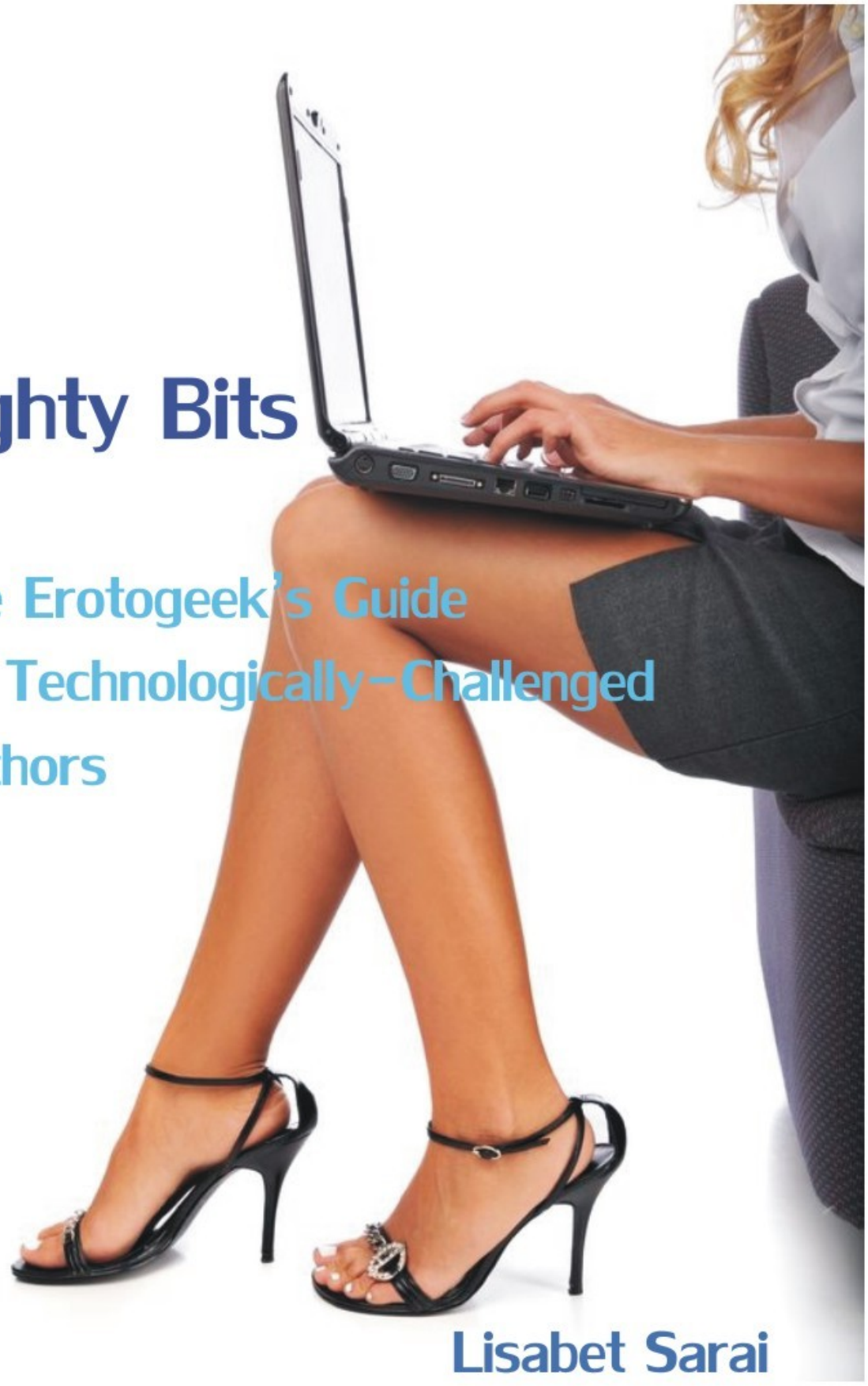
Naughty Bits

The Erotogeek's Guide for Technologically-Challenged Authors



Lisabet Sarai

Naughty Bits:
The Erotogeek's Guide
for Technologically-Challenged Authors

By

Lisabet Sarai
(The Erotogeek)

Naughty Bits: The Erotogeek's Guide for Technologically-Challenged Authors

http://www.lisabetsarai.com

This book is based on a series of articles that appeared at the Erotica Readers & Writers Association website
http://www.erotica-readers.com

Contents

If I Close My Eyes, Will It Go Away? or What the Heck are Bits, Anyway?

Quick! What does a software error have to do with an orgasm?

If you're scratching your head, consider this: for an author of erotica, the former can prevent the latter. Maybe your word processing program is more reliable than mine, but I've certainly experienced *coitus interruptus* when my computer couldn't quite keep it up.

We erotic writers struggle to overcome many obstacles, internal and external: writer's block, recalcitrant characters, families who don't understand that writing ***is*** working, plagiarism and piracy, unscrupulous or stingy publishers, prudes and ideologues who believe we're responsible for the decline of society, and so on. These days, though, not a few of the barriers to success are technological.

If you don't care whether you're published, you can perhaps choose to ignore computers. You can write your grand opus longhand on legal pads, the way I wrote my dissertation. You can transcribe your poems in parchment notebooks in turquoise and purple ink, as I did in my youth, and share manually-produced chapbooks with your small coterie of admirers.

If you want a wider circle of readers, however, you can't afford to be a complete Luddite. You need adequate word processing competence to prepare your manuscript with the margins, fonts, and formatting required by the publisher to whom you plan to submit it. You have to have a website, a blog, or both, and update them regularly, to do even the minimum level of marketing. Email signature lines; Yahoo or Google groups; cover images; buy links; book trailers; Facebook, Twitter, Goodreads and all the other social media; you've got to be at least somewhat tech-savvy just to survive.

I know many authors, though, who are technophobic. When faced with a problem involving computers or software, they feel terrified and totally inadequate, not to mention annoyed. *Why should an author have to worry about mysterious, incomprehensible stuff like anchors and style sheets, image resolution and blog feeds?* they think. *Can't I just write?*

Alas, not anymore. Hence this book. In this nifty little volume, I'm going to make it my business to explain some area of computer technology relevant to authors. My goal is to demystify some of the topics you might find puzzling or obscure, to teach you enough that you will feel competent performing the technical tasks all modern writers need to undertake.

Don't expect in-depth discussions of the fine points of HTML or Blogger. Once you're comfortable with the basics, you can find out more on your own, if you're so inclined. I'm trying to get

you over the hump, so to speak, to gently introduce you to technologies you currently find impenetrable and hopefully to demonstrate that they really make some sense after all.

And what qualifies me to write this book? I have to agree I don't **look** like a geek!

Well, I'm not necessarily the most computer-savvy author out there. I don't Tweet and I'm rarely on Facebook. I don't own a smart phone. In fact, I'm pretty skeptical about technology - partially because I work with it on a daily basis. For more decades than I want to admit, I've been involved designing and writing software. I also have ten years experience teaching these topics to various audiences.

Finally, I've been publishing my erotica since the beginning of the Internet age, long before Google, YouTube, blogs and social networks existed. I edited the galleys for my first novel on paper! I've had a website and domain since 1999 - a website which I now update by hand-coding all my own pages. I've seen the gradual evolution - and digitalization - of publishing, and managed, at least to some extent, to adapt.

Now I want to help you do the same. In the chapters that follow, I'll be talking about HTML (the language of the web) and other aspects of web interaction such as images and animation. I'll also tackle questions like backing up your work and keeping your identity secure, and I'll try to explain why the Web sometimes seems so screwed up.

By the way, I'm not going to talk about computer issues related to self-publishing at all. I really don't have much experience in that realm. I'm sure you can find other valuable resources that cover this topic. (Check out, for example, the excellent series of articles by my friend and colleague William Gaius entitled "Kill Electrons Not Trees", at the Erotica Readers & Writers Association.

So - let's get started. I said that I wanted to teach you the basics. We'll begin with the most fundamental computer concept of all: a **bit**. Just what **is** a bit - naughty or otherwise?

The word "bit" is short for "binary digit". "Binary" is a fancy word for "two". Basically, a binary digit is a number that can have only one of two values: 0 or 1. Ultimately, everything in a computer and everything on the web is just a collection of bits - ones and zeros.

With one bit, we can represent two different alternatives. True or false. Heads or tails. Male or female. Gay or straight. Cunnilingus or fellatio. The person using the bit defines the alternatives. A bit by itself has no meaning - *the person writing the software or creating the data assigns the meaning*. This is a critical point. All a bit provides is a way of signaling one of two different states. Those states could be pretty much anything at all.

If we have more than two alternatives, we need more than one bit. For instance, to represent "Straight", "Gay", or "Bisexual", we'd need two bits.

00 = Straight

01 = Gay

10 = Bisexual

Note that we can also treat these patterns of bits as numbers:

00 = 0

01 = 1

10 = 2

11 = 3

Note also that we have room for one more sexual orientation, which we could map to the pattern 11, if anyone wants to propose one!

With two bits we can represent up to four alternatives. With three bits, there are eight possible combinations. With four bits, there are sixteen. And so on. Each time we add a bit, we double the number of possible meanings we can assign to a bit string of that length.

Eight is a particularly useful number of bits. With eight bits we can represent 256 alternatives. Lots of things in the world fall into that range, enough that groups of eight bits have a special name: a ***byte***. In particular, one byte gives us enough possibilities to assign one pattern of bits to each letter (character) in the Latin alphabet, upper and lower case, plus punctuation and a range of special symbols (like the all important copyright symbol!). There are standards that map each character to a number between 0 and 255 (that is, to patterns of bits).

For example, according a widely used standard mapping called ***ASCII***, the bit pattern 0100 0001 (65) corresponds to a capital **A**. The pattern 0111 1010 (122) corresponds to a lower case **z**. The pattern 0001 0000 (32) corresponds to a space. And so on.

So if I were to look at the bits used to represent this article, I might be able to recognize the individual letters of each word and sentence. I say "might" because this number-to-character mapping is only one way to represent text.

And don't forget that those eight bits could be used to mean something completely different - for example, all the different e-publishers that have opened their virtual doors in the past five years, or the number of rejection letters the typical author receives over the course of her career.

The core concept here is that everything is a computer - the data and the software that

manipulates that data - is composed of strings of bits. Your latest work-in-progress, your head shot, your book trailer, your word processing program - all are composed of bits.

And it's we humans who give those bits meaning. I've demonstrated above that the same pattern of bits can have different meanings. You can also have the same meaning, represented by different patterns of bits. For example, your story has the same meaning whether it is stored in Microsoft Word format or Adobe PDF format, but the underlying bit patterns will be completely different.

A corollary is that software that works with some data (some collection of bits) needs to "know" how to interpret those bits. If the software makes incorrect assumptions, nonsense (or worse!) will result. Just try taking a PDF file of your latest story, renaming it to .DOC and then opening it in MicrosoftWord, to see what I'm talking about.

The digital world is composed of a bewildering array of different standards for arranging bits. When these standards related to how the bits are stored on a hard disk or other persistent medium, they are often called ***file formats***. Don't allow yourself to become confused or discouraged by this variety. After all, it's the meaning that's important.

You may find computer technology mysterious. The real mystery, to me, is how an author can build living, breathing characters and a plausible world out of nothing but his or her imagination. We create meaning and evoke emotion. We challenge and change our readers, leading them to new places and perhaps new understanding. And all of that begins in our minds.

I find that far more marvelous than any silly combination of bits.

HTML 101: Web Basics for Authors

I'm not normally known as the dominant type, but in my role as the Erotogeek, I'm going to make you do things you never believed you'd agree to. Believe me, it's for your own good. And I promise it won't hurt - much! In this chapter, we're going to tackle the basic underpinnings of the World Wide Web.

As I argued in the last chapter, every author has to know something about the Internet. Even if you work with a traditional print publisher, you have to worry about your web site, your blog, your Goodreads, Shelfari, and Facebook pages, your Yahoo groups, and so on, *ad nauseum*. Whether you like it or not, promotion has moved to cyberspace. You have to adapt or die.

So, you need to know something about the mechanics of disseminating information on the Internet. I'm not suggesting that you should be creating your web site by yourself, from scratch. That's a major effort. There are professionals to help with this task, not to mention lots of fancy tools. However, there are times when you need to do a little bit of web coding on your own. Your publisher is running a contest, and you have to add an image or a link to your website. You want to create an email signature. You want to do some quick and dirty modifications to a Yahoo group message or add some fancy formatting to a blog entry or comment.

Every author, in my opinion, needs to know at least of bit of HTML, just to survive. This month, I'll introduce you to some basics of web coding and hopefully, help to demystify the amazing but remarkably simple World Wide Web.

How the Web Works

The Web is much less complicated than you might imagine. Basically, the World Wide Web depends on two types of software programs, plus two languages or ***protocols***.

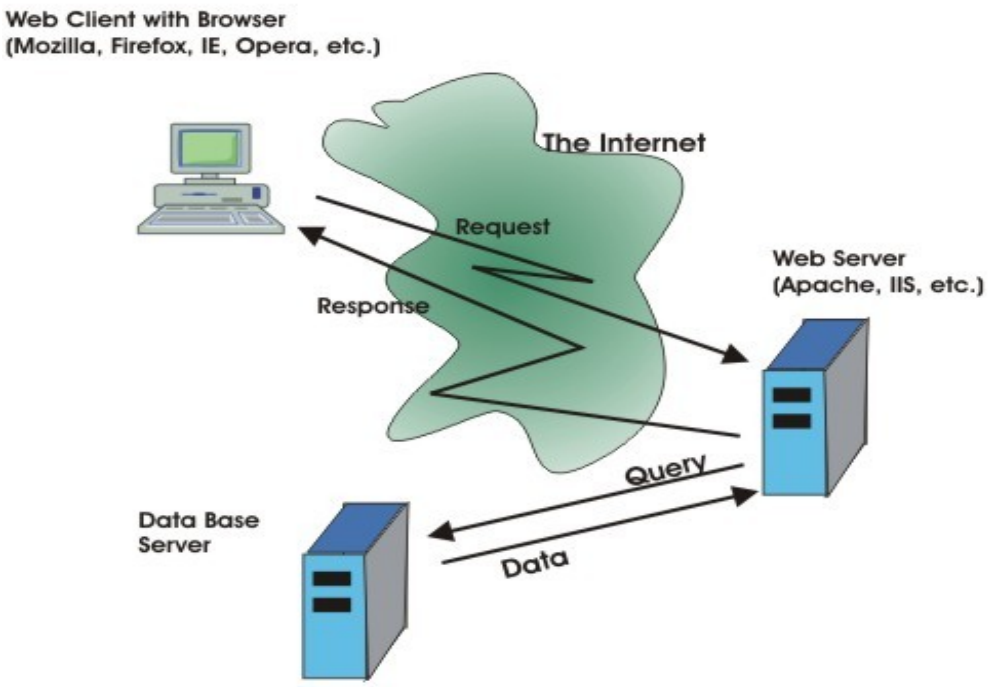
The first kind of software is a ***web server***. In the most straightforward case, a web server runs on the computer where your web page content is located. If you use a web hosting company, the web server runs on that company's computers.

A web server receives requests from the second type of software, a ***web browser***. Firefox, Safari, Opera, Chrome and Internet Explorer are common examples of web browsers. A web browser runs on a user's computer and allows the user to get information from any web server that is connected to the Internet.

A web browser creates requests, which are sent to a web server. Which server? That depends on the ***URL*** (***Universal Resource Locator***) included in the request. The URL identifies not only the network address of the web server, but also what content on the web server is desired. For example, the URL **http://www.lisabetsarai.com/news.html** specifies that the server can be identified as **www.lisabetsarai.com**, and that the user wants content named **news.html** from this server.

The language used to make requests is called ***Hypertext Transfer Protocol*** (***HTTP***). You don't need to worry much about this language. However, it's remarkably simple. There are only half a dozen possible commands in the HTTP language.

When a web server receives an HTTP request, it tries to create a response. If the URL specifies a particular item of content that is stored on the web server computer (e.g. the file named **news.html**), the web server will read that file and send its content across the network, back to the web browser. Normally the retrieved content will be expressed in a language called ***Hypertext Markup Language*** or ***HTML***.



The main function of a web browser is to make requests, receive content expressed as HTML, and display that content in the web browser window. All Web browsers understand HTML and know how to follow its commands in order to display a nicely-formatted document. Thus, if you want to create your own content for your website, blog or social networking site, you need to know at least a little bit of HTML.

What the Heck is HTML?

HTML is an example of what is called a "markup language". An HTML document includes the text or other information that you want to show on a web page, plus commands that tell the browser (in general terms) what to do with this content: how it should be displayed or what actions it should cause.

HTML markup commands are always enclosed in angle brackets (<>). For example, the markup command used to indicate the start of a paragraph is ** **.

Actually, most (but not all) HTML markup commands come in pairs. The first member of the pair indicates the start of some section to be treated in a particular way. The second member of the pair indicates the end of the section. For signaling a paragraph, the ending markup is ** **. In general, the ending command is the same as the starting command, with a forward slash after the opening angle bracket (the "less than" sign).

Some of the most common HTML commands are used simply to format text. For example, you can create section headings of various sizes using the markup **<h1>**...**</h1>**, **<h2>**...**</h2>**, and so on. The text of the heading replaces the .... The smaller the number, the larger the heading text. The subtitle of this section could be coded in HTML as:

```
<h3>What the Heck is HTML?</h3>
```

Here are some other common text formatting markup commands:

Markup	Effect
...	Makes the enclosed text bold
...	Makes the enclosed text italic
<pre>...</pre>	Encloses text that should not be formatted or paragraph-wrapped. The text will appear exactly as typed.
 	Causes a line break. Note that this markup does not have a corresponding end command.
<hr>	Displays a horizontal rule (a dividing line, often displayed as raised)
...	Makes the enclosed text be displayed in a larger font than "normal"
...	Makes the enclosed text be displayed in red.

