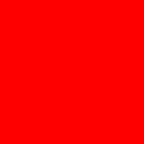




ORACLE[®]

JSR-353: Java API for Processing JSON

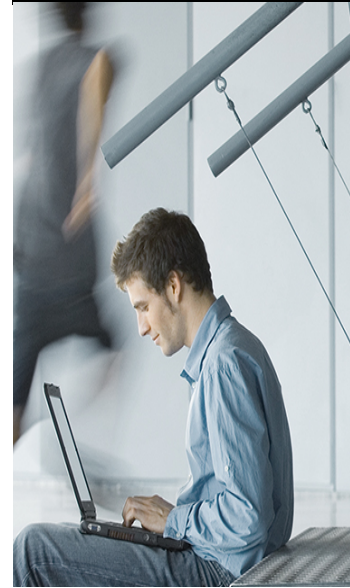
Jitendra Kotamraju



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.

Agenda

- Overview
- Use Cases : JAX-RS
- Standardization
- API



Overview

JSON

- JSON is a light-weight data exchange format
 - Easy for humans/machines to read and write
 - For e.g.:

```
{"name": "Bob", "age": 20, "phone": ["276 1234", "123 4567"]}
```
- JSON is used by popular web sites in their RESTful web services
 - Facebook, Twitter, Amazon, ...
 - Twitter Streaming API discontinues XML

Overview

JSON usages : Amazon CloudSearch

<http://search-domainname-domainid.us-east-1.cloudsearch.amazonaws.com/2011-02-01/search?q=star+wars>

```
{
  "rank": "-text_relevance",
  "match-expr": "(label 'star wars')",
  "hits": {
    "found": 7,
    "start": 0,
    "hit": [
      {"id": "tt0086190"},
      {"id": "tt0120915"},
      {"id": "tt0121766"}]
  },
  ...
}
```

Overview

JSON usages: Twitter Search

```
http://search.twitter.com/search.json?q=JSON
```

```
{  
  "created_at": "Thu, 06 Sep 2012 21:45:04 +0000",  
  "from_user": "loggly",  
  "metadata": {"result_type": "recent"},  
  "text": "Good news if you log JSON. (And another reason to  
switch to JSON if you haven't already.) http://t.co/  
9Dz2JP41",  
  ...  
}
```

JAX-RS

XML Usage

- JAX-RS applications handle XML using JAXP API

```
@Produces("application/xml")
public Source getBook(String id) {
    return new StreamSource(...);
}
```

JAX-RS

XML Usage

- JAX-RS applications handle XML using JAXB API

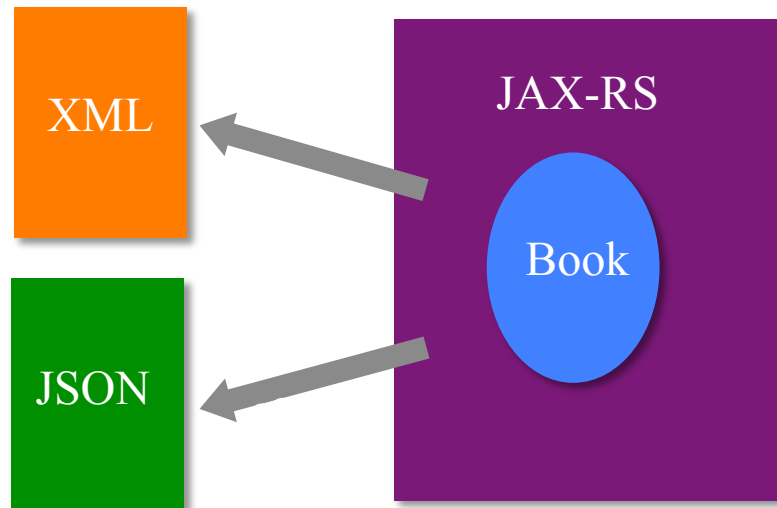
```
@Produces("application/xml")  
public Book getBook(String id) {  
    return new Book(...);  
}
```


