Compartir código en plataformas

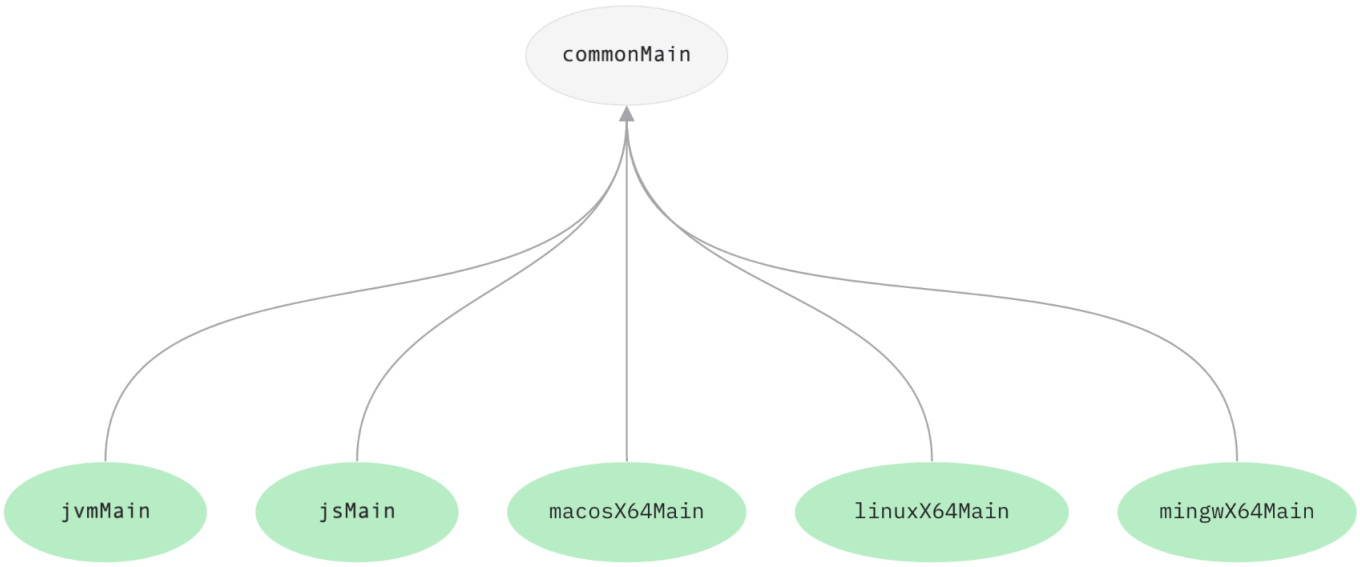
Con Kotlin Multiplatform, puedes compartir el código utilizando los mecanismos que Kotlin proporciona:

- Comparte el código entre todas las plataformas utilizadas en tu proyecto. Úsalo para compartir la lógica empresarial común que se aplica a todas las plataformas.
- Comparte código entre algunas plataformas incluidas en tu proyecto, pero no todas. Puedes reutilizar el código en plataformas similares con la ayuda de la estructura jerárquica.

Si necesita acceder a las API específicas de la plataforma desde el código compartido, utilice el mecanismo Kotlin de declaraciones esperadas y reales.

Compartir código en todas las plataformas

Si tiene una lógica empresarial que es común para todas las plataformas, no necesita escribir el mismo código para cada plataforma, simplemente compártalo en el conjunto de fuentes comunes.



Algunas dependencias para los conjuntos de fuentes se establecen de forma predeterminada. No es necesario especificar ninguna relación de dependencia manualmente:

- Para todos los conjuntos de fuentes específicos de la plataforma que dependen del conjunto de fuentes común, como jvmMain, macosX64Main y otros.
- Entre los conjuntos principales y de fuentes de prueba de un objetivo en particular, como androidMain y androidUnitTest.

Si necesita acceder a las API específicas de la plataforma desde el código compartido, utilice el mecanismo Kotlin de declaraciones esperadas y reales.

Comparte código en plataformas similares

A menudo necesitas crear varios objetivos nativos que potencialmente podrían reutilizar gran parte de la lógica común y las API de terceros.

Por ejemplo, en un proyecto multiplataforma típico dirigido a iOS, hay dos objetivos relacionados con iOS: uno es para dispositivos iOS ARM64, el otro es para el simulador x64. Tienen conjuntos de fuentes separados específicos de la plataforma, pero en la práctica rara vez hay una necesidad de un código diferente para el dispositivo y el simulador, y sus dependencias son muy iguales. Así que el código específico de iOS podría ser compartido entre ellos.

Evidentemente, en esta configuración sería deseable tener un código fuente compartido establecido para dos objetivos de iOS, con código Kotlin/Native que aún podría llamar directamente a cualquiera de las API que son comunes tanto al dispositivo iOS como al simulador.