The last two examples show another wrinkle concerning HTML markup. Many HTML command can include attributes that modify their normal behavior or appearance. If I wanted my heading, above, to be centered, I could add the attribute **align**, as follows:

```
<h3 align="center">What the Heck is HTML?</h3>
```

There are a few hundred HTML markup commands, but don't worry about learning them all. You can look them up when you need them. By the way, HTML markup commands are often called "tags". That's what I will call them from now on.

Just remember that whatever tags you include in your HTML document will be interpreted by the browser and used to format the content of the document. You need to recognize that different browsers may produce somewhat different-looking displays from the same HTML document. There are some commands that are not standardized, which will only work in some browsers. However, there is a standard definition of HTML; if you start to write a significant amount of HTML, you should probably get yourself a book that covers the standard.

Images and Links

Formatting text is obviously important. After all, we're writers - text is our primary tool. However, graphics and links are what make the Web so effective and attractive as a way to communicate.

Including a graphic image in a web page requires adding a single markup command, the **<img>** tag. This tag does not have a corresponding ending tag. Inside the tag, you must specify the **src** attribute to indicate what image should be displayed. For example, when I originally wrote this article for my website, the HTML that I used to display the web architecture image above was as follows:

```
<img src="./WebArch1.jpg" align="left" hspace="5" border="1" alt="Basic Web Architecture">
```

Most web browsers can display JPEG or GIF images. Some browsers can display other formats. Remember, though, that a typical computer screen and browser window may be as small as 1024 dots across (even smaller for smartphones and other special devices). You will often need to reduce the

resolution of your images so that they will fit on the screen. We'll talk about how to do that in a few months.

The **<img>** tag provides **width** and **height** attributes that will tell the browser to change the size of the image. However, the visual quality of the result is often poor.

There are many attributes that can be used with **<img>.** The ones that I used above define the location of the image in the browser window, allow the text of the article to wrap, and create a border around the image. The **alt** attribute defines a text string that will be displayed if the image cannot be found or if a user accesses the web page with a text-only browser.

What about links? Anyone who uses the web knows what a link (more correctly, a "hyperlink") is: some text that you click on, which will cause the browser to take you to a different location. The new location (called the "link target") can be a page on a different web site, another page on the current site, or even a different location on the same page.

So, to create a link, the first thing that you need to know is the URL of the page that should be displayed when someone clicks the link. There is a short cut method for specifying pages within the same site, but you can always use the full URL (see above).

Once you know the URL, you must decide what the text of the link should say. What text will the user click on? For example, if I have a link to my latest newsletter, "my latest newsletter" is the link text.

The tag to create a link is called the ***anchor*** tag, and is written as **<a>**. This tag has a required attribute, **href**, which indicates the target URL. The anchor tag also has an ending tag, ****. In between the begin anchor and end anchor tags, you put the link text.

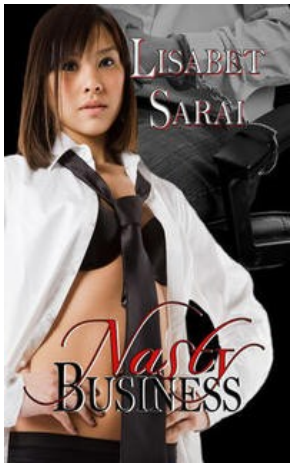
So, putting this all together, the code for the newsletter link above is:

```
<a href="http://www.lisabetsarai.com/news.html">my latest newsletter</a>
```

What about combining a hyperlink and an image? How do you do that? It's easy! Instead of putting link text between the begin anchor and end anchor, you put the appropriate **img** tag. So you would could use the following HTML code:

```
<a href="http://tinyurl.com/apbhych">
    <img src="http://www.lisabetsarai.com/NastyBusinessCover200x320.jpg">
</a>
```

to create a link like this:



This example demonstrates a very important feature of HTML. You will often need to nest one set of tags inside another. When you do, it is essential that the ending tags occur in the opposite order from the beginning tags, that is, that the inner markup is completely enclosed by the outer markup. If, for example, you want to make some text be both bold and italic, you could do so as follows:

```
<b><i>Some Text</i></b>
```

Note that you must end the italic before the bolding. If you put the ending tags in the wrong order, you may get some strange results. Some browsers might not even be able to display your page.

Creating Lists

If you use a modern word processing program, you are probably accustomed to capabilities for automatic numbering and formatting of lists. These features are convenient since you can insert or rearrange list items without having to worry about manually revising the numbering.

HTML offers similar capabilities, though you don't necessarily have as much detailed control over the appearance of the list. To create a list in HTML, you need to use two sets of tags, one to define the list itself, and another to define individual items on the list.

Here's an excerpt from my To Do list, an example of a numbered list:

1.Write 10K on “Her Secret Weapon”
2.Edit “Coming Together Presents: Amanda Earl”
3.Prepare blog article for ERWA Blog, December 21
4. Finish “Cat Toy”

5.Submit review for Erotica Revealed

To code this example in HTML, I would use two sets of tags. The tags **
** and **
** begin and end an ordered list. The tags **
** and **
** mark the start and finish of a list item. HTML automatically assigns numbers according to the order of the list items. So the code for the list above looks as follows:

```
<ol>
        <li>Write 10K on “Her Secret Weapon”</li>
        <li>Edit “Coming Together Presents: Amanda Earl”</li>
        <li>Prepare blog article for ERWA blog, December 21</li>
        <li>Finish “Cat Toy</li>
        <li>Submit review for Erotica Revealed</li>
</ol>
```

Would you rather have your list items preceded by letters? You can control this using the **type** attribute with the **<ol>** tag. For instance, to change the list above to use lower case letters, I would begin it with **<ol type="a">**:

```
<ol type=”a”>
        <li>Write 10K on “Her Secret Weapon”</li>
        <li>Edit “Coming Together Presents: Amanda Earl”</li>
        <li>Prepare blog article for ERWA blog, December 21</li>
        <li>Finish “Cat Toy</li>
        <li>Submit review for Erotica Revealed</li>
</ol>
```

Uppercase letters and other formats are also available.

Like most HTML constructions, lists can be nested (included) inside other lists. Thus, it is easy to create an outline, or a multiple choice poll:

1.Who is the sexiest actor?
    a. Brad Pitt
    b. Hugh Jackman
    c. Clive Owen
    d. Michael Caine

2.Who would you most like to have endorse your book?
    a. Oprah Winfrey
    b. Margaret Atwood
    c. Portia da Costa
    d. Salman Rushdie

You'll find the HTML code for this nested list on the next page.

```
<ol>
  <li>Who is the sexiest actor?
    <ol type="a">
      <li>Brad Pitt</li>
      <li>Hugh Jackman</li>
      <li>Clive Owen</li>
      <li>Michael Caine</li>
    </ol>
  </li>
  <li>Who would you most like to have endorse your book?
    <ol type="a">
      <li>Oprah Winfrey</li>
      <li>Margaret Atwood</li>
      <li>Portia da Costa</li>
      <li>Salman Rushdie</li>
    </ol>
  </li>
</ol>
```

I have indented to show the different levels of list; this is not required. Notice that the nested list does need to be completely inside the list element tags for the outer list. It's quite easy to get confused; if you put the end tags in the wrong places, you can get some strange results.

In addition to ordered lists, HTML also provides "unordered lists", which use the tags **
** and **
**.

- Sexy actors
    - Brad Pitt
    - Hugh Jackman
    - Clive Owen
    - Michael Caine
- Book endorsers
    - Oprah Winfrey
    - Margaret Atwood
    - Portia da Costa
    - Salman Rushdie

The HTML for this nested list would be the same as for the numbered list example, except that I qouls replaced **
** and **
** with **
** and **
**, and take out the type attribute. On my browser, at least, HTML automatically uses a different style of bullet for the nested list, which looks quite nice. I suspect that it is possible to control the style of bullets directly, but I've never tried.

Tables

Tables are an extremely powerful and useful construct in HTML. Obviously they can be used to present data in rows and columns, like the following simple (and fanciful!) example:

Book	Jan Sales	Feb Sales	Mar Sales
Raw Silk	12,040	15,231	22,765
Incognito	23,309	18,756	32,605
Exposure	17,652	20,121	22,811
Nasty Business	26,122	39,100	41,615

However, tables can be used in other contexts, to organize both graphical and textual data. For instance, the array of book covers and commentary at http://www.lisabetsarai.com/books.html is implemented as a table, as is the display of "Lisabet's Friends and Colleagues" on my Links page.

There are many tags that can be used in creating tables, and each of them has a variety of possible attributes. However, to code a basic table, you need only four sets of tags:

<table> and **</table>** mark the beginning and the end of the table.

<tr> and **</tr>** mark the beginning and the end of a table row.

<th> and **</th>** are used inside a table row, to define column headers.

<td> and **</td>** are used inside a table row, to define normal table data.

Putting these four sets of tags together, we can create the dream book sales table shown above, with the following HTML code:

```
<table width="100%" cellpadding=”5” border=”5”>
 <tr>
   <th>Book</th>
   <th>Jan Sales</th>
   <th>Feb Sales</th>
   <th>Mar Sales</th>
 </tr>
 <tr>
   <td>Raw Silk</td>
   <td>12,040</td>
   <td>15,231</td>
   <td>22,765</td>
 </tr>
 <tr>
   <td>Incognito</td>
   <td>23,309</td>
   <td>18,756</td>
   <td>32,605</td>
 </tr>
```

```
  <tr>
    <td>Exposure</td>
    <td>17,652</td>
    <td>20,121</td>
    <td>22,811</td>
  </tr>
  <tr>
    <td>Nasty Business</td>
    <td>26,122</td>
    <td>39,100</td>
    <td>41,615</td>
  </tr>
</table>
```

Once again, I've used indenting to make the structure of this code clearer. Notice that the table description is purely hierarchical. Table rows are contained within the table tags. Table header or table data tags are contained within the table rows.

You might also notice that you do not need to explicitly tell the browser how many columns you want in your table (although you can). The maximum number of table data or table header tags in any table row will determine the number of columns for the table as a whole. If some rows have fewer table data cells than others, the table will look odd.

You can control many aspects of the table appearance using various attributes, including alignment of the table cell contents, width of the table columns, and thickness of the border, if any. You can also create table rows in which some of the data span multiple columns. I've inserted a spanning heading in the example below, just to demonstrate.

Book	**Q1 2008 Sales Figures**		
	Jan Sales	**Feb Sales**	**Mar Sales**
Raw Silk	12,040	15,231	22,765
Incognito	23,309	18,756	32,605
Exposure	17,652	20,121	22,811
Nasty Business	26,122	39,100	41,615

In this modified example, I added a new table row above the previous heading row, with the following code:

```
<tr><th>Book</th><th colspan=3>Q1 2008 Sales Figures</th></tr>
```

The **colspan** attribute makes the header extend over three columns.

The really powerful thing about tables is that the table cells (material between the **<td>** and **</td>** tags) can contain pretty much any HTML code. This means that you can embed an image inside a table cell (as I did on my Books page), or a list, or even another table. The possibilities are pretty much limitless.

If you spend some time experimenting, you will find that you can create some very sophisticated information layouts using tables.

What's Next?

I think that I'll stop here. I don't want to overwhelm you! However, HTML provides many more useful capabilities. There are image maps, for example, images with multiple sensitive areas that take you to different places. Of course, it is also possible to embed multimedia content such as a video or audio clip in a web page. HTML also provides tags that allow you to create forms for input, with text fields, drop down lists, push buttons and so on.

I'll leave you with one last suggestion. If you are ever curious as to what the HTML for a web page looks like, you can find out. Most browsers have a ***View*** menu which includes an item called ***Page Source***. (In the latest version of Firefox, you have to go to ***Tools->Web Developer*** for this.) If you select this menu item, the browser will open a new window and show you the HTML used to create the page.

Warning! For many web sites, you'll find this to be totally incomprehensible! This is because many sites are created by tools which do not format the HTML to be easily read and understood by a human. (Also, some sites are generated by special programs running on the web server, rather than simply being a stored HTML document.) However, if you go to http://www.lisabetsarai.com/html101.html and choose ***View->Page Source***, you should be able to see lots of examples of the points that I've made in this chapter.

Also, you can find many excellent HTML tutorials on the web. For reference purposes I generally use W3Schools (http://www.w3schools.com/html/). This site has a live demo capability, so you can try things out and see the results. Fun!

Image Problems, or,
I Haven't Looked This Bad Since My High School Yearbook

If you're an author (and I assume you must be, or you wouldn't be wasting your time on this book), words are your primary medium of expression. You use them to set the mood, to evoke emotion, to help readers imagine your characters' actions. Your success depends on your skill in fashioning worlds with words.

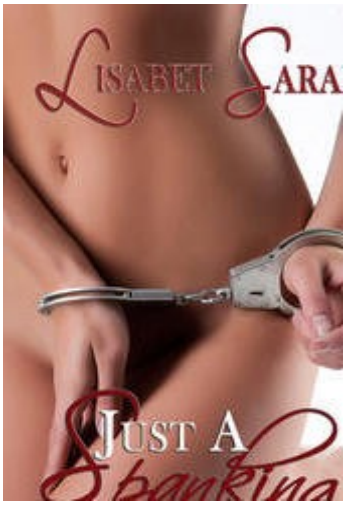
In this age of digital media, though, authors need to deal with pictures as well as text. Surveys have shown that readers pay at least as much attention to a book's cover as they do to the blurb in making purchase decisions. Adding photos or images to your blog posts or Facebook page helps liven up the content and "grab eyeballs", in e-Marketing terminology. Reader-oriented websites like Goodreads, Whipped Cream Reviews, The Romance Studio and so on sell advertising space for authors to post book covers or graphical banners, luring viewers to click through and make a purchase. PR types advise that in addition to showcasing your covers, you should have a "head shot", so readers can associate a face with your name.

All the images you're likely to deal with in your marketing efforts are ***digital images***, that is, images represented by patterns of bits. (We'll talk about what that means in a minute.) Unlike photos taken with film cameras, or drawings on paper or canvas, digital images are almost infinitely malleable. They can be enhanced or modified in a wide variety of ways to satisfy different needs.

Unfortunately, this flexibility can come back to bite you. You might submit a head shot to a review site that is showcasing your work and have it come out looking as though you really need to go on a diet:

Or as if you're an alien from some other star system:

Your cover image might end up being displayed like this, with critical information being hacked off.



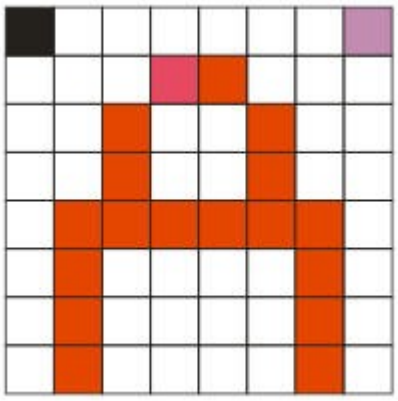
Or you might discover that the size of the advertising slot you've purchased is just plain too small for your cover to be read.

My goal in this chapter is to help you understand the source of these image problems, as well as to provide an overview of how digital images are constructed and represented in a computer and how you can get better control over how they look.

I won't be talking about how to *create* cover images, by the way, although that's an important and somewhat related topic. That's really outside my area of expertise. However, after reading this chapter, you may be able to better understand why some covers look great even at very small sizes, while others are unreadable.

What is a Digital Image?

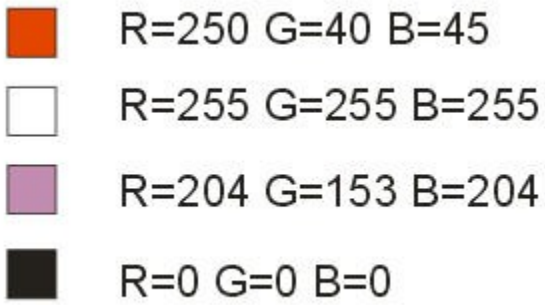
An image that can be manipulated by a computer is structured as a two-dimensional matrix of dots, or ***pixels***. Each pixel holds information about a particular location in the image - specifically, information about its color. The figure below shows an image of the letter **A** which is 8 pixels wide by 8 pixels high.



Digital images use numbers to describe the color of a particular pixel. Actually, a full color image associates three numbers with each pixel, one of which measures the amount of red in the pixel, one the amount of green and one the amount of blue. You can think of each of these numbers percentages. A pixel that is 100% red, 100% green and 100% blue will be white. A pixel with 100% red and 0% of green and blue will be pure red. A pixel with 100% red, 0% green and 100% blue will be magenta. And so on. The three components mix to produce the final color.

Here's where the bits come in! You may recall from the first chapter that computers use a series of bits to represent numeric values, where more bits allows a larger range of values. Digital images often (though not always) use one byte (8 bits) for red, one byte for green and one byte for blue. Each byte can hold a value between 0 and 255.

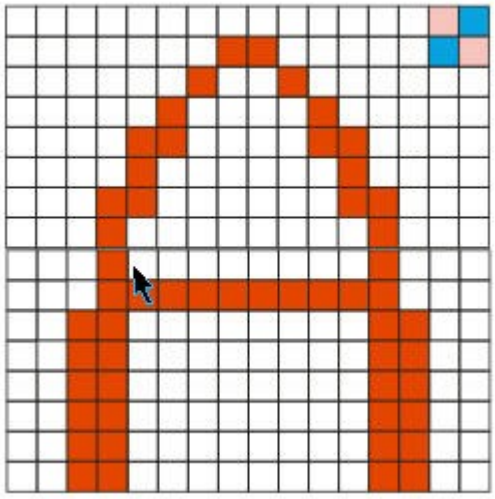
The colors in the figure above translate to the following values of red, green and blue.



If we allow one byte for each of the three components, we can specify more than 16 million different combinations. This is actually more than the human visual system can distinguish.