JAX-RS

DataBinding

- JAX-RS content negotiation

```
@Produces({"application/xml", "application/json"})  
public Book getBook(String id) {  
    return new Book();  
}
```



JAX-RS

JSON Solutions & Limitations

- A custom MessageBodyWriter that converts to JSON
 - JSONObject (For e.g. json.org's API) → JSON
 - JAXB → StAX → JSON (For e.g. using jettison)
 - POJO/JAXB → JSON (For e.g. using jackson, eclipseLink etc.)
- No standard API
- Some solutions have technical limitations
- Applications/Frameworks need to bundle the libraries



Standard API

Advantages

- Application can use standard types
- Leaner, portable applications

Standard API

Contents

- Parsing/Processing JSON
- Data binding : JSON text <-> Java Objects
- Two JSRs: Processing/Parsing (JSON-P), Binding (JSON-B)
 - Similar to JAXP and JAXB

Java API for Processing JSON (JSON-P)

JSR-353

- Streaming API to produce/consume JSON
 - Similar to StAX API in XML world
- Object model API to represent JSON
 - Similar to DOM API in XML world

JSR-353: Java API for Processing JSON

EG

- Corporate Members
 - RedHat, Twitter, Oracle
- Individual Members
 - Christian Grobmeier, Jorn Horstmann, Werner Keil
- And, user community !!

