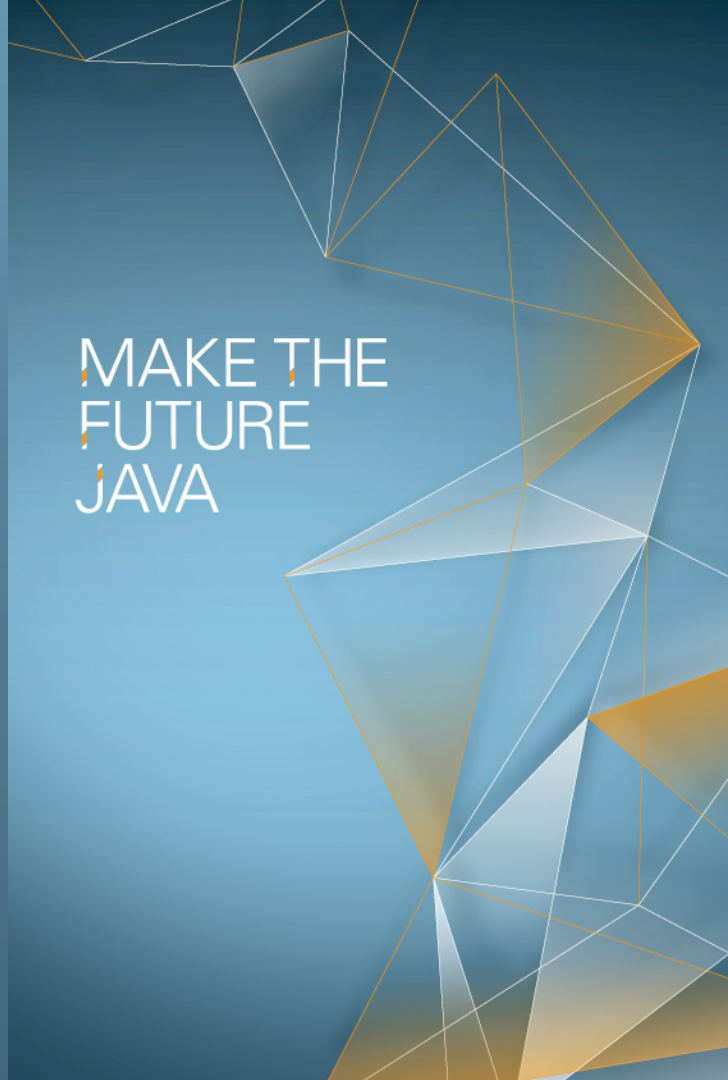
ORACLE

Overview of JSR 353 : Java API for JSON Processing

Anissa Lam (梁鈺儀)
Principal Member of Technical Staff



The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Overview

- HTML5 is a key theme of Java EE 7
  - WebSocket, RESTful webservices, JSON etc
- JSON Processing JSR
  - New to Java EE (and SE too)
  - JAX-RS native support for JSON Processing
  - In general, many uses in web applications

JSON Overview

- JSON is a light-weight data exchange format
  - Minimal, textual and a subset of JavaScript
  - Easy for humans/machines to read and write
  - For e.g.:

```
{"name":"Bob", "age":20, "phone":["276 1234", "123 4567"]}
```

  - Used heavily in RESTful web services, configuration, databases, browser/server communication

ORACLE

JSON Overview

- JSON is used by popular web sites in their RESTful web services
  - Facebook, Twitter, Amazon, …
  - Twitter Streaming API discontinues XML

JSON Standard API

- Parsing/Processing JSON
  - Similar to JAXP
  -
- Data binding : JSON text <=> Java Objects
  - Similar to JAXB

- Two JSRs
  - Processing/Parsing part of Java EE 7
  - Binding targeted Java EE 8

ORACLE

JSR 353 :Java API for JSON Processing

- Streaming API to produce/consume JSON
  - Low-level, efficient way to parse/generate JSON
  - Similar to StAX API in XML world

- Object model API to represent JSON
  - Simple, easy to use high-level API
  - Similar to DOM API in XML world

