to XML (eXtensible Markup Language). His “What is XML and Why Should Humanists Care? An Even Gentler Introduction to XML”\(^8\) is a prime example of a well-formed gentle introduction. Not only does he give clear examples of the language’s functional utility, but he also presents a rich depiction of the language’s materiality with elegance. Take the following excerpt from Birnbaum’s piece:

XML is a formal model that is based on an ordered hierarchy, or, in technical informatic terms, a tree. It consists of a root (which contains everything else), the components under the root, which contain their own subcomponents, etc. These components and subcomponents are called nodes. (Birnbaum 2012)

Birnbaum uses his rhetorical skills to draw a clear picture of foundational rules of XML, framing information clearly and deeply, serving both novice and expert. I compared this piece of open scholarship to the results I found merely typing “XML” into the Google search engine. The first deliverable reads as follows:

XML stands for eXtensible Markup Language.

XML is designed to transport and store data.

XML is important to know, and very easy to learn.

---


Start learning XML now!9

This block of text from W3Schools.com tells the reader very little, and is colloquial to a fault. Such information appearing at the top position in a Google keyword search shows how important it is that resources beyond the search engine are provided for DH researchers. What Birnbaum’s gentle introduction proves is that the relevant scholarship is available; it just needs to be collected and surfaced.

I was able to find a number of resources aimed at aggregating tools and software purportedly useful in DH research. One such resource was Bamboo DiRT wiki,10 which had a very comprehensive list of software titles, but without a sharp focus on rules of use and technical guidelines, this resource appeared as a grab-bag of software titles (many proprietary) and a mis-aligned aggregation of topics without malleable infrastructure or a proper vetting system. Because many of my peers (PhD and MA students in the humanities) were quick to point out and describe failed attempts at discovering useful/readable explanations of computational techniques, it became apparent that addressing such issues could prove beneficial. Difficulty with comprehending of these facets of the digital field often led to negative opinions about relevant tools and concepts. I concluded that it would be beneficial to create a platform that allowed students in a class, working together in a physical space, to have their own micro-network that did not encourage an overly lengthy contribution void of constraint and susceptible to opinionated rhetoric, when merely a link and sound metadata might

---


better serve the community. By framing an entire network as directly linked to a theme, in this case gentle introductions, users create a specific resource fit to a specific purpose: collecting exterior sources of technical instruction seen as useful to both expert and novice. Merely streamlining a micro-network instance to bare essentials may seem underwhelming for such a crucial goal, but I would argue that such minimalist pursuits are important pedagogical tools. DHers have a history of learning by doing, and engaging a large, hyper-dynamic network has the potential to overwhelm. Open construction of a simple deliverable provides the opportunity for engaged researchers to experiment with function in the digital space. My goal was to create a web application that could stand apart from other forms of digital scholarship. I wanted to build something that would solve some of the issues suffered by Bamboo DiRT wiki and could serve as a supplement to classes making use of the CUNY Academic Commons ecosystem.\footnote{City University of New York Academic Commons. http://commons.gc.cuny.edu/}

The two projects that probably had the biggest influence on my construction of The GIR were Bamboo DiRT wiki and the CUNY AC. In my initial proposal, I made several points explaining why the Bamboo DiRT wiki\footnote{Project Bamboo, Bamboo DiRT wiki, 2008-2012. http://dirt.projectbamboo.org/} was incapable of solving issues I found to be undermining a significant amount of research in the DH community. The site collects relevant links to software relevant to DH research, but I felt the breadth of content was more exhaustive than the site’s dynamic filtration system could handle. I did not critique the CUNY AC in the proposal, but certainly had it in mind, as most of my
DH-centric courses made use of the network. The CUNY AC is a WordPress/BuddyPress-powered social network comprised of groups, profiles, discussion forums, and blogs connected to one another and composed by academics within the City University of New York system. I believe the CUNY AC to be a major step toward an evolved academic digital space. However, the familiarity of the AC's blogging platform can prove to be as problematic as it is beneficial to classrooms utilizing it for scholarship and technical instruction in tandem. Blogs are an excellent way to exercise intermediate research productivity, allowing researchers to ask questions, keep a record of sources, and consider positions held by their peers in relation to the topic (or topics) at hand. The practice of blogging also allows scholars to share their research with the public swiftly. Yet, when blogs become a catchall resource, manifestations of digital competency can appear in a less productive context. Questioning why specific tools, prepackaged software, or computer languages are used to address Digital Humanities based goals does require a significant amount of introspection and communication, which a blog structure is well suited for, but teaching expertise in relation to such topics within the same virtual space becomes challenging. Graduate students in the humanities are well equipped to analyze and critique traditional humanist texts upon acceptance into graduate programs in the humanities, but technical expertise among such students varies significantly. I saw this to be the most burdensome handicap of CUNY AC blog use. Students who garnered enlightening insight from texts would attempt to participate in technical competencies discussed in such assigned research. These students, myself included, would embed
links to external tutorials and introductions in their text discussion. Such resources, both
good and bad, would be treated primarily as asides, easily skimmed over when
surrounded by engaging debates, more focused on textual analysis than functional
instruction.

