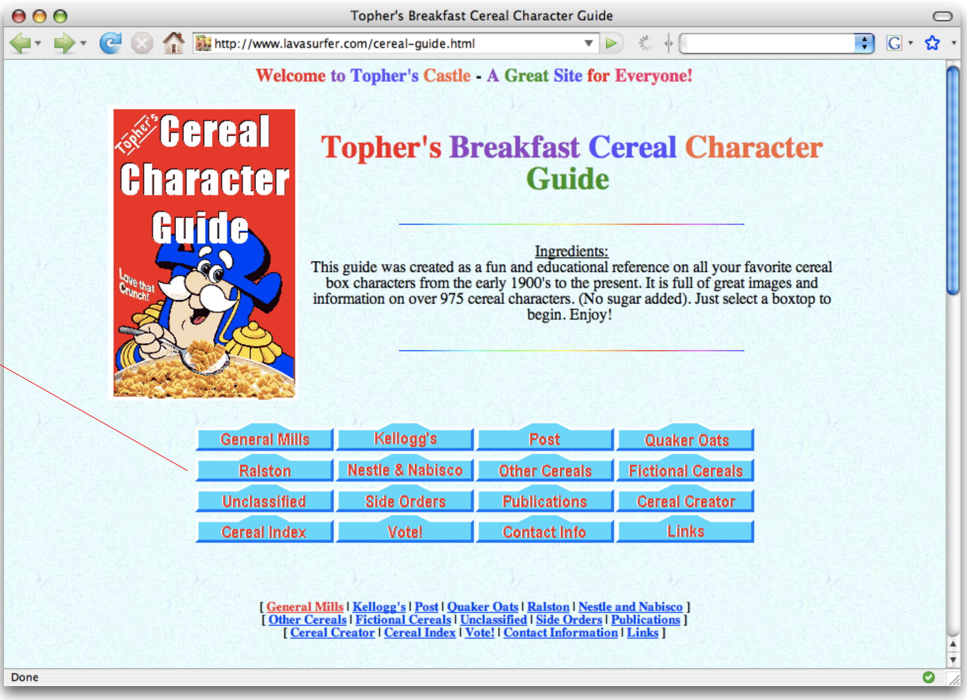
# 1. Introduction

When we encounter data on the Web, most of the time it can only be read on the original page, at the original site, and in its original form. For example, if a few dozen names and addresses are found in a list on a web page, then it is very difficult to plot these addresses all on a map (to visually compare their relative distances to some location) or even just to store their contact information into one's address book. The publisher of that site has decided to present the names and addresses in a list and there is nothing the site's users can do to use that data differently without incurring a great deal of time and effort.

Similarly, it is difficult to publish data to the Web beyond making the data accessible as a spreadsheet or a text file, or encoding the data in hand-written HTML that offers no advanced browsing features or rich visualizations. Consider the home-made site in Figure 1.1 that show breakfast cereal characters. The publisher of that site, Topher, has organized the characters by brand. If Topher decided to let his users browse the characters by year of introduction to market—a reasonable choice, then he would have to completely reorganize his web site. In contrast, the commercial site in the same figure offers several ways of browsing for cameras, and in order to offer more ways, the site can just formulate more queries to its existing database. To catch up with commercial sites, Topher would have to acquire many skills: setting up a database, designing a data schema, designing the user interface, implementing the user interface, implementing the application logic, and testing the resulting three-tier web application on many browsers.

My thesis demonstrates that tools can be built to let casual users—those without programming skills—interact with today's Web in more data-centric ways, to effortlessly retrieve and reuse data from the Web as well as publish data into the Web in a browsable and reusable form. These tools can be built and used today without needing the Web to become more data-centric first by itself.

A home-made web site offers only one way to browse the data, and it would be extremely hard for the publisher to offer other ways, as that requires complete reorganization of the web site.


Topher's Breakfast Cereal Character Guide

http://www.lavasurfer.com/cereal-guide.html

Welcome to Topher's Castle - A Great Site for Everyone!

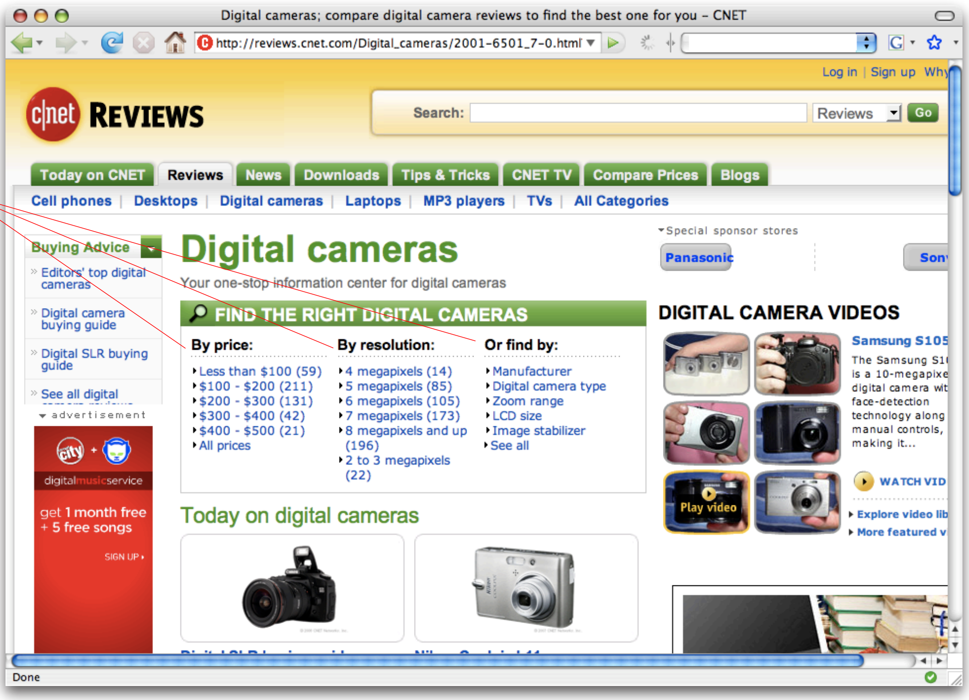
Topher's Breakfast Cereal Character Guide

Ingredients:
This guide was created as a fun and educational reference on all your favorite cereal box characters from the early 1900's to the present. It is full of great images and information on over 975 cereal characters. (No sugar added). Just select a boxtop to begin. Enjoy!

