Resumen de las versiones semánticas 2.0.0

El software que utiliza el control de versiones semántico debe declarar una api pública. una vez que se ha lanzado un paquete versionado, el contenido de esa versión no debe modificarse. cualquier modificación debe publicarse como una nueva versión. La versión principal cero es para el desarrollo inicial.

la api pública no debe considerarse estable. La versión 1.0.0 define la api pública. la forma en que se incrementa el número de versiones después de este lanzamiento depende de esta api pública y de cómo cambia. La versión del parche z debe incrementarse si solo se introducen correcciones de errores compatibles con versiones anteriores.

debe incrementarse si alguna funcionalidad de la api pública está marcada como obsoleta. puede incrementarse si se introducen nuevas funcionalidades o mejoras sustanciales en el código privado. La versión del parche debe restablecerse a 0 cuando se incrementa la versión secundaria. La versión principal x debe incrementarse si se introducen cambios incompatibles con versiones anteriores en la api pública.

También puede incluir cambios menores y de nivel de parche. el parche y las versiones secundarias deben restablecerse a 0 cuando se incrementa la versión principal. Una versión previa al lanzamiento puede indicarse agregando un guión y una serie de identificadores separados por puntos inmediatamente después de la versión del parche. los identificadores deben contener solo guiones y caracteres alfanuméricos ascii.

Los identificadores no deben estar vacíos. Los identificadores numéricos no deben incluir ceros a la izquierda. las versiones preliminares tienen una precedencia menor que la versión normal asociada. Una versión preliminar indica que la versión es inestable y es posible que no satisfaga los requisitos de compatibilidad previstos, como lo indica su versión normal asociada.

Los metadatos de compilación pueden indicarse agregando un signo más y una serie de identificadores separados por puntos inmediatamente después del parche o la versión preliminar. Los metadatos de compilación deben ignorarse al determinar la procedencia de la versión.

Por lo tanto, dos versiones que difieren sólo en los metadatos de compilación tienen la misma precedencia.

¿Por qué utilizar el control de versiones semántico?

Sin el cumplimiento de algún tipo de especificación formal, los números de versión son esencialmente inútiles para la gestión de dependencias. Al dar un nombre y una definición clara a las ideas anteriores, es fácil comunicar sus intenciones a los usuarios de su software. Un ejemplo simple demostrará cómo el control de versiones semántico puede hacer que la dependencia sea cosa del pasado. Considere una biblioteca llamada «Firetruck». Requiere un paquete Versionado Semánticamente llamado «Ladder». En el momento en que se crea Firetruck, Ladder tiene la versión 3.1.0.

Ahora, cuando las versiones 3.1.1 y 3.2.0 de Ladder estén disponibles, puede publicarlas en su sistema de administración de paquetes y saber que serán compatibles con el software dependiente existente. Lo que puede hacer es dejar que Semantic Versioning le brinde una forma sensata de lanzar y actualizar paquetes sin tener que implementar nuevas versiones de paquetes dependientes, lo que le ahorrará tiempo y molestias. Si todo esto suena deseable, todo lo que necesita hacer para comenzar a usar el control de versiones semántico es declarar que lo está haciendo y luego seguir las reglas.