The GIR seeks to address this problem by creating an infrastructure that
separates instructional/technical expertise from dense rhetoric. This is accomplished by
initializing a platform that limits user submissions to posts with no more than 140
characters. Using a micro-post model file (a type of Rails file responsible for validation
of user actions), one can allow not only for more or less characters in a post
submission, but also for customized “gems” (precompiled open source Ruby packages),
such as dynamic hashtags and star rating systems. I will discuss the need for such
features in the “Evaluating Initialization of The GIR” section of this paper. With The GIR,
students are able to minimize noise and rhetorical discussion surrounding crowd-
sourced external material, and can quickly move between micro-posts of instructional
material that share a theme. Instead of nesting exterior technical instruction within long-
form reactions to readings, scholars are presented with The GIR, a micro-network
designed specifically for sharing links to sources of instruction.

Certainly the CUNY AC is comprehensive enough to allow for two blogs per
classroom, one for analysis and one for technical instruction links, but by initiating a
microblog data stream, students can explore making/building without anxiously
suspecting they are to share far more than should be expected before competency is
achieved. While blog posts can be short, typically they are longer. A micro-blog
format’s aesthetic must be short, and its aesthetic makes this very clear. No doubt the AC is also capable of accommodating a microblog data stream from within, but I wanted to implement something that could exist outside that system. By building something outside the system, I could create a deliverable I had complete power over; I would be better equipped to produce transparent code in full that might benefit those engaging with The GIR on a meta-pedagogical level.

While collecting introductions to technical instruction was the foundation of the application, it was equally important to construct a code-based deliverable with constraints that were transparent and malleable. I had seen far too many researchers spend a great deal of time concentrating on mastering particular proprietary software packages. Such software has the tendency to hide the bulk of its infrastructure within an impenetrable, inaccessible black box. This makes it nearly impossible to seamlessly move content from the original platform to a newer platform during rapid paradigm shifts of technological innovation that frequently occur. From the start, I knew that all aspects of my resource needed to be immediately visible and immediately alterable in a form graspable by a researcher with a humanities background, but not necessarily a technical one. In the end, by using the Ruby computer language and Ruby-based web framework, Rails, it became possible for me to emulate both the aesthetics and function of popular social networks reliant on the aggregation of data streams (like Twitter). It was not an effortless task, but a task I completed nonetheless. The GIR provides an intuitive web space where students can crowdsource learning relating to any theme the class instructor (or admin) dictates. The proof-of-concept on the live web asks
registered users specifically to share introductions to technical materials, but the free/open-source code allows alteration for any topic an instructor might wish to focus on.

The GIR homepage\textsuperscript{13} provides a case study that helps illustrate how a user might engage the site:

Here is a story to illustrate how this web app can be utilized in a classroom setting:

Jack is a student seeking a Master’s Degree in History at Everyman’s University. Jack comes from a traditional Humanities background, but is taking a course in Digital Humanities. He hopes to learn how he can inject digital, computational, and web-based components into his research. Jack has owned a computer all of his life, so he figured it shouldn't be too hard for him. At first it wasn't, but about halfway into the semester his homework began to seem overly challenging...

[transcribed from comic illustration of “Jack’s” speech bubble: “What the devil?! XML? I've barely even heard of XML and I'm supposed to write a blog post explaining how I can use it in MY research? By next week?! What the H-E-Double-Hockey-Sticks am I gonna do?! GAWD! Wait a sec... what was that site the professor said to check out? A Gentle Introductions Resource where we can share helpful material with each other or something? Man I hope somebody knows more about this crap than me!”]

Judy is in the same class Jack is in. Unlike Jack, she minored in Computer Science during her years as an undergrad. Not only that, but she actually used XML syntax to markup a Shakespeare play for her undergraduate thesis. She was in Jack's position for quite a while back then (fearful, utterly confused) until she found a paper by a Professor at CBSU (Comic Book State University) that was far more clear and much less dense than most of the material she'd seen...

[transcribed from comic illustration of “Judy’s” speech bubble: “Like, WOW! XML, never thought I’d see this stuff again. Maybe it is useful after all, but that reading was confusing. Hmmmm, I wonder if somebody in my class might benefit from that CBSU Introduction to XML I read last year. I think I’ll link to it...”]

Judy logs into the classes private Gentle Introduction Resource. It looks very similar to some of her favorite social networks and news feeds, except only members of her class are allowed membership. Several other people had posted some XML resources, but Judy is sure they can't possibly be as helpful as hers!

[transcribed from comic illustration of “Jack’s” speech bubble: “Whoa! This looks a lot like the social networking sites I use every day! And hey, three different posts to XML introductory material! There’s my pal Judy. She seems like she’d link to some good stuff. Oh wow, and it’s by a Prof from CBSU! Those guys in Pen & Ink Valley are great with digital concepts! This first sentence is already more readable.”]

And Jack wasn’t the only student in class who benefitted from Judy's submission
to the Gentle Introduction Resource. Several other students also felt that the complicated homework material became far clearer after reading the Gentle Introduction suggested by Judy.

This case study attempts to briefly convey how The GIR can be put to use in the classroom. It provides an anecdote illustrating one student struggling with a technical topic (eXtensible Markup Language or XML) and a second student with prior experience with the topic sharing a relevant Gentle Introduction on their classroom’s GIR instance. The illustrations and infusion of colloquial dialog allows those that access the site to immediately think about the “gentle” or non-threatening nature of the resource’s approach: fun and familiarity in the face of difficult or foreign material.

