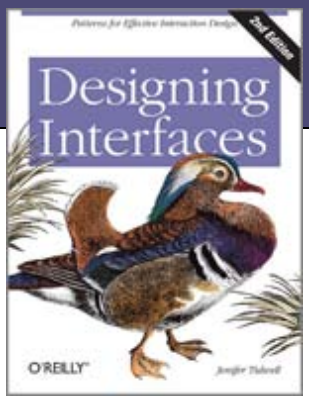
Excerpts from



2nd Edition

Home

About the book

What's new in the second edition

Blog

Patterns
- Picture Manager
- News Stream
- Wizard
- Settings Editor
- Alternative Views
- Many Workspaces
- Fat Menus
- Sitemap Footer
- Animated Transition
- Two-Panel Selector
- One-Window Drilldown
- List Inlay
- Grid of Equals
- Radial Table
- Infinite List
- Password Strength Meter
- Liquid Layout
- Deep Background

Buy from Amazon

Search

Categories

DI Second Edition (4)
Meta (1)
Patterns (4)
Uncategorized (1)
UX Randomness (1)

Wizard

What

Lead the user through the interface step by step to do tasks in a prescribed order.

Use when

You are designing a UI for a task that is long or complicated, and that will usually be novel for users—not something that they do often or want much fine-grained control over (such as the installation of a software package). You're reasonably certain that the designer of the UI will know more than the user does about how best to get the task done.

Tasks that seem well suited for this approach tend to be either branched or very long and tedious—they consist of a series of user-made decisions that affect downstream choices.

The catch is that the user must be willing to surrender control over what happens when. In many contexts, that works out fine, since making decisions is an unwelcome burden for people doing certain things: "Don't make me think, just tell me what to do next." Think about moving through an unfamiliar airport—it's often easier to follow a series of signs than it is to figure out the airport's overall structure. You don't get to learn much about how the airport is designed, but you don't care about that.

But in other contexts, it backfires. Expert users often find wizards frustratingly rigid and limiting. This is particularly true for software that supports creative processes such as writing, art, or coding. It's also true for users who actually do want to learn the software; wizards don't show users

Archives

June 2011 (1)
May 2011 (1)
April 2011 (5)

RSS feed

what their actions really do, or what application state gets changed as choices are made. That can be infuriating to some people. Know your users well!

Why

Divide and conquer. By splitting up the task into a sequence of chunks, each of which can be dealt with in a discrete “mental space” by the user, you effectively simplify the task. You have put together a preplanned road map through the task, thus sparing the user the effort of figuring out the task’s structure—all he needs to do is address each step in turn, trusting that if he follows the instructions, things will turn out OK.

But the very need for a wizard indicates that a task may be too complicated. If you can simplify a task to the point where a short form or a few button clicks can do the trick instead, that’s a better solution. (Keep in mind, too, that wizards are considered a bit patronizing in some Asian cultures.)

How

“Chunking” the task

Break up the operations constituting the task into a series of chunks, or groups of operations. You may need to present these groups in a strict sequence, or not; sometimes there is value in breaking up a task into steps 1, 2, 3, and 4 just for convenience.

A thematic breakdown for an online purchase may include screens for product selection, payment information, a billing address, and a shipping address. The presentation order doesn’t much matter because later choices don’t depend on earlier choices. Putting related choices together just simplifies things for people filling out those forms.

You may decide to split up the task at decision points so that choices made by the user can change the downstream steps dynamically. In a software installation wizard, for example, the user may choose to install optional packages that require yet more choices; if she chooses not to do a custom installation, those steps are skipped. Dynamic UIs are good at presenting branched tasks such as this, because the user never has to see anything that’s irrelevant to the choices she made.

In either case, the hard part of designing this kind of UI is striking a balance between the sizes of the chunks and the number of them. It’s silly to have a 2-step wizard, and a 15-step wizard is tedious. On the other hand, each chunk shouldn’t be overwhelmingly large, or you’ve lost some benefits of

this pattern.

Physical structure

Wizards that present each step in a separate page, usually navigated with Back and Next buttons, are the most obvious and well-known implementation of this pattern. They're not always the right choice, though, because now each step is an isolated UI space that shows no context—the user can't see what went before or what comes next. But an advantage of such wizards is that they can devote each page to that step completely, including illustrations and explanations.

