Janis Gonzalez

WEB 420 RESTful APIs

Discussion 7.1 Semantic Versioning

April 27,2023

Bellevue University

When something like an application has many dependencies, a developer should be releasing new package versions. Having many dependencies could become an issue if it is not handled efficiently and with care. This is why Semantic Versioning is used for these new releases of versions. Semantic Versioning provides easy management for all the updates that are sure to come along with releasing a public API for use. The format for Semantic Versioning is major.minor.patch. Major version is for when incompatible API changes are made. Minor version is for when an added functionality for backwards compatibility is added (Preston-Werner). Patch version is for when backwards compatible bug fixes are made. Any additional labels are added to this extension. There are eleven specifications for Semantic Versioning. One, the software that is using Semantic Versioning must declare the public API. When the API is declared, whether through the code or in the documentation, it should be precise and comprehensive. Two, the format used should follow the major.minor.patch format with each element increasing numerically (Preston-Werner). For example, it should be 1.3.0 -> 1.4.0 -> 1.5.0. Three, once a version packahe has been released, it cannot be edited or modified in any way, but instead should be released as a new version. Four, major version zero, 0.minor.patch, should only be used for the initial development. Five, version 1.0.0 defines the public API. Six, patch versions should be incremented when only backwards compatible bug fixes are made (Preston-Werner). Seven, minor versions should only be incremented when new backwards compatible functionality is introduced into the public API (Preston-Werner). Patch versions must be reset back to 0 when minor version is incremented. Eight, major versions must be incremented if any backwards incompatible changes are made to the public API. Patch and minor versions must be reset to 0 when a major version is incremented. Nine, pre-released versions may be shown by adding a hyphen and a series of dot identifiers following the patch version (Preston-Werner). Identifiers should not be empty and should not include leading zeroes. Examples include: 1.0.0-alpha, 1.0.0-alpha.2, 1.0.0-0.2.5, 1.0.0-x.6.y.67, 1.0.0-x-y-z.--. Ten, build metadata can be shown by adding a plus sign and a series of dot separated identifiers that immediately follow the patch or pre-release version. For example: 1.0.0-alpha+001, 1.0.0+20130313144700, 1.0.0-beta+exp.sha.5114f85, 1.0.0+21AF26D3----117B344092BD (Preston-Werner). Eleven, precedence is defined by separating the version into major, minor, patch and pre-release identifiers. It is identified by comparing the identifiers from left to right.

Reference

Preston-Werner, T. (n.d.). *Semantic versioning 2.0.0*. Semantic Versioning. Retrieved from <https://semver.org/>