

Actualización de un código base con tipos de referencia que admiten un valor NULL para mejorar las advertencias sobre diagnósticos nulos

Artículo • 06/04/2023

Los [tipos de referencia que admiten un valor NULL](#) permiten declarar si a las variables de un tipo de referencia se les debe asignar o no un valor `null`. El análisis estático y las advertencias del compilador cuando el código podría desreferenciar `null` son la ventaja más importante de esta característica. Una vez habilitado, el compilador genera advertencias que ayudan a evitar que se genere una excepción [System.NullReferenceException](#) cuando se ejecuta el código.

Si el código base es relativamente pequeño, puede activar la [característica en el proyecto](#), resolver advertencias y disfrutar de las ventajas que ofrecen los diagnósticos mejorados. Los códigos base más grandes pueden requerir un enfoque más estructurado para resolver advertencias a lo largo del tiempo, habilitar la característica para algunos a medida que usted resuelve advertencias en distintos tipos o archivos. En este artículo se describen distintas estrategias para actualizar un código base y las contrapartidas asociadas a estas estrategias. Antes de iniciar la migración, lea la introducción conceptual de los [tipos de referencia que admiten un valor NULL](#). En ella se trata el análisis estático del compilador, los valores *null-state* de *maybe-null* y *not-null* y las anotaciones que admiten un valor NULL. Una vez que se haya familiarizado con esos conceptos y términos, podrá migrar su código.

Planeación de la migración

Independientemente de cómo actualice su código base, el objetivo es que las advertencias y anotaciones que admiten un valor NULL estén habilitadas en su proyecto. Una vez que alcance ese objetivo, tendrá el valor `<nullable>Enable</nullable>` en su proyecto. No necesitará ninguna de las directivas del preprocesador para ajustar la configuración en otro lugar.

La primera opción es establecer el valor predeterminado para el proyecto. Las opciones son:

1. **'Disable' que admite valores NULL como valor predeterminado:** *disable* es el predeterminado si no se agrega ningún elemento `Nullable` a su archivo de proyecto. Use este valor predeterminado cuando no esté agregando activamente nuevos archivos al código base. La actividad principal es actualizar la biblioteca para que utilice tipos de referencia que admiten un valor NULL. Si se usa este valor predeterminado, se agrega una directiva de preprocesador que admite un valor NULL a cada archivo a medida que actualice su código.
2. **'Enable' que admite valores NULL como valor predeterminado:** Establezca este valor predeterminado al desarrollar activamente características nuevas. Quiere que todo el código nuevo se beneficie de los tipos de referencia y análisis estáticos que admiten un valor NULL. Si usa este valor predeterminado, deberá agregar un valor `#nullable disable` en la parte superior de cada archivo. Quitará estas directivas de preprocesador a medida que aborde las advertencias de cada archivo.
3. **Advertencias que admiten valores NULL como valor predeterminado:** Elija este valor predeterminado para una migración de dos fases. En la primera fase, resuelva las advertencias. En la segunda fase, active las anotaciones para declarar el valor *null-state* que se espera de una variable. Si usa este valor predeterminado, deberá agregar un valor `#nullable disable` en la parte superior de cada archivo.
4. **Anotaciones que admiten valores NULL** como valor predeterminado. Anote el código antes de resolver las advertencias.

Al habilitar una opción que admite un valor NULL como predeterminada, se genera más trabajo previo para agregar las directivas del preprocesador a cada archivo. La ventaja es que todos los archivos de código nuevos que se agreguen al proyecto estarán habilitados para aceptar valores NULL. Cualquier trabajo nuevo admitirá valores NULL; solo se debe actualizar el código existente. Si se deshabilita la opción que admite un valor NULL, este proceso funcionará mejor si la biblioteca es estable y el objetivo principal del desarrollo es adoptar tipos de referencia que admiten un valor NULL. Los tipos de referencia que aceptan valores NULL se habilitan al anotar las API. Cuando haya terminado, habilite los tipos de referencia que aceptan valores NULL para todo el proyecto. Al crear un archivo nuevo, debe agregar las directivas del preprocesador y hacer que sea compatible con los valores NULL. Si alguno de los desarrolladores de su equipo se olvida de hacerlo, ese nuevo código se sumará al trabajo pendiente para hacer que todo el código admita valores NULL.

La elección de una estrategia u otra dependerá de la cantidad de desarrollo activo que haya en el proyecto. Cuanto más desarrollado esté y más estable sea su proyecto, más

adecuada será la segunda estrategia. Cuantas más características se estén desarrollando, más apropiada será la primera estrategia.

Importante

El contexto global que admite un valor NULL no se aplica a los archivos de código generado. En cualquier estrategia, el contexto que admite un valor NULL está *deshabilitado* para cualquier archivo de código fuente marcado como generado. Esto significa que las API de los archivos generados no se anotan. Hay cuatro maneras de marcar un archivo como generado:

1. En el archivo `.editorconfig`, especifique `generated_code = true` en una sección que se aplique a ese archivo.
2. Coloque `<auto-generated>` o `<auto-generated/>` en un comentario en la parte superior del archivo. Puede estar en cualquier línea de ese comentario, pero el bloque de comentario debe ser el primer elemento del archivo.
3. Inicie el nombre de archivo con *TemporaryGeneratedFile_*
4. Finalice el nombre de archivo con *.designer.cs*, *.generated.cs*, *.g.cs* o *.g.i.cs*.

