Confessions

Table of contents

1. Overview

Here lays the golden rules, the main principles of what makes a good API. We had tried to follow these rules in the jpl.

Note:

These principles comes from Practical API Design - Confessions of a Java Framework Architect written by Jaroslav Tulach. The following paragraphs are extracted from this book.

2. Comprehensibility

Those who have to use an API must be able to understand it. I have already mentioned that an API is about communication between the programmer who writes the API and the programmer who writes an implementation against that API.

The most similar activity to writing an API is writing a book: one writer and a lot of readers. The readers know something of the writer, but the writer knows little or nothing about the readers. Guessing their skills and knowledge correctly is part of the delicate art of making an API that is easy to understand. Concepts provided by a good API have to lie inside the user's horizon, otherwise they won'be understood.

When coding in Java, it is safe to expect that people know concepts used in most of the java.*libraries. Iterators, enumerations, I/O streams, JavaBeans, listeners, and visual components are objects that nearly any Java programmer has encountered. Use of familiar terminology, classes, and terms reduces the learning curve, which in turn places fewer cognitive demands on an API's user.

The less familiar coding constracts, the more novel the concepts an API uses, the greater the benefit of extensive examples.

3. Consistency

If the users of an API have to invest time to learn a concept, it is important that the concept be applied consistently across the entire API. For example, if there is a way to register, say, factories for objects, then all types of factories should use that one registration mechanism. This reduces the number of caveats and special cases a user of the API needs to track.

From the point of view of the lifetime of an API, it is important to follow the same style of development or at least clearly communicate the evolutionary style of an API. It is inconvenient to find that certain parts of an API have evolved in a different way than others.

4. Discoverability

Even the most beautiful API is useless if those who are supposed to use it cannot find it or easily understand how to use it.

In most cases, the set of classes is not what interests most API users. They are interested in getting their job done. For these purposes it is more important to see examples of API usage, enabling the selection of the idiom that is closest to what you want to do.

Regardless of what type of API you provide, it is important to create a single place that can serve as a starting point and can send people in the directions that solve their problems. As people don't think in terms of classes, it is important to organize this entry point in an optimal manner, based on the actual or at least expected goals and tasks.