

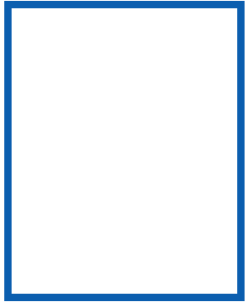
Interoperating with External Services



Antonio Goncalves

@agoncal | www.antoniogoncalves.org

Previous Module



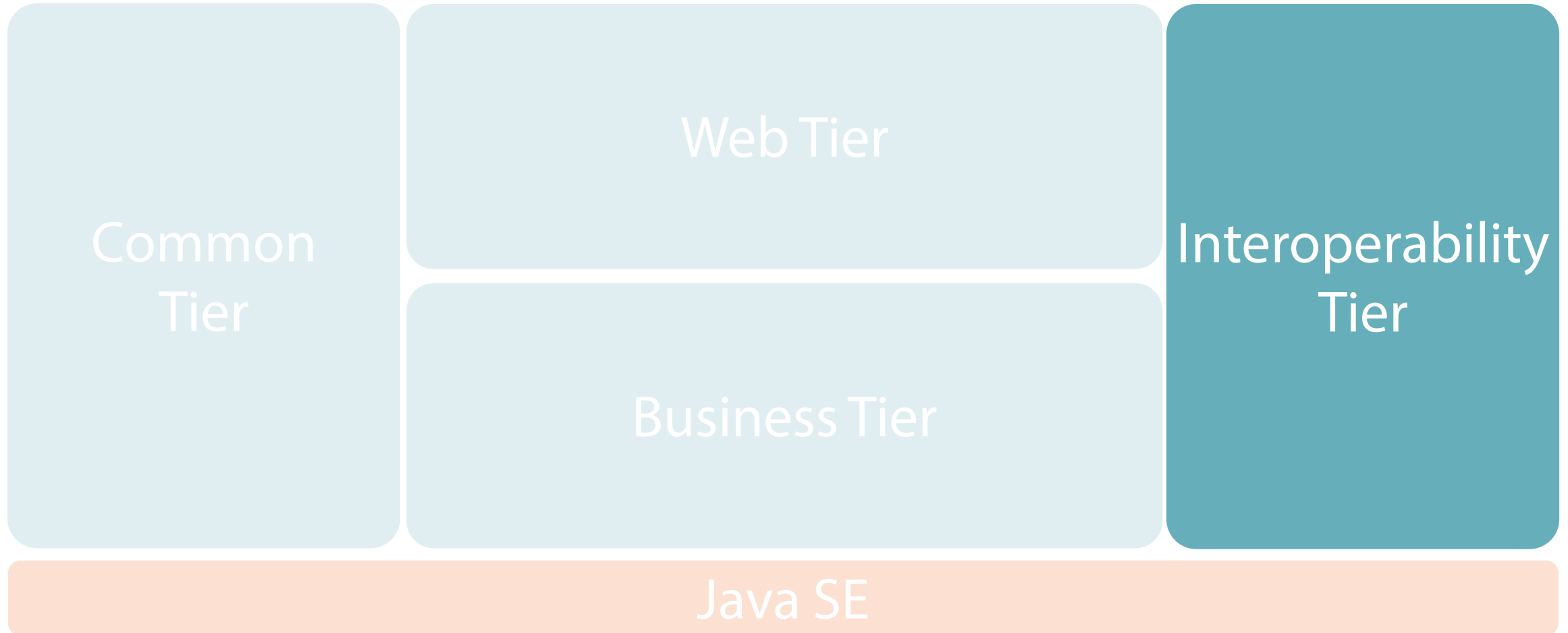
Web tier

Servlets

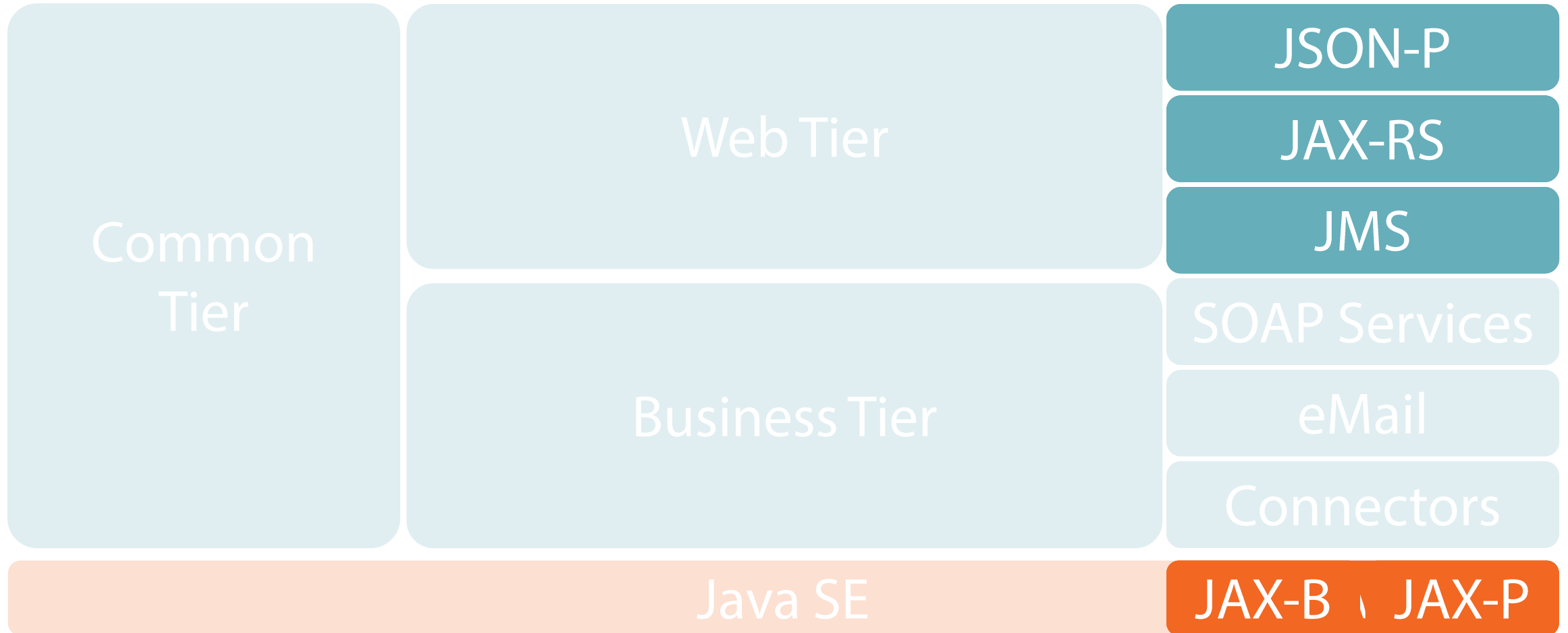
JavaServer Faces

WebSockets

Module Outline



Module Outline

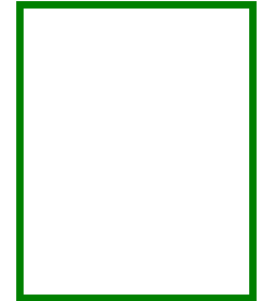


XML & JSON

JAX-B 2.0 & JAX-P 1.2 & JSON-P 1.0

What Is XML?

- eXtensible Markup Language
- World Wide Web Consortium (W3C)
- Data independence and interoperability
- Interchange data
- Meta-language

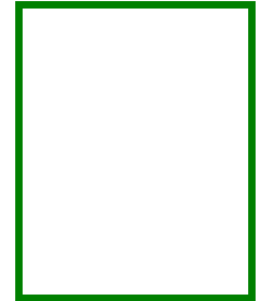


XML Document

```
<?xml version="1.0" encoding="UTF-8" ?>
<book isbn="1234" title="H2G2" unit-cost="63.98">
  <description>Best Scifi book</description>
  <nb-of-pages>123</nb-of-pages>
  <language>ENGLISH</language>
  <category name="IT"/>
  <publisher name="Megadodo"/>
  <authors>
    <author>
      <first-name>Douglas</first-name>
      <last-name>Adams</last-name>
    </author>
    <author>
      <first-name>John</first-name>
      <last-name>Cleese</last-name>
    </author>
  </authors>
</book>
```

What Is JSON?