Los generadores pueden optar por usar la directiva de preprocesador **#nullable**.

Información sobre contextos y advertencias

Si se habilitan las advertencias y las anotaciones, se controlará cómo el compilador visualiza los tipos de referencia y la nulabilidad. Cada tipo tiene una de las tres nulabilidades:

- *oblivious*: todos los tipos de referencia son de tipo *oblivious* que admiten un valor NULL cuando el contexto de anotación de deshabilita.
- *nonnullable*: tipo de referencia no anotado; `C` es *nonnullable* cuando el contexto de anotación se habilite.
- *nullable*: tipo de referencia anotado; `C?` es de tipo *nullable*, pero puede que se genere una advertencia cuando el contexto de anotación se deshabilite. Las variables declaradas con `var` son de tipo *nullable* cuando el contexto de anotación se habilita.

El compilador genera advertencias en función de esa nulabilidad:

- Los tipos *nonnullable* generan advertencias si se les asigna un valor `null` potencial.

- Los tipos *nullable* generan advertencias si se desreferencian en el caso *maybe-null*.
- Los tipos *oblivious* generan advertencias si se desreferencian en el caso *maybe-null* y el contexto de advertencia está habilitado.

Cada variable tiene un estado predeterminado que admite un valor NULL que depende de su nulabilidad:

- Las variables que admiten un valor NULL tienen un estado predeterminado *null-state* de *maybe-null*.
- Las variables que no admiten un valor NULL tienen un estado predeterminado *null-state* de *not-null*.
- Las variables de tipo "oblivious" que admiten un valor NULL tienen un estado predeterminado *null-state* de *not-null*.

Antes de habilitar los tipos de referencia que aceptan valores NULL, todas las declaraciones del código base son de tipo *nullable oblivious*. Esto es importante porque significa que todos los tipos de referencia tienen in estado predeterminado *null-state* de *not-null*.

Resolución de advertencias

Si su proyecto usa Entity Framework Core, debe leer sus guías sobre cómo [trabajar con tipos de referencia que admiten un valor NULL](#).

Al iniciar la migración, debe empezar habilitando solo las advertencias. Todas las declaraciones siguen siendo de tipo *nullable oblivious*, pero se le mostrarán advertencias cuando desreferencie un valor después de que su estado *null-state* cambie a *maybe-null*. A medida que se resuelvan estas advertencias, realizará comprobaciones de NULL en más ubicaciones, y su código base será más resistente. Para obtener información sobre técnicas específicas para diferentes situaciones, vea el artículo sobre [Técnicas para resolver advertencias que admiten un valor NULL](#).

Puede resolver advertencias y habilitar anotaciones en cada archivo o clase antes de continuar con otro código. Sin embargo, a menudo resulta más eficaz resolver las advertencias generadas mientras el contexto es *warnings* para poder habilitar las anotaciones de tipo. De esta forma, todos los tipos serán *oblivious* hasta que haya resuelto el primer conjunto de advertencias.

Habilitación de anotaciones de tipo

Después de resolver el primer conjunto de advertencias, podrá habilitar el *contexto de anotación*. Así, se cambiarán los tipos de referencia de *oblivious* a *nonnullable*. Todas las variables declaradas con `var` son de tipo *nullable*. Con este cambio se suelen generar nuevas advertencias. El primer paso para resolver las advertencias del compilador es usar anotaciones `?` en los tipos de parámetros y de valores devueltos para indicar si los argumentos o los valores devueltos pueden ser `null`. A medida que realiza esta tarea, su objetivo no es solo resolver las advertencias; el objetivo más importante es hacer que el compilador entienda su intención de admitir posibles valores NULL.

Atributos para complementar a las anotaciones de tipo

Se han agregado varios atributos para expresar información adicional sobre el estado NULL de las variables. Es probable que las reglas de sus API sean más complicadas que *not-null* o *maybe-null* para todos los parámetros y valores devueltos. Muchas de las API tienen reglas más complejas para cuando las variables pueden ser `null` o no. En estos casos, usará atributos para expresar dichas reglas. Los atributos que describen la semántica de su API se encuentran en el artículo sobre los [Atributos que afectan a los análisis que admiten un valor NULL](#).

Pasos siguientes

Una vez que haya resuelto todas las advertencias después de habilitar las anotaciones, podrá establecer el contexto predeterminado para su proyecto en *enabled*. Si ha agregado alguna pragma a su código para la anotación que admite un valor NULL o el contexto de advertencia, la podrá eliminar. Con el tiempo, es posible que se le muestren nuevas advertencias. Puede escribir código que introduzca advertencias. Una dependencia de biblioteca se puede actualizar para los tipos de referencia que aceptan valores NULL. Esas actualizaciones cambiarán los tipos de esa biblioteca de *nullable oblivious* a *nonnullable* o *nullable*.

También puede explorar estos conceptos en nuestro módulo de aprendizaje sobre [Seguridad sobre la aceptación de valores NULL en C#](#).


Colaborar con nosotros en GitHub


El origen de este contenido se puede encontrar en GitHub, donde también puede crear y revisar problemas y solicitudes de incorporación de cambios. Para más información, consulte [nuestra guía para colaboradores](#).



Comentarios de .NET

.NET es un proyecto de código abierto. Seleccione un vínculo para proporcionar comentarios:

 [Abrir incidencia con la documentación](#)

 [Proporcionar comentarios sobre el producto](#)