For a more in-depth explanation of this project's initial goals, including cases for the use of Gentle Introductions in Humanities research, open-source and open-access infrastructure, and coding pedagogy, please see A Proposal for a Gentle Introduction Resource.14

A Blank Canvas or Learning by Doing

Because I have experience in developing static web pages, when I began this project in December 2012, I thought that I could build a useful implementation of pedagogical resources for Digital Humanists using only HTML (HyperText Markup Language) and CSS (Cascading Style Sheets). I initially concerned myself with the

creation of a ludic pathway toward self-collected Gentle Introductions. When I began to search for tools that would allow me to implement a registration/membership system on the site, I realized I had two options: cross my fingers and hope I could find an easy-to-implement, open-source package that I could inject into a vanilla HTML site; or tighten my belt and learn something new. The first option seemed to be a dead end, or if not a dead end, an extremely risky option void of scholarship but oozing with labor. Because I had begun experimenting with Python libraries and had been extensively working with the command line interface via a course in the R Programming Language, I felt I had enough time to dedicate myself to learning how to utilize a MVC (Model-View-Controller) framework. An MVC framework is a design pattern that can be utilized with several Object Oriented programming languages and has become popular because it “provide[s] the illusion of a direct connection from the end user brain to the computer ‘brain’—its memory and processor.” Because I had some hobbyist experience with the Python language, the books I cracked first were those specifically dedicated to implementing Python on the web and reviewing the Python language in general.

Zed Shaw’s *How to Learn Python the Hard Way* was a very useful collection of exercises that taught practical Python through repetition. I found this to be an excellent pedagogical tool. Shaw’s writing seemed very boxed in, however; rarely did he make

---

17 Lev Manovich, *IDS 81650: Big Data, Visualization, and Digital Humanities*, CUNY Graduate Center, Spring 2012.
any references to competency that extended outside of his strict rubric.\textsuperscript{21} That said, I still appreciated Shaw's approach over the popular “virtual terminal” code instruction sites like Codeacademy.\textsuperscript{22} Such dynamic Flash-based sites virtualize all aspects of coding to the point where the logic of a language like Python can be considered, but practical use is in many ways illusory. Creating truly functional code demands one understand how to install a proper version of the computer language, write the code in a text editor, save the code to an appropriate directory, and run the code on an actual machine—Shaw addressed all such issues in his writing. After working through his tutorial, I had become comfortable enough to write a basic text-adventure style game.\textsuperscript{24} This game could be run in a command-line terminal with relative ease, but I was unable to translate terminal-based hypertext programming into something readily engaged with on the web.

I began searching the internet for ways Python code could be utilized in building/making web based applications. It was during this period that I discovered a video tutorial on Django,\textsuperscript{25} a popular MVC framework that used Python to add more complex components to a static webpage.\textsuperscript{26} I attempted to work through the tutorial, but found it to be beyond my comfort zone. In the video’s provided comment section I praised the author’s delivery and asked if he had produced anything that might serve as

\textsuperscript{21} Ibid.
\textsuperscript{24} Anderson Evans, “Seinfeld Quest,” July, 2013, Sample on Appendix 2 page ii-iii.
\textsuperscript{25} Michael Herman, “Setting up a Django Project,” June 25, 2013, http://www.youtube.com/watch?v=D0MoGRZRtcA.
a prerequisite. He suggested I read his book *Real Python*, which I promptly did.

Aimed at Python novices, the book deconstructed multiple frameworks, most of them considered “microframeworks” or frameworks that had a limited range of things that they were capable of compared with more robust frameworks like Django. Chief among the microframeworks were Flask and PythonAnywhere. These microframeworks had a modality reminiscent of risky implementations I was trying to avoid: packaged software that had a very limited community as the distributors attempted to satisfy at best the needs of a niche audience, at worst the lowest common denominator. Experimentation with these frameworks was useful, but ultimately I tried to speed through them and get to the author's section on Django. I was able to deploy a simple test site with limited complexity to my local server, but when trying to deploy the site to my personal online server, I had no luck.

At this point in my research I had become incredibly frustrated and overwhelmed. A colleague suggested I take a look at some basic writings on the Ruby language and its popular MVC framework Rails. Purely out of desperation, I began to query these terms, and much to my surprise transitioning to this different computer language was not as absurd a suggestion as I had initially believed it to be. The numerous programmers and designers that had dedicated themselves to the Ruby language and the Rails framework far outweighed the amount of attention the Python community was giving purely to rapid web application development. Because I had spent months trying

---

27 Michael Herman, *Real Python*, Self Published, August 1, 2013.
to master Django, the Rails framework was immediately familiar; the two frameworks shared much of the same functionality. At the core of the functionality was the manipulation of a database. The database is what separates singular users from one another so that they can communicate data within an enclosed network. Frameworks like Django and Rails make the difficulties of database administration far more digestible, allowing the architect to worry about a more palatable language like Ruby or Python for his or her projects instead of necessitating very high-level understanding of relational database management systems and Structured Query Language (SQL).32

