

# 1 What Is zembly?



widgets      Web 2.0      one-click      reuse      social  
casual      APIs      publishing           networking  
technologist      clone      **zembly**      services      collaboration  
Facebook      situational      mashups  
applications

Imagine writing web applications where your own creativity is the only limiting factor. The challenge becomes thinking up that killer application, not implementing it. No matter where you get your material (from web site content, API calls, or simply hooking one data stream to another), you can fashion that cool, customized widget or full-blown social application. If you can imagine it, you can build it. And if you can build it, you can deploy and scale it, too.

**zembly** is the first step to realizing this goal. Imagine a “programmable web,” where the browser is your development environment and your colleagues are thousands of perhaps unknown but similarly-minded developers. In the **zembly** environment, developing social applications in itself becomes social. Developers create building blocks that can be shared and reused. The rich variety of web APIs out there (the Web 2.0 promise) populates your palette and lets you construct widgets and applications that suit your immediate needs.

This book shows you how to use **zembly** to further this goal. You’ll learn how to create services and widgets and deploy applications with a one-touch Publish button (no deployment descriptors required). But before you start with menus and tabs, news

feeds and contacts, let's discuss what **zembly** is and how it fits into the big picture of web development.

## 1.1 Social Programming

**zembly** is all about building social applications in a social networking environment. **zembly** is not only the world's first Facebook application development environment, but OpenSocial and Meebo application development are just around the corner. What you're doing with **zembly** is *social programming*. That is, developing applications with other people using social networking-type features. Not only can you reuse pieces and parts that other people have previously built to construct new applications, you collaborate with your "contacts" (friends and colleagues). You see what your colleagues are working on via news feeds. You keep up with what they publish and even with changes they make.

**zembly**'s environment is browser-based. This means that activities such as editing, testing services, previewing widgets, documenting chunks of your application, and deploying all happen within your browser, potentially with the collaboration of other developers. By inviting and collaborating with others, you can reuse what they have built to create your own services and widgets. You can also "clone" **zembly** pieces. This lets you build your own version that you can modify to fit your particular situation.

All this makes developing web applications easier. If the barrier for building social applications is lowered, you open the door for more participants to fuel the "long-tail" of social application development. When less expert knowledge is required from developers, you let them create "situational applications." These types of applications are often narrowly focused and can be thrown away or retooled as situations change. When the cost to enter the fray is negligible, more people will join and enrich the field for everyone else. With **zembly**, you can move beyond developing large applications for thousands of users that need care and maintenance. You can target a small group (less than a hundred, say) who share a common interest addressed by a very specialized application. This collaboration and reuse feeds on itself. The more people who develop with **zembly**, the richer its environment becomes. And, the more useful **zembly** becomes, the easier it is to create (perhaps only incrementally different) applications. More developers then flock to **zembly**.

Let's look at an example of collaboration on **zembly**. One of **zembly**'s engineers, Jiri Kopsa, created an iPhone-friendly Facebook application called Pixelife. (To try this out, search for the application on **zembly** as shown in Figure 1.1.)



Figure 1.1 Using zembly's Search mechanism to find Facebook application Pixelife

Pixelife is a fun iPhone web application that lets you create pixel-based designs using the iPhone touch screen. Figure 1.2 shows a screen shot of this application. (Okay, so we may not have the next Picasso here.)

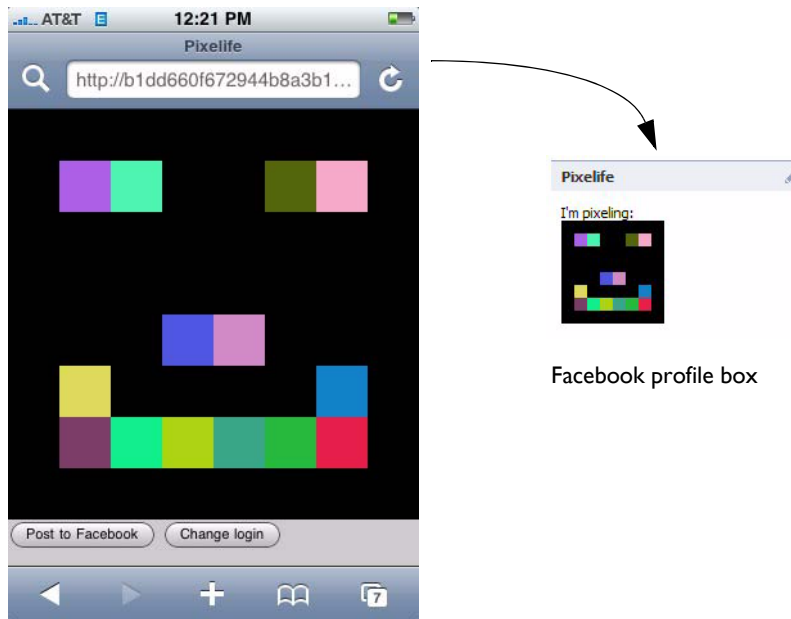


Figure 1.2 Be an artist (or scribbler) with iPhone-friendly Pixelife Facebook application

Along comes another developer named Yutaka Hosoi, who builds a brand-new Facebook application called Guess Who. This application displays a randomly selected friend's profile picture completely blacked out. You have a limited time to guess whose picture you're looking at. You see more and more of the picture as you remove the black pixels obscuring your friend's face as shown in Figure 1.3. Guess Who adds motivation: if you don't guess your friend's name within the allotted time period, a dire notice informs your friend that you don't know their name.