[ General Mills | Kellogg's | Post | Quaker Oats | Ralston | Nestle and Nabisco ]
[ Other Cereals | Fictional Cereals | Unclassified | Side Orders | Publications ]
[ Cereal Creator | Cereal Index | Vote! | Contact Information | Links ]

Done

A commercial web site offers many ways to browse the data, and more ways can be added just by formulating new queries to the database.


Digital cameras; compare digital camera reviews to find the best one for you – CNET

http://reviews.cnet.com/Digital_cameras/2001-6501_7-0.html

Log in | Sign up | Why
cnet REVIEWS
Search: Reviews Go
Today on CNET | Reviews | News | Downloads | Tips & Tricks | CNET TV | Compare Prices | Blogs
Cell phones | Desktops | Digital cameras | Laptops | MP3 players | TVs | All Categories
Buying Advice
Editors' top digital cameras
Digital camera buying guide
Digital SLR buying guide
See all digital
advertisement
digitalmusicservice
get 1 month free + 5 free songs
SIGN UP
Special sponsor stores
Panasonic
Son

Digital cameras

Your one-stop information center for digital cameras

FIND THE RIGHT DIGITAL CAMERAS

| By price: | By resolution: | Or find by: |
|---|---|---|
| Less than $100 (59) | 4 megapixels (14) | Manufacturer |
| $100 - $200 (211) | 5 megapixels (85) | Digital camera type |
| $200 - $300 (131) | 6 megapixels (105) | Zoom range |
| $300 - $400 (42) | 7 megapixels (173) | LCD size |
| $400 - $500 (21) | 8 megapixels and up (196) | Image stabilizer |
| All prices | 2 to 3 megapixels (22) | See all |

DIGITAL CAMERA VIDEOS

Samsung S105
The Samsung S1 is a 10-megapixe digital camera wit face-detection technology along manual controls, making it...

Today on digital cameras

Done

**Figure 1.1.** Using just HTML, a small publisher cannot match the advanced browsing features and rich visualizations offered by commercial and institutional sites, or compete with the flexibility and speed at which such features can be implemented by those sites.

## 1.1 The Case for Data-Centric Interactions

Being able to reuse data on the Web creates new opportunities:

• The same data can show different insights when it is presented differently. Plotting the US presidents on a time line color-coded by their political parties shows the interleaving pattern of Democrats and Republicans coming into power, whereas plotting the presidents on a map by their birth places reveals that all except one were born in eastern United States. Listing the presidents in a table sorted by their surnames highlights two Adams and two Bushes. If a web site provides only one of these presentations and no means to retrieve and reuse its data so that it can be visualized differently, its audience has been denied some of these insights.

• Several sources of data can provide more insights when combined than they can individually. Plotting wars, economic performance, artistic movements, scientific discoveries, etc., alongside the presidents' terms on a time line can reveal each president's and each political party's effects on the country. Plotting cholera cases and water pumps on the same map did bring insight to John Snow, suggesting obvious actions that would stop the cholera epidemic in London in 1854 [71]. As long as it remains hard to combine data from several sources, such insights can easily be missed.

• One important source of data is a user's own information. Seeing several potential houses to buy on a map is useful, but seeing on that same map where one's friends and relatives live and where one's workplace is located can be even more useful in making a house purchasing decision. As often a user's own information is private and cannot be made known to any web site, the data on web sites (which the user is already allowed to see) must leave those sites and come to the user's computer where it can be combined with the user's own information.

These opportunities make the case for getting data on the Web into a form conducive to reuse for casual users' benefits. Once *reusing* data is just as easy as *using* data, there will not be a need to distinguish the two cases. In that future, casual users will always have at hand whatever functionality is appropriate to apply on whatever data encountered wherever on the Web. Such an experience with the Web can be said to be more *data-centric* than *text-centric*.

Not only should casual users be able to easily reuse data from the Web, they should also be able to easily publish data into the Web in ways that satisfy their publishing requirements. For example, if a casual user with no programming experience has gathered all the data about her genealogy and wishes to share that data for her living relatives' enjoyment, she should be able to publish a web site presenting that data. If her family is dispersed geographically, a map would be an appropriate visualization. A timeline can superimpose the lives of her ancestors against a historical backdrop with certain events that she deems important. A thumbnail view shows everyone's portrait together and highlights facial feature resemblance. Grouping or filtering by artistic abilities might reveal which side of her family is more artistic than the other. If many of her kindred have had military careers, browsing and sorting by their military ranks and plotting the military bases where they have served might appeal to her. How the data is displayed and browsed should be left entirely to her—the data owner's—discretion. Just as she can structure and lay out a *text document* however she wishes using HTML, she should also be able to present her *data* however she likes.

But that is easier said than done. Even to support as basic a feature as sorting a few hundred people by name immediately requires database setup, schema design, and server-side programming—skills that a casual user does not possess and would take years to acquire. Whereas it *was* easy to become a first-class citizen of the *early* Web by authoring HTML in a text editor, it is no longer easy on today's Web to possess the same publishing power as large companies who have teams of engineers dedicated to building three-tier web applications.

That it is difficult to publish data into the Web and reuse data from the Web is not surprising. The Web was designed for publishing hypertext—*text*—rather than *data*. Web standards (HTML) were specified for publishing text, and web browsers were built for viewing text. Thus we find the majority of web content to be text documents written in natural human languages, unfavorable to data-centric interactions.

This text-centric Web is showing its limits as the demand for data-centric interactions rises. In response, the Semantic Web project [31, 41] holds out the vision of a future Web wherein most if not all data encountered on the Web is described in a standard data model and data-centric interactions can be easily supported. Getting the Web to such a state is a grand challenge, and the slow adoption of semantic web technologies keeps that vision elusive even after a decade of the effort's existence.

In this thesis, I demonstrate that existing tools designed for experts to publish data to the Web and extract and reuse data from the Web can be adapted for casual users by taking into consideration the casual users' needs and abilities. These tools allow them to interact with today's Web in data-centric ways without having to wait for the coming of the Semantic Web.

## 1.2 Approach

My research approach to providing data-centric interactions on today's Web to casual users—those without programming skills—can be divided into three aspects:

- publishing data into the text-centric Web and providing data-centric features on that data;
- extracting data from the text-centric Web so that missing data-centric features can be applied; and
- integrating data from several sources to gain value that no single source alone can offer.