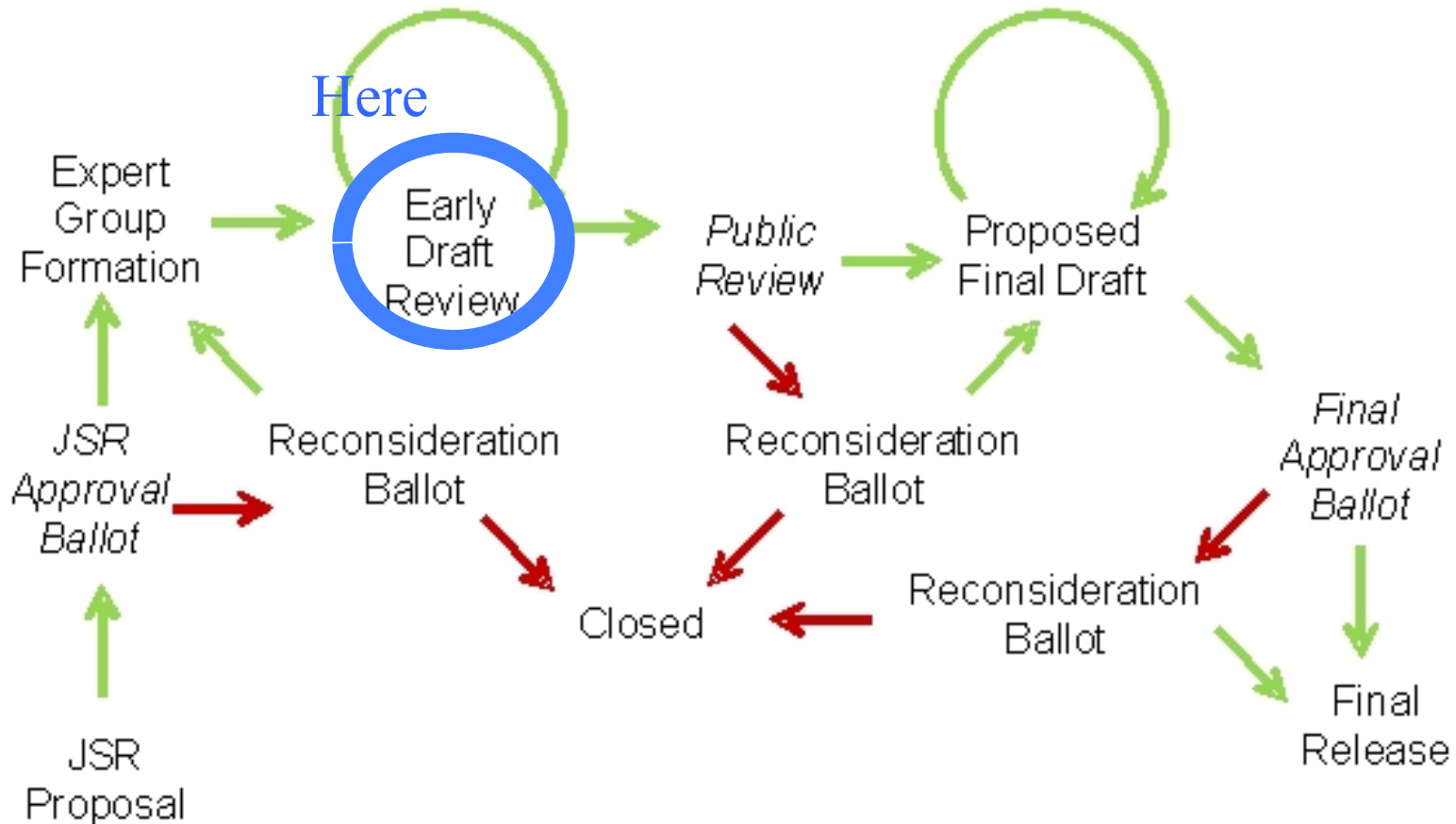
JSR-353: Java API for Processing JSON

Transparency

- json-processing-spec java.net open source project is used for JSR-353
- Mailing lists:
 - users@json-processing-spec.java.net
 - jsr353-experts@json-processing-spec.java.net
 - Lists are archived (publicly readable)
- Issue Tracker:
 - http://java.net/jira/browse/JSON_PROCESSING_SPEC

JSR-353: Java API for Processing JSON

Status



JSR-353

Schedule

- Align with Java EE 7 schedule
 - Early Draft – Sep 2012
 - Public Review – Dec 2012
 - Proposed Final Draft – Mar 2013
 - Final Release – Apr 2013