As you might have guessed, Yutaka uses the same code (cloned from Pixelife) to uncover the profile picture in small pixel-based chunks. Yutaka was able to build the Guess Who application very quickly because of the following reasons.

- You can build the basic Facebook application structure in **zembly** in about two minutes.
- **zembly**'s collaboration and code reuse let you pull together and assemble pieces developed by others.

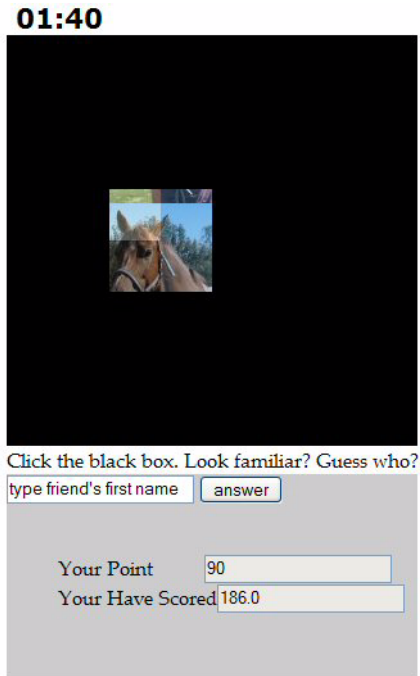


Figure 1.3 Guess Who is a zembly-based Facebook application that tests your ability to recognize friends' profile pictures

## 1.2 zembly's Environment

**zembly** looks at the entire web as an API and attempts to bring together disparate sources of data and functionality. **zembly**'s platform is the web and its goal is to make the programmable web environment more consistent for the developer. This approach not only gives developers a richness, but makes it possible to build applications based on different APIs: Twitter, Yahoo!, and Google Maps, to name a few. Note

that each API is different in how you invoke services and how you learn about them. **zembly** provides a level of consistency because it manages platform-specific API keys, provides a consistent calling interface, and provides a consistent link to API documentation. Through the **zembly** Find & Use facility, you can search for available services and platform-specific adapters from all **zembly** sources, including services written and published by other developers.

One of **zembly**'s long term goals is to help developers architect applications across multiple platforms. It helps you break up applications into reusable pieces (generally services and widgets). **zembly** offers common tools that help modularize your applications by providing a common model or abstraction for different environments like Facebook, Meebo, and MySpace, for example. The reality is that those different platforms are actually very heterogeneous, even within OpenSocial (a common API for social applications). The OpenSocial environment is fractured and each platform has its own extensions and capabilities. **zembly** lets you easily reuse common code and helps you isolate the differences among the platforms.

You can, of course, use platform-specific API calls. If you decide, for instance, that the Facebook data store API is appropriate for your storage mechanism, you can program directly to the Facebook API.

Let's look at another example. The myPicks U.S. Election 2008 application (see Figure 1.4), designed by Pramati and built with **zembly**, is a social networking application that works in both MySpace and Facebook. This application lets you voice your opinions on the 2008 U.S. presidential campaigns and discuss issues surrounding the election. After installing this application on your Facebook or MySpace profile, you can do the following.

- Participate in a poll indicating how you will vote in the presidential election.
- Provide your opinion on issues being debated by the candidates.
- State what you believe are the views of your friends. You can also invite your friends to play, guess their views on various issues, and give them an opportunity to agree, disagree, or comment on the issues.



Figure 1.4 Application myPicks U.S. Election 2008 runs on both MySpace and Facebook

Why is this application significant? First, myPicks U.S. Election 2008 aggregates data between Facebook and MySpace users of the game.

But there is a second significant point here. Pramati developed myPicks U.S. Election quickly so the application would be relevant during the rapidly changing politics of the campaign. Note that this application has a built-in shelf-life, since its relevancy disappears after November 4. (The predecessor application developed by Pramati focused on the Beijing Olympics—also an application with a built-in ending time.) Clearly, if building a compelling social application quickly is not possible, these “situational applications” would not exist.

### 1.3 zembly’s Audience

So, who exactly is the target audience for **zembly**? If the previously stated goal is to make application development easy (or at least easier), then part of the realization of that lofty premise is to attract application builders who might not really think of themselves as traditional software developers. Indeed, one group of **zembly** application builders could be termed “Casual Technologists.” This is a group of developers who are comfortable with the web, use Facebook or MySpace every day, use Twitter or instant messaging, but may not be classically-trained engineers. These casual technologists are comfortable manipulating flickr data and searching for interesting photos, for example. With this in mind, the next step might be to write a flickr-based slide show widget. (And we show you how. See “Creating a Flickr User Slide Show Widget” on page 71.) **zembly** aims to accommodate both the professional application builder and the casual technologist.