- JavaScript Object Notation
- Originated with JavaScript
- Less verbose than XML
- Browsers have native JSON support
- Data interchange format



JSON Document

```
{
  "book": {
    "isbn": "1234", "title": "H2G2", "unit-cost": "63.98", "nb-of-pages": "123",
    "description": "Best Scifi book", "language": "ENGLISH",
    "category": { "name": "IT" },
    "publisher": { "name": "Megadodo" },
    "authors": [
      { "first-name": "Douglas", "last-name": "Adams" },
      { "first-name": "John", "last-name": "Cleese" }
    ]
  }
}
```

When to Use XML Over JSON

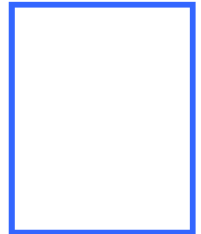
- XML

- Validation
- Contract between systems
- Transform
- Desktop applications



- JSON

- Directly consumable by JavaScript
- Documents are shorter



JAX-B, JAX-P and JSON-P Specifications

- JAX-P 1.3 (JSR 206)
- JAX-B 2.2 (JSR 222)
- JSON-P 1.0 (JSR 353)



Java
Community
Process

The Java Community Process

https://jcp.org/en/jsr/detail?id=353

Rechercher

Press Room | Get Java Here | Search JSRs

Community Development of Java Technology Specifications

JSR Community Expert Group

Summary | Proposal | Detail (Summary & Proposal)

Search JSRs

JSRs: Java Specification Requests

JSR 353: Java™ API for JSON Processing

Stage	Access	Start	Finish
Final Release	Download page	23 May, 2013	
Final Approval Ballot	View results	26 Mar, 2013	08 Apr, 2013
Proposed Final Draft	Download page	27 Feb, 2013	
Public Review Ballot	View results	22 Jan, 2013	04 Feb, 2013
Public Review	Download page	22 Dec, 2012	21 Jan, 2013
Early Draft Review	Download page	07 Sep, 2012	07 Oct, 2012
Expert Group Formation		04 Jan, 2012	07 Jun, 2012
JSR Review Ballot	View results	20 Dec, 2011	03 Jan, 2012
JSR Review		06 Dec, 2011	19 Dec, 2011

My JCP

Sign-in

Register for Site

Use of JCP site is subject to the JCP Terms of Use and the Oracle Privacy Policy

JCP Info

» About JCP

» Get Involved

» Community Resources

» Community News

» FAQ

» Contact Us

Status: Final

JCP version in use: 2.8

Java Specification Participation Agreement version in use: 2.0

Description:

The Java API for JSON Processing (JSON-P) JSR will develop a Java API to process (for e.g. parse, generate, transform and query) JSON.

Expert Group Transparency:

Public Communications

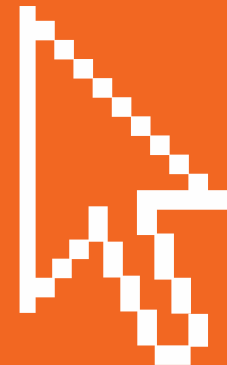
Issue Tracking

XML and JSON

XML

JSON

REST consumers



Understanding Processing

- JAX-P and JSON-P
- Read
 - Object model
 - Streaming
- Write
 - Build
 - Stream