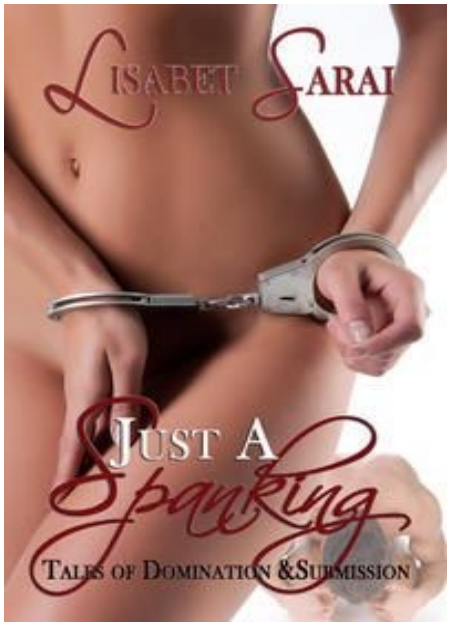
Image Resolution

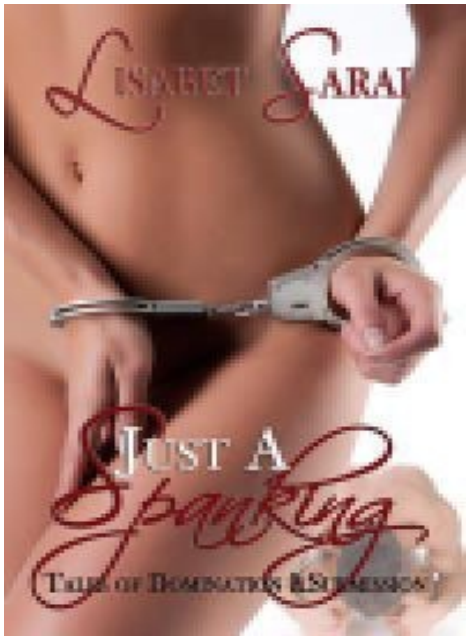
You may notice that my sample letter **A** above looks pretty poor. A matrix only 8 pixels wide and 8 pixels high makes it difficult to show a lot of detail. In general, the larger the number of pixels in an image, the clearer the image will appear, and the more detail you will see. If I double the number of pixels in each direction, I can create a somewhat better looking **A**, with smoother curves.



The dimensions of a digital image are called the ***image resolution***. Resolution is usually expressed as ***W x H***, where **W** is the width in pixels and **H** is the height in pixels. In general, the higher the resolution, the better the image quality. Compare the clarity of the two covers below. The one on the

left is 200 pixels wide by 320 pixels high (200 x 320 resolution). The one on the right has a resolution of 100 x 160, but has been adjusted to take up the same amount of screen area.





The difference in quality is quite obvious. The lower resolution image looks blocky and blurry.

So does this mean you should always use the highest resolution image possible? Not necessarily. There are two additional issues you must take into consideration: screen resolution and file size.

Most of the images you'll use for your marketing - for websites, blogs, or to embed in emails - are intended to be displayed on the screen associated with some computational device: a stand-alone monitor, laptop screen, tablet or phone. The display capability of computer screens is also expressed in pixels. A typical modern LCD screen will have a ***screen resolution*** of about 1600 pixels across and 1200 pixels high. The (fairly old) laptop that I'm using to write this chapter has a 1024 x 780 screen.

When you display a digital image on a computer screen, the resolution of the image and the resolution of the screen combine to determine the visual size of the image. For example, if I display a 500 x 400 image on my laptop, the image will take up about a quarter of the screen area - about half the width and half the height. If I had a screen that was 1600 pixels wide, the same image would look smaller; it would take up less than a third of the screen width.

Thus, when you're deciding on an effective resolution for an image, you want to consider what else will be displayed at the same time. You want readers to be able to see the text on your blog or web

page, as well as the illustrations! Websites that offer cover or banner ads normally have a standard, required image resolution - usually quite small. They want to be able to fit many images on a single page.

The other issue you should think about is how much memory and disk space your image will consume. The higher the resolution, the more space required. For example, if we have a 500 x 500 image, this means we must store 250,000 pixels. Using one byte each for red, green and blue, the basic storage required will be 750,000 bytes - three quarters of a megabyte. If we double the resolution in both directions, to 1000 x 1000, we quadruple the number of pixels, to one million, and we will need three megabytes of space to store the image.

Now you may be shaking your head, saying "So what? What difference does it make?" It's true that these days memory and disk space are relatively cheap. However, in most cases where you're likely to be using digital images - on a blog or website, or embedded in an email - the image data will be transmitted over the Internet. Including a 1000 x 1000 image on a web page will definitely slow down the display of that page. If your readers are using mobile devices, which have slower download speeds, the impact will be even more serious.

In addition, some web hosting companies have limits on how much space you can use. Even Google limits your space on Picasa to one gigabyte (1000 megabytes). That may seem like a lot, but at three megabytes per image, that's fewer than 400 photos. I took nearly that many on my last vacation!

So you may need to balance your desire for clearer, more detailed images against practical issues such as space requirements, download time, and the limits imposed by advertisers or other consumers of your images.

Manipulating Images

Suppose that you've decided to purchase a cover ad at some website. The site requires an image that is 100 x 160 in size. Your publisher has send you a file that is 800x1280. What do you do?

To resize an image, you need appropriate software. You could use a graphics arts program like PhotoShop, but for the most common tasks you don't need anything nearly as elaborate or expensive.

If you work in the Windows operating system, get yourself a copy of IrfanView (http://www.irfanview.com/). This free program is one of the best designed pieces of software I've ever encountered. It can reduce or enlarge images, change the color balance or the contrast, sharpen and do other enhancements, convert from one format to another, run a scanner, create a slide show... And it's so

easy to use that even the most dedicated technophobe will find it a snap.

If you're a Linux geek like me, there are a variety of options, including Image Magick (http://www.imagemagick.org) and Gimp (http://www.gimp.org). These packages have most of the same capabilities as IrfanView, though in my opinion, they're not nearly as user friendly.

On the Mac, I believe you can use iPhoto to modify your images.

To change the resolution (and thus the size) of an image, you must specify the new resolution, either in terms of pixels or as a percentage of the original. Be sure you understand which of the two values is the width and which is the height. Most programs display the results, so you can check whether your assumptions are correct, and start over if you guessed wrong.

You need to decide whether or not you will maintain the ***aspect ratio***. Aspect ratio specifies the relationship between the width and the height. If you change the aspect ratio, your image may look stretched or squished, like my head shots at the beginning of this chapter.

To determine whether you need to change the aspect ratio, compare the ratio of width to height for the original size and the desired new size. In my example, the original aspect ratio is 800/1280 or 0.625. The ratio for the required dimensions is 100/160, which is also 0.625. To perform this resizing operation, we need to reduce the resolution to 12.5% of the original (100/800) in both width and height.

What if the required resolution implies a different aspect ratio than the original? Both IrfanView and ImageMagick (the two programs I use most often for this) have a check box to control the aspect ratio. If the box is checked, you can only change one dimension (either width or height). The other dimension will vary automatically to keep the same ratio. If you uncheck the box, you can vary the width and height independently.

A slight change in aspect ratio usually will not cause problems with appearance. I routinely resize the 200x300 covers provided by one of my publishers to the standard 200x320 I use on my website. However, if you discover the resized image looks bad, your other option is to maintain the aspect ratio and then ***crop*** the result.

Cropping involves selecting a region of an image (usually rectangular) to keep, while throwing away the rest. If you're working with a cover image, you may be able to identify background areas that you can cut out without seriously harming the readability or the composition.

Suppose I have the 200x320 cover image shown in Figure 1 below.

I sign up for a cover ad and learn that the website wants a 200x240 image. This is a significant change in aspect ratio (0.625 to 0.833).

When I try do the resizing operation, not maintaining the aspect ratio (Figure 2), the woman on the cover looks pretty chunky – especially her butt!

As alternative, I can crop off the top and bottom of the cover to reduce the height from 320 to 240 (Figure 3).

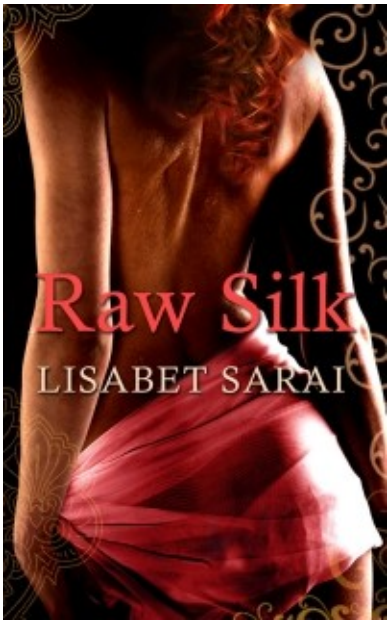


Figure 1 – Original Cover (200x320)

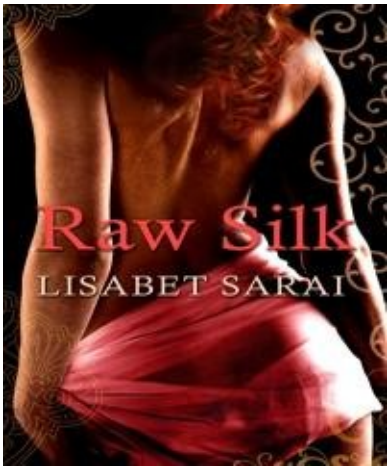


Figure 2 – Resized Cover (200x240)

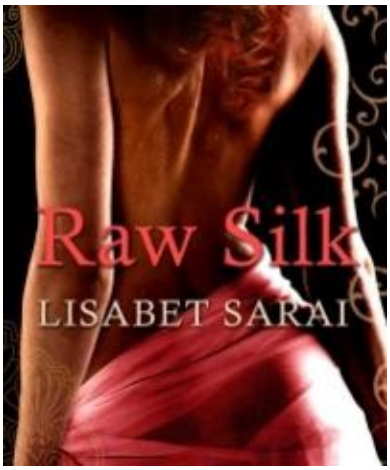


Figure 23 – Cropped Cover (200x240)

Does this look better? In the case of this cover, it's a matter of taste. However, in some cases, the choice between changing the aspect or cropping will be more clear cut.

Be sure that you save all the different versions of your image, with different file names. That will allow you to compare different approaches and decide which ones work best.

There are other situations where cropping is useful. If you want to include a photo from your travels, but you don't want to show your husband's face (for reasons of anonymity, not because he's ugly!), you can crop him out. I often use cropping when I download a cover image from the Amazon website to use in a guest's blog post. The cover images on Amazon include large areas of white background; cropping lets me create an image of just the cover.

Remember that earlier I said higher resolution images have more detail. What happens to the detail when you reduce the resolution? Looked at another way, how does the software determine what values to use for the new pixels in the reduced image?

The process of calculating new values when changing the image size is called ***resampling***. There are many mathematical techniques for resampling. The simplest will simply average the red, green and blue values for old pixels that are contributing to a new pixel. This generally will not give very good results; the reduced size image will look blurry. Different software uses different methods, some more effective than others. In particular, web browsers have the ability to resample images, but the resulting quality is usually poorer than you'd get with a dedicated graphics program like IrfanView. This is one reason why you should avoid using the **width** and **height** attributes on the HTML **<img>** tag and create your images at the desired size instead.

Alphabet Soup: Image File Formats

Before I finish, I want to say a few words about different image file formats. You've probably heard of JPG and GIF formats. There are dozens of other formats for storing images as well.

What is an image format? It is a standard scheme for storing the bits that make up the image. Two different formats may both use one byte each for red, green and blue, but they may pack this information into a disk file in different ways. Usually when you use an image manipulation program like IrfanView, you get to decide what format to use when you save the file.

For images that will be used on the web, the three most important formats are JPEG (JPG), GIF and PNG. Modern browsers can display all three.

The advantage of JPEG files is that they can reduce the size of an image dramatically, through a process called ***compression***. Compression takes advantage of the fact that pixels in the same region of an image tend to have similar values, so we don't necessarily need to store multiple copies of the same data. I said earlier that a 1000 x 1000 image would take up three megabytes of disk space, but if stored in JPEG format, the same image is likely to be much smaller. The disadvantage of JPG is that it actually throws away some information as part of the compression, resulting in an image that may be less clear or detailed than the uncompressed version.

GIF files also use compression, but without losing information. The main weakness of the GIF format is that a specific GIF file can only hold 256 different colors (as opposed to 16 million). On the other hand, the GIF format has two big advantages. It allows you to specify one color as "transparent", so that the background will "show through". This makes it possible to create graphical elements that look as though they have an irregular outline. Also, GIF supports simple frame-based animation, so you can (for example) create a banner that acts like a slide show. (To see an example, go to http://www.lisabetsarai.com/TheUnderstudyAnimated.gif).

The PNG file format were invented because of patent restrictions that originally encumbered GIF files. PNG files have similar capabilities to GIF (including transparency) but offer more control over color. On the other hand, since they are less common, you may find programs that do not handle them correctly.

They say that a picture is, but I've taken several thousand to describe how you can take control of yours. I hope I haven't bored you. As usual, you need to experiment yourself with some simple image manipulation tools to become comfortable with these basic but important concepts.

Did the Earth Move?
A Quick Tour of Active Web Content

Back in Chapter 2 (HTML 101), I explained the basic way the World Wide Web works. It's a remarkably simple paradigm. A program running on your computer called a ***web browser*** (Firefox, Chrome, Safari, Internet Explorer, etc.) sends a request to a ***web server*** - a program running on some distant computer connected to yours via a network. The request includes an item called a ***URL***, which identifies the content you are requesting. The web server locates the desired content and sends that data back to your browser over the network. Then your browser ***renders*** (formats and displays) the newly received page on your screen.

In the most straightforward case, the requested content already exists as a file (usually holding HTML) stored on the web server. The server merely retrieves that file from disk and sends it back, unmodified. Websites that operate this way are known as ***static*** web sites. On a static website, a request for a particular URL will always display the same information.

Very few modern web sites are completely static. The world has decreed that static sites are just not cool. Pundits proclaim that every site needs to be dynamic: displaying animation or videos, changing in response to user interaction, allowing viewers to upload photos, add comments, “like” content or forward links to friends. After reading about the incredibly simple request-response cycle that underlies all web communication, you may wonder how all this complex, flashy active content works. That's my topic for this chapter. I'm going to give you a brief survey of some of the different mechanisms web sites use to move beyond static page displays.

Why do you need to know about this? In order to make intelligent choices about the active content you want on your own sites and blogs. There's a temptation to add as much active content as possible to your sites. After all, static web sites are boring, right? However, there are risks and costs associated with active content, and as I discuss the various approaches you might adopt, I want to point these out.

And lest you make the erroneous assumption that static websites are useless and hopelessly old-fashioned, let me point out that my author website, http://www.lisabetsarai.com uses primarily static content. Certainly, dynamic content is essential for some kinds of web applications. I've yet to be convinced that an author site is one of them.

I'm going to quickly discuss six mechanisms for adding active content to a web site: server-side

programs, JavaScript, animated GIFs, plug-ins (including Flash), applets and ActiveX controls, and HTML5. I'll conclude by offering my own recommendations for using active content.

Server-Side Programs

The most powerful and flexible way to make a web site behave dynamically is ***server-side programming***. In this scenario, when the web server receives a URL, it doesn't look up a file. Instead, it runs some software. This software will create the content to be returned to the requester.

Server-side software can be enormously complex. It can do calculations, retrieve or store information in data bases, even send requests to other web servers. E-commerce sites (like Amazon or Barnes & Noble), social network sites, blogging platforms (like Blogger and Wordpress), and most other large scale web applications depend heavily on server-side programming.

One commonly used technique for invoking server-side programs is ***HTML forms***. HTML includes tags for creating text fields, drop down lists, radio buttons, push buttons and other sorts of user interface controls as part of a web page. Every HTML form is associated with an ***action*** that is invoked when the user clicks the Okay or Submit button. A form action is a special URL that identifies the program to be run (on the server) when the button is clicked. The browser automatically sends the material the user has entered in the form as part of the request. Then the server program can use that information to do something intelligent (like adding a user to your address list database, or recording a comment).

Most authors probably won't be writing any server-side code, although creating scripts to do simple tasks isn't all that hard. Still, you can always hire someone to do this sort of work for you, or find free server-side scripts that you can incorporate into your site. Server-side programming has two huge advantages. First, it has pretty much unlimited power and flexibility. Second, there's generally little security risk to the user because the programs are executing on a different computer.

On the negative side, sites that rely entirely on server-side programs may not feel very responsive or natural. The browser has to wait for the reply to its request. The delay includes not only transmission time over the network but also processing time on the server. Long waits can be annoying and frustrating for users.

JavaScript

JavaScript is a programming language that runs on your computer, inside your browser. Web

pages sent from the server can include JavaScript programs, embedded in the HTML. When the browser receives the page data, it will process and possibly execute the JavaScript code, which may modify the appearance or the content of the page.

Often, JavaScript functions are written to respond to user actions, like mouse movement or button clicks, by changing the page content. For instance, if you go to my web site (http://www.lisabetsarai.com/welcome.html) and run your mouse over the menu buttons on the left, you'll notice they change color. This little bit of activity is handled by JavaScript. What is actually going on is that JavaScript is replacing one button image with another one in response to the mouse event. (If you want to look at this in detail, just select your browser's “View Page Source” menu item.)

JavaScript is often used to validate data entered in forms (e.g. to make sure that all required fields have been filled in, or that a phone number has the correct structure.) It makes it possible to implement drop down menus and other flexible UI behavior on a mostly static page. JavaScript can draw graphics, change background or text, switch from one page or tab to another, even (under certain circumstances) send a new request to the server without redrawing the entire page.

JavaScript is surprisingly powerful. The “widgets” available on Blogger and many other sites are mostly chunks of JavaScript code that are designed to be embedded in a web page. For instance, if you go to my blog (http://lisabetsarai.blogspot.com) you'll notice a slide show of cover images in the upper left. I'm pretty sure this is handled by a JavaScript module downloaded from Google's servers.

