

Java-Sprachserver in Visual Studio Code

Hey! Die Einstellung `java.jdt.ls.java.home` in Visual Studio Code (VSCode) ist eine spezifische Konfigurationsoption, die von dem Java Language Server (Teil der Erweiterung "Language Support for Java(TM) by Red Hat", die im Java Extension Pack enthalten ist) verwendet wird. Lassen Sie uns das genauer betrachten:

Was es bedeutet

- **JDT**: Steht für "Java Development Tools", ursprünglich Teil der Eclipse IDE. Der Java Language Server in VSCode basiert auf Eclipse's JDT, das Funktionen wie Code-Vervollständigung, Debugging und Kompilierung bietet.
- **LS**: Steht für "Language Server", ein Hintergrundprozess, der die Java-Unterstützung in VSCode ermöglicht (z.B. IntelliSense, Fehlerprüfung).
- `java.jdt.ls.java.home`: Diese Einstellung gibt dem Java Language Server genau an, welche Java Development Kit (JDK) für seine Operationen verwendet werden soll. Sie ist von der allgemeineren `java.home`-Einstellung getrennt, obwohl sie miteinander verbunden sind.

Kurz gesagt, spezifiziert `java.jdt.ls.java.home` den JDK-Pfad, den der Java Language Server verwendet, um:

- Ihren Java-Code zu analysieren.
- Sprachfunktionen bereitzustellen (z.B. Autovervollständigung, Gehe zu Definition).
- Code in einigen Fällen zu kompilieren und auszuführen (obwohl die Kompilierung oft von anderen Einstellungen oder Build-Tools abhängt).

Unterschiede zu `java.home`

- `java.home`: Eine allgemeine VSCode-Einstellung, die auf das JDK für alle Java-bezogenen Erweiterungen und Aufgaben in VSCode verweist. Sie wird verwendet, es sei denn, sie wird durch spezifischere Einstellungen überschrieben.
- `java.jdt.ls.java.home`: Eine spezifischere Einstellung, die `java.home` für den Java Language Server überschreibt. Wenn diese nicht gesetzt ist, greift der Language Server auf `java.home` zurück.

Wenn Sie also `java.jdt.ls.java.home` setzen, hat diese Einstellung Vorrang für die Operationen des Language Servers, sodass Sie ein anderes JDK für Sprachfunktionen verwenden können als z.B. für Ausführungs- oder Debugging-Aufgaben.

Wie man es konfiguriert

Da Sie Windows verwenden und VSCode 1.96.4 mit dem Java Extension Pack, hier ist, wie Sie es einstellen:

1. Einstellungen öffnen:

- Drücken Sie `Ctrl + ,` für die Einstellungen-Benutzeroberfläche oder verwenden Sie `Ctrl + Shift + P` und geben Sie **“Open Settings (JSON)”** ein, um `settings.json` zu bearbeiten.

2. Pfad setzen:

- **In der Einstellungen-Benutzeroberfläche:**

- Suchen Sie nach **“java.jdt.ls.java.home”**.
- Geben Sie den vollständigen Pfad zu Ihrem JDK ein (z.B. `C:\Program Files\Java\jdk-17.0.1`).

- **In `settings.json`:**

- Fügen Sie diese Zeile hinzu (ersetzen Sie durch Ihren JDK-Pfad):

```
"java.jdt.ls.java.home": "C:\\Program Files\\Java\\jdk-17.0.1"
```

- Beispiel `settings.json`:

```
{  
  "java.jdt.ls.java.home": "C:\\Program Files\\Java\\jdk-17.0.1"  
}
```

3. JDK-Pfad überprüfen:

- Stellen Sie sicher, dass der Pfad auf ein JDK-Verzeichnis zeigt, das ein `bin`-Verzeichnis mit `javac.exe` und `java.exe` enthält.

4. VSCode neu laden:

- Laden Sie das Fenster neu (`Ctrl + Shift + P` > **“Reload Window”**) oder starten Sie VSCode neu, um die Änderung zu übernehmen.

Wann man es verwendet

Sie würden `java.jdt.ls.java.home` typischerweise setzen, wenn:

- Sie mehrere JDKs installiert haben und möchten, dass der Language Server ein bestimmtes verwendet (z.B. JDK 17), während andere Java-Aufgaben (wie das Ausführen von Code) möglicherweise ein anderes JDK über `java.home` verwenden.
- Sie Language Server-Probleme beheben (z.B. er erkennt das richtige JDK nicht automatisch).
- Ihr Projekt spezifische JDK-Versionen für Sprachfunktionen erfordert.

Wenn Sie es nicht setzen, verwendet der Language Server das JDK aus `java.home` (oder die Systemumgebungsvariable `JAVA_HOME` als letztes Mittel).

Praktisches Beispiel

Angenommen, Sie haben JDK 8 und JDK 17 installiert: - Setzen Sie "java.home": "C:\\Program Files\\Java\\jdk1.8.0_ für die allgemeine Verwendung. - Setzen Sie "java.jdt.ls.java.home": "C:\\Program Files\\Java\\jdk-17.0.1", um sicherzustellen, dass der Language Server JDK 17 für die Unterstützung moderner Java-Syntax verwendet.

Auf diese Weise können Sie mit JDK 8 debuggen/ausführen, aber Sprachunterstützung für JDK 17-Funktionen erhalten.

Klarer? Wenn Sie sich fragen, wie das zu Ihrer Einrichtung passt oder Hilfe bei der weiteren Anpassung benötigen, lassen Sie es mich wissen! 😊