After having moved through all the aforementioned steps, I had equipped myself with a toolkit for implementing the kind of infrastructure I desired: I could now build and maintain an online space that would take advantage of a database of users within a framework that allowed for community-based improvements and additions. At this point my initial expectations for what a site that was capable of realizing my desired variables seemed malformed. Before the intense exploration into higher-level web development, it was quite hard to visualize what the site could look like, much less how the user might interact with it. I had initially imagined it would be a helpful application at one basic level, that is to say, I believed that it would be most helpful as a central source of gentle introductions that could be moderated. Once I had moved into the realm of understanding the foundation I was building upon, my mind changed. Even if I set out to build constraints into a single deliverable, it could be possible to find myself at the mercy of the users who would basically push functionality toward whatever the evolved consensus demanded. This now looked more to me like simply a rehashing of what had

led to the problems I was addressing in my initial proposal in relation to Bamboo Dirt Wiki: an exponential growth of users that could undermine my attempt to add Ruby packages focused on content filtration long after pushing the production site live.33

It seemed more suitable to construct a generic social data-stream that took its aesthetic cues from corporate social network infrastructure, while being explicitly framed to espouse its use in individual classrooms as a singular instance. This would solve the social-media researcher's dilemma: rather than trying to wrangle infinite data with infinite users on a single page (as implemented by Facebook and Twitter), one could have that infinite data mediated by an individual classroom and controlled by themes presented in the live classroom sessions. Such a goal is attempted as a response to the concept of "information overload," which NYU Professor Clay Shirky points out is more appropriately considered as "filter-failure." Shirky has brought this point up in many instances, such as in the Columbia Journalism Review, where he explains that as the cost of publishing goes down the need for proper filters goes up. Shirky explains this is not a new phenomenon, likening our current internet-based feelings of information overload with the information overload suffered in the world of literature before the populous had adapted to the organization of genres.34 With a social network like Twitter, a popular social network platform The GIR is designed to emulate in many ways, the perpetual growth of content in the data stream has the capacity to overwhelm the longer a user is engaging with the system. Even with the use of filtration systems

like dynamic hashtags, built to support filtration of specific topics, the infinite capacity of the network makes personally gratifying filtration more and more difficult. By constraining such an infrastructure to a specific group (in the case of The GIR, a classroom of students), an organic constraint mechanism can serve to alleviate the strain of data mining for the most appropriate material the ever-growing web has to offer. By default The GIR will allow any user with a valid e-mail address to register for the implemented GIR instance, however an administrator can easily change this restraint in the code by allowing only users with a specific e-mail suffix the ability to log in. Such a technique can be taken further by only allowing only specific e-mail addresses access to the network. This allows the administrator control over the application, allowing for as small or as large a group as he or she wishes to involve in The GIR instance.

The aforementioned construction of The GIR would not have been possible without experts willing to share open source code in open repositories like GitHub.\textsuperscript{35} One such expert, Michael Hartl, has written extensive tutorials on Ruby on Rails. Hartl is a very popular voice in the Ruby on Rails community, and his definitive \textit{Ruby on Rails Tutorial} that has been published many times over.\textsuperscript{36} This resource has been released in both free and commercial print versions. The tutorial included a “sample app” to be built from the ground up, which proved to be an excellent foundation for the application I had in mind. The GIR would not have been possible for me to produce if instructors like Hartl were not willing to share their expertise without direct compensation. Any and all


“forks” (changes) I have made to Hartl’s foundational code, I too am sharing. My hope is that I have added another level of open scholarship that finds an audience coming from a different background than those that are apt to discover Hartl’s texts.

My initial effort to utilize an alpha class of users fell short of my expectations. I reached out to an Intro to DH class of approximately thirty students in person with a handout providing a link to The GIR landing page along with a brief explanation of purpose; I also posted a digital copy of this handout on a Digital Humanities group forum on the CUNY AC. Several students created a membership account on the site, but only three of them opted to use the site for posting introductions. While some of this failure could be chalked up to the fact that my request for their engagement was very broad and of no direct consequence to their studies, I felt that there was something missing from my initial implementation. In a meeting with my thesis adviser, he critiqued my site as having some major usability problems. While I had spent much of my time concerned with building a working fork of a generic Rails-based Twitter clone, I had neglected the aspects of my web application that did not necessitate the technical expertise I had been obsessing over. My advisor made it clear that the site’s landing page needed far more work. Instead of expecting students to navigate around the collection of apps pages until they found a dense explanation of the project buried in an “About” section, I should have paid some attention grabbing bullet points and creative illustrations to give a general idea of the site’s purpose as soon as a user opened the home screen. With these changes made, the GIR now provides a more welcoming introduction to new users.
Evaluating Initialization of The Gentle Introduction Resource

The final implementation of The GIR still requires several caveats in terms of actually implementing the resource in the classroom. While I have been able to provide all the necessary ingredients for the application to function, a significant amount of documentation is still needed that can provide instructors with proper step-by-step instructions for deploying the site freely. I believe a strong case for why The GIR can be a useful supplement to the digital humanities classroom has now been appropriately communicated through the home page and FAQ sections of the application. The proof-of-concept demo has also had all of the basic kinks worked out, and while it is in need of some improvements, it is functional and I have cleared up any bugs that have been reported to me by alpha testers.

There are four main additions to the working implementation of The GIR that would improve the resource in significant ways; they are listed here in order of importance:

1. Hyperlinks - Users of the application are still required to cut and paste shared links from the application into the URL box of their web browsers. This needs to be remedied so that anything written as a hyperlink takes the user directly to the location without time spent cutting and pasting the links.