However, there are certain things a JavaScript program cannot do. In particular, JavaScript cannot read or write files on your local computer. This is a deliberate design decision which makes it less likely that a JavaScript routine might corrupt or compromise your computer.

Most websites these days (even my antediluvian site!) use at least a bit of JavaScript. However, there are risks in relying on JavaScript too heavily. Users can choose to disable JavaScript in their browsers (and some do). Furthermore, JavaScript doesn't necessarily behave the same way in different browsers. You should be sure that your web site functions adequately with JavaScript turned off. A static site may be boring, but a broken site that prevents user navigation is far worse.

Animated GIFs

Animated GIFs are a simple, safe way to add a bit of activity to a page. As I explained in the previous chapter, an animated GIF is an image file that includes multiple frames plus timing information. Browsers treat an animated GIF like any other image, downloading it to the browser

memory the first time it is referenced by a page. This may cause some delay. After that, no further communication with the server is needed. However, switching frames does take some of the browser's processing power. This may slow rendering on older computers.

You can make your own animated GIFs. I use an ancient but reliable Windows program called GIF Construction set, from Alchemy Mindworks. You can also find animated GIFs on the web. For instance, while I was researching this book, I happened upon the sexy pole dancer you can view at http://www.lisabetsarai.com/poledancing.gif. Check it out!

Animated GIFs are effective for things like click-through ads. They get a bit tiresome if you have too many of them, since they repeat forever. Also, it's difficult to make a really complex animation using an animated GIF (although I gather there are now programs on the web that will turn video clips into animated GIFs.) With a dozen frames, even the pole dancer is more complicated than anything I've ever created.

Plug-ins and Flash

Real-time video content or complex animation requires more powerful technology than animated GIFs. To display video content, browsers use add-on program modules called *plug-ins*. A plug-in extends the functionality of a browser so that it can render additional kinds of content. Popular plug-ins are usually free to download and simple to install. However, you (or your visitors) must get the correct version for your operating system and browser. Also, many plug-ins have frequent updates, so managing them can be a chore.

Plug-ins are available for various audio/video formats such as MPEG, AVI, QuickTime and Adobe Flash. Flash is probably the most popular format for video and animation, partly because it was designed to be streamed over the Internet. Adobe also provides sophisticated authoring tools for creating Flash animations. If you'd like to see what Flash can accomplish in the hands of a true artist, visit Jacquie Lawson's e-greeting card site and preview some of her gorgeous cards. Currently, YouTube streams its videos in Flash format.

Unfortunately, the Flash system and plug-ins have proven to have many flaws that make them popular vehicles for malware (spyware, viruses, trojans, worms and other evil programs). Flash content is actually a downloaded program that runs, inside the plug-in, on your computer. Apparently it's quite easy to embed malicious code in a Flash object. This code can subvert the security mechanisms of the browser and gain access to your files and operating system: stealing your personal data, compromising

your identity, or turning your computer into a zombie that sends out spam messages or viruses upon command. The SANS security institute (http://www.sans.org) reports new Adobe Flash vulnerabilities on the average of once a month.

Some users (including your humble author) will refuse to install the Flash plug-in because of these security issues. Thus, you should make sure that your site or blog works well in the absence of a Flash player. Don't rely on animations of any type to convey critical information or for navigation. You cannot assume that all your visitors will play your videos.

Java Applets and ActiveX Controls

Most of the plug-ins you're likely to encounter are oriented toward rich media. However, you can also download and install a plug-in that can run generic applications called “applets” inside the context of your browser. Applets are written in a programming language called Java. You might use an applet to embed a drawing application or whiteboard, a dynamic book reader, or an interactive puzzle on a web page.

ActiveX controls are similar to Java applets in concept. However, they work only in Microsoft's Internet Explorer.

Using either of these mechanisms involves downloading a chunk of executable code from the web server. The HTML page hosting the applet or control includes a specification of where the code should come from. Once the program has been downloaded, the browser starts it running.

Applets and Active-X controls can do far more complicated things than any of the browser-based mechanisms I've discussed so far. However, the richer the functionality of an applet, the bigger the code object (usually) and the longer it takes to download. Also, for applets at least, you must install the plug-in, and the plug-in version needs to be compatible with the Java version of the applet. There was a great deal of excitement when applets were first introduced in the late nineties. These days, though, they do not appear to be all that widely used.

HTML 5

An effort is currently underway to define and standardize a new version of HTML that will provide new, platform-independent mechanisms for supporting active content. HTML 5 includes a **<canvas>** tag that will allow drawing on the page, **<video>** and **<audio>** tags to support rich media, and support for a powerful graphics format called Scalable Vector Graphics (SVG). The goal is to

reduce the need for specialized plug-ins as well as to enhance the security of web applications.

Although the HTML 5 standard isn't yet final, major browsers already implement many of its new features. If you're starting from scratch building a new website, you may want to examine whether HTML 5 will give you a better way to support active content.

Active Content and You

When you see all the amazing things that can be done with Internet technology, you may lust after similarly sexy features for your own site or blog. Certainly, you want your users to enjoy the experience of visiting your site. Keep in mind, though, that every one of the active content techniques I've discussed above (except perhaps HTML5) has a downside.

Consider the performance implications of active content. Not every reader has a fast computer or even broadband Internet. I have people on my email list who still connect via dial-up. If you're trying to lure readers from outside your continent, remember that your site is likely to load more slowly for them than it does for you. Every video, animation, widget, and even image that you add to your page slows the load time. I live in Asia and I can tell you, I've given up trying to access some sites. The content isn't worth the frustration of waiting. Presumably you don't want your readers to feel this way.

Consider security as well. Some active content techniques carry significant risk of transmitting malware. You don't want your visitors catching viruses from your site! And if your visitors are more concerned about security than you are, and turn off Flash or JavaScript, make sure your message still gets through.

In short – think carefully about active content before you add it to your site. A simpler site will be clearer, faster and more reliable. Your ultimate objective is to get readers interested in your writing and hopefully, to buy your books. If you think you really need some dynamic content to achieve that goal, that's fine. Just don't be dazzled by pure coolness, to the exclusion of functionality.

Backup Blues, or
Murphy was a freakin' optimist

Ephemeral. That's the nature of an author's work, especially today when every stage of the writing and publishing process is digital. The products of our creativity and sweat exist primarily as files in computer storage, fragile collections of bits that are highly susceptible to corruption or loss. In the first chapter of this book, I explained how strings of ones and zeroes can encode any sort of meaning, including the wondrous products of a writer's imagination. At the time, I didn't emphasize the dark side of this technological marvel. Let just a few ones flip over to zeros, or vice versa, and a manuscript becomes unreadable!

The media we use to store our precious stories are subject to a wide range of threats. Dust, electromagnetic radiation, power surges, manufacturing defects, and controller malfunctions can all cause hard disks to fail. Indeed, **every** hard disk **will fail**, eventually; the tiny, intricate mechanisms used to read and write data will simply wear out.

Thus, it's not really a question of whether you'll lose data, but when. Murphy won't be ignored. When that dreaded day arrives and your work in progress disappears, what will happen? If you've been disciplined and diligent about backing up your work, you'll face some inconvenience, but you'll be able to recover most if not all of your efforts. If you've been lazy or disorganized – if you've closed your eyes to the grim realities of the computer world – you might well have to start from scratch.

In this chapter, I want to discuss general backup issues and strategies. I'm not going to recommend specific programs, services or processes, because there's not one single approach that will work for every author. If you always write on same computer, in the same location, your backup needs are different than those of an author who travels constantly and writes on the train or in coffee shops. If you generate gigabytes of content weekly, you can't use the same approach as someone who produces only a few megabytes. A Linux geek like me, comfortable typing command lines and writing scripts, is going to use different tools than someone who runs Windows and wants to do everything through a graphical user interface.

My goal is to outline the dimensions of the problem and sketch some possible solutions, with their advantages and disadvantages.

What's the worse that can happen?

There are three categories of threat you should consider when choosing a backup strategy:

1. ***Media or computer failure***. Your hard disk crashes. Your tablet gets run over by a truck. Your kid spills Coke all over your laptop keyboard. In these cases, you need a local copy of your data so you can quickly replace what you've lost.
2. ***Major disasters.*** Your house burns down, or is devastated by a tornado, or is washed away in a flood. Assuming that you and your family escape unscathed, what happens to your stories? This sort of threat suggests a need for storing copies of your data at some other location, away from your computer.
3. ***Errors or carelessness.*** You're not exactly sure how, but in the process of your work this afternoon, you managed to clobber the three chapters you wrote this morning. I've done this more times than I care to admit, often by "cleaning up" what I thought was a redundant copy of a file or by mistyping some common command. In this situation, even the best backups will often not allow you retrieve your most recent work. Some text processing software can create automatic backup versions while you write, or you can train yourself to do this manually. As frustrating as this situation may be – my poor husband has become accustomed to my swearing and melodramatic threats of suicide – the damage done is likely to be less extensive than in the other two scenarios. With decent backups, you should at least be able to return to where you were yesterday.

The basic idea behind every backup strategy is that you want to make copies of your important files, which can be retrieved if something bad happens to your original data. The decisions you face involve the medium used to store the copy, the frequency and mechanisms of making the copy, where the copies should be kept, and how long a particular backup copy should be retained.

Backup Media Options

Let's consider medium first. For local backups, you can make copies on:

- Another hard disk attached to the same computer (including an external hard drive);
- A disk on a different, networked computer;
- A flash drive or memory stick;

- Writable non-volatile media, like CD-ROMs or DVDs;
- Paper.

The first option is probably most convenient. However, if your entire computer is destroyed, you run the risk of losing both your original and your backup copies. The second option is more complicated, since you need to figure out how to get the data from the primary disk to the backup disk, but is likely to provide a more robust backup.

I know many authors who use flash drives as their primary backup media. This solution has attractive aspects. Flash drives are cheap and highly portable, so they're great for authors writing on the go. They're also really easy to use on most platforms, though copying the data is likely to be manual rather than automated. However, using flash memory as your main backup medium has two serious problems. First, flash memory supports a limited number of write operations (in the tens of thousands). So a flash drive will become non-functional much sooner than a hard drive (possibly without warning). Second, memory sticks are so small that it's really easy to lose them. What would happen if somebody else got hold of copies of your stories? Personally, I use flash drives for backup when I'm traveling, but as soon as I get back to home base, I'll update the primary backup disk with the work I did on the road.

CDs and DVDs are less likely that hard drives or flash drives to be corrupted by environmental factors such as magnetism or power surges (though they are vulnerable to dust and scratches). The primary disadvantages of these media are their relatively small capacity (though they may well be large enough for some users), the physical space they require for storage (assuming you accumulate your backups over time), and the fact that standards change and formats become obsolete. (I have some backups of my early work on floppy disks. That means we need to retain at least one computer that has a floppy drive – something that is becoming increasingly rare!)

Using printed paper copies for backup is better than not having any backup at all. However, if you ever want to recover your work, you will need to scan it, subject it to OCR (Optical Character Recognition), and then correct the OCR errors – a time-consuming and labor-intensive process. In addition, you can't use paper to backup non-text data such as book trailers or cover images. Finally, paper requires a lot of physical storage space.

Of course, you don't have to settle on a single medium for your backups. For instance, you could do a nightly backup to a networked hard drive and a weekly or monthly backup to DVD. This kind of hybrid approach is more complex but generally more reliable than a single-medium solution.

Backup Sequence, Timing and Location

One serious error many people make is to use the same device or medium over and over, with each day's backups replacing those from the previous day. If you're doing this, and a file gets corrupted today, but you don't discover it until the day after tomorrow, you are – pardon my crudeness – screwed. You probably had a good copy of the file in yesterday's backup. But today's backup will overwrite that good copy with the bad one – before you realize what you've lost.

For this reason, an alternating or tiered backup strategy is often a wise choice. In an alternating backup, you copy your work to one drive on Monday, Wednesday and Friday, and to another one on Tuesday, Thursday and Saturday. In a tiered backup, your work gets copied to one disk or computer every day, but then that computer is backed up periodically (maybe once or twice a week) to another computer/disk. This arrangement increases the likelihood that after your delayed realization of disaster, you'll still have a copy of the uncorrupted data somewhere.

What about backing your data up to “the cloud”, storing your files on some server on the Internet? This has become a popular solution, and it does have some advantages. For one thing, it reduces the risk from major disasters, since your backups aren't stored in your home or at your office. It's also cheap (sometimes free) and convenient, especially with the automated backup clients provided by some storage services.

However, you should read the terms and conditions for such services pretty carefully. In the cases I checked:

- The service has no liability if your data gets lost, corrupted or stolen;
- The service makes no guarantee that it will remain available. Indeed, the service has the right to terminate your account at any time and will not necessarily allow you to retrieve the data you've stored with them if they do terminate you.

Beware, in particular, of free services, which can change or disappear at any time. When you're not paying anything, you also have no leverage at all with a service provider.

And what if you lose Internet connectivity? Our Internet went down for a week last May. I shudder the horrible recollection! If I'd been dependent on the 'Net for my backups, I would have risked losing an entire week's work.

So, as you probably gather, I personally don't trust cloud-based backup services. However, I do use the Internet on occasion for off-site storage of important files. I pay for hosting my website, and

this includes a good-sized chunk of disk space. When I want to be sure a copy of some data exists, physically distant from my home, I'll copy it up to the web server. I have to be careful, though, to put the data in a private directory. Otherwise any visitor to the site could (potentially) access that data.

How often should you backup your work? If you're like me, you're using your computer every day. Thus every day offers a new opportunity to mess things up! Daily backups are essential for me. Your situation might differ, of course.

And how should you make the backup copies? If you can find a way to automate the backup process, rather than relying on manual backups, it's likely to be more sustainable in the long run. It's all too easy to forget to copy your work when you're tired, or excited, or suddenly interrupted. Many software solutions for automated or semi-automated backup are available for different platforms. If you have geekish tendencies, you can roll your own.

The question of where to keep backup copies pits convenience against safety. Probably you want recent backups to be immediately available when Murphy strikes. On the other hand, it's highly desirable to have some sort of off site backup facility as well, in case your premises are destroyed. We keep a backup hard drive with our most important files in our safe deposit box at the bank.

Backup Longevity

How long should you keep backups? Clearly it's not feasible, even with today's cheap storage, to save full copies of all daily backups. However, I can tell you from experience that it's worth making periodic snapshots of your work to keep indefinitely. More than once I've found myself scanning through CD-based backups from five or even ten years ago, trying to locate an old file. Sometimes I'm successful, sometimes not. In the balance, though, it can be a lot cheaper and easier to keep permanent copies of files than to recreate those files *de novo*. The latter is often simply impossible.

Testing Your Backup Strategy

Suppose you choose a backup strategy and put it into practice. You might be lucky. You could go months or even years without losing a file. Then Murphy strikes, and you discover (for example) that none of your backup CDs can be read.

The lesson here is that you should test your backups periodically, to make sure that you actually have the information you need. I worked for a software company (back in the dark ages) where every night our systems were backed up using an expensive automated mechanism onto tape cartridges. Things went swimmingly for several months. Then our server had a disk crash. When the system

administrator tried to restore from the backups, he found that every single tape was blank.

What does the Erotogeek do?

You might be wondering about my personal backup strategy for my writing and marketing data. I have to admit that my husband, who's even more of a geek than I am, was a major force in its design and implementation. Furthermore, we have refined our approach over the years.

Here's a summary of what I'm doing right now.

1. Every night when I've finished working, I run a script that creates a zip file of all the content in the directories where I do most of my work, then encrypts that archive to keep its contents private.
2. The script keeps three days' worth of archives on my computer's temporary drive. Each day it renames the one from the previous day. So if today's archive is (for example), *backup.zip*, then yesterday's will be *backup-1.zip* and the previous day's, *backup-2.zip*. When I run the script tomorrow, *backup-1.zip* will be renamed to (and thus overwrite) *backup-2.zi*p. Today's *backup.zip* will become *backup-1.zip*. Thus I always have three days worth of backups available immediately.
3. Early each morning, an automatic process on our one of our backup servers pulls the encrypted archives from my computer to a big central storage area. We have two different backup servers, which alternate.
4. Every month or two, my husband updates a big hard drive with our latest files, and takes that to his office so that it will provide off-site backup.
5. Every six months or so, my husband replaces the hard drive in our safe deposit box with an updated version of the drive from the office. The previous drive from the safe deposit box rotates back to our home.

In addition to this regular process, I sometimes create CD-based snapshots of important directories to save. We have several binders of CD backups. Also when I'm actively working on a new book, I'll manually send each day's work to a different computer, just to have an extra copy.

A Minimal Strategy

You're probably shaking your head at this point, thinking that there's no way you could handle a process this complex. Very likely you can satisfy your needs with something simpler.

Here's a very easy procedure that will provide the primary benefits:

1. Every week, on the same day, copy all your working directories to a CD or DVD.
2. Check that you can read the DVD. Then store it somewhere outside your home.

If you do this regularly, you're not likely to lose more than a week's worth of work.

Don't delude yourself. Sooner or later, your disk will crash – or something worse – and you will need your backups. Now is the time to consider the issues I've raised in this chapter and adopt a process that makes sense for you.

Don't let Murphy get the better of you!

I Want to be Alone:
Safeguarding Your Identity, Your Privacy and Your Sanity in the Digital Age

Writing used to be a fairly private endeavor. An author would spend weeks or months alone with her typewriter or word processor, creating a work of fiction. Nobody other than her immediate family or friends knew much about her work in progress. After the book was published, luckier authors might do readings, signings, interviews, or perhaps even end up on some television talk show. Most authors shrugged, answered the occasional fan letter, and pounded away on the next book.

It's hard to believe how much things have changed. Successful authors in the digital age are required to be very public indeed. We expose ourselves on our websites and blogs. We hang out our shingles on Facebook, Goodreads, or Google+ and try to wheedle readers into “liking” or “friending” us. We populate the ether with pithy hundred character messages designed to attract attention to ourselves and our books. Many authors I know post current word counts on their blogs, so readers can track the status of their WIPs. Some set up chat groups or mailing lists to provide readers with up-to-the-minute information about everything from plot issues to their kids' school performances.

