Using Action(s) to expand the market of your software

Action(s), the powerful automation tool, brings developers some powerful new business opportunities. If you haven't yet considered how Action(s) can work for you, this article may give you some ideas for writing your own actions or variables, either by adding them to your application or as stand-alone products, and some business suggestions that could help you find new ways to generate revenue using Action(s).

Action(s) lets users automate time-consuming, repetitive chores using workflows that link together discrete functionalities (called elements) of various applications. This means a user can run an Action(s) workflow that downloads images from the family's shared summer reunion webpage, add them to their personal pictures library, apply a filter that changes the photos to sepia tone images, then upload the sepia versions to another website.

The opportunities for developers lie in creating elements and workflows. For application developers, this is a great way to expose key features and extend their applications to reach new markets, as well as to increase perceived value and loyalty among their existing customer base. Action(s) also opens the door to creating actions and workflows that provide powerful functionalities that developers can sell into a growing market as stand-alone products or as plug-ins for existing applications. There are similar markets for products such as Photoshop plug-ins or Internet Explorer extension bars, which range from simple functionality to complex and industrial-strength professional workflows and automation tools.

Opportunities for Application Developers

It's the very nature of elements and workflows that make them powerful for expanding the market for an application. Actions focus attention on specific functionality, which can be a powerful marketing tool, because customers are often seeking specific functions. And workflows, which link applications together and can be easily shared, can bring visibility to an application in a crowded marketplace.

Let's take the case of a self-employed developer who creates a cool new Java application. He needs to compete against the promotions and ads of a myriad of other products to get the attention of the typical user. As a small business person with a tiny marketing budget, it's difficult for him

to establish brand recognition and mindshare for his product. He has solved the tough technical problems, but marketing his product is intimidating, as marketing is not core skill—in fact, he'd rather spend his time on the technical aspect anyway. He needs an effective way to get above the crowd and get some attention for his product.

Enter Action(s). To stand out in the market, our developer decides to expose one killer feature of his application — a single, time-saving task that his application does exceptionally well — through an Action(s) element. He then makes his element widely available, for free or a for a nominal fee, on various Action(s) sites. By focusing and drawing attention to on one key capability of his application, has made it easy for people to immediately grasp its value, and it begins to get attention. People begin building it into their workflows—anyone can save the workflow containing this Action as an application—and it starts getting shared among friends and colleagues. Knowing it comes from a trusted source, recipients double-click on the workflow application and it fires up Action(s). To those users who don't have the missing application in their system, Action(s) displays a message indicating that the workflow can't run until the application is installed.

Once the Action is part of a workflow that does something valuable, the missing application becomes desirable, even necessary.

A Market for Actions and Workflows

There's another kind of opportunity with Action(s) as well. Consider the case of another developer who has an idea for some very compelling new functionality, but isn't yet ready to build a fully-fledged, multi-feature product. Instead, she'd like to get just that functionality—essentially, a single feature or task—out there quickly to gauge the market and start generating revenue early. She can quickly create an Action(s) element that delivers her feature and lets users build it into workflows that rely on other applications.

The Action(s) API make creating elements easy, and there are various technology options. Depending on the resource integration needs of the Action and the developer's personal skill set, she can create her element in Java or in any other developer language that can be run on the Java VM, such as PHP, JavaScript, Python, Ruby or JavaFX.

When it comes to UI design, our developer's work is made light. Unlike a full-blown application, she just needs to design a minimal UI for the Action's pane. Figure 1 shows the Create Thumbnail Images action element, a good example of a UI that provides enough information to be useful, but not so much that it's busy or confusing.

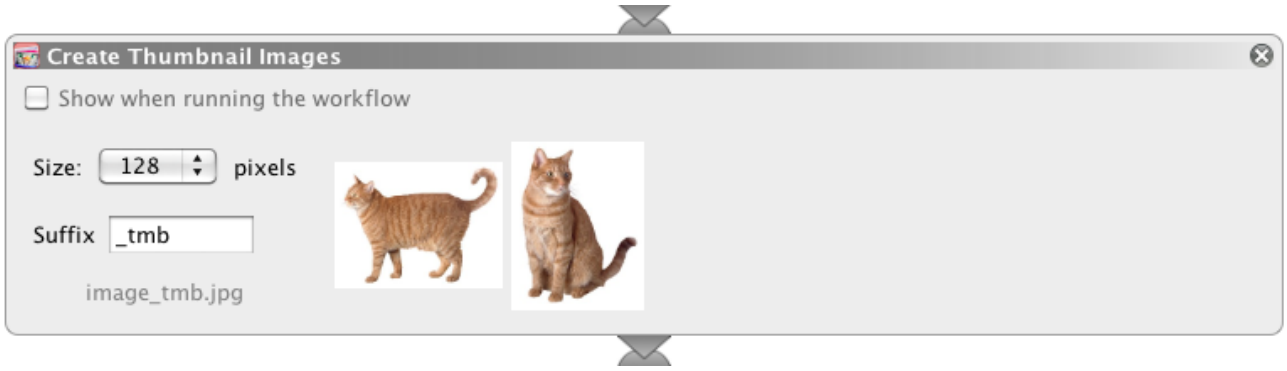

Figure 1: Just Enough User Interface for an action element pane.

Simply put, Action(s) gives developers the freedom to build and sell smaller components. For example, instead of spending a year building one application with 100 features that sells for $29.95, a developer could release one or two elements right away, and then incrementally build up a suite of dozens of elements that sell for a few dollars each. This model generates essential start-up revenue, and eventually the developer can bundle a set of elements with a discounted price for the package. Users get the choice of buying one or many elements, as they see fit, and the developer stays productive while still generating sales.

Building Customer Loyalty

A workflow that does something critical for a user is not likely to be abandoned. If that workflow includes elements bound to one or more applications, the user is likely to stay loyal to those applications, because the user has developed a dependency on the workflow. If the workflow does what they need time and again, there's simply no reason to change. They are also likely to recommend the workflow to others.

Equally worth mentioning, for businesses that rely on automation, workflows become part of their business processes and infrastructure. They invest internal resources for developing and fine-tuning them, train people around them, and are thus generally very committed to them.

Conclusion

Action(s) is significant because it's more than a technology—it's a delivery mechanism for a developer to increase sales and gain mindshare with customers who might not otherwise realize that the product has value. Whether you're selling a brand-name application or have an idea for the next killer feature, the path to more customers starts with that robot icon in your desktop.

Getting Started

With the support of Java and the flurry of activity around Action(s), there's no better time to get started. These steps will help you start taking advantage of Action(s) today:

- Make sure your application is scriptable or provides a Java API. Developers out in the world can't write Action(s) elements to leverage your application unless the application is scriptable.
- Write your own Action(s) elements. This puts you in the driver's seat to make sure the element is bound to your hosting application for maximum revenue potential. Developers don't have as much incentive to do that for you. To learn how to write elements and create Workflows, check out the Resources for Action(s) Developers at http://app.jbbres.com/actions/developers. You can also have a look at existing Action(s) elements for ideas: http://app.jbbres.com/actions/more.

For more information on using Action(s), visit the Action(s) website http://app.jbbres.com/actions.