### 1.2.1 Assumptions and Limitations

To scope the research in these three aspects, I have made two assumptions:

• *Casual users most often deal with small data sets, each containing at most about a thousand records.* That is, when a casual user publishes data to the Web, her data most probably consists of no more than a thousand records. When she extracts data from existing web sites, she only extracts from similarly limited data sets. And when she combines data from several sources, she only combines a few sources and each source contains a small data set to begin with. This size limit does not tremendously reduce the benefits of data-centric interactions. For instance, finding the winner in a race among as few as 20 runners is most efficiently and effortlessly done by sorting them by their timings rather than visually scanning over a list of unordered timings. We can also look to the fact that commercial spreadsheet programs have kept their limit of 65,536 rows for over two decades as evidence that up to some size, scale is not relevant to casual users.

• *Casual users most often deal with simple data made up of records consisting of property/value pairs and binary relationships rather than complex class hierarchies with n-ary relationships.* The popularity of commercial spreadsheet programs is evidence that simple tabular data models can carry the usefulness of data-centric interactions a long way for casual users. For instance, being able to align the "home address" field in one table of records with the "residential address" field in another table of records is already useful, such as for plotting all records together on a single map.

In making these assumptions, I have clearly imposed some limitations on my research results. My research contributions may not immediately apply on large, complex data sets, which I argue are rarely encountered by casual users.

### 1.2.2 Publishing Data

A typical web application consists of three layers: data, presentation, and application logic that connects them. This separation of data from presentation has two advantages.

• First, it allows mixing and matching data with presentation. The same presentation template can be applied to many different data records, making the process of presenting data very efficient. The same data can be shown in different presentations—say, as a list in one case and on a map in another, increasing the richness of the user interface.

• Second, the separation of data from presentation makes it easier to build tools specialized for dealing with either data or presentation rather than both at once. Databases and query languages are designed for manipulating data en-masse while WYSIWYG editors and templating languages are tailored toward specifying visual layouts and styles. It would be much harder to design a piece of software and a standard for manipulating both data and presentation together.

This principle of separating data from presentation has been built into tools and frameworks designed for large publishers—online retailers and institutions—who were the first to run into the need for publishing large data sets and offering advanced features. Built to meet the needs of these large publishers, such as to accommodate complex site structures, to allow for heavily customized looks and feels, and to handle secured online transactions, these technologies are far too complex for casual users to adopt. However, the benefits of separating data from presentation embodied in these technologies are applicable to casual users. Unfortunately, such users have so far only been offered HTML as the generic web publishing technology, and in HTML, data and presentation are mixed together.

To let casual users benefit from the separation of data from presentation, the costs of authoring data, authoring presentation, and connecting them up must be lowered.

The cost of authoring presentation can be lowered in two ways by assuming that the presentation needs of casual users are not so complex:

• First, a variety of common features such as sorting, grouping, searching, filtering, map visualization, timeline visualization, etc. can be provided out-of-the-box so that the user does not need to re-implement them herself. (As her presentation needs get more sophisticated, she can plug in more and more third parties' extensions.)

• Second, customization to the presentation can be specified in an HTML-based syntax right inside the HTML code used to layout the web page. This is so that the publisher can work on every part of the page's presentation inside a single file in a single syntax. (As the publisher's needs to customize the presentation get more sophisticated, the HTML-based syntax will no longer satisfy her.)

The cost of authoring data can be lowered in two ways by assuming that casual users publish only small data sets:

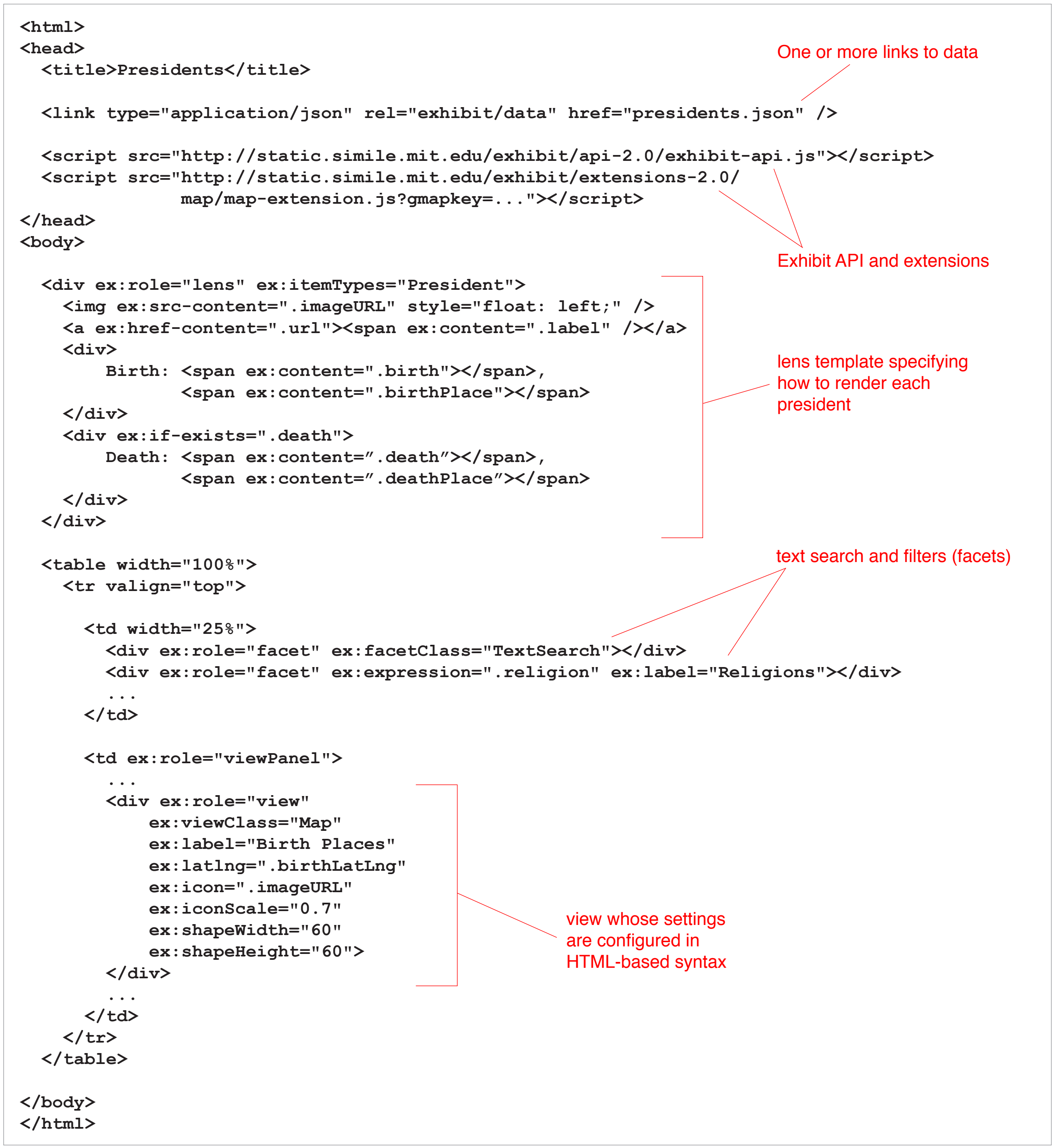
• First, if a publisher is already editing her data in some particular format and some particular editor convenient to her, she should not need to load that data into a database and then maintain the data through the unfamiliar and inconvenient user interface of the database. She should be able to keep managing her data however she likes, and the data only gets loaded into the database when it needs to be rendered. This is realizable if the data is small and loading it into the database is quick. (As the data gets larger, there is a point when the data should already be loaded into a database to ensure a responsive web site.)