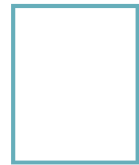
Object model



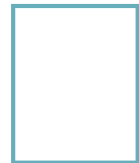
Streaming



← Read



Write →



Reading JSON

```
Path file = Paths.get("books.json");
InputStream stream = Files.newInputStream(file);

try (JsonReader reader = Json.createReader(stream)) {

    JsonArray array = reader.readArray();
    for (int i = 0; i < array.size(); i++) {
        array.getJsonObject(i).getInt("id");
        array.getJsonObject(i).getString("title");
    }
}
```

```
[
  { "id": 1001, "title": "AAAAAAAAAA" },
  { "id": 1002, "title": "BBBBBBBBBB" },
  { "id": 1003, "title": "CCCCCCCCCC" },
  { "id": 1004, "title": "DDDDDDDDDD" },
  { "id": 1005, "title": "EEEEEEEEEE" },
  { "id": 1006, "title": "FFFFFFFFFF" },
  { "id": 1007, "title": "GGGGGGGGGG" },
  { "id": 1008, "title": "HHHHHHHHHH" },
  { "id": 1009, "title": "IIIIIIIIII" },
  { "id": 1010, "title": "JJJJJJJJJJ" },
  { "id": 1011, "title": "KKKKKKKKKK" },
  { "id": 1012, "title": "LLLLLLLLLL" },
  { "id": 1013, "title": "MMMMMMMMMM" },
  { "id": 1014, "title": "NNNNNNNNNN" },
  { "id": 1015, "title": "OOOOOOOOOO" },
  { "id": 1016, "title": "PPPPPPPPPP" },
  { "id": 1017, "title": "QQQQQQQQQQ" },
  { "id": 1018, "title": "RRRRRRRRRR" },
  { "id": 1019, "title": "SSSSSSSSSS" },
  { "id": 1020, "title": "TTTTTTTTTT" },
  { "id": 1021, "title": "UUUUUUUUUU" }
]
```

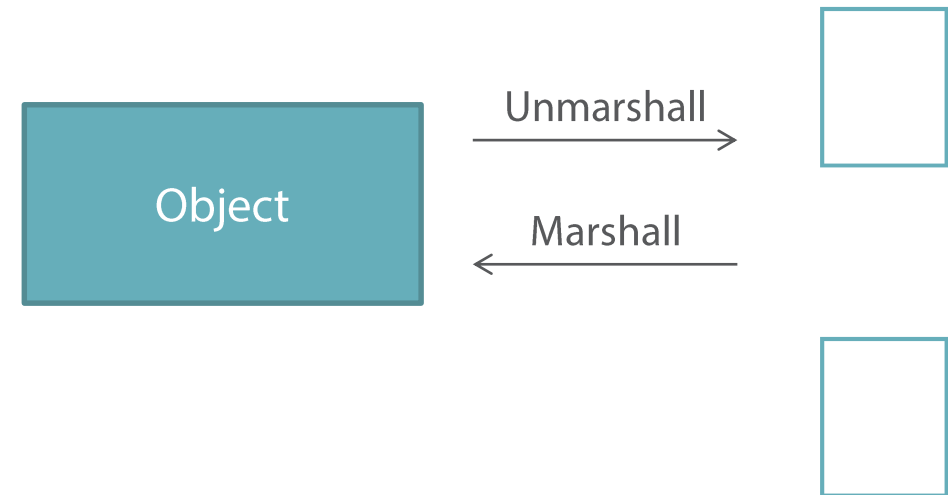
Writing JSON

```
JsonObject jsonBook = Json.createObjectBuilder()
    .add("book", Json.createObjectBuilder()
        .add("isbn", "1234")
        .add("title", "H2G2")
        .add("unit-cost", "63.98")
        .add("nb-of-pages", "123")
        .add("description", "Best Scifi book")
        .add("language", "ENGLISH")
        .add("category", Json.createObjectBuilder()
            .add("name", "IT")
        )
        .add("publisher", Json.createObjectBuilder()
            .add("name", "Megadodo")
        )
        .add("authors", Json.createArrayBuilder()
            .add(Json.createObjectBuilder()
                .add("first-name", "Douglas")
                .add("last-name", "Adams")
            )
            .add(Json.createObjectBuilder()
                .add("first-name", "John")
                .add("last-name", "Cleese")
            )
        )
    ).build();
```

```
{
  "book": {
    "isbn": "1234",
    "title": "H2G2",
    "unit-cost": "63.98",
    "nb-of-pages": "123",
    "description": "Best Scifi book",
    "language": "ENGLISH",
    "category": {
      "name": "IT"
    },
    "publisher": {
      "name": "Megadodo"
    },
    "authors": [
      { "first-name": "Douglas", "last-name": "Adams" },
      { "first-name": "John", "last-name": "Cleese" }
    ]
  }
}
```