En este caso, puede compartir código entre objetivos nativos de su proyecto utilizando la estructura jerárquica utilizando una de las siguientes maneras:

Uso de la plantilla de jerarquía predeterminada

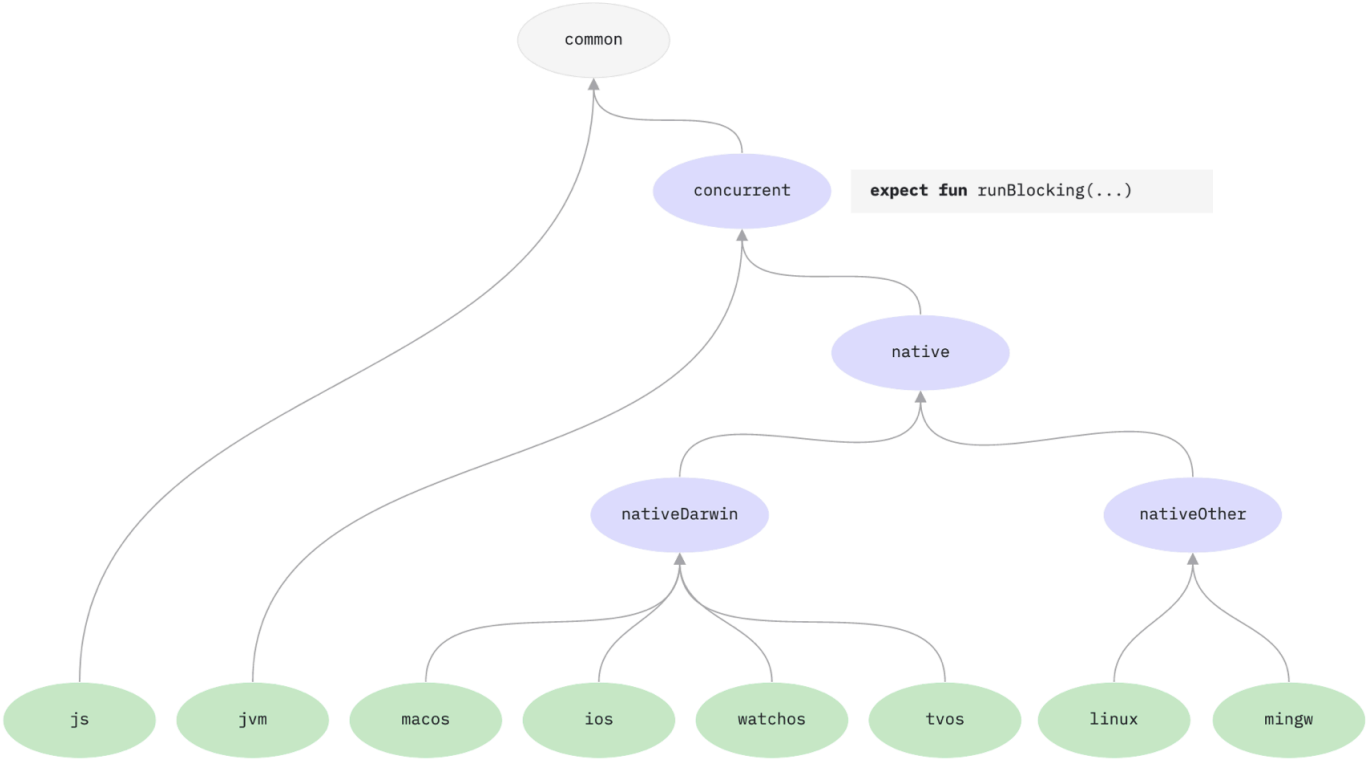
Configurar la estructura jerárquica manualmente

Obtenga más información sobre el intercambio de código en las bibliotecas y la conexión de bibliotecas específicas de la plataforma.

Compartir código en las bibliotecas

Gracias a la estructura jerárquica del proyecto, las bibliotecas también pueden proporcionar API comunes para un subconjunto de objetivos. Cuando se publica una biblioteca, la API de sus conjuntos de fuentes intermedias se incrusta en los artefactos de la biblioteca junto con información sobre la estructura del proyecto. Cuando utiliza esta biblioteca, los conjuntos de fuentes intermedias de su proyecto solo acceden a las API de la biblioteca que están disponibles para los objetivos de cada conjunto de fuentes.

Por ejemplo, echa un vistazo a la siguiente jerarquía de conjuntos de fuentes del repositorio kotlinx.coroutines:



El conjunto de fuentes concurrentes declara la función runBlocking y se compila para la JVM y los objetivos nativos. Una vez que la biblioteca kotlinx.coroutines se actualice y se publique con la estructura jerárquica del proyecto, puede depender de ella y llamar a runBlocking desde un conjunto de fuentes que se comparte entre la JVM y los objetivos nativos, ya que coincide con la "firma de objetivos" del conjunto de fuentes concurrentes de la biblioteca.

Conectar bibliotecas específicas de la plataforma

Para compartir más código nativo sin estar limitado por dependencias específicas de la plataforma, conecte bibliotecas específicas de la plataforma. Las bibliotecas enviadas con Kotlin/Native (como Foundation, UIKit y POSIX) están disponibles en conjuntos de fuentes compartidas de forma predeterminada.

Además, si utiliza el complemento Kotlin CocoaPods Gradle en sus proyectos, puede trabajar con bibliotecas nativas de terceros consumidas con el mecanismo de cinterop.

¿Qué sigue?

- Echa un vistazo a los ejemplos de intercambio de código utilizando el mecanismo Kotlin de declaraciones esperadas y reales
- Más información sobre la estructura jerárquica del proyecto
- Vea nuestras recomendaciones sobre el nombre de archivos fuente en proyectos multiplataforma