• Second, data schemas can be made optional if their benefits do not justify their costs. While schemas are advantageous on large data sets for database optimizations and for managing data at a higher level of abstraction, their benefits on small data sets are much less apparent. (As the data gets larger or more complex, there is a point when schema abstractions benefit both the publisher as well as the publishing engine.)

Finally, all the costs of setting up software (database, web server, and application server) can be eliminated if the software is packaged as a Web API to be included into a web page on-the-fly. Such a Web API can also easily allow for extensions, which accommodate the increasing sophistication in a casual user's publishing needs.

I have built a lightweight publishing framework called Exhibit based on these ideas. To publish an *exhibit*—a web page powered by Exhibit—a publisher uses any text editor to lay out her web page in an HTML file (Figure 1.2) and to enter her data into one or more JSON files (Figure 1.3). The end result is a richly interactive web page such as the one shown in Figure 1.4. If the publisher already has data in some other format, then she can continue to maintain it in that format. Currently, Exhibit can import Bibtex, Excel files, live Google Spreadsheets feeds, RDF/XML, and N3.

When a casual user publishes to the Web using Exhibit, she herself benefits from the separation of data from presentation. Moreover, anyone who visits her web site can easily extract her data for reuse because her data is already in structured form and as publicly accessible as her HTML code. Exhibit kills two birds with one stone: addressing the data publishing need as well as making available in reusable form data that would otherwise be encoded in text-centric HTML.



```
<html>
<head>
  <title>Presidents</title>

  <link type="application/json" rel="exhibit/data" href="presidents.json" />

  <script src="http://static.simile.mit.edu/exhibit/api-2.0/exhibit-api.js"></script>
  <script src="http://static.simile.mit.edu/exhibit/extensions-2.0/
               map/map-extension.js?gmapkey=..."></script>
</head>
<body>

  <div ex:role="lens" ex:itemTypes="President">
    <img ex:src-content=".imageURL" style="float: left;" />
    <a ex:href-content=".url"><span ex:content=".label" /></a>
    <div>
        Birth: <span ex:content=".birth"></span>,
               <span ex:content=".birthPlace"></span>
    </div>
    <div ex:if-exists=".death">
        Death: <span ex:content=”.death”></span>,
               <span ex:content=”.deathPlace”></span>
    </div>
  </div>

  <table width="100%">
    <tr valign="top">

      <td width="25%">
        <div ex:role="facet" ex:facetClass="TextSearch"></div>
        <div ex:role="facet" ex:expression=".religion" ex:label="Religions"></div>
        ...
      </td>

      <td ex:role="viewPanel">
        ...
        <div ex:role="view"
            ex:viewClass="Map"
            ex:label="Birth Places"
            ex:latlng=".birthLatLng"
            ex:icon=".imageURL"
            ex:iconScale="0.7"
            ex:shapeWidth="60"
            ex:shapeHeight="60">
        </div>
        ...
      </td>
    </tr>
  </table>

</body>
</html>
```

One or more links to data
Exhibit API and extensions
lens template specifying how to render each president
text search and filters (facets)
view whose settings are configured in HTML-based syntax

**Figure 1.2.** An HTML-based syntax is used for configuring the user interface of an Exhibit-powered web page.

```
{
  "items": [
    { "label":      "George Washington",
      "type":       "President",
      "birth":      "1732-02-22",
      "birthPlace": "Westmoreland Country, Virginia, USA",
      ...
    },
    // ... more presidents
  ],
  "properties": {
    "birth":        { "valueType": "date" },
    // ... more properties
  },
  "types": {
    "President":    { "pluralLabel": "Presidents" },
    // ... more types
  }
}
```

**Figure 1.3.** Exhibit provides by default a simple JSON syntax for specifying data, but it has an extensible importer architecture for importing other formats including Excel, Bibtex, RDF/XML, N3, and live Google Spreadsheet feeds.