2. Hashtags - It would be helpful if any topic recorded with a hashtag (#) next to it would display to the user all other posts in The GIR with the shared hashtag topic.
3. Organized Topics Page – It would be useful to have a page that can be accessed from the navigation bar that collects all recorded links in an organized fashion in a hierarchy based on hashtagged topics (See supplementals for illustrated explanation). This would require more expertise than I currently have as a lone researcher working on all aspects of the application.

4. Star Rating System - Using what is known as a Ruby “gem” (a packaged set of commands that can be embedded in a Rails page to provide special functionality), I would like to implement a j-query star rating system next to each user post. This way, when researchers read posts suggested by their peers, they can anonymously rate the posts as helpful or not helpful. This is something that I might be able to do, but without the help of a more experienced programmer I am wary of implementing it now for fear of breaking the application.

**Moving Forward**

I hope to move the project toward completion, but this will depend on outside interest and possible funding. I have released the project with an Open-Source MIT License that allows any interested party complete access to the code in such a way that it can be forked, altered, and changed by any user. If I am not able to move forward with improvements myself, the project is accessible to any interested researcher or group of researchers that has the time and the funds. I am also experimenting with a Bitcoin donation QR code. If enough donations are transferred I will publicize this fact and put the donations back into the project.
Brief Aside on Long-Term Impact

While I have no plans to incorporate this project into a broader research topic, I think it would be a great platform for even two or three researchers working in tandem to share resources. In and of itself, the application can serve many uses, even on the meta level where such a project can be very helpful in assisting students interested in learning how they might approach pragmatic use of a programming language. By highlighting the project's transparent and accessible open-source code with as much emphasis as the functional final product, the academic community has the opportunity to experiment with, or to analyze, any and all portions they find to be most personally beneficial. My forking of Michael Hartl’s Ruby on Rails tutorial was essential to my ability to build The GIR in the first place. In time, when the interface becomes dated, its total foundation can be reassessed and altered as any researcher, experimenter, or curious person might see fit.

Reforming Digital Pedagogy

As far as future distribution, I hope that this DH thesis will be helpful to future students in the CUNY MALS Digital Humanities track. By providing a significant amount of documentation that relates to the process of exploring a computational field, I hope that others coming from traditional humanities backgrounds might find a springboard into analyzing computational processes from whatever vantage point makes them
comfortable. My hope is that this project removes some of the stigma from working “under the hood” with digital tools.

There is far more tweaking and customizing to be done to perfect the vision I have for The GIR, but in this weakness is a foundational strength of this web application. My purpose for distributing this project under an Open Source license rests on a belief that people should have complete access to the code that makes the site function. It would have been possible for me to experiment more deeply with some of the customizations I mentioned in a previous section of this piece, but such an action had the potential to weaken my final deliverable. Many of the packages that purported to add the functionality I wanted were sparse in their instructional documentation. Because I am still a novice when it comes to object oriented programming, these experimental additions to The GIR had the potential to add bugs to my code that I did not wish to initiate given my limited remaining time until completion.

The application is presently being hosted on a free Heroku server\textsuperscript{37} and in its more raw form on GitHub. These services allow anyone with a computer and access to the internet the ability to deploy a high-level web application without having to worry about spending money. The live site can currently be accessed at http://boiling-wildwood-9939.herokuapp.com, while the raw data that the site is made up of can be accessed at https://github.com/EliCash82/sample_app.

Appendices

Appendix 1: Illustrations

Figure 1.1: Gentle University ludic interface map
This map was designed with the initial intent of making the foundation of a Gentle Introduction Resource a game-like interface for concept exploration.
Figure 1.2: Model View Controller illustration from http://www.tonymarston.net/php-mysql/model-view-controller-01.png
This illustration presents how the MVC framework is initialized by the application architect and utilized by the user.

Appendix 2: Code Examples

Sample from Seinfeld Quest build:
The following is a sample from a practice “text adventure” game I wrote while experimenting with exercises from Shaw’s book. Run through the terminal, the user’s commands move a character through a narrative with a rough map structure.

print"Seinfeld Quest by Anderson Evans I Version 0.0.1"
print"(c) 2013"
print"
print"##########################################
print "The year is 1998, and you are iconic comedian Jerry Seinfeld."
print "You live in The Upper West Side in a one bedroom apartment."
print "Last night you went on a date with a woman that ate with what"
print "Can only be described as man hands. Her breasts were real"
print "and amazing, but this simply did not make up for those hands."
print "\n"
print "Your phone rings. It could be her... but then again it could"
print "also be your friend George Costanza, who you sent to the comic"
book store less than an hour ago to check on an issue of Superman
That may or may not exist. Should you answer?
Decisions, decisions.

action = raw_input(">>> ")

if action == "answer phone":
    print 'You pick up the phone, "George?!!" you say "No Jerry, its'
    print 'me, Jillian. I left something at your place, mind if I'
    print 'stop by?" Ugh, you think, this cant be happening "Sure,'
    print 'whatever." You hang up the phone.'
    print "n"
    print "Moments later Jillian arrives. You let her in, she heads"
    print "For the couch, she pulls out A GUN?!!! YES A GUN!"
    print "**BANG BANG**"
    return 'death'

elif action == "don't answer" or action == "dont answer" or action == "don't answer phone":
    print "You decide not to answer the phone, and take a nap instead.".
    print "Thank God for these new earplugs you picked up at the"
    print "pharmacy. Man this person really wants to talk, maybe you"
    print "should answer? Naaaah. You fall asleep with the faint sound"
    print "Of ringing in the background.".
    print "n"
    print "You wake up to the sound of a pounding on your door.".
    print '"Wellll, I guess I should see what this person wants,"'
    print "you think to yourself, you approach the door, and open it.".
    print "It is Jillian (AKA Man-Hands), and OH JESUS! SHE'S HOLDING"
    print "A GUN! *BANG BANG***'
    return 'death'

elif action == "leave" or action == "leave apartment":
    print "You don't want to answer the phone, but man, the ringing"
    print "is annoying. You decide to head next door to Kramer's"
    print "place. Why not? He's constantly invading your space.".
    return 'kramersApt'

else:
    print "What does that even mean?"
    return 'jerrysApt'
Samples of Application Code (4 page samples)