If you do this, allow the user to move back and forth at will through the task sequence. Offer a way for the user to step backward, or to otherwise change her mind about an earlier choice. Additionally, many UIs show a selectable map or overview of all the steps, getting some of the benefits of a **Two-Panel Selector**. (In contrast to that pattern, a wizard implies a prescribed order—even if it's merely suggested—as opposed to completely random access.)

If you instead choose to keep all the steps on one page, you could use one of several patterns from Chapter 4:

- Titled Sections, with prominent numbers in the titles. This is most useful for tasks that aren't heavily branched, since all steps can be visible at once.
- Responsive Enabling, in which all the steps are present on the page, but each one remains disabled until the user has finished the previous step.
- Responsive Disclosure, in which you wait to show a step on the UI until the user finishes the previous one. Personally, I think this is the most elegant way to implement a short wizard. It's dynamic, compact, and easy to use.

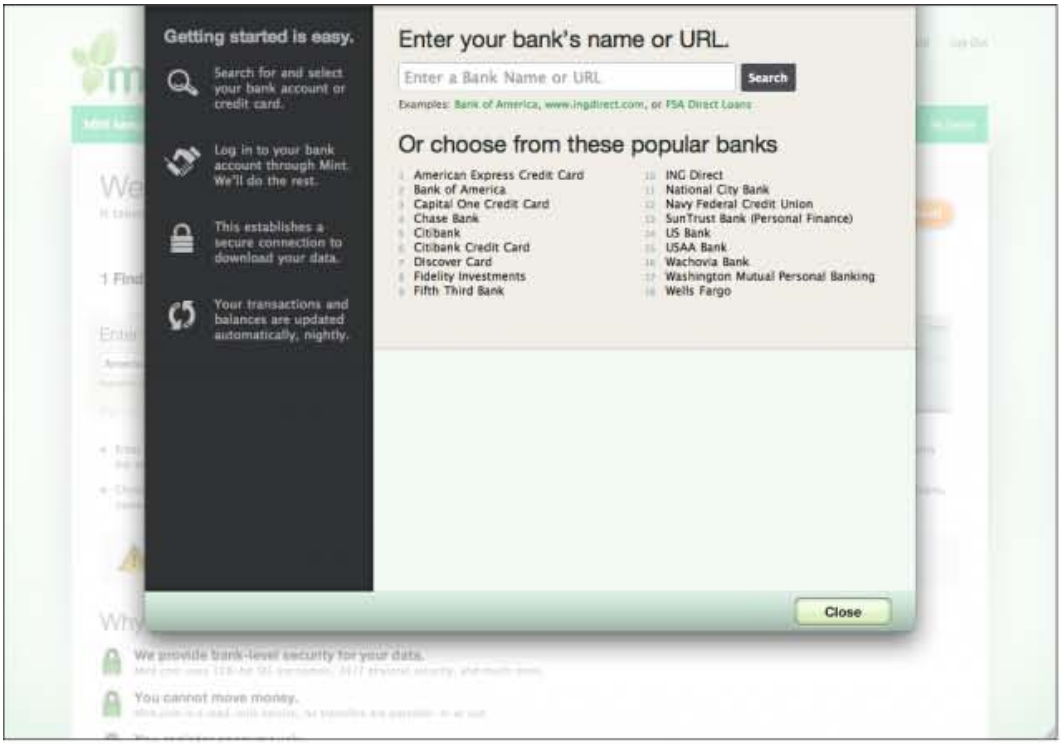
Good Defaults are useful no matter how you arrange the steps. If the user is willing to turn over control of the process to you, odds are good she's also willing to let you pick reasonable defaults for choices she may not care much about, such as the location of a software installation.