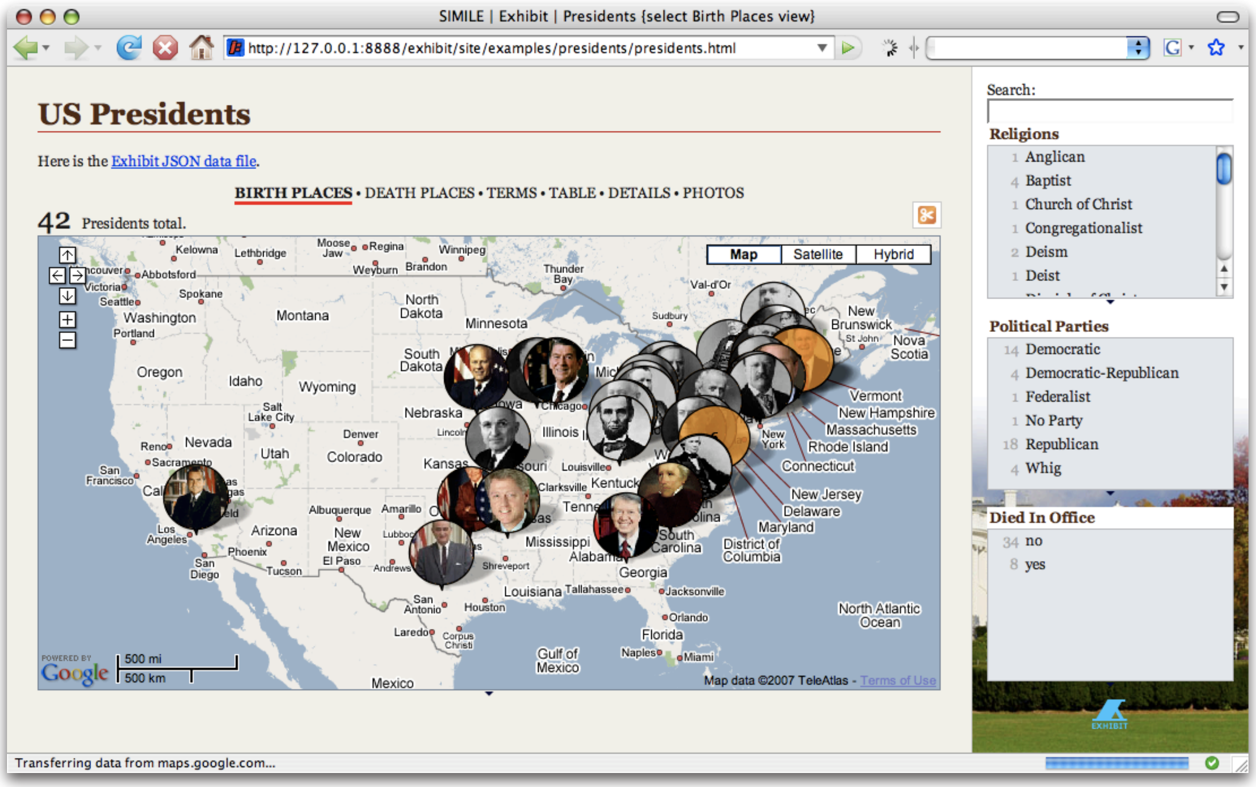

SIMILE | Exhibit | Presidents {select Birth Places view}
http://127.0.0.1:8888/exhibit/site/examples/presidents/presidents.html
US Presidents
Here is the Exhibit JSON data file.
BIRTH PLACES • DEATH PLACES • TERMS • TABLE • DETAILS • PHOTOS
42 Presidents total.
Map
Satellite
Hybrid
Map data ©2007 TeleAtlas - Terms of Use
500 mi
500 km
Search:
Religions
1 Anglican
4 Baptist
1 Church of Christ
1 Congregationalist
2 Deism
1 Deist
Political Parties
14 Democratic
4 Democratic-Republican
1 Federalist
1 No Party
18 Republican
4 Whig
Died In Office
34 no
8 yes
Transferring data from maps.google.com...

**Figure 1.4.** A web page powered by Exhibit provides text search and filtering features as well as rich visualizations such as maps and timelines.

### 1.2.3 Extracting Data

Just like conventional publishing technologies, conventional web data extraction technologies are unsuitable for casual users as they also have been designed for large data sets and complex use cases. If a casual user encounters a few dozen street addresses on a site that offers no map feature, she will not spend much effort to learn these advanced technologies to scrape the data and then build a "Web 2.0 mash-up" site just to map those few dozen addresses.

Web data extraction has been mostly employed by web sites that scrape other web sites for data, aggregate the data, and then offer new services on that data. It makes sense that the scraping be tailored to each source web site to ensure the quality of the result, and that the scraping then be automated to keep the data up-to-date in an efficient manner. As the data is offered again in completely new web sites, there is no need to retain the presentation elements of the original sites.

Although casual users also make use of web data extraction technologies for repurposing data on the Web, their requirements differ from those of large data aggregation sites. The typical task of, say, an online flight reservation site is to accumulate as complete and up-to-date as possible flight information from several airlines, amounting to thousands of records. In contrast, a casual user might just want to pull out street addresses of a few dozen of private schools from their school board web site, plot them on a map, make a one-time printout, and not bother to update the map ever again. That is, the user deals with a lot less data, cares only for a few fields (e.g., street address) rather than all fields, and does not need to keep the extracted data up-to-date since she only needs the printout once. Thus, casual users might put less demand on web data extraction algorithms with respect to scalability, accuracy, and automation.

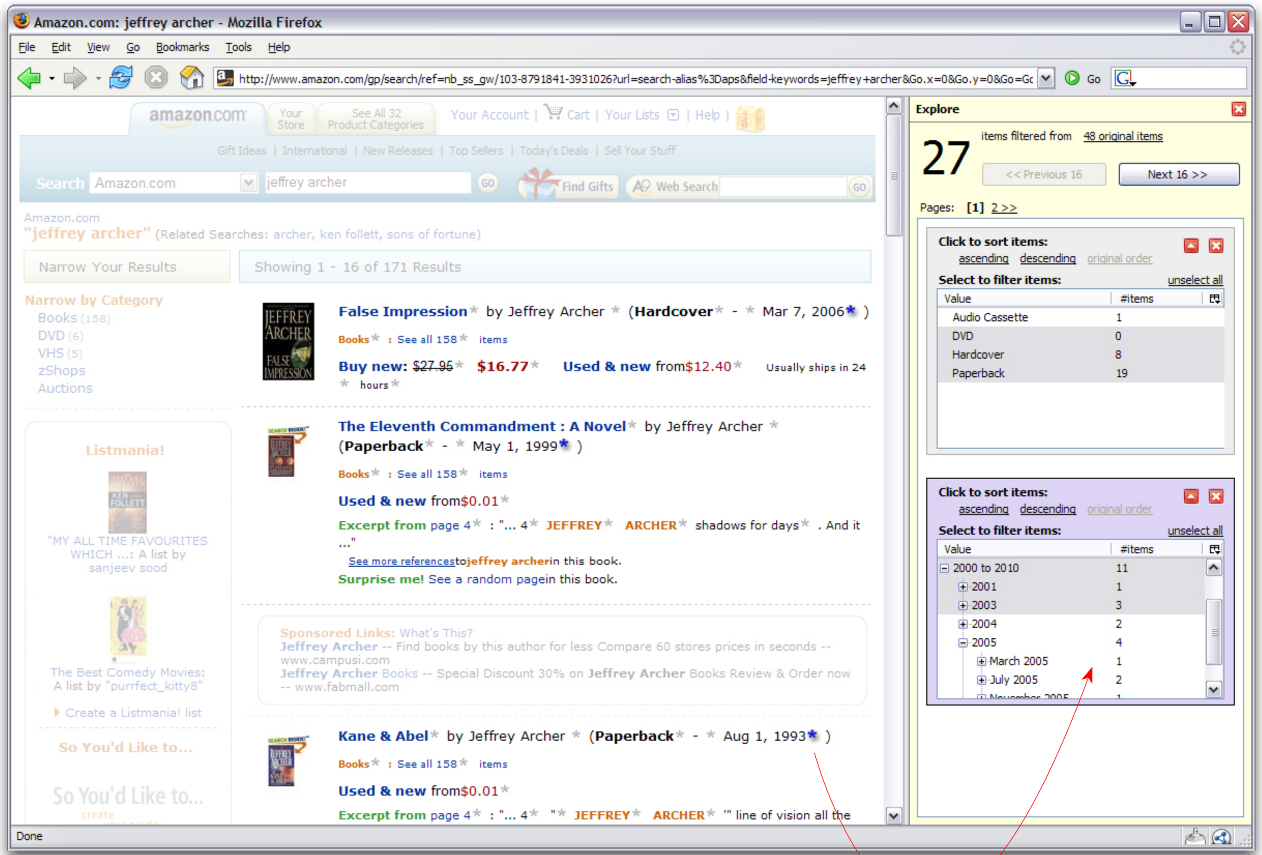
In other aspects, however, casual users might have more demands than large sites. Skills and resources that a data aggregation site has at hand to repurpose scraped data into a new three-tier web application are not available to casual users: no casual user can be expected to set up, design, and maintain a database, to design and implement a rich user interface, and to connect them with application logic, *especially just to plot a few dozen street addresses on a map*. Thus, web data extraction tools built for casual users must offer as complete an experience as possible. Instead of simply returning raw data, they must behave like web applications themselves, offering appropriate presentations and features that casual users need to accomplish their tasks. That is, it is more about adding in missing features rather than taking out data.