Understanding Binding

- JAX-B (and JSON-B)
- Higher level of abstraction
- Based on annotations
- Unmarshall
- Marshall



Binding XML

```
Book book = new Book("1234", "H2G2", "Best Scifi book", 63.98F, 123, ENGLISH);
book.setCategory(new Category("IT"));
book.setPublisher(new Publisher("Megadodo"));
book.addAuthor(new Author("Douglas", "Adams"));
book.addAuthor(new Author("John", "Cleeese"));

JAXBContext context = JAXBContext.newInstance(Book.class);

Marshaller m = context.createMarshaller();
StringWriter writer = new StringWriter();
m.marshal(book, writer);

System.out.println(writer.toString());
```

Binding XML

```
public class Book {  
  
    private Long id;  
  
    private String isbn;  
  
    private String title;  
  
    private Float unitCost;  
    private String description;  
  
    private Integer nbOfPage;  
    private Language language;  
    private Category category;  
    private Publisher publisher;  
  
    private Set<Author> authors = new HashSet<>();  
}
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<book>  
  
    <isbn>1234</isbn>  
    <title>H2G2</title>  
    <unitCost>63.98</unitCost>  
    <description>Best Scifi book</description>  
    <nbOfPage>123</nbOfPage>  
    <language>ENGLISH</language>  
    <category name="IT"/>  
    <publisher name="Megadodo"/>  
    <authors>  
        <first-name>Douglas</first-name>  
        <last-name>Adams</last-name>  
    </authors>  
    <authors>  
        <first-name>John</first-name>  
        <last-name>Cleese</last-name>  
    </authors>  
  
</book>
```

Binding XML

```
@XmlElement
public class Book {

    @XmlTransient
    private Long id;

    private String isbn;

    private String title;

    private Float unitCost,
    private String description;

    private Integer nbOfPage;
    private Language language;
    private Category category;
    private Publisher publisher;

    private Set<Author> authors = new HashSet<>();
}
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<book isbn="1234" title="H2G2" unit-cost="53.98">

    <description>Best Scifi book</description>
    <nbOfPage>123</nbOfPage>
    <language>ENGLISH</language>
    <category name="IT"/>
    <publisher name="Megadodo"/>
    <authors>
        <first-name>Douglas</first-name>
        <last-name>Adams</last-name>
    </authors>
    <authors>
        <first-name>John</first-name>
        <last-name>Cleese</last-name>
    </authors>

</book>
```

Binding XML

```
@XmlElement
public class Book {

    @XmlTransient
    private Long id;
    @XmlAttribute
    private String isbn;
    @XmlAttribute
    private String title;
    @XmlAttribute(name = "unit-cost")
    private Float unitCost;
    private String description;

    private Integer nbOfPages;
    private Language language;
    private Category category;
    private Publisher publisher;

    private Set<Author> authors = new HashSet<>();
}
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<book isbn="1234" title="H2G2" unit-cost="63.98">
```

```
    <description>Best Scifi book</description>
    <nb-of-pages>23</nb-of-pages>
    <language>ENGLISH</language>
    <category name="IT"/>
    <publisher name="Megadodo"/>
    <authors>
        <first-name>Douglas</first-name>
        <last-name>Adams</last-name>
    </authors>
    <authors>
        <first-name>John</first-name>
        <last-name>Cleese</last-name>
    </authors>
```

```
</book>
```