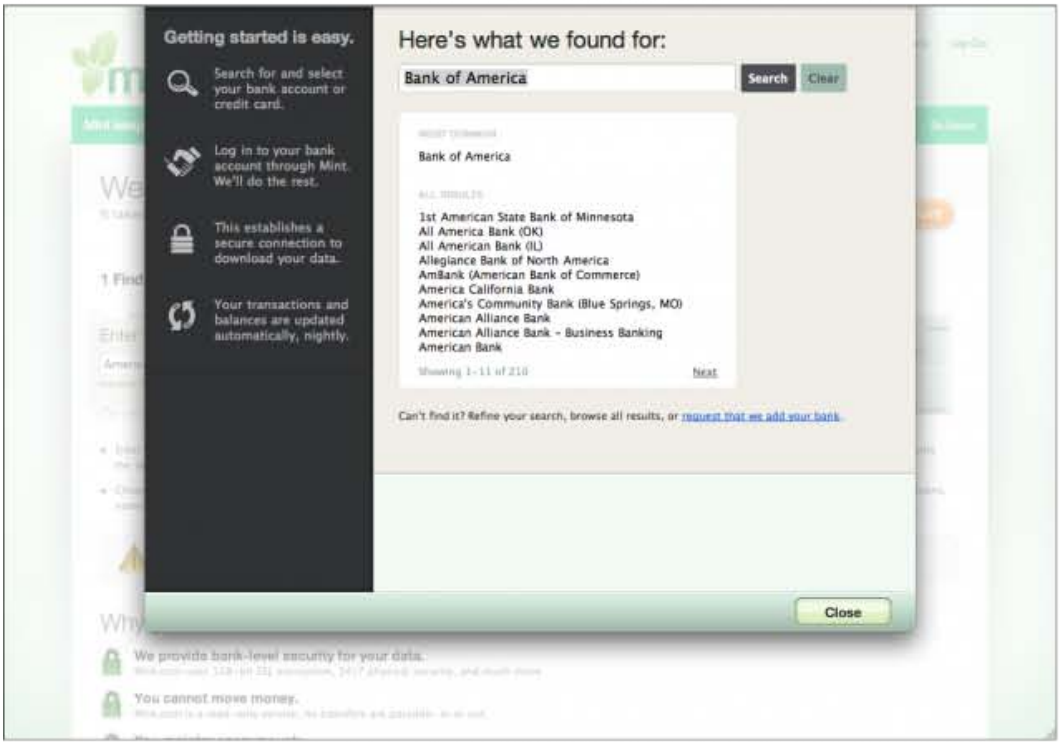
Examples

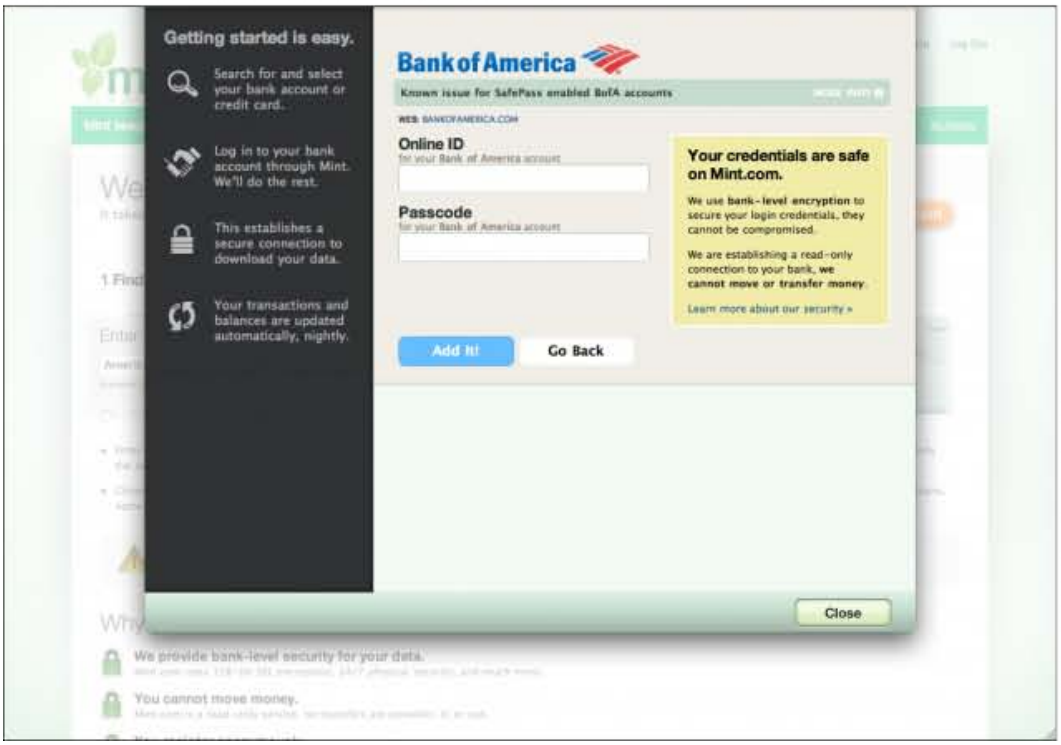
The My Yahoo! example at the top of the pattern illustrates many good features of a contemporary wizard. It uses a "lightbox" technique to focus attention on the modal dialogs; it lays out a clear Sequence Map (Chapter 3) of steps to show the user what will happen; it's short, easy to use, and

visually interesting; and it has a Cancel button in the upper right, as an Escape Hatch from the whole thing.