To offer a complete web application-like experience over extracted data, all three layers of a typical web application must be automated as much as possible.

• First, the user interface can be "designed" with zero user intervention just by reusing as much as possible the presentation elements already in the original web site. In particular, when a data record is extracted from a part of an original web page, that fragment of the web page is also extracted so that in order to show that data record later on, we can simply show the fragment again. The rest of the web page, which does not contain data to extract, can also be kept as-is so to preserve the original visual context. This is novel since existing web data extraction algorithms throw away original presentation elements (because they are not needed for the purposes of large data aggregation sites).

• Second, in the application logic layer, some set of commonly needed features such as sorting and filtering can be provided out-of-the-box so that there is no need for a casual user to implement them.

• Finally, the extracted data can be loaded immediately into a database without any user intervention if the extracted data can be used as-is without further processing by the other layers. The biggest roadblock to using the extracted data as-is seems to be the need to label fields because field labels are used in conventional user interfaces to provide affordance for features like sorting and filtering (e.g., "sort by publication date"). Unfortunately, field labels can be hard to recover from


Amazon.com: jeffrey archer - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.amazon.com/gp/search/ref=nb_ss_gw/103-8791841-3931026?url=search-alias%3Daps&field-keywords=jeffrey+archer&Go.x=0&Go.y=0&Go=Gc
Go
amazon.com
Your Store
See All 32 Product Categories
Your Account | Cart | Your Lists | Help |
Gift Ideas | International | New Releases | Top Sellers | Today's Deals | Sell Your Stuff
Search Amazon.com jeffrey archer GO
Find Gifts
Web Search GO
Done

Amazon.com
"jeffrey archer" (Related Searches: archer, ken follett, sons of fortune)

Narrow Your Results

Showing 1 - 16 of 171 Results

Narrow by Category
Books (158)
DVD (6)
VHS (5)
zShops
Auctions

False Impression* by Jeffrey Archer * (Hardcover* - * Mar 7, 2006* )
Books* : See all 158* items
Buy new: ~~$27.95~~* $16.77* Used & new from$12.40* Usually ships in 24 * hours*

The Eleventh Commandment : A Novel* by Jeffrey Archer * (Paperback* - * May 1, 1999* )
Books* : See all 158* items
Used & new from$0.01*
Excerpt from page 4* : "... 4* JEFFREY* ARCHER* shadows for days* . And it ..."
See more referencestojeffrey archerin this book.
Surprise me! See a random pagein this book.

Listmania!
"MY ALL TIME FAVOURITES WHICH ...: A list by sanjeev sood
The Best Comedy Movies: A list by "purrfect_kitty8"
Create a Listmania! list

Sponsored Links: What's This?
Jeffrey Archer -- Find books by this author for less Compare 60 stores prices in seconds -- www.campusi.com
Jeffrey Archer Books -- Special Discount 30% on Jeffrey Archer Books Review & Order now -- www.fabmall.com

So You'd Like to...
So You'd Like to...

Kane & Abel* by Jeffrey Archer * (Paperback* - * Aug 1, 1993* )
Books* : See all 158* items
Used & new from$0.01*
Excerpt from page 4* : "... 4* "* JEFFREY* ARCHER* '" line of vision all the

Explore

27 items filtered from 48 original items

<< Previous 16    Next 16 >>

Pages: [1] 2 >>

Click to sort items: ascending descending original order
Select to filter items: unselect all

| Value | #items |
|---|---|
| Audio Cassette | 1 |
| DVD | 0 |
| Hardcover | 8 |
| Paperback | 19 |

Click to sort items: ascending descending original order
Select to filter items: unselect all

| Value | #items |
|---|---|
| 2000 to 2010 | 11 |
| 2001 | 1 |
| 2003 | 3 |
| 2004 | 2 |
| 2005 | 4 |
| March 2005 | 1 |
| July 2005 | 2 |
| November 2005 | 1 |

An asterisk is inserted after each field value.
Clicking on an asterisk adds sorting and filtering controls for that field.

**Figure 1.5.** By keeping presentation elements from the original web site, Sifter can apply direct manipulation techniques on the extracted data fields to support sorting and filtering without requiring the user to label the fields.

web pages. For example, in Figure 1.5, nowhere on the web page says that "Mar 7, 2006" is the "publication date" of "False Impression." However, direct manipulation techniques can be applied to avoid the need for field labels altogether. If the user can interact directly with the text "Mar 7, 2006" to invoke sorting and filtering operations, it does not matter if "Mar 7, 2006" is the publication date of the book or the birth date of the author. It is *just* a date field and it can be sorted and filtered mechanically without any regard for its actual semantics.

I have built these ideas into a browser extension called Sifter (Figure 1.5) that can augment a web site *in-place* with filtering and sorting functionality while requiring from the user as few as two clicks. The added features work inside the site's own pages, preserving the site's presentational style, as if the site itself has implemented the features.