1. **home.html.erb**

   This page makes it possible for the user that is not signed in to the application to access all the site’s documentation and introductory material. If the user has signed up, creating his or her space in the database, and has signed in, this page redirects to a view of the working web application wherein the user sees all of his or her posts.

   ```erb
   <% if signed_in? %>
     <div class="row">
       <aside class="span4">
         <section>
           <%= render 'shared/user_info' %>
         </section>
         <section>
           <%= render 'shared/stats' %>
         </section>
         <section>
           <%= render 'shared/micropost_form' %>
         </section>
       </aside>
       <div class="span8">
         <h3>Micropost Feed</h3>
         <%= render 'shared/feed' %>
       </div>
     </div>
   <% else %>
     <div class="center hero-unit">
       <h1>The Gentle Introduction Resource</h1>
       <h2>The only thing to fear is a bad explanation.</h2>
       <p>
         <%= image_tag("gentle_fred2.png", alt: "Gentle Logo") %>
       </p>
       • Let students crowdsourse the research that works for them!
       • Use a familiar UI to drive engagement with important information!
       • Introduce coworkers and staff to difficult material!
     </div>
   <% end %>
   ```
The Gentle Introduction Resource is a private online social datastream designed for classrooms. It allows students to aggregate instructional material that is directly relevant to classroom topics.

Want to try it out? <%= link_to "Click here!", '#foobar', class: "btn btn-large btn-danger" %>

Problem #1: DIGITAL IS DIFFICULT

Some of us are in denial about it, but its true. Just because we use a computer every day does not mean we are technical wizards. Even technical wizards have a benchmark they haven't passed.

Problem #2: THE FIELD IS INFINITE

Search engines and social networks both pose a similar problem: When there is no constraint students and researchers get lost in the massive onslaught of content, voices, and hyperlinks.

Problem #3: IT'S HARD TO TRANSLATE PHYSICAL SPACE INTO DIGITAL SPACE

Many digital translations of the physical classroom get crowded quickly and attempt to facilitate far too many users in one small space. When the digital classroom becomes the digital school or university it can be hard to find your way around.

How about a little more detail?

The web is a great place for research, but it does have its own brand of drawbacks. Back when one did research in your typical brick-and-mortar library, the problem was a LACK of access to definitive material. With the constantly evolving internet of today researchers have the opposite problem. There is what seems like an infinite amount of information about almost any given subject littering the web. How can we find the research we want? How do we KNOW it actually is the research we want? Is there a way classrooms can embark on this search together, constraining the infinite in a way that suits any one classroom appropriately? The answer is YES! Thanks to The Gentle Introductions Resource that is!

WAIT! Hold it! Hold it right there...
A "Gentle Introduction" is not strictly defined, but the moniker is typically used to describe introductions to difficult material simplified to a point in which researchers new to a given subject might be able to understand core conceptual features. Still confused? Here are a few Gentle Introductions our alpha users have submitted:

- [A Gentle Introduction to Version Control](http://chronicle.com/blogs/profhacker/a-gentle-introduction-to-version-control)
- [An Even Gentler Introduction to XML](http://dh.obdurodon.org/what-is-xml.xhtml)
- [An Introduction to Photoshop](http://guides.lib.unc.edu/photoshop)

The Gentle Introduction Resource is a proof-of-concept for a Digital Humanities based resource foundationally interested in giving instructors a tool allowing students to crowdsource materials on any subject, concept, or theme. The "Gentle Introductions" moniker was chosen for this prototype because this title illustrates a specific need in the growing field of Digital Humanities research.

The G.I.R. is a Rails based web app that hopes to collect a specific collection of crowdsourced academic resources. This app is maintained by Anderson Evans as the core of his thesis for the MALS degree in Digital Humanities at CUNY Graduate Center.

How can you use The Gentle Introduction Resource in your class?

Here is a story to illustrate how this web app can be utilized in a classroom setting:
Jack is a student seeking a Masters Degree in History at Everyman's University. Jack comes from a traditional Humanities background, but is taking a course in Digital Humanities. He hopes to learn how he can inject digital, computational, and web based components into his research. Jack has owned a computer all of his life, so he figured it shouldn't be too hard for him. At first it wasn't, but about halfway into the semester his homework began to seem a little overly challenging...<br/>

Judy is in the same class Jack is in. Unlike Jack, she minored in Computer Science during her years as an undergrad. Not only that, but she actually used XML syntax to markup a Shakespeare play for her undergraduate thesis. She was in Jack's position for quite a while back then (fearful, utterly confused) until she found a paper by a Professor at CBSU (Comic Book State University) that was far more clear and much less dense than most of the material she'd seen...