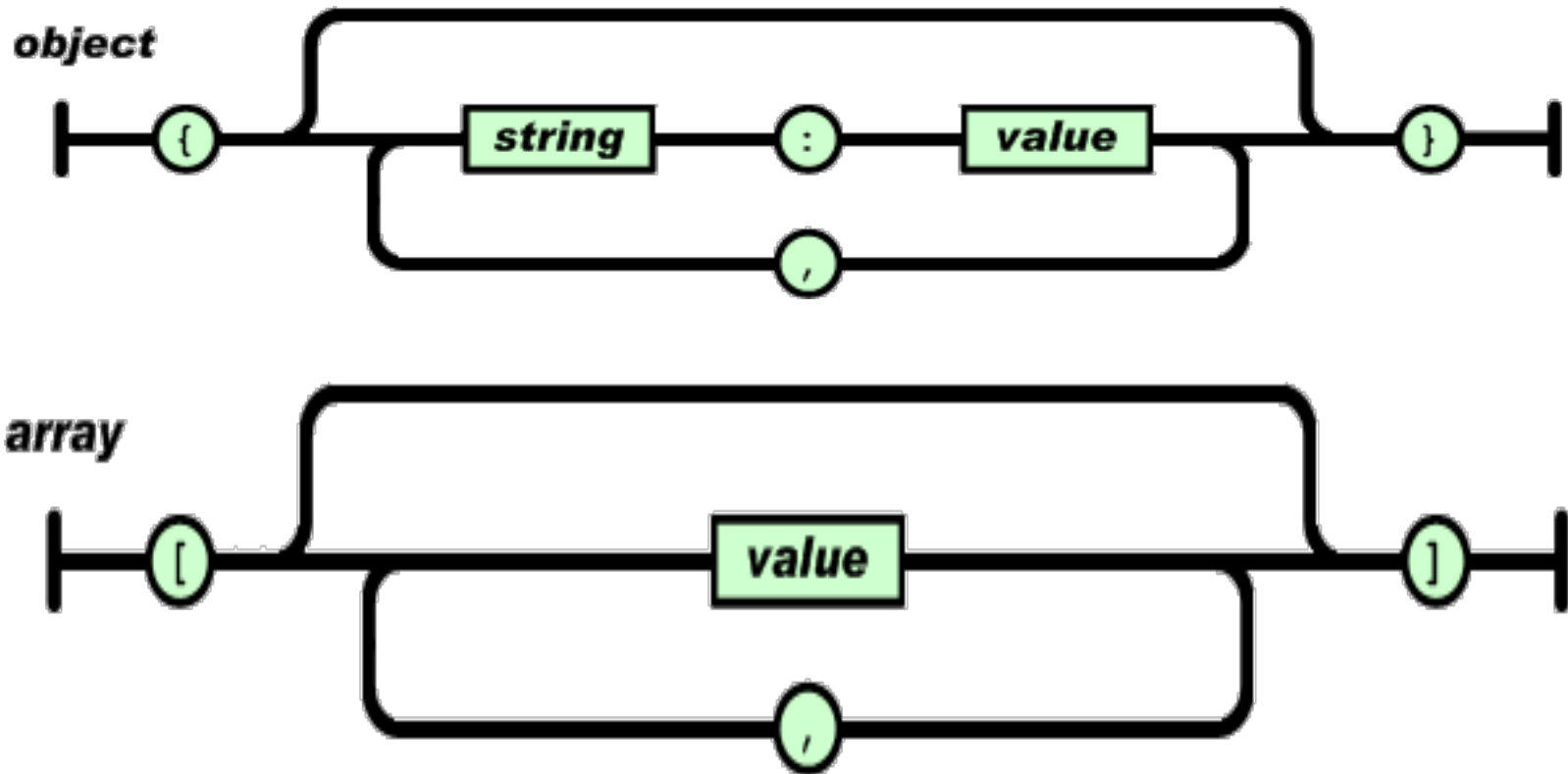
JSR-353 RI

Open Source Project

- jsonp java.net open source project is used for JSR-353 RI
- Up-to-date w.r.t spec