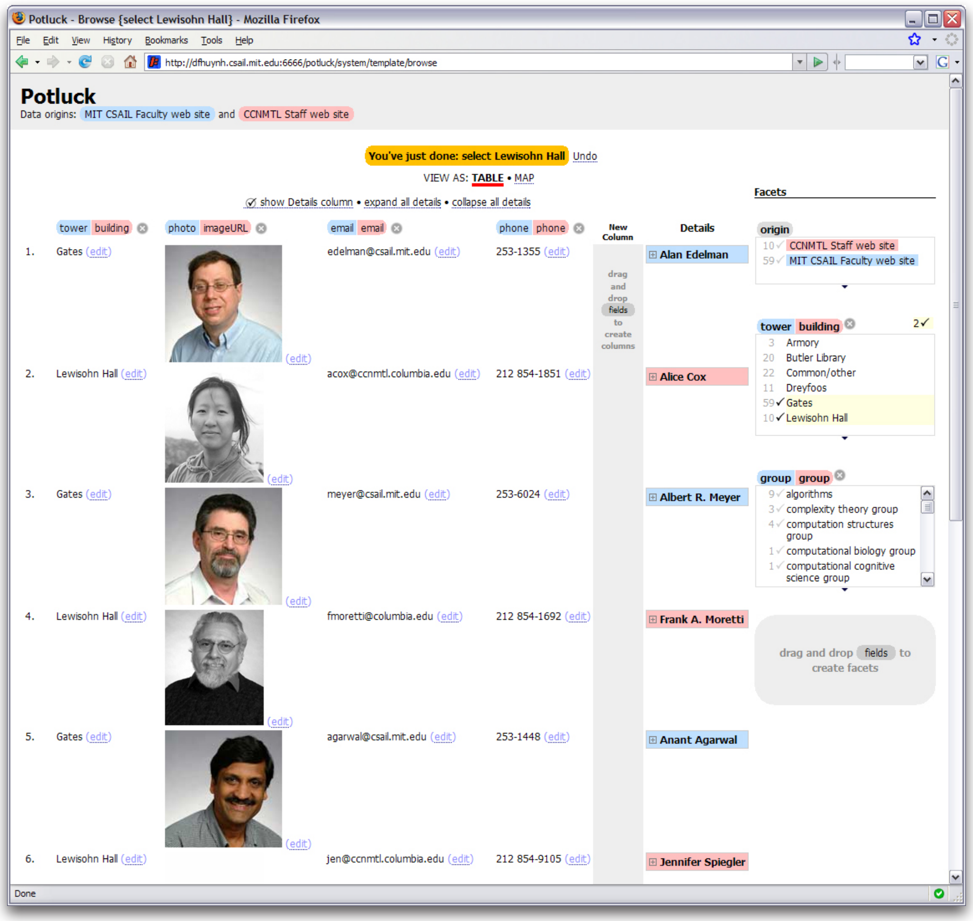
### 1.2.4 Integrating Data

Data integration tools have also been built for a different audience than casual users. For a typical conventional data integration task, such as merging huge databases of two institutional libraries together to serve the combined data through a new web site, several experts and programmers are involved to handle different aspects of the task, including aligning the schemas, cleaning up the data, and then building the web site. Each tool employed is specialized for one stage of the process and designed to handle large, complex data sets. In contrast, a casual user might just want to merge two lists of a few dozens of addresses from two web sites together to plot them on a common map. There is much less data and the data is much simpler. Power tools used by experts are too advanced for casual users.

Whereas a team of experts and programmers can work most efficiently by dividing a data integration task into stages and letting each team member specialize on one stage, each casual user only has herself to deal with her own data integration task. She needs not a set of highly specialized tools like those for experts but a single tool that lets her work on the whole task. Furthermore, unlike experts experienced in dividing a clearly defined task cleanly into stages, the casual user might have to deal with different aspects of the task in an interleaving manner, switching from cleaning up data to constructing presentation and back again, as she gains more and more understanding of what she needs, of what the data is like, and of how to handle the task.

Compared to experts, casual users lack both data modeling skills and programming skills. However, for small, simple data sets, neither skill set may be crucial. First, when the data set is small, schemas—useful for efficiently reasoning about a massive quantity of data in the abstract—are not as useful and can introduce overhead cognitive costs. Instead of having to learn about theoretical concepts like schemas, casual users can manipulate data instances directly. Second, instead of using programming to process data, casual users can just use direct manipulation techniques. For example, two fields can be aligned by dragging and dropping one onto the other.

I have demonstrated these ideas in a tool called Potluck that lets casual users pool together data from several sources, supports drag and drop for merging fields, integrates and extends the faceted browsing paradigm for focusing on subsets of data to align (Figure 1.6), and applies simultaneous editing [58] for cleaning up data syntactically (Figure 1.7). Potluck also lets the user construct rich visualizations of data in-place as the user aligns and cleans up the data. This iterative process of integrating the data while constructing useful visualizations is desirable when the user is unfamiliar with the data at the beginning—a common case—and wishes to get immediate value out of the data without having to spend the overhead of completely and perfectly integrating the data first.


Potluck - Browse {select Lewisohn Hall} - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
http://dfhuynh.csail.mit.edu:6666/potluck/system/template/browse
Done

Potluck

Data origins: MIT CSAIL Faculty web site and CCNMTL Staff web site

You've just done: select Lewisohn Hall Undo

VIEW AS: TABLE • MAP

✓ show Details column • expand all details • collapse all details

| | tower building | photo imageURL | email email | phone phone | New Column | Details |
|---|---|---|---|---|---|---|
| 1. | Gates (edit) | (edit) | edelman@csail.mit.edu (edit) | 253-1355 (edit) | drag and drop fields to create columns | ⊞ Alan Edelman |
| 2. | Lewisohn Hall (edit) | (edit) | acox@ccnmtl.columbia.edu (edit) | 212 854-1851 (edit) | | ⊞ Alice Cox |
| 3. | Gates (edit) | (edit) | meyer@csail.mit.edu (edit) | 253-6024 (edit) | | ⊞ Albert R. Meyer |
| 4. | Lewisohn Hall (edit) | (edit) | fmoretti@columbia.edu (edit) | 212 854-1692 (edit) | | ⊞ Frank A. Moretti |
| 5. | Gates (edit) | (edit) | agarwal@csail.mit.edu (edit) | 253-1448 (edit) | | ⊞ Anant Agarwal |
| 6. | Lewisohn Hall (edit) | | jen@ccnmtl.columbia.edu (edit) | 212 854-9105 (edit) | | ⊞ Jennifer Spiegler |