Judy logs into the classes private Gentle Introduction Resource. It looks very similar to some of her favorite social networks and news feeds, except only members of her class are allowed membership. Several other people had posted some XML resources, but Judy is sure they can't possibly be as helpful as hers!

And Jack wasn't the only student in class that benefitted from Judy's submission to the Gentle Introduction Resource. Several other students also felt that the complicated homework material became far more clear after reading the Gentle Introduction suggested by Judy.
The next day in class Jack, Judy, and the rest of their peers

The Gentle Introduction resource is still in its alpha stages, but has more innovations coming if support is found. See the FAQ section for a list of additions that are currently under construction.

Want to help the project reach the next phase of development? Please donate Bitcoin!

Every little BTC helps!

Want to help the project reach the next phase of development? Please donate Bitcoin!
This site was designed with <strong>Ruby on Rails</strong>.<p></p>

This site is hosted on the live web using the PaaS <strong>Heroku</strong>.

Full code hosted and accessible on <strong>GitHub</strong>.

PostgreSQL is the app's underlying database.

This site is the foundation for Anderson Evans' MALS thesis at <strong>CUNY Graduate Center</strong>.

---

2. routes.rb

The routes file tells the applications which parts of the application are accessible to users and to non-users. It gives rules to the accessible portions of the application and pathways for those accessing the site to move through. Areas that are commented out with hashtags were autogenerated and can be initialized by the architect if specific general rules are needed.

SampleApp::Application.routes.draw do
  resources :users do
    member do
      get :following, :followers
  end

end
resources :sessions, only: [:new, :create, :destroy]
resources :microposts, only: [:create, :destroy]
resources :relationships, only: [:create, :destroy]

root to: 'static_pages#home'

match '/signup', to: 'users#new'
match '/signin', to: 'sessions#new'
match '/signout', to: 'sessions#destroy', via: :delete
match '/faq', to: 'static_pages#faq'
match '/license', to: 'static_pages#license'
match '/contact', to: 'static_pages#contact'
match '/downloads', to: 'static_pages#downloads'

# The priority is based upon order of creation:
# first created -> highest priority.

# Sample of regular route:
# match 'products/:id' => 'catalog#view'
# Keep in mind you can assign values other than :controller and :action

# Sample of named route:
# match 'products/:id/purchase' => 'catalog#purchase', :as => :purchase
# This route can be invoked with purchase_url(:id => product.id)

# Sample resource route (maps HTTP verbs to controller actions automatically):
# resources :products

# Sample resource route with options:
# resources :products do
#   member do
#     get 'short'
#     post 'toggle'
#   end
# # collection do
#   get 'sold'
# # end
# end
# Sample resource route with sub-resources:
# resources :products do
#   resources :comments, :sales
#   resource :seller
# end

# Sample resource route with more complex sub-resources
# resources :products do
#   resources :comments
#   resources :sales do
#     get 'recent', :on => :collection
#   end
# end

# Sample resource route within a namespace:
# namespace :admin do
#   # Directs /admin/products/* to Admin::ProductsController
#   # (app/controllers/admin/products_controller.rb)
#   resources :products
# end

# You can have the root of your site routed with "root"
# just remember to delete public/index.html.
# root :to => 'welcome#index'

# See how all your routes lay out with "rake routes"

# This is a legacy wild controller route that's not recommended for RESTful applications.
# Note: This route will make all actions in every controller accessible via GET requests.
# match ':controller(/:action(/:id))(/:format)'
end

### 3. Gemfile

The Gemfile is composed by the architect and contains all of the additions to the basic Rails framework that need to be accessed for the application to function properly. Gemfiles can be separated by environments so that the development version might use a different tool than the production version.

source 'https://rubygems.org'

gem 'rails', '3.2.7'
gem 'bootstrap-sass', '2.0.0'
gem 'bcrypt-ruby', '3.0.1'
gem 'faker', '1.0.1'
gem 'will_paginate', '3.0.3'
gem 'bootstrap-will_paginate', '0.0.6'

group :development, :test do
  gem 'sqlite3', '1.3.5'
  gem 'rspec-rails', '2.10.0'
end

gem 'annotate', '2.5.0', group: :development

# Gems used only for assets and not required
# in production environments by default.
group :assets do
  gem 'sass-rails', '3.2.4'
  gem 'coffee-rails', '3.2.2'
  gem 'uglifier', '1.2.3'
end

gem 'jquery-rails', '2.0.1'

group :test do
  gem 'capybara', '1.1.2'
  gem 'factory_girl_rails', '1.4.0'
end

group :production do
  gem 'pg', '0.12.2'
end

4. sessions_helper.rb
The sessions_helper.rb file uses Ruby code to help define what a user is within the application rather than a visitor that has not yet initiated the site's database. It also separates each user from every other user.