These days, authors who aren't visible in the cybersphere might as well not exist.

In some ways, this is a positive development. Publicity has never been easier or cheaper (though you have to really shout to be heard above the cyber-din). However, there's a serious downside. It has become far more difficult – close to impossible – to keep **anything** private.

I'm sure you all have read about the myriad threats to privacy lying in wait for anyone who logs on, anywhere. From hackers who steal your personal data for financial gain to trolls who persecute you purely out of malice, the Internet is full of villains. Identity theft and privacy breaches are significant concerns for everyone – but authors, especially authors of erotica, have more at stake than the average citizen.

What would happen if your neighbors – your children's teachers – your boss – your pastor, priest or rabbi – learned that you write sexually explicit fiction? Sure, there are some erotica authors who write under their own names and are up front about what they do. Most of them live in New York City or San Francisco, and most of the ones I can think of, at least, make their living from erotic writing and related activities.

The rest of us toil away at our day jobs in some less cosmopolitan and less tolerant locale,

hiding our dirty little vocation from almost everyone around us. I know authors of erotic content who have been ostracized by their communities, lost their jobs, or even been forced to move because they were “outed”.

Aside from the problem of having your mask torn away, authors are more susceptible to other privacy threats as well. Online stalkers often tend to latch on to female erotica authors, confusing the fantasies we create with the realities of who we are. Most are just creepy cranks, but we've all read about people whose obsessions become dangerous. Keeping your real identity secure can help reduce the potential danger.

Then there's the risk to your career of having your email hacked or your website or blog defaced. The former seems to happen almost daily on the reader lists where I hang out. I'll see a message from the email address of some author I know, advertising cheap iPads, posting links to Russian dating sites, or pleading for money because her passport has been stolen in a foreign city. Of course, it's usually clear that these weren't sent by the real author – but can readers always tell? A day or two later these authors always pop in and apologize. Even if nobody gets fooled, one can't help feeling hacked authors should have been more careful.

So what can you do if you really want to keep your author and real world identities separate, make things harder for stalkers and trolls, and save yourself from the embarrassment (or worse) of being hacked? To be bluntly honest, nothing can guarantee your on-line safety. However, there are a number of actions you can take to reduce your vulnerability.

Don't use free email services like Hotmail, Yahoo or Gmail

Yes, I know. Most of us don't make much money from our writing. Free is synonymous with “good”. However, hackers know how to exploit the weaknesses of these large systems. Microsoft, Yahoo and Google are constantly playing catch-up as the bad guys break in and harvest hundreds or thousands of email addresses every year.

Not only can hackers break in to steal your email address and password, they can suck up your contact list, too, when that list is hosted on a public server. Then everybody you know may receive messages from you with embarrassing content, dodgy links, or dangerous attachments.

An email address “@lisabetsarai.com” is considerably less likely to be hacked than a Hotmail, Yahoo or Gmail account.

Aside from security concerns, using free services can make you look less professional. For ten

or fifteen dollars per year, you can get yourself your own domain name, linked to your pseudonym or some other aspect of your writerly persona. You don't need a website to use your domain for email. Many web hosting companies will “park” your domain and give you half a dozen email addresses associated with that domain for five or ten bucks additional annual cost.

Create different email addresses for different purposes

You may want to have one address for communications with editors, for example, another for communication with readers, and yet a third for commenting on blogs. Multiple email addresses do complicate things, but on the other hand, if one address is compromised, you don't have as much work to do. Having several addresses also makes it easier to filter your email based on the “To” field.

Another advantage of having your own domain for email is that you can create separate, special purpose addresses that still reflect your branding. For instance, when I put together my call for submissions for *Coming Together: In Vein*, I created a special email address, still “@lisabetsarai.com”, to receive submissions.

Don't publish your email in text form anywhere

When I say “in text form”, I mean, don't ever include a string like “myname@mydomain.com” anywhere. It's relatively easy for software to recognize the patterns of email addresses. (In fact my word processing program automatically turned the fake email in the previous sentence into a link!)

It's not at all difficult to write a program that “crawls” the web, looking for email addresses. These programs, sometimes called “spiders” or “bots”, are very common.

Remember how the web works? You send a request with a URL to the web server, and it responds, sending back the data for the web page to be displayed. Spiders basically impersonate users sitting at browsers. They send URLS, grab the text returned, and then go looking in that text for email addresses (or other information). (They also recognize links, so that they can send out new URLs.)

The collected email addresses get sold to Internet marketers (who will then send spam to your address) or to criminal rings (who may use your address as the apparent source of emails that carry viruses or other malware).

But if you don't publish your email, how will readers communicate with you?

There are several options. Some people simply take the “@” and replace it with the word “at”, e.g.”myname [at] mydomain.com”. I wouldn't be surprised is some spiders are now looking for this

pattern as well.

Another possibility, one that I use on my links page (http://www.lisabetsarai.com/links.html) is to create an image showing your email address.

A third strategy is to not publish your email address at all, but to include a contact form on your web page or blog. This requires a simple server-side script (remember active content?) that gathers information from your reader and then, behind the scenes, sends you email. It's true that your email address in text form will be embedded in the script itself, but script files are somewhat more difficult for spiders to access and analyze. You don't have to write this script yourself. Most hosting services can provide one for you, that you can customize.

Purchase domain privacy services

If you have your own domain, go to the following URL:

http://www.networksolutions.com/whois

Type your domain name into the text box at the top and click "Search". What do you see?

The domain registry companies maintain ***a public database*** that lists the name, address, phone and email of the people associated with each domain. Thus, your contact information could be exposed to anyone who cares to look it up.

This is intended to be beneficial, to allow consumers, law enforcement and other affected parties to discover who is responsible for web content. However, for an author trying to remain anonymous, this can spell disaster (or at least, disclosure).

Many web hosting/domain registration companies sell a service called "domain privacy". When you purchase this service, your details will not appear in the WhoIs listing. It's still possible for people to ferret it out, but it takes a good deal more work. My hosting company charges $10 per year for this service. I feel this is worth paying – and of course, it's tax deductible.

Remember that everything is public

You leave a trail of evidence as you traverse the web, engaged in authorly pursuits. Your comments on Facebook, the photos you post on your blog, the reviews you write for Goodreads, even your Amazon purchasing history, may be visible to others. Legitimately visible, I might add. When

you sign up for many social networks and eCommerce platforms, you have to accept their privacy policies. In many cases you're agreeing to expose more than you may realize. Furthermore, most privacy policies reserve the right to make changes at any time. And the fine print always says that they're not legally responsible for unintentional breaches to their systems (although there's new legislation being considered in many countries to change this).

It used to be that you could rely on the size and diversity of the Web to cover your tracks. With all the millions of blogs out there, who was going to realize that you posted one picture here, a similar one there, mentioned your birth place in a third location, noted your birth date in a fourth? Nobody was going to look at all these different posts and put that information together to identify you.

Now, though, companies are spending literally millions of dollars to connect all these dots, to build an integrated picture of who you are, where you live, what you like and dislike, how much money you make, and much, much more. These so-called data mining efforts are, I believe, more of a threat to privacy than the villains trying to steal credit card data or social security numbers.

A vigorous effort is underway to integrate and unify social networking and eCommerce platforms. For instance, when I post a review on Goodreads, I'm then invited to share it on Facebook, Google+ and Twitter. How convenient! This makes everything easier for the users! But every time you take advantage of one of these convenient cross-linkages, you're adding more data to that personal profile companies are building. If you're an author trying to hide her real identity, you're increasing the risk that you'll somehow expose the relationship between your pseudonym and your real name.

The linkages don't have to be explicit. The software developed by the data mining companies can infer these connections by matching various items of data. They're getting better at it all the time. Meanwhile, even pictures can reveal information you'd rather hide. Facebook just bought a company that can scour thousands pictures posted online and find all the faces that match a target face - in a matter of minutes.

If you have one Facebook account for your real identity, and another for your author identity, it's only a matter of time before Facebook figures out that you're the same person. Maybe that doesn't bother you, but you should be aware of the risk. Do you really want your grandmother to hear about the BDSM ménage story you just published?

Every time you include some real world fact about yourself in an author blog post, consider whether you really need to share that item of data. I don't publish pictures of myself (other than my head shot, which honestly doesn't look much like the real me), or photos of my husband or family, or

my home, or my cats... In fact, I don't even publish my cats real names. They have pseudonyms just like I do.

You may think I'm being paranoid. Perhaps. But I'd rather be paranoid than knowingly risk exposure.

Use deliberate misdirection

As an author, you may want to keep your personal information private. But many readers sincerely would like to know you better. They get a kick out of communicating with their favorite writers and learning about the details of those writers' lives. If you're too stingy with your personal data, readers may come to feel that you're insincere or stuck up.

One solution I've found is to employ a certain amount of deliberate misdirection in my blog posts and other public communications as Lisabet Sarai. I alter the facts I share, just a bit. I give the experiences I recount a twist, change the locations or the times, make up a subplot or a few characters. I don't view this as dishonest - I'm terrible at lying - but rather, as a bit of fictional embroidery. My job, after all, is to tell stories. Meanwhile, every time I slant the truth, I'm helping to lead the data snoops astray.

Don't piss people off

Online, you never know who's paying attention to you. It pays to think long and hard before you post anything negative, about anyone. This is true in general, but especially when you're wearing your author hat. It's all too easy to take revenge on the 'Net. Piss someone off and you may motivate him to do some detective work, trying to uncover your real identity. And despite all your care, he just might succeed.

Rants may be emotionally satisfying - especially when they're directed at individuals or organizations that are obviously evil and wrong...! You may be right in believing the object of your criticism deserves every word. However, ask yourself whether the catharsis is worth the risk.

Consider the privacy of others

As you work to defend your own identity and privacy, remember that we all face the same threats. Don't share private information about colleagues without getting express permission. I know it's difficult. Industry gossip is one of the most enjoyable aspects of getting together with one's author

colleagues. It's so tempting to tell your great stories about the big names you might have met - but hold back. Those stories belong to the protagonists, not to you.

One simple action you can take to enhance everyone's privacy is to use blind CC's (BCC) when you're sending a message to a list of email addresses. You don't have the right to expose anyone else's email to the world - and that includes other authors. Readers, too. If you have a reader email list, be sure that you keep each member's identity private.

Actually, even storing emails on your hard drive is something of a privacy risk. Many viruses have the ability read your address book and then forward their evil payload to every single person listed there. Unfortunately, it's nearly impossible to function these days without a list of contacts that integrates with your email client. About the only thing you can do is switch to a client that's less susceptible to attacks.

Conclusion

You may get the feeling from my comments above that I'm pretty pessimistic about privacy issues. You're right. The forces trying to tear down the walls that protect your anonymity are powerful and well-funded. There may come a time, in the not too distant future, when pseudonyms will be totally transparent, despite our intentions.

Until that time, though, you can reduce the likelihood of disclosing information you want to keep private by following the recommendations in this chapter. At least let's make the hackers, the trolls and the data snoops work a bit harder.

Social Butterfly:
Databases, Graphs and Connection-based Marketing

"You're nobody till somebody friends you..."

That could be the theme song of the twenty first century. The explosion of social networking has transformed entertainment, commerce, and most certainly, publishing. To someone (like me!) who remembers the world BF (Before Facebook) the proliferation of social networking platforms can be bewildering and disturbing. However, the revealed wisdom from the MP's (Media Pundits) states that you cannot succeed as an author without a well-orchestrated plan for marketing yourself through Facebook, Twitter, Google+, Triberr, LinkedIn, Goodreads, and so on *ad nauseum*.

This so-called truth generates huge amounts of anxiety in many authors I know. One of my objectives in this chapter is to allay some of that anxiety. Another is to give you an idea of how social network platforms work. In fact, if you've been following this book so far, you already know most of the basic principles. I'll be talking about a couple of additional technical concepts on which social networks depend. Finally, I want to suggest that core concepts associated with social media marketing do not depend on today's platforms. You can apply what I call connection-based marketing even if you're as Facebook-phobic as I am.

Let me start with my own revealed truth: the purpose of social networks is to make money for the people who run them. This money can come from selling ads to sponsors or services to members. It can come from selling data about members to third parties. The money may also come from sources (for instance, investors) who view the attention of a large group of members – the "eyeballs", in media parlance – as a pool for future sales of products or services.

Facebook may make it possible for families to keep in closer touch, sharing photos, news, and love over long distances Twitter may help emergency responders find affected individuals in an earthquake or assist political activists in opposing what they perceive as tyranny. There is no question that social networking can have social benefits. Just don't forget: that's not the fundamental objective, at least for the big, for-profit sites. At the end of the day, it's all about money.

Furthermore, the more members a network has, and the more active they are, the larger the potential payoff.

I will leave you to work out the implications on your own. Let's move on to the tech stuff.

How Do Social Networks Work?

At the most basic level, Facebook is nothing more than a very complicated web site. Like any other web site, it's based on HTTP and HTML – and you already know how they work! Of course, social network sites involve huge amounts of active content. The specific approaches used will vary, but most of the capabilities of FB and similar sites depend on server-side programs. That is, most of the computations are happening somewhere inside FB's computers, which create new content and send it back to your browser. The responsive user interface, which pops up a dialog when your mouse moves over a certain area, or provides a list of possible completions when you start to type in a search string, most likely depends on Javascript programs which run inside your browser and manipulate the appearance of the page.

What distinguishes a social network from other web applications? The core idea behind social networking is the notion of connections between users - “friending”. Much of the appeal of social networking derives from these connections. When you add some information to your page, or engage in other activities on the site, people whom you have friended may be notified – and vice versa. You can also in many cases add information or record your opinions on the pages belonging to your friends. You can share all sorts of material, including images, videos or links, and you can make recommendations to the people with whom you are connected. The possibly interactions aren't limited by time and distance, and in fact go far beyond the capabilities of face-to-face conversation.

Furthermore, the site itself constantly suggests new users whom you might want to add as your friends. These are frequently individuals who have some indirect relationship with you, via a common friend, workplace, educational institution, and so on. Sometimes these suggestions will “magically” unearth someone from your past, someone you haven't heard from in decades. Since having a large number of friends is often viewed as a measure of status or popularity, members are quite likely to act on at least some of these suggestions.

In order to handle these social functions, social networking platforms rely on two types of computational structures: relational data bases and graphs.

Relational Data Bases

Even if you're seriously tech-challenged, you probably have an intuitive idea about data bases. A data base is simply a collection of information, stored in some persistent form. Your address book could be considered to be a data base. For our Oh Get a Grip blog, we have a data base of previous

topics with the year we used them, so that we can avoid repetitions.

Many authors, including me, maintain a spreadsheet to keep track of work we've submitted. These are data bases, too. Mine has columns labeled *Title*, *Submitted To*, *Submission Date*, *Status*, and *Publication Date*.

Many data bases are like my spreadsheet. They can viewed as a set of rows, each of which represents an example of a particular sort of object or thing – in computer jargon, an ***entity***. Each column in the row stores the value of some property or attribute of the entity, so a row in some sense "describes" a specific entity. In my submissions data base, each row is a "submission", that is, an event in which I sent a book or story to someone in an attempt to get it published. As I get news about a particular submission (for instance, if a story is rejected), I try to remember to update the relevant column in that row.

Social network applications (as well as e-commerce sites and many other types of computer programs) use a somewhat more complex variant on this scheme, called a ***relational data base*** (or RDB). As suggested by the name, relational data bases don't just store information about entities. They also keep track of relationships between entities, especially entities that represent different kinds of objects or concepts. So usually a relational data base has more than one set of rows (more than one ***table***, in RDB terminology). Relationships are represented by matching up information in the different tables.

Let's consider an example. Suppose we have a table of information about authors. (I apologize for making up some of the information below about my good friends Ashley Lister and M. Christian...!)

AuthorID	Name	Nationality	FirstPubbed	BirthYear
A1000	Lisabet Sarai	American	1999	1953
A1001	Ashley Lister	British	1998	1959
A1002	M. Christian	American	2001	1962
A1003	….			

This is fine, but we'd really like to know more about these authors. For example, what genres do they write? We can create a second table (let's call it ***AuthorGenre***) to store that information.

AuthorID	Genre
A1000	Erotica
A1000	GLBT
A1000	BDSM
A1000	Paranormal
A1000	ScienceFiction
A1000	Romance
A1001	Erotica
A1001	Horror
A1001	Nonfiction
A1001	Humor
A1001	ScienceFiction
A1001	Mystery
A1002	BDSM
A1002	Horror
A1002	GLBT
A1002	SocialCommentary
A1002	LiteraryFiction
…	…

Why don't we just store the genre information as part of the ***Author*** table? Mainly because many authors write in more than one genre. We could create columns named *Genre1, Genre2,* etc. but how many slots should we provide? Storing the relationship between author and genre in a separate table allows us to have an unlimited number of genres for any author. Furthermore, we can add new genres at any time. Supposed that I wrote a novel targeted at young adults. We'd just add one more row to the ***AuthorGenre*** table, as follows.

AuthorID	Genre
A1000	YoungAdult

And why are we using these funny “author ID” strings like “A1000” instead of the author's name? The primary reason is that author names aren't guaranteed to be unique. There could be another author somewhere named “Lisabet Sarai”. (Although I hope not!) If we added her to our data base, she'd get a new row in the ***Authors*** table, with a brand new, unique *AuthorID* value. There will be no possibility that anyone will confuse her sweet romances with my steamy smut.

So what else might we want to store about our authors? Well, clearly it would be nice to know something about the books they've written. Let's add two new tables, ***Books*** and ***BookAuthor***.