JSR-353: API

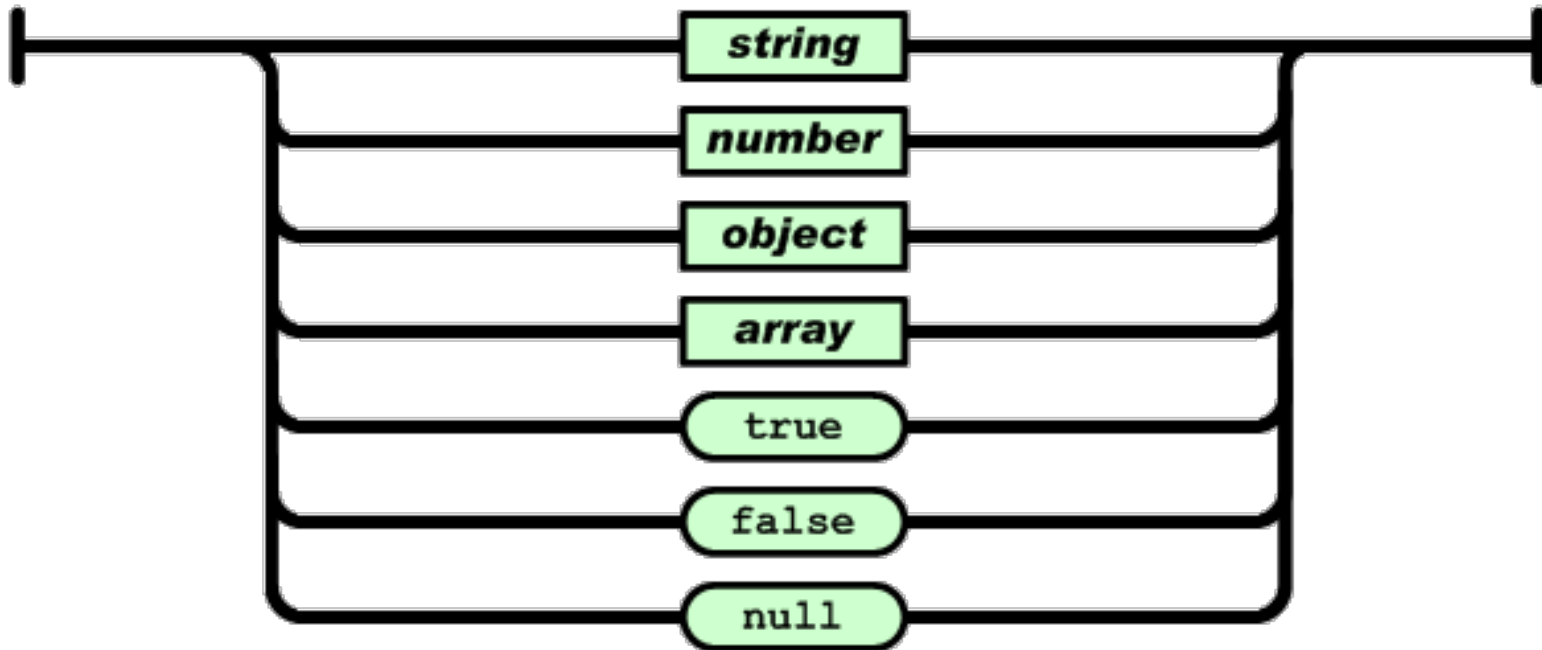
JSON Grammar



JSR-353: API

JSON Grammar

value



JSR-353: API

Streaming & Object Model

- Streaming API
 - Low-level, efficient way to parse/generate JSON
 - Provides pluggability for parsers/generators
- Object Model API
 - Simple, easy to use high-level API
 - Implemented on top of streaming API

JSR-353 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(...)`
- 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`

JSR-353 Streaming API

JsonParser

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

JSR-353 Streaming API

JsonParser

START_OBJECT



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


JSR-353 Streaming API

JsonParser

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

JSR-353 Streaming API

JsonParser

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

VALUE_STRING



JSR-353 Streaming API

JsonParser

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

↑ VALUE_NUMBER

JSR-353 Streaming API

JsonParser

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

START_ARRAY



JSR-353 Streaming API

JsonParser

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

END_ARRAY ↑

JSR-353 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"
```

JSR-353 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
- Cannot mix array and object methods

JSR-353 Streaming API

JsonGenerator

```
JsonGenerator gene = Json.createGenerator(...);
gene.beginArray()
    .beginObject()
        .add("type", "home").add("number", "212 555-1234")
    .endObject()
    .beginObject()
        .add("type", "fax").add("number", "646 555-4567")
    .endObject()
    .endArray()
.close();
[
    { "type": "home", "number": "212 555-1234" },
    { "type": "fax", "number": "646 555-4567" }
]
```


JSR-353 Object Model API

Core classes

- **JsonObject/JsonArray** – JSON object and array structures
 - **JsonString** and **JsonNumber** for string and number values
- **JsonBuilder** – Builds **JsonObject** and **JsonArray**
- **JsonReader** – Reads **JsonObject** and **JsonArray** from input source
- **JsonWriter** – Writes **JsonObject** and **JsonArray** to output source

JSR-353 Object Model API

JsonObject

- Holds name/value pairs and immutable
- Name/value pairs can be accessed as `Map<String, JsonValue>`

```
JsonObject obj = ...;
Map<String, JsonValue> map = obj.getValues();    // as a map

JsonNumber num = obj.getValue("foo", JsonNumber.class);

Set<String> names = obj.getNames();    // all names
```

JSR-353 Object Model API

JsonObject

- Holds name/value pairs and immutable
- Name/value pairs can be accessed as `Map<String, JsonValue>`