ORACLE

Streaming API - `JsonParser`

- `JsonParser`
  - Parses JSON in a streaming way from input sources
  - Similar to StAX's `XMLStreamReader`, a pull parser

- Created using
  - `Json.createParser(…)`
  - `Json.createParserFactory().createParser(…)`

Streaming API - `JsonParser`

- Optionally, configured with features

- Parser state events
  - START_ARRAY
  - START_OBJECT
  - KEY_NAME
  - VALUE_STRING VALUE_NUMBER VALUE_TRUE, VALUE_FALSE, VALUE_NULL
  - END_OBJECT
  - END_ARRAY

ORACLE

Streaming API - `JsonParser`

```
{
    "firstName": "John", "lastName": "Smith", "age": 25,
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

Streaming API - JsonParser

START_OBJECT

```
{
    "firstName": "John", "lastName": "Smith", "age": 25,
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

Streaming API - JsonParser

```
{                    KEY_NAME
    "firstName": "John", "lastName": "Smith", "age": 25,
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

Streaming API - JsonParser

```
{
    "firstName": "John", "lastName": "Smith", "age": 25,
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

VALUE_STRING

Streaming API - JsonParser

```
{
    "firstName": "John", "lastName": "Smith", "age": 25,
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

VALUE_NUMBER

Streaming API - JsonParser

```
{
    "firstName": "John", "lastName": "Smith", "age": 25,
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

START_ARRAY

Streaming API - JsonParser

```
{
    "firstName": "John", "lastName": "Smith", "age": 25,
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

END_ARRAY

Streaming API - JsonParser

```
{
    "firstName": "John", "lastName": "Smith", "age": 25,
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
Iterator<Event> it = parser.iterator();
Event event = it.next();                    // START_OBJECT
event = it.next();                          // KEY_NAME
event = it.next();                          // VALUE_STRING
String name = parser.getString();           // "John"
```

Streaming API - `JsonGenerator`

- `JsonGenerator` – Generates JSON in a streaming way to output sources
  - Similar to StAX's **`XMLStreamWriter`**

- Created using
  - `Json.createGenerator(…)`
  - `Json.createGeneratorFactory().createGenerator(…)`

- Optionally, configured with features
  - For e.g. pretty printing
- Allows method chaining

Streaming API - JsonGenerator

```
JsonGenerator gene =  …
gene.writeStartArray()
    .writeStartObject()
        .write("type", "home")
        .write("number", "212 555-1234")
    .writeEnd()
    .writeStartObject()
        .write("type", "fax")
        .write("number", "646 555-4567")
    .writeEnd()
.writeEnd()
.close();
```

```
[
  {
    "type": "home”,
    "number": "212 555-1234"
  },
  {
    "type": "fax”,
    "number": "646 555-4567"
  }
]
```

Object Model/Tree API

- JSON builders – Builds `JsonObject` and `JsonArray`
  - `JsonObjectBuilder`
  - `JsonArrayBuilder`

- `JsonReader`
  - Reads `JsonObject` and `JsonArray` from input source
- `JsonWriter`
  - Writes `JsonObject` and `JsonArray` to output source

Object Model API - `JsonObject`

- Immutable **`Map<String, JsonValue>`** to hold name/value pairs
- Convenient accessor methods

```
JsonObject obj = …;
Set<String> names = obj.keySet();      // standard map method

String str = obj.getString("name");     // convenient string accessor
int num = obj.getInt("age");           // convenient number accessor
```

Object Model API - `JsonObject`

- Immutable `Map<String, JsonValue>` to hold name/value pairs
- Convenient accessor methods

```
JsonObject obj = …;
Set<String> names = obj.keySet();      // standard map method

String str = obj.getString("name");     // convenient string accessor
int num = obj.getInt("age");           // convenient number accessor
```

Object Model API - `JsonObject`

- Immutable `Map<String, JsonValue>` to hold name/value pairs
- Convenient accessor methods

```
JsonObject obj = …;
Set<String> names = obj.keySet();      // standard map method