What skills do application builders need to use **zembly**? The development environment of **zembly** is based on scripting languages. Currently, **zembly** developers use HTML for markup, CSS for styling, and JavaScript for program logic. Facebook appli-

cations can also use Facebook Markup Language (FBML) and Facebook JavaScript (FBJS), which are Facebook-specific flavors of HTML and JavaScript, respectively.

Note that you don't have to start from scratch with **zembly**. **zembly** provides widget "templates" for categories of widgets (widgets for polling, slide shows, or text translations, to name several) that you can select and configure. This provides "no-coding" widget building. Or, if you find a useful service, widget, or application, **zembly** lets you clone it. You iterate off of other people's work and reuse code. **zembly** then gets better as more people participate and contribute. Todd Fast, one of **zembly**'s architects has said, "It's like a wiki, but for live, executable code."

**zembly** also reduces the complexities of web programming. If you use **zembly** to call an Amazon or Twitter service, it's a one-line JavaScript function call. You can then mashup the returned data in any number of ways. You can put this data on Facebook or use Google Maps, if it applies. In contrast, to access the Amazon Simple Storage Server with Java, for example, requires 135 lines of code!

What is the allure in building social applications for the web? Simply put, social applications are a great way to reach a lot of people (Facebook alone has more than 100 million active users). Social applications are also a potential source of revenue and can be a vehicle for advertising (for example, a game based on a movie) or community organizing. After 2008, political campaigns will all take advantage of social networking sites. **zembly** provides the development environment for all these situations.

## 1.4 Publishing and Scaling with zembly

Ok, you've programmed your application, widget, or service with **zembly**, now how do you deploy it? How do you make it accessible to the world? What about servers, scaling, and deployment issues?

**zembly** lets you skip the traditional architectural issues of enterprise computing by providing all of the back-end deployment infrastructure transparently. That is, you bring your code live by simply clicking the Publish button. **zembly** does all of the work of pushing the code to **zembly**'s containers and running the servers on the back-end for you. As a developer, this means you don't have to worry about how millions of people might use your application or access millions of rows of data.

How does **zembly** work then? Behind the scenes, Sun Microsystems supplies the infrastructure. Supporting **zembly**, and applications powered by **zembly**, is a full Sun Microsystems stack, including MySQL, Glassfish application server, Java, Apache, and Memcached. All this is hosted within `network.com`, Sun Microsystem's cloud computing infrastructure. The idea is that when you build something, you get the backing

of Sun Microsystems. You know it's going to scale properly when you go viral and acquire millions of users.

## 1.5 Monetizing with zembly

Today's idea of a successful application is no longer measured in lines of code or number of downloads. In a social application environment, people expect to see analytics that measure how many people use their application, how it's used, and how it travels through the social network. **zembly** has formed a partnership with Sometrics to provide this capability. Developers who create social applications with **zembly** automatically receive Sometrics social analytics data to help them understand their audience and boost traffic and revenue.

Reliable, in-depth analytics data is the critical piece developers need to help them build a thriving community and increase engagement with their applications. Sometrics specializes in helping developers get a unique perspective on the usage and spread of their applications. **zembly's** long term goal is to let developers monetize their applications. Deep social analytics is the first step.

## 1.6 Coming to zembly Soon

**zembly** has grown tremendously in the last year and will continue to evolve as it supports more platforms and programming environments. Because of **zembly's** rapid development, parts of **zembly** are being finalized as this book's printing deadline looms. Other parts are on the drawing boards or still in the works. Here are some of the near-term platform targets on the horizon for **zembly**.

### **Meebo**

Meebo is a web site that lets you chat with the most widely used instant messaging networks from a browser. Meebo also offers an interesting environment for games and other applications that you can launch in your chat window.

**zembly** lets you build Meebo applications. After registering your application with Meebo, you can use **zembly** to build widgets and services.

### **OpenSocial**

OpenSocial defines a common API for social applications across multiple web sites. Built from standard JavaScript and HTML, developers can create applications with OpenSocial that access a social network's friends and update feeds. Applications

implementing the OpenSocial APIs are interoperable with any social network system that supports them, including features on sites such as Hi5, MySpace, orkut, Netlog, Sonico, Friendster, Ning and Yahoo!.

zembly anticipates supporting OpenSocial target sites soon (myPicks U.S. Elections 2008 already runs on both MySpace and Facebook platforms).

## Scripting Languages

zembly is planning to target other scripting languages besides JavaScript. Target languages include PHP, Ruby, and Python. Scripting languages are important in zembly, because they simplify the programming environment. With scripting, you don't have to worry about compilation and linking, deployment, or other software building-related tasks.

## A Final Thought

### zembly Axiom

---

*The more people who use **zembly**, the better **zembly** becomes. And that's good for everyone!*

---