AGENDA

Java Streams Basics – Praxis-Grundlagen für Einsteiger:innen

1 Tag · Individueller Termin · Vor Ort oder Online · Deutsch / Englisch · Anpassbare Agenda · Teilnahmezertifikat

Wir wechseln im Kurs zwischen kurzen Theorieeinheiten, praktischen Codebeispielen und interaktiven Übungen. So lernt ihr die Grundlagen der Stream API strukturiert und praxisnah kennen.

Funktionale Programmierung in Java:

- Lambda-Ausdrücke: Syntax, Typen und Verwendung
- Funktionale Interfaces (`Consumer`, `Supplier`, `Function`, `Predicate`, etc.)
- Methodenreferenzen: Static, Instance, Constructor References
- Praktische Übungen: Lambda-Ausdrücke vs. anonyme Klassen

Stream-Grundlagen:

- Was ist ein Stream? Konzeptuelle Unterschiede zu Collections
- Stream-Pipeline Anatomie: Source → Intermediate Operations → Terminal Operations
- Lazy Evaluation und ihre Auswirkungen
- Stream-Erstellung: von Collections, Arrays, Files, Ranges
- Primitive Streams: `IntStream`, `LongStream`, `DoubleStream`
- Praktische Übungen: Erste Stream-Pipelines mit primitiven Streams

Intermediate Operations Deep Dive:

- Transformationen mit `map()`, `flatMap()`, `filter()`
- Auswahl und Sortierung: `distinct()`, `sorted()`, `limit()`, `skip()`
- Debugging mit `peek()` – richtig eingesetzt
- Chaining von Operations und Performance-Aspekte
- Praktische Übungen: Komplexere Datenverarbeitung

Terminal Operations:

- `collect()` – der vielseitigste Terminal Operator
- Eingebaute Collectors: `toList()`, `toSet()`, `toMap()` – Stream to Map leicht gemacht
- `Stream.toList()` und wie es sich von `Collectors.toList()` unterscheidet
- `forEach()`, `forEachOrdered()`
- `reduce()` für Aggregationen
- find-Operationen: `findFirst()`, `findAny()` – Rückgabetyp `Optional` verstehen und richtig verwenden
- Matching-Operationen: `allMatch()`, `anyMatch()`, `noneMatch()`
- Praktische Übungen: Terminal Operations mit Standard-Collectors