Javax JSON-Verarbeitung

Ich erkläre, wie man das `javax.json`-Paket verwendet, das Teil der Java API für JSON-Verarbeitung (JSON-P) ist, um mit JSON-Daten in Java zu arbeiten. Hier ist eine umfassende Anleitung:

1. Fügen Sie zunächst die Abhängigkeit zu Ihrem Projekt hinzu (wenn Maven verwendet wird):

```
<dependency>
    <groupId>javax.json</groupId>
    <artifactId>javax.json-api</artifactId>
    <version>1.1.4</version>
</dependency>
<!-- Implementierung (wie GlassFish) -->
<dependency>
    <groupId>org.glassfish</groupId>
    <artifactId>javax.json</artifactId>
    <version>1.1.4</version>
</dependency>
```

2. Grundlegende Beispiele zur Verwendung von `javax.json`:

a) Erstellen eines JSON-Objekts:

```
import javax.json.Json;
import javax.json.JsonObject;
import javax.json.JsonArray;

public class JsonExample {
    public static void main(String[] args) {
        // Erstellen eines JSON-Objekts
        JsonObject jsonObject = Json.createObjectBuilder()
            .add("name", "John Doe")
            .add("age", 30)
            .add("isStudent", false)
            .add("address", Json.createObjectBuilder()
                .add("street", "123 Main St")
                .add("city", "New York"))
            .build();

        System.out.println(jsonObject.toString());
    }
}
```

b) Erstellen eines JSON-Arrays:

```
JsonArray jsonArray = Json.createArrayBuilder()
    .add("Apple")
    .add("Banana")
    .add(Json.createObjectBuilder()
        .add("name", "Orange")
        .add("price", 0.99))
    .build();
```

c) Parsen von JSON aus einer Zeichenkette:

```
import javax.json.JsonReader;
import java.io.StringReader;

String jsonString = "{\"name\":\"Jane\",\"age\":25}";
JsonReader jsonReader = Json.createReader(new StringReader(jsonString));
JsonObject parsedObject = jsonReader.readObject();
jsonReader.close();

// Werte abrufen
String name = parsedObject.getString("name");
int age = parsedObject.getInt("age");
```

d) Schreiben von JSON in eine Zeichenkette:

```
import javax.json.JsonWriter;
import java.io.StringWriter;

JsonObject jsonObject = Json.createObjectBuilder()
    .add("title", "Example")
    .build();

StringWriter stringWriter = new StringWriter();
JsonWriter jsonWriter = Json.createWriter(stringWriter);
jsonWriter.writeObject(jsonObject);
jsonWriter.close();

String result = stringWriter.toString();
```

3. Wichtige Klassen und Schnittstellen:

- `Json`: Fabrikklasse zum Erstellen von JSON-Buildern und -Lesern
- `JsonObject`: Repräsentiert ein JSON-Objekt
- `JsonArray`: Repräsentiert ein JSON-Array
- `JsonObjectBuilder`: Erstellt JSON-Objekte
- `JsonArrayBuilder`: Erstellt JSON-Arrays
- `JsonReader`: Liest JSON aus einer Eingabequelle
- `JsonWriter`: Schreibt JSON in eine Ausgabequelle

4. Umgang mit verschiedenen JSON-Werttypen:

```
JsonObject obj = // ... ein JSON-Objekt
String stringValue = obj.getString("key");
int intValue = obj.getInt("key");
boolean boolValue = obj.getBoolean("key");
JsonObject nestedObj = obj.getJsonObject("key");
JsonArray arrayValue = obj.getJsonArray("key");

// Überprüfen, ob der Schlüssel existiert und null-Handling
if (obj.containsKey("key") && !obj.isNull("key")) {
    String value = obj.getString("key");
}
```

5. Beispiel für Streaming-API (für große JSON):

```
import javax.json.stream.JsonParser;
import java.io.StringReader;

String json = "{\"name\":\"John\",\"scores\":[1,2,3]}";
JsonParser parser = Json.createParser(new StringReader(json));
while (parser.hasNext()) {
    JsonParser.Event event = parser.next();
    switch (event) {
        case KEY_NAME:
            System.out.println("Key: " + parser.getString());
            break;
        case VALUE_STRING:
            System.out.println("String: " + parser.getString());
            break;
        case VALUE_NUMBER:
            System.out.println("Number: " + parser.getBigDecimal());
            break;
```

```
    }
}
parser.close();
```

Tips: - Schließen Sie `JsonReader`, `JsonWriter` und `JsonParser` immer, wenn Sie fertig sind - Behandeln Sie `javax.json.JsonException` für mögliche Parsing-Fehler - Verwenden Sie die Streaming-API (`JsonParser`) für große JSON-Dokumente - Die API ist unveränderlich - Builder erstellen neue Instanzen

Dies deckt die Grundlagen der Verwendung von `javax.json` ab. Lassen Sie mich wissen, wenn Sie spezifischere Beispiele benötigen oder einen bestimmten Anwendungsfall im Sinn haben!