BookID	Title	Year Published	Pages	Primary Genre	Secondary Genre	….
B1000	Raw Silk	1999	220	Erotica	BDSM	
B1001	Bodies of Light	2011	70	Romance	ScienceFiction	
B1002	Quarantine	2012	198	Romance	GLBT	
B1003	Swingers: Female Confidential	2010	167	Nonfiction		
B1004	Death by Fiction	2010	360	Mystery	Horror	
B1005	The Bachelor Machine	2003	270	Erotica	ScienceFiction	
B1006	Finger's Breadth	2011	284	Erotica	Horror	
B1007	Coming Together Presents: M. Christian	2010	210	Erotica	GLBT	
B1008	….					

BookID	AuthorID
B1000	A1000
B1001	A1000
B1002	A1000

BookID	AuthorID
B1003	A1001
B1004	A1001
B1005	A1003
B1006	A1003
B1007	A1003
B1007	A1000

The ***Books*** table records information about individual books. The ***BookAuthor*** table stores the relationship between books and authors.

Why don't we just store the authors in the book table? Well, many books have more than one author. Since I edited and wrote the introduction to *Coming Together Presents: M. Christia*n, I've put two rows in the ***BookAuthor*** table for that book, one linking the book to M. Christian, and one to me.

There's another, more fundamental reason for storing relationships separately, however. Relational data bases, if structured correctly, make it fairly easy to do complicated searches that combine information from multiple tables. In techie terms, this is called "querying the data base". For instance, I might want to know the names of all the authors in the data base who have written science fiction books. We could execute this query in several steps as follows:

- Find the *BookId* for every book in the ***Books*** table that has "ScienceFiction" as the primary or secondary genre.
- For each *BookId* returned from the first search, find the corresponding *AuthorId* values in the ***BookAuthor*** table.
- Look up each of these *AuthorId* values in the **Authors** table and print the *Name*.

Relational data bases provide the ability to make this kind of request in a single "question", using something called Structured Query Language, or SQL. In SQL, we could specify this query as follows:

```
Select Author.Name from Author where Author.AuthorID = BookAuthor.AuthorID
                                  and  BookAuthor.BookID = Books.BookID
                                  and  Books.PrimaryGenre = 'ScienceFiction' or
                                       Books.SecondaryGenre = 'ScienceFiction'
```

Don't worry too much if you don't follow this. The point is that by matching up the values of columns in different tables, we can pull out exactly the information we want from our four tables. For example, we could answer the following questions:

- Who wrote BDSM books that were published between 1999 and 2003?
- What is the longest (in terms of pages) book that was written by Ashley Lister?
- What genres do M. Christian and Lisabet Sarai have in common?

It turns out that this works much better if we store all relationships between our primary entities (Books, Authors and Genres) in separate tables from the entities themselves.

Of course, we could expand our data base by adding more tables. Suppose we added a ***Readers*** table, and then a ***BookReader*** table, that linked each reader to the books he or she had read. Then we could ask:

- Who has read BDSM books by Lisabet Sarai?

  For each reader R returned by the first query, we could then ask

- What BDSM books by Lisabet Sarai has reader R **not** read?

As you see, I've already used our data base to do targeted marketing, by discovering readers who seem to like BDSM and have sampled some of my work in that genre but not all of it. Imagine how much more intelligent our suggestions could be if we also knew what ratings the reader had assigned to each book she had read, or which books she had "liked".

I think you can see how relational data bases can support some of the features of social networks, like finding and suggesting other people who went to school with you (Facebook) or suggesting new books by authors you've already read (Goodreads). But what about the connections between people? How does Facebook figure out that I might know Bob, from the fact that Bob is friends with Alice and Alice is my friend?

We could store friend connections in a relational data base. We would have a ***Member*** table that assigned *PersonID*s to every member of the site, and then a ***Friends*** table that held pairs of *PersonID* values.

Member

PersonID	Name	… (more columns)
P00001	Bob	
P00002	Alice	
P00003	Fred	
P00004	Lisabet	
P00005	June	

Friends

Person1	Person2
P00001	P00002
P00001	P00004
P00002	P00004
P00002	P00005
P00005	P00003
….	…..

The ***Friends*** table would quickly become pretty enormous, however. If there are 1000 members of the network, there could be as many as 999,000 rows in the table. More important, however, is the fact that this structure makes it difficult (complicated and time consuming) to answer questions like the following:

- Who is friends with both Alice and June?
- Which of Lisabet's friends has friends who are not also friends of Lisabet?

Instead of using a relational database, this sort of network of connections is usually represented as a graph.

A Brief Introduction to Graphs

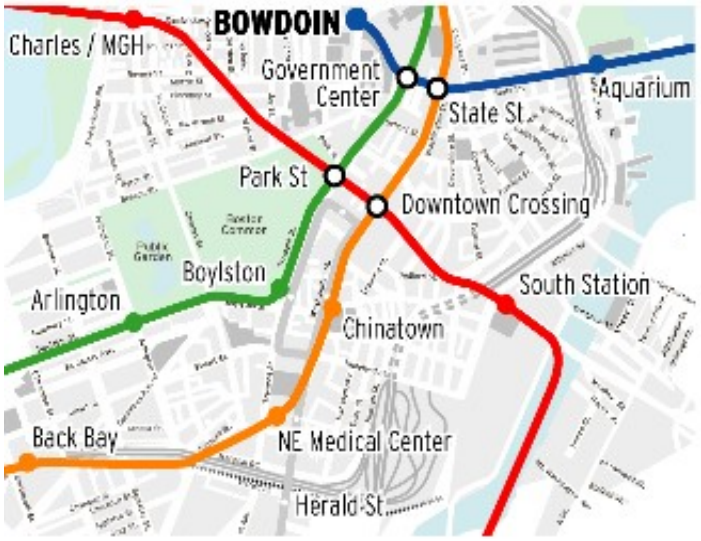
A graph is a structure for holding information about connections. A graph is composed of ***nodes***

(also called ***vertices***) and ***links*** (also called ***edges***). (I'll use the node/link terminology from now on.)

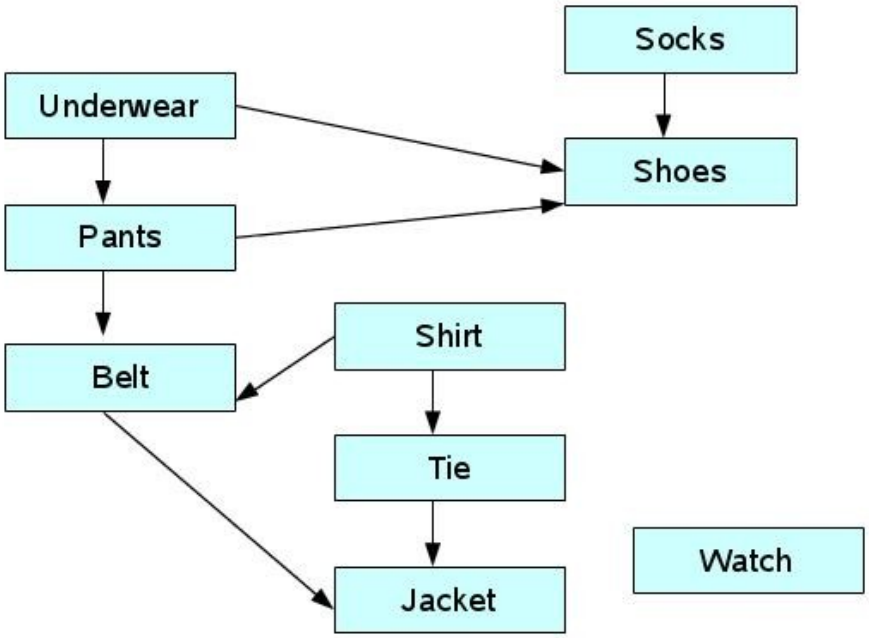
A node is an object, person, thing or concept. A link indicates a relationship or connection of some type between two nodes.

You could use a graph to represent a subway system, in which case the nodes would be subway stations and the links would be the tracks running between them.

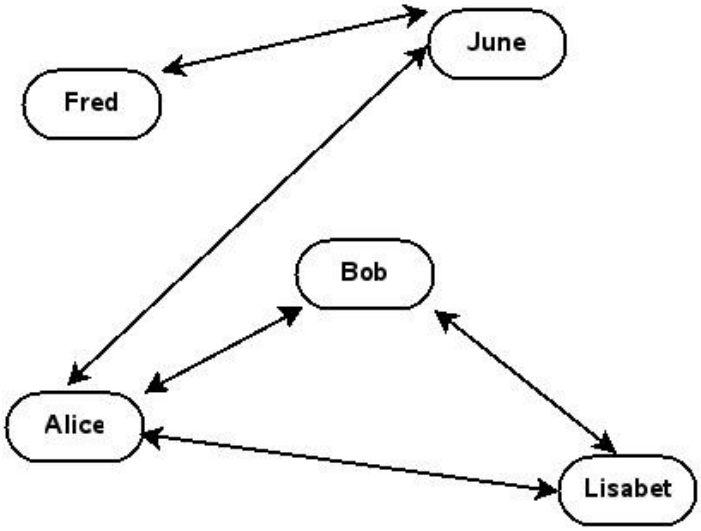
Note that junction stations, where you can switch from one subway line to another (e.g. Park Street, Downtown Crossing), will have more incoming/outgoing links than stations that belong to a single line (e.g. Arlington, South Station).



A graph might represent something more abstract, like the steps in some process. The graph below represents the dependencies a guy faces when getting dressed.



The nodes are the various items of clothing (which are also stages in getting dressed). The links (arrows) mean that our hero must put on the item at the start of the arrow before the item at the end of the arrow. If he dons his shoes before his pants, he's going to have problems! Notice that in this case, unlike the subway example, links are directional. Also there is more than one sequence of steps that will work; that is, there is more than one way to “traverse the graph”.



In a social network, the nodes are members. The links indicate friend relationships. The diagram below shows a graph that holds the same information as our ***Friends*** table in the previous section.

I've drawn the links as bidirectional. In most social networks, if Lisabet is Bob's friend, that means that Bob is also Lisabet's friend.

When we store graphs in a computer, we don't represent the relationships one by one, as we did in our ***Friends*** table. Instead, for each node (each member, in our social network), we store a list of all the other nodes that are adjacent – that is, directly connected – to that node. It turns out that this makes it much easier to follow a path through the set of connections. You don't need to know the details, but it is simple and efficient to start at one node (e.g. Lisabet) and find all the friends of her friends.

Real social networks probably use multiple sets of graphs for different kinds of relationships. They perhaps add numerical weights to the links that indicate the strength of a relationship, based on the frequency of interaction. For example, if Lisabet frequently visits Alice's page or comments on Alice's photos, but rarely interacts with Bob, the system might be more likely to suggest that Lisabet should add Alice's friends to her own set of connections, than Bob's friends. There are also computational techniques to identify mutually connected clusters in graphs, such as the group of Bob, Alice and Lisabet. A social network might use this kind of information to decide what email notifications to send to whom.

Marketing Using Social Networks

Why have social networks become important tools for marketing? I believe there are two reasons (other than hype):

1. The number of people who might potentially see your marketing material is very large – in the hundreds of millions.
2. The connections between individuals and the ease with which information can be shared means that one direct marketing impression may result in many indirect impressions. If someone sees your book cover, likes it, and sends the link to her friends, your marketing activity has a larger chance to make an impact on your popularity and sales.

Social networks facilitate connection-based or referral-based marketing. They can be a highly efficient method for diffusing information about you and your work. However, to market effectively using social media, you need to attract the attention of the right people in the first place: people who might like your books and whose friends are also target readers.

Ultimately, there's nothing special about social network platforms that helps with this critical point. If anything, all the random noise on Facebook – the billions of transactions that occur daily – may interfere with your message. You need to focus your own efforts on the readers most likely to buy your books and then spread the word. But how do you find them?

If I knew the answer to this question, I'd be a lot more popular than I am. However, I believe that you can use the concept of connection-based marketing outside of the social networking environment. The basic message needs to be: *if you like my books, tell your friends*.

I blog frequently, both at Beyond Romance and at as a guest at other blogs. There's a cadre of readers who follow me around, leaving comments – especially when I'm running a contest! They've been known to rave about my books, and of course, I'm delighted when they do so, because the other readers see their opinions and consider giving my books a try.

(I had a reader comment recently: “If you published your Grocery List I'd buy because that's how talented a Writer you are.” Now how do I find more readers like her?!)

I do at least two or three giveaways a month, where the prize is some title from my back list. I've tried contests where you get extra credit for getting someone else to enter. I'm always hoping to spark the interest of a new reader. And I do add fans, slowly, but surely, even though I have neither a personal nor a fan page on Facebook.

There's another reason I haven't invested much time or energy in social media. Today's hot site can easily become tomorrow's digital ghost town. My abandoned page on MySpace is testimony to the fickleness of the crowd. After its disastrous IPO, there have been some indications that Facebook is losing popularity. What will be the next big thing? I won't venture to predict that. However, I'm certain that whatever it is, there will be plenty of self-styled experts exhorting us poor harried authors to jump onto the band wagon.

It's up to you to decide whether or not to listen. Either way, I hope that this chapter had helped demystify the technology and made you realize there's no magic going on – just the usual bits.

Head in the Clouds, or,
Even a Submissive Might Think Twice

If you're technologically-challenged, take heart. According to the digital diviners, in the relatively near future, personal computers as we know them will disappear. No more confusing software installation! No more frustrating crashes and lock ups! The only program you'll need is your web browser and you can run that on your tablet or telephone.

Pythian pundits predict that very soon, all the applications you need will be provided on an as-needed basis by “the cloud”. You'll access the cloud to write your stories, to handle your edits, to store your author's copies and covers, to keep track of your royalties, to interact with your readers. You'll never have to buy or install any new programs, though there might be small fees for some of the services you use. You'll never run out of memory or disk space. In the brave new cloud-based world, the computing power and data storage available to you will be unlimited, and you'll pay for only what you need. No software obsolescence or upgrades! No inter-application incompatibilities! No hours on the phone with tech support! Everything will be clean, simple, handled for you by your cloud service provider. You can concentrate on your writing instead of on trouble shooting or system administration.

Does this sound too good to be true? Well, you can call me a curmudgeon, but these days I simply tune out these sorts of exaggerated claims. Cloud-based computing definitely is a current trend, one that has me somewhat worried to be honest. Each of you has to make your own decision, but it will be a long time before I'll give up ***my*** PC!

In this chapter, I'll explain a bit about the technology behind the cloud hype, and explore the positive and negative aspects of ditching your personal computing tools in favor of a cloud-based solution. To preview the punchline, it all comes down to control. How much control are you willing to relinquish in return for convenience?

Load Balancing, Virtual Machines and Dynamic Resource Allocation

Even if you've never heard the term “cloud computing”, you've probably used cloud-based applications. Blogger, Google Docs, Picasa, Yahoo Calendar, DropBox, all run in the cloud. But aren't those just active web applications? you might well ask (if you've been following the previous chapters, that is....). What's special about cloud computing?

When I explained how the web works, I told you that your browser sends HTTP requests to a

web server, whose location in the Internet is specified by the URL in the request. The web server gathers the data necessary to respond to your request (possibly by querying a data base or running some programs) and then returns that information to you in its response.

This simple request-response paradigm is still what underlies the cloud. However, a cloud consists of many servers (hundreds, possibly even thousands). Your request URL doesn't really identify a single server. Instead, management software will allocate your request to whichever server happens to have the computational capacity to handle it at the current moment. This process is called ***load balancing***. Load balancing allows a large web site to use its resources most efficiently, and helps keep the time to respond to your request low.

This type of multi-server configuration, where requests are directed to the most appropriate one of many servers, is sometimes called a ***server farm***. Server farms have been in common use for more than a decade. Cloud configurations take this idea one step further. There are still many real servers to handle requests. However, each physical machine (each computer that acts as a server) may be running multiple ***virtual machines***.

A virtual machine (VM) is a set of software processes that simulates the functionality of a physical computer. It has virtual disks, the contents of which might be stored on the physical computer's hard disks or in memory (so you can have more virtual disk space than you have real space). It has a simulated keyboard (which will probably receive input from the real keyboard), a simulated graphics screen, and so on.

Each VM will run an operating system (like Windows or Linux). Additional programs can then execute within that operating system. Virtual machines allow you to run different operating systems, or multiple copies of the same operating system, on the same computer, at the same time, without any interactions. Basically, a VM encapsulates an entire computing environment, insulating it from the physical machine as well as other concurrently executing virtual machines.

This encapsulation property provides many benefits. You can create a VM that automatically reverts to a previous baseline state whenever you shut it down. This protects the VM from the enduring effects of viruses and other malware. You can have your VM remember its state under normal circumstances, but roll it back manually (for example, if you install some demonstration software and then decide you don't want to keep it).

A VM can create “snapshots” of itself while it is running, basically copying its entire state into a persistent form as disk files. Because of this snapshot or ***checkpoint*** capability, you can easily move a

VM from one physical machine to another – even without shutting it down. You simply checkpoint the VM and ***suspend*** it – that is, temporarily freeze the processing. Copy the checkpoint files to a different physical machine, ***resume*** the VM, and it will continue right where it left off, blissfully unaware that it is on a different host.

A cloud configuration may create a virtual machine to handle your specific requests, and then move that machine to another physical server if the present server becomes overloaded. In the cloud, the allocation of computations to physical hardware becomes extremely flexible and dynamic. Furthermore, hardware resources can be added to or removed from the physical cloud infrastructure transparently, without having any effect at all on the processes that are executing within that infrastructure.

Companies that provide computational services like creating cloud configurations because a cloud helps them deliver good quality of service to a varying number of users with a minimum of resources. A cloud is more efficient than a server farm with the same number of physical computers because the virtual machine concept allows load balancing to occur at a finer level of granularity.

Cloud Computing Business Model

The basic business model for the cloud is sometimes called “software as a service” (SaS). From the user's point of view, this means that all computational capabilities are accessed via a web browser, with no need to buy or install any local programs. The software provider takes responsibility for storing any application-related data and for delivering the functionality the user needs via a web application. Usually the user will pay an access fee, which may be based on time, bandwidth or data storage required, and (for business users), number of people who will access the service. Some SaS applications (e.g. Google Docs) are offered for free. “Free” services generally support themselves by presenting advertisements or gathering personal data that can be used for future marketing.