Binding XML

```
@XmlRootElement
public class Book {

    @XmlTransient
    private Long id;
    @XmlAttribute
    private String isbn;
    @XmlAttribute
    private String title;
    @XmlAttribute(name = "unit-cost")
    private Float unitCost;
    private String description;
    @XmlElement(name = "nb-of-pages")
    private Integer nbOfPage;
    private Language language;
    private Category category;
    private Publisher publisher;

    private Set<Author> authors = new HashSet<>();
}
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<book isbn="1234" title="H2G2" unit-cost="63.98">
```

```
    <description>Best Scifi book</description>
    <nb-of-pages>123</nb-of-pages>
    <language>ENGLISH</language>
    <category name="IT"/>
    <publisher name="Megadodo"/>
    <authors>
        <author>
            <first-name>Douglas</first-name>
            <last-name>Adams</last-name>
        </author>
        <author>
            <first-name>John</first-name>
            <last-name>Cleese</last-name>
        </author>
    </authors>
</book>
```

JAX-P, JAX-P and JSON-P Packages

Package	Description
➡ <code>javax.xml.bind</code>	Marshalling, unmarshalling APIs
➡ <code>javax.xml.bind.annotation</code>	Binding annotations
➡ <code>javax.xml.transform</code>	Transformation APIs
➡ <code>javax.xml.parsers</code>	Object model and streaming APIs
➡ <code>javax.json</code>	Core JSON APIs
➡ <code>javax.json.stream</code>	Streaming APIs

XML and JSON

Binding

Customizing binding

Processing JSON

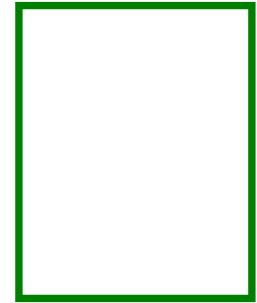


REST Web Services

Java API for RESTful Web Services (JAX-RS) 2.0

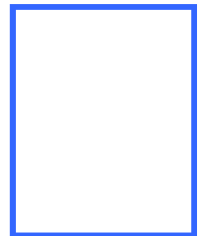
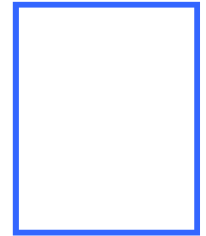
What Is REST?

- REpresentational State Transfer
- Architectural style
- Based on how the Web works
- HTTP
- Very robust transport protocol
- Stateless
- Handle high load and scale better



When to Use REST

- SOAP vs REST Web services
- HTTP is used everywhere
- Many different clients
- Reduce the client/server coupling
- Updating the server regularly
- Easy to build



JAX-RS Specification

- JAX-RS 2.0
- JSR 339
- <http://jcp.org/en/jsr/detail?id=339>



Java
Community
Process

A screenshot of a web browser displaying the Java Community Process (JCP) website. The browser's address bar shows the URL "https://jcp.org/en/jsr/detail?id=356". The page header includes the Java logo, the text "The Java Community Process", and a navigation bar with links like "Press Room", "Get Java Here", and a search bar. Below the header, there's a diagram illustrating the "Community Development of Java Technology Specifications" process. The main content area is titled "JSRs: Java Specification Requests" and "JSR 356: Java™ API for WebSocket". It features a table with columns for "Stage", "Access", "Start", and "Finish", listing various milestones from "Maintenance Release" to "JSR Review". To the left of the table, there are links for "JSRs by Platform", "JSRs by Technology", "JSRs by Stage", "JSRs by Committee", and "List of All JSRs". Below the table, there's a section for "My JCP" with a "Sign-in" button and a "Register for Site" link. At the bottom, there's a "JCP Info" section with links for "About JCP", "Get Involved", "Community Resources", "Community News", "FAQ", and "Contact Us". The status of JSR 356 is listed as "Maintenance" with JCP version 2.8 and Java Specification Participation Agreement version 2.0 in use. A description at the bottom states: "The Java API for WebSocket JSR will define a standard API for creating WebSocket applications."