```
JsonObject obj = ...;
```

```
Map<String, JsonValue> map = obj.getValues(); // as a map
```

```
JsonNumber num = obj.getValue("foo", JsonNumber.class);
```

```
Set<String> names = obj.getNames(); // all names
```

JSR-353 Object Model API

JsonObject

- Holds name/value pairs and immutable
- Name/value pairs can be accessed as Map<String, JsonValue>

```
JsonObject obj = ...;
Map<String, JsonValue> map = obj.getValues();    // as a map

JsonObject num = obj.getValue("foo", JsonNumber.class);

Set<String> names = obj.getNames();    // all names
```

JSR-353 Object Model API

JsonObject

- Holds name/value pairs and immutable
- Name/value pairs can be accessed as `Map<String, JsonValue>`

```
JsonObject obj = ...;
Map<String, JsonValue> map = obj.getValues();    // as a map

JsonNumber num = obj.getValue("foo", JsonNumber.class);

Set<String> names = obj.getNames();    // all names
```

JSR-353 Object Model API

JSONArray

- Holds a list of values and immutable
- Values can be accessed as `List<JsonValue>`

```
JSONArray arr = ...;
```

```
List<JsonValue> list = arr.getValues(); // as a list
```

```
JsonNumber num = arr.getValue(0, JsonNumber.class);
```

JSR-353 Object Model API

JSONArray

- Holds a list of values and immutable
- Values can be accessed as `List<JsonValue>`

```
JSONArray arr = ...;
```

```
List<JsonValue> list = arr.getValues(); // as a list
```

```
JsonNumber num = arr.getValue(0, JsonNumber.class);
```

JSR-353 Object Model API

JSONArray

- Holds a list of values and immutable
- Values can be accessed as `List<JsonValue>`

```
JSONArray arr = ...;
```

```
List<JsonValue> list = arr.getValues(); // as a list
```

```
JsonNumber num = arr.getValue(0, JsonNumber.class);
```


JSR-353 Object Model API

JsonBuilder

- Builder to build JsonObject and JsonArray from scratch
- Allows method chaining
- Type-safe (cannot mix array and object building methods)

```
// builds empty JSON object  
JsonObject obj = new JsonBuilder().beginObject().endObject().build()
```

JSR-353 Object Model API

JsonBuilder Example

```
JsonArray arr = new JsonBuilder()
    .beginArray()
        .beginObject()
            .add("type", "home").add("number", "212 555-1234")
        .endObject()
        .beginObject()
            .add("type", "fax").add("number", "646 555-4567")
        .endObject()
    .endArray()
    .build();
[
  { "type": "home", "number": "212 555-1234" },
  { "type": "fax", "number": "646 555-4567" }
]
```

JSR-353 Object Model API

JsonReader

- Reads JsonObject and JsonArray from input source
 - i/o Reader, InputStream (+ encoding)
- Optionally, configured with features
- Uses pluggable JsonParser

```
// Reads an empty JSON object
JsonReader reader = new JsonReader(new StringReader("{}"));
JsonObject obj = reader.readObject();
reader.close();
```

JSR-353 Object Model API

JsonWriter

- Writes JsonObject and JsonArray to output source
 - i/o Writer, OutputStream (+ encoding)
- Optionally, configured with features. For e.g. pretty printing
- Uses pluggable JsonGenerator

```
// Writes a JSON object
Writer sw = new StringWriter();
JsonWriter writer = new JsonWriter(sw);
writer.writeObject(obj);
writer.close();
```

JSR-353 API

Configuration

- Configuration is a set of parser/generator features
 - Pretty Printing, Single-Quoted strings
- Supports extensibility (custom features)
- Can be used in streaming & object-model API

```
// Writes a JSON object prettily
Writer sw = new StringWriter();
JsonConfiguration config = new
    JsonConfiguration().withPrettyPrinting();
JsonWriter writer = new JsonWriter(sw, config);
writer.writeObject(obj);
...
```

JSR-353 API

TODO

- Defining equals/hashcode() semantics for JSONArray/JsonObject
- Exception handling

Resources

- <http://json-processing-spec.java.net>
- <http://jsonp.java.net>

Q&A

ORACLE®