String str = obj.getString("name");     // convenient string accessor
int num = obj.getInt("age");           // convenient number accessor
```

Object Model API - `JsonObject`

- Immutable `Map<String, JsonValue>` to hold name/value pairs
- Convenient accessor methods

```
JsonObject obj = …;
Set<String> names = obj.keySet();      // standard map method

String str = obj.getString("name");     // convenient string accessor
int num = obj.getInt("age");           // convenient number accessor
```

Object Model API - `JsonArray`

- Immutable `List<JsonValue>` for sequence of values
- Convenient accessor methods

```
JsonArray arr = …;
JsonValue value = arr.get(0);      // standard List method

String str = arr.getString(0);        // convenient string accessor
int num = arr.getInt(2);           // convenient number accessor
```

Object Model API - `JsonArray`

- Immutable `List<JsonValue>` for sequence of values
- Convenient accessor methods

```
JsonArray arr = …;
JsonValue value = arr.get(0);      // standard List method

String str = arr.getString(0);        // convenient string accessor
int num = arr.getInt(2);           // convenient number accessor
```

Object Model API - `JsonArray`

- Immutable `List<JsonValue>` for sequence of values
- Convenient accessor methods

```
JsonArray arr = …;
JsonValue value = arr.get(0);      // standard List method

String str = arr.getString(0);        // convenient string accessor
int num = arr.getInt(2);           // convenient number accessor
```

Object Model API - `JsonArray`

- Immutable `List<JsonValue>` for sequence of values
- Convenient accessor methods

```
JsonArray arr = …;
JsonValue value = arr.get(0);      // standard List method

String str = arr.getString(0);        // convenient string accessor
int num = arr.getInt(2);           // convenient number accessor
```

Object Model API – JSON Builders

- Builders to build `JsonObject` and `JsonArray` from scratch
- Allows method chaining
- Can also use existing `JsonObject` and `JsonArray` in a builder

```
//Creates a builder factory
JsonBuilderFactory  factory = Json.createBuilderFactory(config);
factory.createObjectBuilder().build;

// builds empty JSON object
JsonObject obj = Json.createObjectBuilder().build();
```

Object Model API – JSON Builders

```
JsonArray value =
  Json.createArrayBuilder()
    .add(Json.createObjectBuilder()
      .add("type", "home")
      .add("number", "212 555-1234"))
    .add(Json.createObjectBuilder()
      .add("type", "fax")
      .add("number", "646 555-4567"))
  .build();
```

```
[
  {
    "type": "home”,
    "number": "212 555-1234"
  },
  {
    "type": "fax”,
    "number": "646 555-4567"
  }
]
```

Object Model API - `JsonReader`

- Reads `JsonObject` and `JsonArray` from input source
  - I/O Reader, InputStream (+ encoding)
- Optionally, configured with features

```
// Reads a JSON object
try(JsonReader reader = Json.createReader(io)) {
    JsonObject obj = reader.readObject();
}
```

Object Model API - `JsonWriter`

- Writes JsonObject and JsonArray to output source
  - I/O Writer, OutputStream (+ encoding)
- Optionally, configured with features.
  - pretty printing, single quoted

```
// Writes a JSON object
try(JsonWriter writer = Json.createWriter(io)) {
    writer.writeObject(obj);
}
```

JAX-RS JSON Support

- `JsonObject/JsonArray/JsonStructure` can be used as parameter and return types of resource methods

```
@Produces("application/json")
public JsonObject getBook(String id) {
 return …;
}
```

API Summary

- API provides :
  - Parsing input streams into immutable objects or event streams
  - Writing event streams or immutable objects to output streams
  - Programmatically navigating immutable objects
  - Programmatically building immutable objects with builders

- API becomes
  - A base for building data binding, transformation, querying, or other manipulation APIs

Download the Java EE 7 SDK

http://www.oracle.com/javaee

Stay Current

https://twitter.com/glassfish

https://www.facebook.com/glassfish

https://blogs.oracle.com/theaquarium

http://www.youtube.com/GlassFishVideos

http://www.glassfish.org