A number of large companies, including Amazon, Google, and IBM, allow customers to “rent” cloud capabilities in order to configure and manage their own applications. Businesses, in particular, are attracted by the notion of getting flexible computing resources while decreasing or eliminating costs for purchasing hardware and software, and for paying system administrators and other IT personnel. This aspect of the cloud business model is not likely to be relevant to individual authors.

What Can Authors Get from the Cloud?

Some of the web-based applications you use on a daily basis may be cloud-based, without your even realizing it. There are a variety of cloud-based services an author might find useful.

Data storage and backup. Services such as DropBox, Google Drive, and Carbonite will keep your files stored "in the cloud". Some of these services also provide automated backup and synchronization capabilities. Cloud based storage can allow you to access your WIP anywhere there's Internet connectivity. In addition, it's extremely unlikely you'll ever run out of space.

Document creation and editing. Google Docs is an example of a free cloud-based office suite that lets you create text documents, spreadsheets, presentations and so on. Microsoft Office 365 provides similar capabilities, for a fee. Either one (and there are probably additional alternatives) eliminates the need to install Microsoft Office or OpenOffice on a local computer.

Accounting and other financial functions. Most authors I know don't need much more than a spreadsheet to keep track of their earnings and expenses, but if you happen to be at the heart of a lucrative publishing empire, you can find both generic and customizable accounting applications in the cloud. Salesforce.com is a long-time leader in this area.

Mailing list and bulk email management. ConstantContact and YMLP are two bulk mail services I've personally investigated for dealing with newsletters, contest announcements and so on. Both provide varying levels of service (i.e. different numbers of messages or contacts) for a monthly fee.

Photo storage and editing. Photo editing in the cloud appears to be the latest hot application for SaS computing. Even Adobe, the company that sells Photoshop, is getting into the act with their Carousel photo editing service.

Blogging and Social Networking. Most platforms for blogging and social networking are probably based on a cloud configuration now, or will be very soon.

Pros and Cons of Working in the Cloud

As you've probably realized by now, I take a rather jaundiced view of the hottest new computing trends. I've been in the tech biz long enough to know that many supposedly innovative concepts are really just old wine in new bottles. I also know that the overarching goal of every commercial company is to make money – not to "improve the user experience", or "serve the customers' needs", or any other such nonsense. Thus, I'm quite skeptical about cloud computing.

Nevertheless, there are some obvious advantages to moving your computational tasks into the cloud.

Ubiquity. With cloud computing, you don't need to worry whether you copied your current WIP onto your laptop before you left home, or scratch your head trying to figure out whether the version on your thumb drive is the most recent one or not. You can work with your files anywhere, moving seamlessly from one location to another. For someone who travels a lot, this can be a big plus.

Economy. If you commit to working in the cloud, you may save money on both software and hardware.

Convenience. Many of you might breathe a huge sigh of relief at the notion of not having to deal with all the daily hassles of managing one's own computer. I can sympathize. If you don't have a lot of IT knowledge, working in the cloud may free you from the administrative tasks even the simplest home computing environment entails, and give you more time to write.

Functionality. Sometimes a cloud-based solution may provide powerful capabilities that just aren't available in your local software. I recently signed on with YMLP for just that reason. The service allows me to pre-schedule mailings, track message opens and link clicks, and filter out duplicate email addresses, all nice features I can't get from my (very capable) email client. On the other hand, I'm still maintaining my address lists in parallel on my own PC, just to be safe!

Ranged against these acknowledged advantages are several serious risks.

Internet dependence. If you use the cloud to store your manuscripts or actually do your writing, what happens when the Internet isn't available? Many of us have gotten accustomed to unlimited connectivity, all the time, but disasters and emergencies or equipment failure can interrupt connectivity for hours or days – possibly preventing you from writing.

Service reliability. Even if you're up and running on the Internet, your service provider might not be. In June 2012, one of Amazon Cloud Service's major facilities went down due to intense storms. High profile web services such as Netflix, Pinterest and Instagram went down with it. Indeed, that incident provided a graphic demonstration of how much popular web functionality already depends on the cloud.

Service longevity. What if the cloud company you depend on goes out of business? What will happen to your data? Does the company have any liability? Can you sue them if you lose your 400 page breakout novel because of their financial incompetence?

Security. Cloud advocates often claim that web-based software execution is more secure than

personal computer software, because personal computers are more susceptible to malware. However, as far as I know, this claim has yet to be demonstrated empirically. Furthermore, the more popular cloud software becomes, the more it will become a target for malware developers, especially since the payoff for hacking a cloud could be huge. It's true that virtual machines provide encapsulated environments, but configuration errors could still make your private data vulnerable.

Control. The latest version of a software package is not always the best. When you have your own computer, you can choose to ignore upgrades in favor of running your familiar tools, rather than having to learn an entirely new version, with a different (and possibly horrible) user interface. For example, I still use Adobe Acrobat Version 6 because Version 7 removed a critical feature I really needed. I have a good friend who's still running Windows 2000 because he has a whole suite of utilities that aren't supported by later versions.

When you opt for computing in the cloud, you lose that control. If your software provider decides to provide a new and improved version of their offerings, you're usually stuck with accepting it. Google's recent update of the Blogger platform is a case in point. Although I've discovered a number of excellent new capabilities, it took me weeks to learn the ins and outs of the new post editor.

It's Like Dominance and Submission...

Ultimately, you have to decide what you trust more – your own (possibly minimal) computer skills, or those of a faceless corporation. I know my answer. I'm keeping control of most of my data and computational functions, for the moment at least. Though surrender might be sweet, I'm not convinced that the cloud folks will do a better job at managing my data and my writing than I can do myself.

Your answer may be different than mine, but make a decision with your eyes open. Don't let the seductive promises of the cloud vendors blind you to the risks.

App-y Together:
Straight Talk about Mobile Madness

Do you need an author app?

Do you have any idea what I'm talking about?

The ecology of computing has changed dramatically in the past two or three years. Touch-screen cell phones and tablets have taken the world by storm. Sales of notebook and desktop computers have stagnated as consumers opt for convenient, trendy mobile devices that promise constant access to information, entertainment and productivity tools.

If you're a technologically-challenged author, you may well wonder how you're going to deal with yet another mystifying and frustrating computing platform. Take heart. Almost everything I've discussed over the past ten months applies to mobile devices as well as to traditional personal computers. Although mobile applications look different than other kinds of software, they're based on the same principles. In this chapter, I'll talk about what distinguishes mobile devices from classic PCs as well as what the two platforms have in common. Then I'll consider the implications of mobility for your writing and marketing.

Power in Your Palm

Mobile computing devices have actually been around for quite a while. Personal Digital Assistants (PDAs) with touch screens and wireless connectivity became available in the early 1990s. These early “palmtops” didn't catch on, partly due to their high prices and limited functionality. The recent explosive growth of mobile devices has resulted from several converging factors:

- New mobile data transmission protocols;
- Widespread expansion of cellular phone networks;
- Sharp drops in the price of computer processors and memory;
- Fierce competition between device providers.

Mobile devices have many features in common with more familiar types of computers. Like PCs, mobile devices have a CPU (Central Processing Unit), dynamic memory, and persistent storage for files, as well as facilities for input (touch screen, microphone, accelerometer, etc.) and output (screen, speakers, etc.). Today's smart phones and tablets use gestures for interaction and thus offer an experience quite different from typical personal computers. However, as important as this is to users,

the difference is largely superficial. Under the hood, mobile applications look a lot like any other computer programs.

Like PCs, modern mobile devices use an ***operating system*** (e.g. Android, iOS or Windows Mobile) to control end-user software and manage resources such as memory and disk space. In many cases these operating systems are simply variants of a desktop OS. Android, for instance, is derived from Linux.

Smart phones and tablets derive much of their utility from the fact that they can connect to the Internet regardless of their location. However, most mobile applications use the same Internet protocols – especially HTTP – that are employed by more traditional software. Thus most of what you've learned from reading this book applies to mobile devices as well.

So why is everyone so excited about the so-called mobile revolution? What's so special about these gadgets? There are at least three critical features of mobile devices that distinguish them from traditional computers.

- **Mobile devices are always available and can be used under almost any circumstances.** The other day I watched a woman in a diner interacting with her iPad with one hand while eating strips of bacon with the other. I cringed, imagining what bacon grease might do to a touch screen, but the scene drove the point home. People take their phones and tablets everywhere.
- **Mobile devices are fundamentally multimedia platforms.** Every smart phone and tablet includes a camera, microphone, speaker, and a high quality color display. And every one of these devices can send and receive data over the Internet. Mobile devices are designed from the ground up for recording, displaying and exchanging video and audio content. While other computing platforms can handle multimedia, these capabilities are seamlessly integrated into mobile devices.
- **Most mobile devices are location-aware.** Global Positioning System (GPS) receivers allow them to “know” where they are on the face of the earth. Software can take advantage of this location information. Applications running on the device can “geotag” content gathered by the device. Server-based software communicating with the device can send content that is customized based on the device's current location.

The three features above make mobile devices an extremely attractive platform for advertising and marketing. Users (that is, customers) are more or less constantly accessible. Marketing material can be delivered using lively and engaging video and audio formats, and furthermore, can be tailored based

on where the user is currently located. Furthermore, software communicating with mobile devices can potentially gather detailed data on the user's patterns of movement and daily routines, which can be exploited to target advertising even more precisely.

Because of these characteristics, creating a mobile app to promote your brand, win customers and build loyalty has become *de rigueur.* But what, exactly, is an app?

Anatomy of an App

"App" is short for "application", which in turn is a fancy name for a computer program designed for end-users (as opposed to a program that is run by the operating system or that is intended mainly for technicians). On mobile devices, apps tend to be small programs that perform only a few functions. This is partly because the computational power of today's mobile devices is still quite limited compared to desktop or notebook computers.

Some apps provide the same essential capabilities available on desktop platforms: calendar, address book, messaging, memo pad, and so on. Games are another popular app category. The apps of most interest to marketers, however, involve access to and exchange of information over the Internet.

Most mobile devices ship with a web browser installed. This browser can be used to view web sites and interact with web applications like those we've considered in previous chapters. In many cases, however, browser-based interaction on a mobile device will be less than satisfactory. Web pages designed for desktop access are hard to read on the smaller screens provided by many mobile devices. Even the fastest mobile Internet is still considerably slower than a wired network, so web applications that transmit large amounts of data to the client will feel sluggish and unresponsive. The problem is compounded by the relatively low computational power of the processor chips used in mobile devices. The faster and more powerful a CPU, the more heat it will produce and the more electricity it will consume – both big problems for mobile computers.

Because of these characteristics of browser-based access, many organizations create specialized mobile apps for interacting with their web applications. These apps communicate via HTTP, just like a generic browser. However, they display information in a simpler format that is more appropriate for mobile screens. They also may be designed to minimize the amount of data exchanged via the 'Net.

A generic browser is deliberately isolated from the system on which it is running, in order to protect against malware. For instance, most browsers are not allowed to read or write files except in certain specific directories, or to trigger the execution of other programs. These restrictions do not

apply to mobile apps. Apps can create and use local databases, maintained in the mobile device's persistent storage, to hold information that would have to be transmitted from the server in a traditional browser-based configuration. Apps are also free to invoke other software residing on the mobile device.

Thus, apps provide the advantages of access to the World Wide Web, without some of the constraints of using a browser. The underlying communication model is still the same – the app sends an HTTP request and the web server sends a response. In some cases the web server may not even be aware that it is communicating with an app as opposed to a general-purpose browser.

There's one additional advantage provided by apps. An app can be designed not only to respond to requests from the mobile device (a “pull” model) but also to broadcast information to the device without a request (a “push” model). This kind of communication uses protocols other than HTTP and depends on the fact that each device has a unique identifier. The ability to send unsolicited information to devices has obvious benefit for advertisers but can be useful in other contexts as well (e.g. emergency notifications).

The big disadvantage of the app approach is the fact that someone has to create a new app for each web application. Furthermore, as server side capabilities change over time, it may be necessary to modify the app to handle differences in the server output. These issues do not arise with a generic browser.

Another problem stems from the fact that currently there are several popular mobile operating systems, most notably Apple's iOS and Google's Android. These two software environments are not compatible. Thus, in general it is necessary to create at least two different versions of every app, one for each platform. Microsoft Windows is trying to gain mobile market share and represents yet a third distinct environment that you might need to support.

Mobile Marketing for Authors

Who has apps these days? Magazines and newspapers create apps that provide access to articles and other content. Airline apps let passengers make reservations, view schedules, and track flight status. Restaurant chains apps let you locate the closest branch, view menus, order food and request delivery. Banks offer apps for reviewing accounts, making transfers and paying bills. Basically, anyone who has a web site might consider producing an app that offers a subset of that site's capabilities in a form appropriate to mobile devices. The “push” capabilities discussed in the previous section allow a business to notify app users about new products, special offers, or other events as they occur.

Some businesses employ mobile-only promotions to encourage use of their apps. For instance, ordering a pizza through an app might automatically entitle you to a discount. For many people, though, the convenience of using their mobile devices provides sufficient motivation, assuming that the app provides functionality that is already important to the user.

So what about authors? Most of us have web sites and/or blogs. Does it make sense to create an author app? And what would such an app do?

Here are some ideas:

- Allow readers to view your covers and read excerpts;
- Allow readers to order your books;
- Notify readers about new releases;
- Feed your blog content to users' devices;
- Allow readers to send and receive messages from you;
- Allow readers to communicate with other readers and fans.

An app with “push” capabilities might do more:

- Send promotional offers customized to readers' previous browsing or buying history;
- Send real-time video of events like readings or conferences;
- Suggest books based on the user's current location.

The possibilities are limited only on by your imagination – and of course by your budget. I'm assuming you're not going to write the app yourself, so you'll have to hire someone to do it for you. Then, of course, you'll need to keep the information provided by the app up-to-date, just as you need to continually update your web site and blog.

I believe there are companies that specialize in creating author apps. I know one very popular author who has such an app. She told me that it costs her a bit under $100 per month. If this sounds steep, remember that this price includes maintaining the core data on a server somewhere, as well as developing and distributing the mobile application itself.

Most of you are probably shaking your heads at this point, figuring you don't have the financial resources to join the mobile revolution. I sympathize. And how important is a mobile presence, anyway?

To try to get a handle on this question, I ran a contest. I asked readers to send me an email answering the following three questions:

1. Do you own and use a smart phone?

2. What kind? iPhone, Android phone, or something else?
3. If you have a smart phone, would you be interested in a free "author" app that let you read excerpts, get news, view covers and generally keep up with what an author is up to?

Each email counted as a contest entry.

I received twelve responses. Five respondents said they had smart phones. Seven did not. Even so, eight told me that they'd enjoy having such an app if it were available. That is, even some of the readers who didn't own smart phones liked the overall idea of an author app. One respondent told me she already had downloaded several such apps and waxed enthusiastic about the latest one she had acquired.

From this small and unscientific survey, I conclude that there may well be a demand right now for author apps. Of course, once a large number of authors do offer such apps, the utility of this approach for marketing will decline dramatically. I think there's a window of opportunity right now, for those of you who can afford to produce an app, but before too long, marketing will move on to the next big thing – whatever that turns out to be!

Don't despair. If you don't want to undertake the bother and expense of creating an app, you can still do something to capitalize on the mobile revolution. Review the design of your web site and blog with the objective of making them more mobile-friendly. Design for a page width no wider than 640 pixels. Reduce the number and size of images. Get rid of animations, videos and other content that requires high bandwidth. (Although mobile devices are optimized for multimedia, mobile browsers usually don't include the necessary plug-ins to handle this content.)

Then check out your site with your own smart phone – or borrow a friend's. If someone accesses your site from their mobile device, will they get your message? Or will they see garbage?

If you really love your fancy, image-laden site too much to change it, there's another simple option. Create a simplified version of the site targeted for mobile devices as a sub-area on your main site, and then provide a link on the home page. This is comparable to the old strategy (which most of you probably don't recall) of having a text-only version of a web site for slow browsers.

Conclusion

I don't own a smart phone. I have a small tablet that I use almost exclusively for reading ebooks; I don't tend to connect it to the Internet because I'm worried about being infected with malware. (Mobile security is a major problem that most users don't even think about.) I am not in any

sense at the forefront of the mobile revolution.

Nevertheless, authors need to be aware of the trend toward mobility and consider how they can exploit it in their marketing. You may not have the money or inclination to build yourself an app, but take some time to think about how you can benefit from the fact that these days your readers carry computers around everywhere they go. Perhaps you'll come up with some new and creative notions for mobile marketing. If you do – let me know!

The Scary Future

2014 is fast approaching, so I thought for this final chapter, I'd indulge in a bit of the prognostication that's so popular at this time of year. In the other chapters in this book, I've tried to demystify various aspects of today's information technology that are important to authors. In this one, I consider how the technologies of writing, publishing and marketing might develop in the future. My predictions are based mostly on intensification of current trends, this year's hot tech taken to its (in some cases absurd) limits.

Multimedia Mania

Words on a page are boring. Mind you, that's not my opinion – to me the words are wondrous and transparent, a window into new worlds – but I believe that many younger readers feel that way.

Today's text to speech technology makes it possible to “read” a book without ever looking at it. Over time I think books will become less and less about written words processed visually, and more about other modalities.

In the future, I expect that visual, auditory, perhaps even tactile, taste or olfactory stimuli, will be integrated into electronic books. Today's standard computing devices already have the capabilities necessary to deliver all but taste and smell. It won't be long until somebody adds those specialized effectors to the mix. (Research on such topics is already underway.)

The experience of “reading” such a multimedia book may be more akin to watching a movie, or maybe playing a video game, than to what we now call reading. I also expect that this experience will be highly customizable. A particular reader (like me) who prefers not to be distracted by audio/video or other ancillary inputs will (hopefully) be able to turn them off. A younger, more visual (less imaginative? But I'm betraying my prejudice here...) reader might accept every modality by default.

