Crystal Web-browsing Framework

Overview

Crystal is an extremely modular web browsing platform. Rather than specifying a monolithic architecture with centralized components, Crystal attempts to allow customization of virtually all parts of the interface, merely providing the loose coupling to create a coherent user experience. In other words, in the reverse of a normal crystal, it takes a collection of colors and presents a single coherent light for the user. It allows for customized plugins for both general functionality and viewing specific content, and allows these plugins to be loaded dynamically at runtime. The system is designed to be highly extensible and loosely coupled, enabling Crystal to handle content that it cannot currently understand, such as RSS, animated images, and more.

Design

Crystal refers to both the framework in general and a particular class inside that framework. For this discussion, the name Crystal will refer to the framework in general unless we specifically indicate otherwise.

Crystal recognizes three major functional components: a shell that must be omnipresent and provide the user with a portal through which to see and interact with the Internet; a collection of services that enable the user to interact with the Internet in general; and a collection of presenters that intelligently display various forms of content encountered on the web. Each of these components is conceptually orthogonal to the other two: the visual interface of the overall shell should not force content to be displayed in a specific way, and Internet-wide functionality should not need to interact with the specific presentation of content or vice-versa. Accordingly, Crystal supplies three interfaces: Shell, which defines a shell; Shard, which defines a presenter; and Service, which defines a service.

1. This is not the origin of the name. The name almost certainly comes from a comment made by Master Shake on Aqua Teen Hunger Force, because I drew up the design documents while watching the show. Unfortunately, I have no idea what the quote was. Anyone who does is welcome to email benjamin.pollack@duke.edu with the subject line “ATTN: I have no life.”
2. Someone pointed out that if I were going to call presenters Shards, I should have called the project as a whole Glass. They were completely correct. Unfortunately, I’m the one who wrote the code, and I say that shards go with crystals, so that’s the way it’s gonna be.
Shells

The Shell, representing the general container for the UI, should present a coherent view of one URL at a time. Accordingly, Shell defines a changeUrl method that allows Services and Shards to change the active URL of the entire application. (Services and Shards are welcome to establish additional connection on their own, but the entire application officially represents only one URL at any given point in time.) As complementary functionality, the Shell also allows any object to register itself as a URLChangeListener to be notified whenever the URL changes. The Shell must also provide some mechanism for components to convey messages to the user in a coherent way, and therefore provides the showStatus method to display custom content in the status bar.

Notably, Shell does not define any public way of adding Shards or Services. Instead, a Shell, in its role as container and general coordinator, should implement its own mechanism for loading and unloading various components. The default Shell provided with Crystal, which is called Crystal, will attempt to load any components specified either in the file components.txt in the directory from which Crystal was launched, or in a file passed on the command line. An example components.txt file has been provided for you. Components should be fully qualified, must be in the classpath, and must adhere to the Shard or Service interface. Please remember that this particular behavior is not specified by the Shell interface in general; this is merely the behavior of the default Shell. The only thing that developers should expect is that the Shell will handle locating, loading, and initializing its components.

Services

Services are designed to be components that are applicable to experiencing the Internet on the whole, and not related to presenting specific content in the browser window. Examples include URL bars, history, back and forward buttons, Google search bars, bookmarks, RSS feed extractors, and more. Any functionality which should always be present and accessible should be implemented as a Service. Services logically need to be able to allow the user some form of interaction. For this purpose, the Service interface provides a mechanism for returning menus and toolbar buttons that the Service would like displayed. These components are always visible and may not be removed once displayed. If a given Service needs to interact with these GUI components, it should retain the components’ handles itself. In order to allow customization, the Service can also provide a preference pane that will be displayed in the Shell’s preference window.

In order to make the Shell’s interface for changing URLs available, and in order to enable listening for URL changes, Service also defines a setShell method that will be called at the time the Service is initially instantiated. This method should only be called once for the lifetime of an instance of Service, but there may be many instances with different Shells at any given point.
Shards

Shards support all of the functionality of Services, but represent presenters that can handle specific content on the Internet. Generally, viewers handle one or more MIME types, such as text/html, image/jpeg, and so on.\(^3\) Shard therefore has the ability to return a set of MIME types that can handle. When a Shell want a viewer to display content, the following sequence of actions occurs:

1. The Shell looks for a Shard that claims it can handle the MIME type of the content at the requested location.

2. Assuming that it finds one, the Shell needs to tell the Shard to display the content at the user’s requested URL, so it passes the URL and data via a URLConnection to the Shard. The Shard should begin loading the content at this time.

3. Occasionally, Crystal will make an inappropriate selection due to a web server incorrectly identifying the content type of the media it’s serving. When this happens, Shards should throw a BadContentException so that the Shell can be made aware of the problem.

4. To handle Shards special role as a viewer, it provides a main content pane that will be displayed when it loads data. In addition, the Shell needs to provide the user with GUI elements that are specific to this particular type of media. The Shell calls the same methods as in Service to load menus and toolbar buttons, and adds them to the window at this time.

5. Finally, when the user goes to a new URL, the menus and buttons are removed from display, the Shard’s view is removed from the main window, and the Shard may either be garbage collected or kept in memory for use at a later time.

Examples of Shards include viewers for SVG, RSS, Atom, images, Flash, and even non-visual formats such as MIDI and MP3. Also note that certain functionality may only be implemented as a Shard even though it initially appears to be a service. For example, searching for content within a webpage requires that the search box (which can be displayed in the main toolbar) be able to communicate with the specific view. A Service cannot do this, so that functionality must be implemented with a Shard.

---

3. For a comprehensive list of MIME types, visit the official media-type list of the Internet Assigned Numbers Authority at http://www.iana.org/assignments/media-types/.