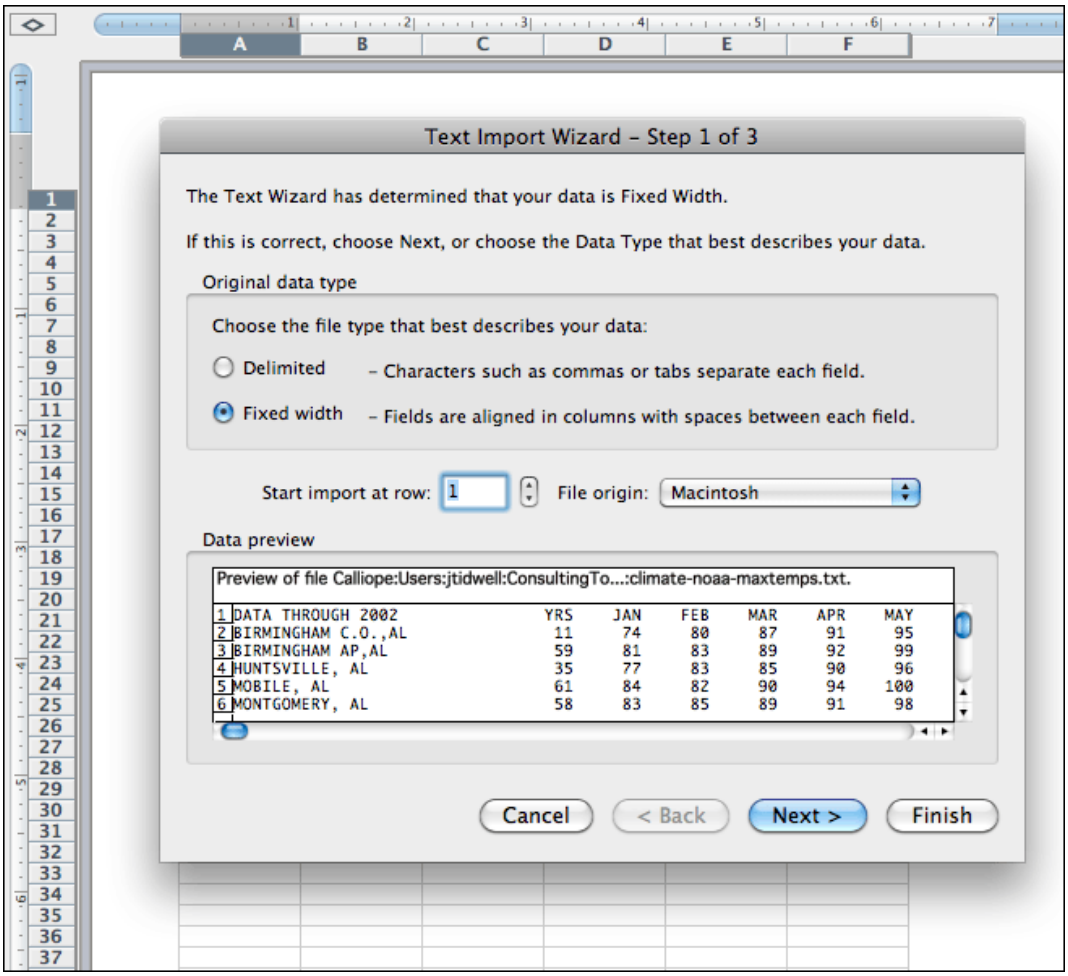
Mint's add-a-bank dialog, below, doesn't use a numbered sequence of steps, nor does it use a permanent Next button. But it still has the quintessential wizard quality of leading the user through a relatively complex series of steps, one screen at a time. Also, the list of steps on the lefthand side (which can't be clicked) gives the user an overview of what to expect.







The Microsoft Office designers have done away with many of its wizards, but a few remain—and for good reason. Importing data into Excel is a potentially bewildering task. The Import Wizard is an old-school, traditional application wizard with Back/Next buttons, branching, and no sequence map. But it works. Each screen lets you focus on the step at hand, without worrying about what comes next.



In other libraries

Wizard at UI-Patterns.com
Wizard at Welie.com
Wizard at Quince
One Page Wizard and **Multiple Page Wizard** at Patternry.com

Wizard is one of the canonical RIA screen layouts described by Bill Scott and Theresa Neil. An **article in UX Magazine** explains these layouts.