Most likely, such multimedia books will also provide hypertext capabilities, allowing a reader to choose his or her own path through the story, rather like Julio Cortazar's famous multi-threaded novel Hopscotch.

As an author, how do you conceptualize a non-sequential tale, in which events could occur in different orders, and the story might end at different points in the characters' lives?

I wonder, too, how this will change the practice of writing, quite aside from the conceptual issues. Someone who wants to produce a multimedia ebook will need new authoring tools, with the

ability to create non-verbal multimedia content and incorporate it into the final product. Alternatively, authors may purchase multimedia development services from companies or consultants who specialize in producing particular content modalities, just as you can hire someone now to create video trailers for your books.

And what will happen to the words? Will the verbal component of our books become abbreviated, as other senses take over the role of painting pictures for the reader? Will the words disappear entirely, as they become redundant to more direct sensory impressions? If they do – can we still call this artifact a book?

Hackers, Fakers, Fools and Pirates

I don't need a crystal ball to predict one unfortunate trend. Scams, plagiarism, identity theft (or mimicry), piracy, and malware will all become more common as writers and readers increasingly depend on digital formats and sources. I personally know one popular author who discovered someone else was using the same pen name as she employs, selling books with titles that are only slight variants of her own. She contacted the individual, asking the other author to cease and desist, with zero success. You can't copyright either a pen name or a book title, and clearly this copycat knew that.

I've already written about email hacks. I expect this to become far more sophisticated and targeted. Most current email hacks are somewhat random, simply taking advantage of weak spots to cull addresses for spam or criminal purposes. In the future, though, hackers will be able to focus on specific individuals – including authors. It will be possible, for instance, for an unscrupulous wannabee author to steal another author's reader list and use it for his own marketing. After all, wouldn't you like to have E.L. James' list of fans?

Fake reviews are already so prevalent that Amazon has begun deleting any review that even hints of a relationship between the author and the reviewer. (This means that Amazon has software that can analyze the semantic content of reviews – a scary thought in itself.) I'm certain this won't stop greedy or ambitious authors from trying to game the system. The digital marketing ecology utilizes sophisticated algorithms for ranking, linking, forwarding, and displaying information, but people who have a stake in cracking these algorithms in order to take (unfair?) advantage are equally sophisticated.

Viruses, trojans and botnets, already a huge problem on traditional computers, are now targeting new mobile platforms, which (unfortunately) have paid far too little attention to security. What might such malware do to hurt an author? Delete your books from the reader's device. Send messages

(perhaps insulting or threatening) through your author app, purporting to be from you. Make it impossible for your readers to contact you through your author app. I'm not even considering the general havoc that such invaders can produce, such as stealing sensitive data or spoofing to gain access to your bank accounts. And to be honest, I don't see any good solution to this problem, aside from constant vigilance on the part of every user (reader and author).

Then there's piracy. I don't believe there is a technological cure for the problem of piracy. Each time publishers deploy a new DRM (Digital Rights Management) scheme, pirates promptly crack it. Meanwhile, honest readers who buy legitimate content are inconvenienced by the constraints of DRM.

In short, the digitalization of publishing and reading has made everything a lot easier for the bad guys, and I foresee the situation getting a lot worse in the future. Be warned!

Pervasive Personalization

When you log in to Amazon or Barnes & Noble, the site suggests books you might enjoy, based on what you've previously viewed or ordered. When you search on Google, the links that get top position are filtered according to what you've looked at recently, while the advertisements that appear depend on your search history, your physical location, and a variety of other factors. I routinely get email notifications from Goodreads, telling me about new books by authors I've read or reviewed.

These are examples of ***personalization***, an attempt to predict your future interests and actions by analyzing information about you, which may be gathered from many sources. In the majority of cases, the goal of personalization is to increase the probability that you'll make a purchase, by offering products that match your needs or desires.

I predict that personalization will become more widespread, more intense, and yet, at the same time, more subtle. A book-selling site where you browse or purchase will base suggestions for new reads not only on superficial data like author, genre or tags, as it does now, but also using additional personal data such as the content of your reviews, what your friends have bought, the current season and current time of day where you're located, your gender, when you were born, where you live, your marital status, where you went on your last vacation, the fact that you recently had Lasik surgery or had a baby... How will the personalization system know so much about you? See the earlier chapter about social networks, which discusses how powerful forces are working to integrate your publicly posted information in order to build a consumer profile.

I am not being paranoid. There are companies that openly provide these data mining and

integration services (there's nothing illegal about it, at least not now) for other organizations. Digital behemoths like Amazon, Google and Facebook employ legions of PhD computer scientists who are working to make personalization more pervasive and more effective.

But personalization is a good thing, isn't it? If a site suggests books that are a good match to my interests, that will save me time and effort. When readers have liked books similar to the ones I write, don't I want the online bookstore to suggest they should purchase my books?

Aside from the privacy issues (which I've considered earlier), the problem with personalization is that it confirms and strengthens readers' existing preferences and prejudices rather than tempting them to try something new. A reader who purchases many paranormal erotic romances will be offered the opportunity to buy more of the same – she won't be confronted with my contemporary BDSM novel. Of course, she might not like my novel, given her past history – but how will she ever know, if her book-buying is limited to the books the site so conveniently offers her?

Some of the best books I've ever read were those I just happened to pick up while browsing in a physical book store. In some cases these books were wildly different from anything I'd read previously, by authors I'd never heard of. A personalization system would never have suggested them to me. (I sometimes find the books Amazon offers me amusing anyway. A recent check of my "recommendations" page included Gordon Dahlquist's *The Dark Volume*, *Professional Javascript for Web Developers*, and several Cleis anthologies in which I have stories.)

Furthermore, I see personalization of books potentially taking a new direction, given the probable future of electronic publishing technology. What if the **content** of books were personalized to suit the reader? Imagine multiple versions of a single erotic novel – the same characters and initial premise, but with different possible instantiations depending on the preferences of the reader. One purchaser with a history of buying happily-ending romance would get a version in which the characters end up together and married. Another, who in the past has purchased more edgy erotica, might see the hero leaving the heroine alone after they enjoyed a period of joint passion - having a psychiatric episode, or becoming an alcoholic, or going back to his first wife. The book delivery software would customize the book content to suit the reader – who might never know there were other stories, other endings, embedded in the book.

Think about writing that kind of customizable novel! The perfect thing for those of us who can't decide on a single genre! Write interwoven variants in multiple genres, and let the software assemble different books for different readers.

On the other hand, for authors who have difficulty keeping consistency and connectivity across a single narrative arc, this might be a nightmare.

You might view this level of personalization as far-fetched, but in fact it's only a small step beyond the notion of the hyperbook discussed earlier – and such books already exist. The fundamental difference is that the narrative path is determined by a personalization algorithm, rather than explicitly chosen by the reader.

Streaming, Serialization, and Real-time Reader/Writer Connections

The final trend I see becoming more important relates to the "mobile madness" I discussed in the previous chapter. Currently, when reader purchases an ebook, the content is transferred as a single file to his or her device (ereader, tablet, etc.). After that point, book resides in local storage on the device, until the reader removes it.

In the future, I expect that more ebook content will be ***streamed***, that is, delivered dynamically to readers in a "just in time" manner. For instance, when I finish one chapter, the book vendor will automatically send the next to my device. This is conceptually similar to streaming video or music content. The entire book never resides on the device. However, the streaming server remembers what each reader has seen already, so you can stop reading and then pick up where you left off.

Why would streaming ebooks be desirable? From the readers point of view, there are three possible advantages that I can see. First, streamed books might be easier and more convenient – like picking a radio channel as opposed to selecting a book from a very crowded shelf. Presumably a reader would be in the midst of perusing only a few books at any one time. Second, streaming would reduce the need for a large amount of local storage. Although both volatile memory and persistent storage continue to drop in price, the space required to store the kind of multimedia books I described earlier will be likely be hundreds of times larger than what's needed for today's text-based books. Third, I think that, like streamed music and movies, streamed books would be cheaper than books purchased via the download-and-own model. (They'd have to be, I think, to compete.)

Publishers, and advertisers, would love streaming (which is why I predict it will become popular relatively soon). For one thing, streaming reduces the risk of piracy. The complete book never leaves the server. It's simply delivered in pieces. So there's no single PDF or mobi file that can be archived and made available for download by the pirates. Of course, I'm sure someone would invent tools for capturing the stream and saving it – you can't stop pirates. But you can slow them down. In

additiona, the book stream offers an ideal medium for delivering advertisements. Streamed books might be priced cheaper for readers who accept ads.

Of course mobile network providers would love streaming ebooks, especially if they include multimedia. More bandwidth usage means higher profits.

What, if any benefits, would streaming provide for authors? One interesting possibility is a revival of the serial novel. Many of the classics in the canon (Dickens, Dumas, Conan Doyle, etc.) were written in installments which were published over a period of time. Serials were wildly popular. Readers waited eagerly for the next chapter, to see what would happen to the characters they'd been following.

Some current authors do serials, but usually as free content, since the current ebook publishing model doesn't support serials very well. It's inconvenient to purchase many small mini-books. Streaming ebooks, though, would be a perfect fit to serialization. Authors could capitalize on the excitement of readers awaiting the next episode, and could also realize revenue from the parts of the book they'd already written, before the book was complete.

On the other hand, serial publishing requires that an author be able to write “on demand” (unless, of course, the entire book is completed before the serialization begins). This could be a problem for some of us.

If you're anywhere close to my age, you've probably noticed how the world just keeps getting faster. The time between submission of a manuscript and its release has dropped from a year and a half when I first began publishing, to a matter of a few months or even weeks. Instead of writing one book a year, we need to write two, or three, or ten.

Given this universal acceleration, I think that the next step after streaming books will be real-time book creation. That is, readers would get the book in pieces – as the author was writing it!

Whoa! What about editing? Would this really work?

Not for multimedia content, but for text, for some authors – maybe. Consider the effect – a real-time connection between the author and her readers! How thrilling! Readers get to see the author's vision, fresh from her imagination. Obviously, the delivery system would capture the text for later, delayed delivery to readers who couldn't be present “at the creation” (and maybe for editing), but think about the intimacy of the experience. Real-time streaming of stories would move authorship into the realm of performance. Personally, I think this could be both popular and profitable. Furthermore, for authors who could pull it off, it would be a distinguishing factor, a way to stand out from the crowd.

Personally, I find this notion fascinating. Because the net effect would be to bring authors full circle to our origins – sitting around the campfire, surrounded by rapt listeners, telling stories.

Happy holidays to you all, and thanks for accompanying me on this digital journey!

About the Author

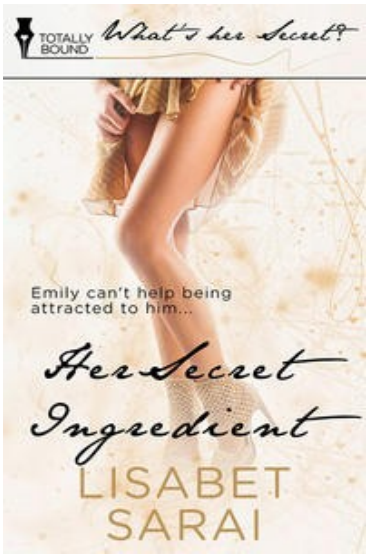
I became addicted to words at an early age. I began reading when I was four. I wrote my first story at five years old and my first poem at seven. Since then, I've written plays, tutorials, scholarly articles, marketing brochures, software specifications, self-help books, press releases, a five-hundred page dissertation, and of course, ***lots*** of erotica and erotic romance – nearly fifty single author titles, plus dozens of short stories in various erotic anthologies, including the Lambda winner *Where the Girls Are* and the IPPIE Best Erotic Book of 2011, *Carnal Machines*. My gay scifi erotic romance *Quarantine* won a Rainbow Awards 2012 Honorable Mention.

In addition to writing, I also edit erotica and erotic romance. My editing credits include the ground breaking anthology *Sacred Exchange*, which explores the spiritual aspects of BDSM relationships, the massive collection *Cream: The Best of the Erotica Readers and Writers Association,* the charity anthology *Coming Together: In Vein*, a collection of vampire tales that benefits Doctors Without Borders, and six volumes of the Coming Together: Presents series of single author charitable erotica books. You'll also find me writing the newsletter and occasional articles for the Erotica Readers and Writers Association (www.erotica-readers.com) and monthly reviews for Erotica Revealed (www.eroticarevealed.com).

I've always loved traveling; my husband seduced me in a Burmese restaurant by telling me tales of his foreign adventures. Since then I have visited every continent except Australia, although I still have a long travel wish list. Currently I live with him and our two exceptional felines in Southeast Asia, where I pursue an alternative career that is completely unrelated to my creative writing.

For more information about me and my writing, visit my website (http://www.lisabetsarai.com) or my blog Beyond Romance (http://lisabetsarai.blogspot.com)

Recent Releases



Stir in a pinch to stir up his passion

When the Tastes of France food channel offers Mei Lee “Emily” Wong a series of guest spots, she jumps at the opportunity to take her culinary career to a whole new level. Ultimately, she wants a show of her own, but first she has to prove herself to Michelin-starred network founder and effective dictator, Etienne Duvalier. A legend in the world of classic French cuisine as well as a domineering perfectionist, Etienne is sceptical about the culinary abilities of a woman from Hong Kong. To make things more difficult, the master chef is also so gorgeous that Emily can't help being attracted to him.

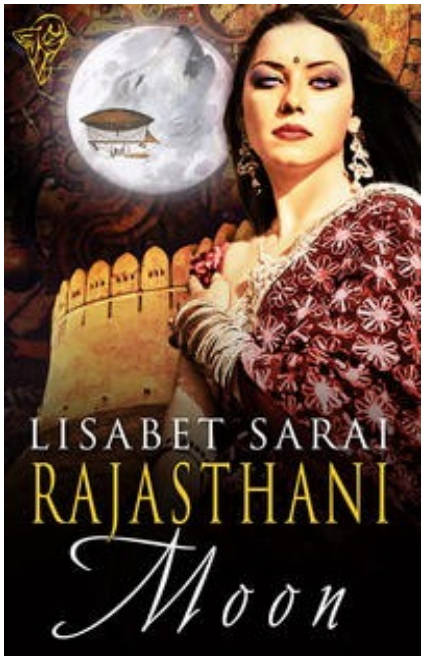
Emily tries to solve both problems by spiking her luscious profiteroles with an ancient Oriental aphrodisiac. Unfortunately, Harry Sanborne, the low-key, bespectacled producer for Emily's show, samples the delicacies she intends for Etienne's consumption. His powerful reaction to her secret ingredient comes as a pleasant surprise to them both. Harry turns out to be far more impressive in bed than on the set. However, he can't do nearly as much to advance her ambitions as Etienne. Emily tries once more to tempt the exacting M. Duvalier with her special cooking as well as her feminine charms. The outrageous results threaten to end her TV career forever - until Harry steps in to save her reputation and claim her heart.

Buy from Totally Bound:
https://www.totallybound.com/her-secret-ingredient

Buy from All Romance:
https://www.allromanceebooks.com/product-hersecretingredient-1348346-147.html

Buy from Amazon:
http://www.amazon.com/Her-Secret-Ingredient-Whats-ebook/dp/B00GOZU4EC/

More Recent Releases



Neither kink nor curse can stop a woman with a mission.

Cecily Harrowsmith, secret agent extraordinaire, is a woman on a mission. When the remote Indian kingdom of Rajasthan refused to remit its taxes to the Empire, Her Majesty imposed an embargo. Deprived of the energy-rich mineral viridium, essential for modern technology and development, Rajasthan was expected to quickly give in and resume its payments. Yet after three years, the rebellious principality still has not knuckled under. Cecily undertakes the difficult journey to that rugged, arid land in order to determine just how it has managed to survive, and if possible to convince the country to return to the Empire's embrace. Instead, she's taken captive by a brigand, who turns out to be the ruler's half-brother Pratan, and delivered into the hands of the sexy but sadistic Rajah Amir, who expertly mingles torture and delight in his interrogation of the voluptuous interloper.

Cursed before birth by Amir's jealous mother, Pratan changes to a ravening wolf whenever the moon is full. Cecily uncovers the counter-spell that can reverse the effects of the former queen's hex and tries to trade that information for her freedom. Drawn to the fierce wolf-man and sympathising with his suffering, she volunteers to serve as the sacrifice required by the ritual—offering her body to the beast. In return, the Rajah reveal Rajasthan's amazing secret source of energy. In the face of almost impossible odds, Cecily has accomplished the task entrusted to her by the Empire. But can she really bear to leave the virile half-brothers and their colourful land behind and return to the constraints of her life in England?

Review Snippet

This book really took me far and wide. At first you have a kidnapping fantasy with rip roaring sex that honestly left me breathless. Then you have a ménage with kinky toys that made me crunch ice. (A lot). Then you add a paranormal twist and I couldn't decide what I enjoyed more. It all worked. Every part of it. The hero was alpha male all the way, but had a soft side too and that made me fall for him too. The heroine was brave and adventurous and embraced every experience that came her way. I loved that. The action was amazing and the sex scenes sizzled through my e-reader. The other thing about this book was the steampunk elements that added spice to an already fun plot. Gadgets that tie you up by oral command, collars that kill and more await in this fast paced and stimulating read. ~ *Thistledown from Long and Short Reviews*

Buy from Totally Bound
https://www.totallybound.com/rajasthani-moon

Amazon.com
http://www.amazon.com/Rajastani-Moon-ebook/dp/B00D47R9E6/

Amazon.co.uk
http://www.amazon.co.uk/Rajastani-Moon-ebook/dp/B00D47R9E6/

Barnes and Noble
http://www.barnesandnoble.com/w/rajasthani-moon-lisabet-sarai/1115449469?ean=9781781843338

All Romance Ebooks
https://www.allromanceebooks.com/product-rajastanimoon-1216568-147.html