module SessionsHelper
  def sign_in(user)
    cookies.permanent[:remember_token] = user.remember_token
    self.current_user = user
  end

  def signed_in?
end
Stack Overflow Question Submission

The following is an example of a Stack Overflow question. Stack Overflow is a site used by programmers from all over the world, and is an invaluable resource, even when working off of a tutorial, or sets of tutorials.
I've been able to jump most hurdles as I move through the Hartl Rails tutorial, but I can't figure out what I'm doing wrong around 10.4. I can get everything to render correctly using this **/static_pages/home.html.erb**

```erb
<% if signed_in? %>
  <div class="row">
    <aside class="span4">
      <section>
        <%= render 'shared/user_info' %>
      </section>
      <section>
        <%= render 'shared/micropost_form' %>
      </section>
    </aside>
  </div>
<% else %>
  <div class="center hero-unit">
    <h1>Welcome to The Gentle Introduction Resource</h1>
    <p>
      This is the home page for the
      web app.
    </p>
    <%= link_to "Sign up now!", signup_path, class: "btn btn-large btn-primary" %>
  </div>
  <br/>
  <%= link_to image_tag("rails.png", alt: "Rails"), "http://rubyonrails.org/" %>
<% end %>
```

But then when I include this code:

```erb
<% end %>
<% if signed_in? %>
  <div class="row">
    <aside class="span4">
      <section>
        <%= render 'shared/user_info' %>
      </section>
      <section>
        <%= render 'shared/micropost_form' %>
      </section>
    </aside>
  </div>
<% else %>
  <div class="center hero-unit">
    <h1>Welcome to The Gentle Introduction Resource</h1>
    <p>
      This is the home page for the
      web app.
    </p>
    <%= link_to "Sign up now!", signup_path, class: "btn btn-large btn-primary" %>
  </div>
  <br/>
  <%= link_to image_tag("rails.png", alt: "Rails"), "http://rubyonrails.org/" %>
<% end %>
```

it breaks.
Full **home.html.erb**:

```erb
<% if signed_in? %>
  <div class="row">
    <aside class="span4">
      <section>
        <%= render 'shared/user_info' %>
      </section>
      <section>
        <%= render 'shared/micropost_form' %>
      </section>
    </aside>
    <div class="span8">
      <h3>Micropost Feed</h3>
      <%= render 'shared/feed' %>
    </div>
  </div>
<% else %>
  <div class="center hero-unit">
    <h1>Welcome to The Gentle Introduction Resource</h1>
    <p>This is the home page for the <a href="http://www.weekendpublisher.com">The Gentle Introduction Resource</a> web app.</p>
    <%= link_to "Sign up now!", signup_path, class: "btn btn-large btn-primary" %>
  </div>
<% end %>
```

Here is my `_feed.html.erb`:

```erb
<% if @feed_items.any? %>
  <ol class="microposts">
    <%= render partial: 'shared/feed_item', collection: @feed_items %>
  </ol>
<% else %>
  <%= will_paginate @feed_items %>
<% end %>
```
And here is my **_feed_item.html.erb**

```html
<li id="<%= feed_item.id %>">
  <%= link_to gravavatar_for(feed_item.user), feed_item.user %>
  <span class="user">
    <%= link_to feed_item.user.name, feed_item.user %>
  </span>
  <span class="content"><%= feed_item.content %></span>
  <span class="timestamp">
    Posted <%= time_ago_in_words(feed_item.created_at) %> ago.
  </span>
  <% if current_user?(feed_item.user) %>
    <%= link_to "delete", feed_item, method: delete, data: { confirm: "You sure?" }, title: feed_item.content %>
  <% end %>
</li>
```

Sorry in advance for my markup, this is my first stack overflow question.

Oh and here is the error I'm getting when I attempt to load my local site:

**SyntaxError in Static_pages#home**

Showing /rails_projects/sample_appOct20_2013/app/views/shared/_feed.html.erb where line #7 raised:

/rails_projects/sample_appOct20_2013/app/views/shared/_feed.html.erb:7: syntax error, unexpected keyword_ensure, expecting $end
Extracted source (around line #7):

```html
4: </ol>
5: <%= will_paginate @feed_items %>
6: <%= end %>
```

Trace of template inclusion: app/views/shared/_feed.html.erb, app/views/static_pages/home.html.erb
Appendix 3: Application Screenshots

Figure 2.1: Landing Page
The landing page is one of the two homepages that might appear when web application is accessed. This appears for first time visitors or members that are signed out of the GIR network. It gives a rundown of why the GIR might be used and presents appropriate links to download all necessary components needed to set up the GIR. It includes several original illustrations.
Figure 2.2: Main Page
This is the page that appears when the “Home” button is pressed on the Navbar. It includes all microposts communicated by members of the network that the user is following. It also has a microform that allows the user to post new microposts to the datastream.
Figure 2.3: Home Page
This is the page that loads once a registered user signs into the application. This is a feed of microposts solely written by the user him/herself. The user can delete any previously communicated micropost.
Figure 2.4: User Page

The user page can be accessed by clicking “Users” in the Navbar. This allows users to decide which members of the network they want to follow. This page also allows the admin to delete any users that do not belong on the network.
Figure 2.6: Downloads Page

This page will host all final pieces of documentation in various formats.
Figure 2.7: Profile Settings Page

This page allows the signed-in user to alter their settings in the network database. It also links to the avatar service, Gravatar.
Figure 2.8: Licensing Page
This page dictates the terms of the MIT License shared by both the current fork of the original code, and the foundational code upon which the GIR is built.
Figure 2.9: FAQ Page

This page addresses frequently asked questions, specifically the original alpha testers of the application.
Bibliography


Chamberlin, Donald. Structured Query Language. 1986.


Herman, Michael. Real Python. Self Published. August 1, 2013.

Herman, Michael. “Setting up a Django Project”. June 25, 2013. http://www.youtube.com/watch?v=D0MoGRZRtcA.