Facets

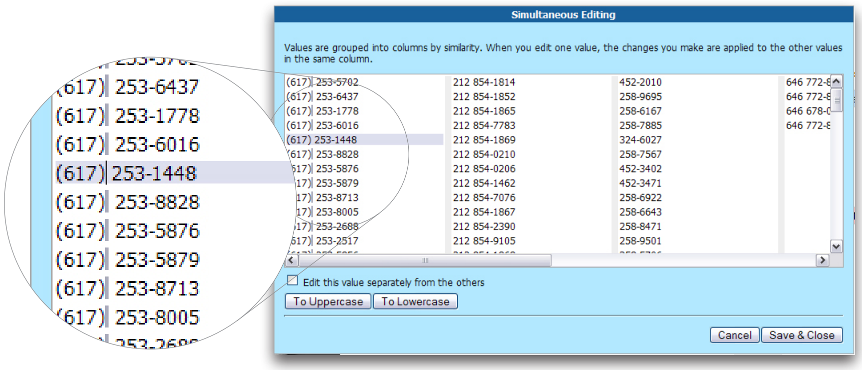
origin
- 10 ✓ CCNMTL Staff web site
- 59 ✓ MIT CSAIL Faculty web site

tower building 2✓
- 3 Armory
- 20 Butler Library
- 22 Common/other
- 11 Dreyfoos
- 59 ✓ Gates
- 10 ✓ Lewisohn Hall

group group
- 9 ✓ algorithms
- 3 ✓ complexity theory group
- 4 ✓ computation structures group
- 1 ✓ computational biology group
- 1 ✓ computational cognitive science group

drag and drop fields to create facets

**Figure 1.6.** Potluck supports drag and drop for merging fields and constructing visualizations. Faceted browsing is available at the same time to help the user isolate subsets of interest or subsets that need further cleaning or alignment.


Simultaneous Editing

Values are grouped into columns by similarity. When you edit one value, the changes you make are applied to the other values in the same column.

| | | | |
|---|---|---|---|
| (617) 253-5702 | 212 854-1814 | 452-2010 | 646 772-8 |
| (617) 253-6437 | 212 854-1852 | 258-9695 | 646 772-8 |
| (617) 253-1778 | 212 854-1865 | 258-6167 | 646 678-0 |
| (617) 253-6016 | 212 854-7783 | 258-7885 | 646 772-8 |
| (617) 253-1448 | 212 854-1869 | 324-6027 | |
| (617) 253-8828 | 212 854-0210 | 258-7567 | |
| (617) 253-5876 | 212 854-0206 | 452-3402 | |
| (617) 253-5879 | 212 854-1462 | 452-3471 | |
| (617) 253-8713 | 212 854-7076 | 258-6922 | |
| (617) 253-8005 | 212 854-1867 | 258-6643 | |
| (617) 253-2688 | 212 854-2390 | 258-8471 | |
| (617) 253-2517 | 212 854-9105 | 258-9501 | |

☐ Edit this value separately from the others

**Figure 1.7.** Potluck applies the simultaneous editing technique to let the user clean up data syntactically in an efficient manner.

## 1.3 Implications for Semantic Web Research

In many ways my thesis aligns with the goal of the Semantic Web project [31, 41]: both aim for a data-centric Web. The philosophies in my research approaches can bring insights to the Semantic Web research community:

• *Before making data meaningful to machines, make it meaningful to people.* In its early days, the Semantic Web research community focused most of its efforts on issues in data modeling, artificial intelligence and agent automation, neglecting to build user interfaces for humans to use semantic web data. Even today, the latest semantic web browsers show raw URIs that make no sense to casual users. In contrast, my research starts out by examining the needs and characteristics of casual users so to build data-centric interfaces that make sense to them.

• *In addition to building a data-centric Web that benefits humanity decades into the future, build data-centric technologies that benefit some individuals right now.* Addressing real needs of people right now motivates the creation of that future data-centric Web and helps identify real needs in that future. If nothing else, this strategy allocates resources to research on data-centric user interfaces, which are severely lacking in Semantic Web research.

• *Before making data reusable, make it useful.* Whereas the Semantic Web researchers encourage people to publish data for the sake of future reuse by other people, my Exhibit framework encourages people to publish data just because publishing data separate from presentation makes the publishing process efficient and the results richly interactive. Immediate, concrete, personal usefulness drives adoption more than prospective, uncertain benefits to someone else.

• *While focusing on data, don't forget presentation.* Exhibit uses rich presentation (map, timeline, etc.) as the motivation for publishing data. Sifter keeps existing presentation elements from original web pages to preserve visual context as well as support direct manipulation on the scraped data, saving casual users from labeling fields. Potluck lets casual users use the presentation they are constructing to guide their data integration process: if the aligned data "looks right" in the user interface, it should be right in the data model.

• *Even while aiming for semantics, take care of syntax.* Exhibit's HTML-based presentation configuration syntax and its ability to import many data formats lower the barrier to its adoption. Sifter's ability to recognize dates and numbers makes its sorting and filtering features more useful. Potluck's simultaneous editing feature lets casual users efficiently fix up data syntactically.

## 1.4 Contributions

My thesis statement is:

> Data-centric interactions with today's Web are useful to and feasible for casual users, and usable tools can be built to support such interactions by gearing for small, simple data sets and common casual needs.

This thesis makes the following contributions:

- First, this thesis combines a simple graph-based data model with simple extensions of the HTML syntax in order to let casual users—those without programming skills—publish web pages offering advanced browsing features and rich visualizations.
- Second, this thesis shows that the cost of using web data extraction technologies can be lowered for casual users by retaining presentation elements from the original web pages. These elements preserve visual context and visual semantics so that direct manipulation techniques can be applied to augment the original pages with more features without requiring users to label fields in the extracted data.
- Third, this thesis proposes that for simple data sets, direct manipulation techniques can be used to let casual users integrate data and construct rich visualizations from it without any need for programming.
- Finally, by letting casual users publish data to the Web in browsable form, extract and reuse data from the Web, and integrate data from several sources without any need for programming, this thesis demonstrates that data-centric interactions can be offered to casual users on today's Web without having to wait for a semantic web.

## 1.5 Thesis Outline

Following this introduction, chapter 2 surveys related work. Then, the main body of this dissertation delves into the three areas identified in the approach:

• Chapter 3 details the experiences of using Exhibit to publish as well as using Exhibit's features on Exhibit-based sites. The chapter also outlines Exhibit's architecture and analyzes its actual use.

• Chapter 4 describes how Sifter delivers web data extraction technologies into the hands of users with no programming experience. Sifter was tested on real web sites and real users, and the results indicated that people could use Sifter to augment existing web sites and then perform sophisticated queries and high-level analyses on sizable data collections.

• Chapter 5 motivates Potluck with a data integration scenario and then explains how Potluck's user interface design meets the needs in that scenario. The chapter also details Potluck's implementation as well as reports the results of a user study that identified usability successes and problems.

Finally, chapter 6 reviews the contributions of this dissertation and outlines future directions for this research.