JAX-RS Implementations

- Jersey
- Apache CXF
- RESTEasy
- Restlet



REST Services

Expose catalog of items

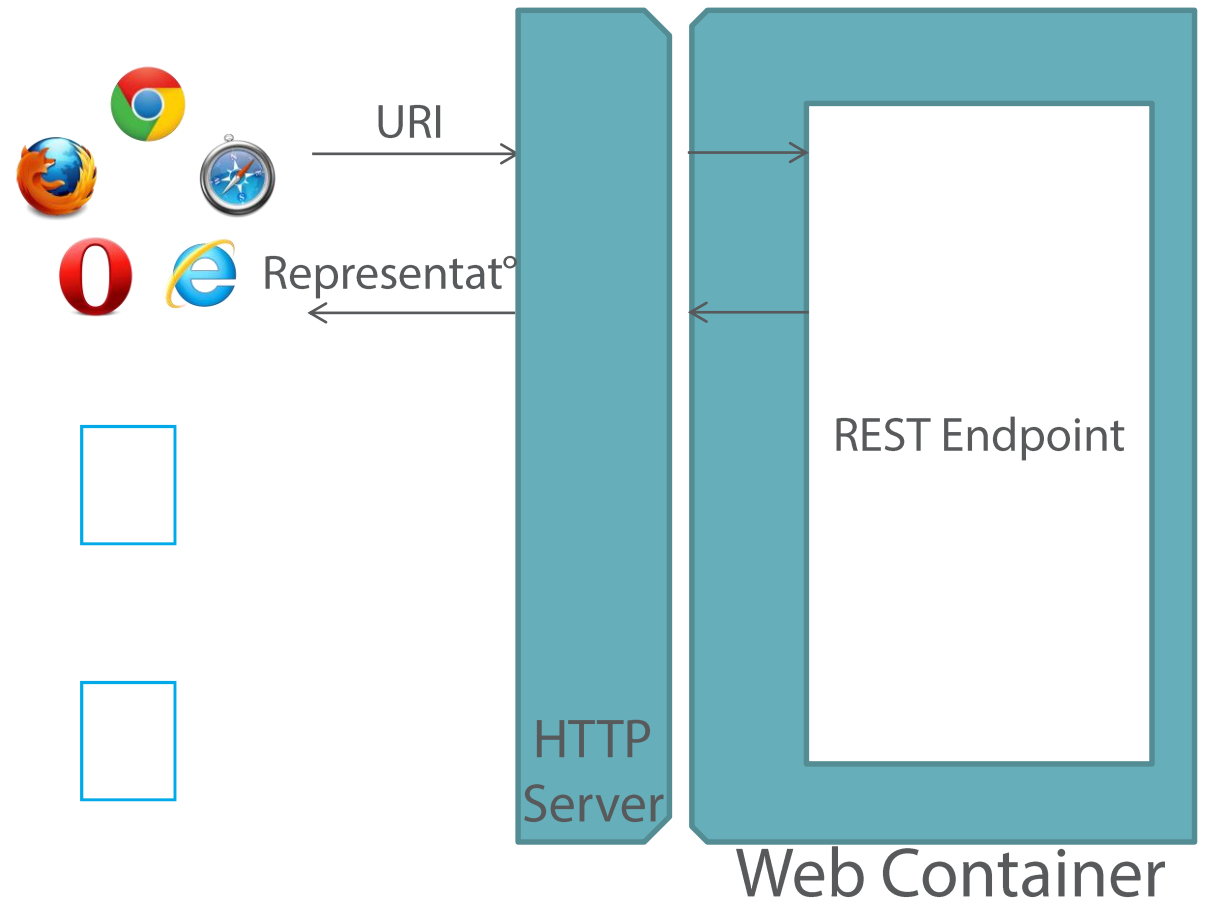
Consume TopRated items

External service



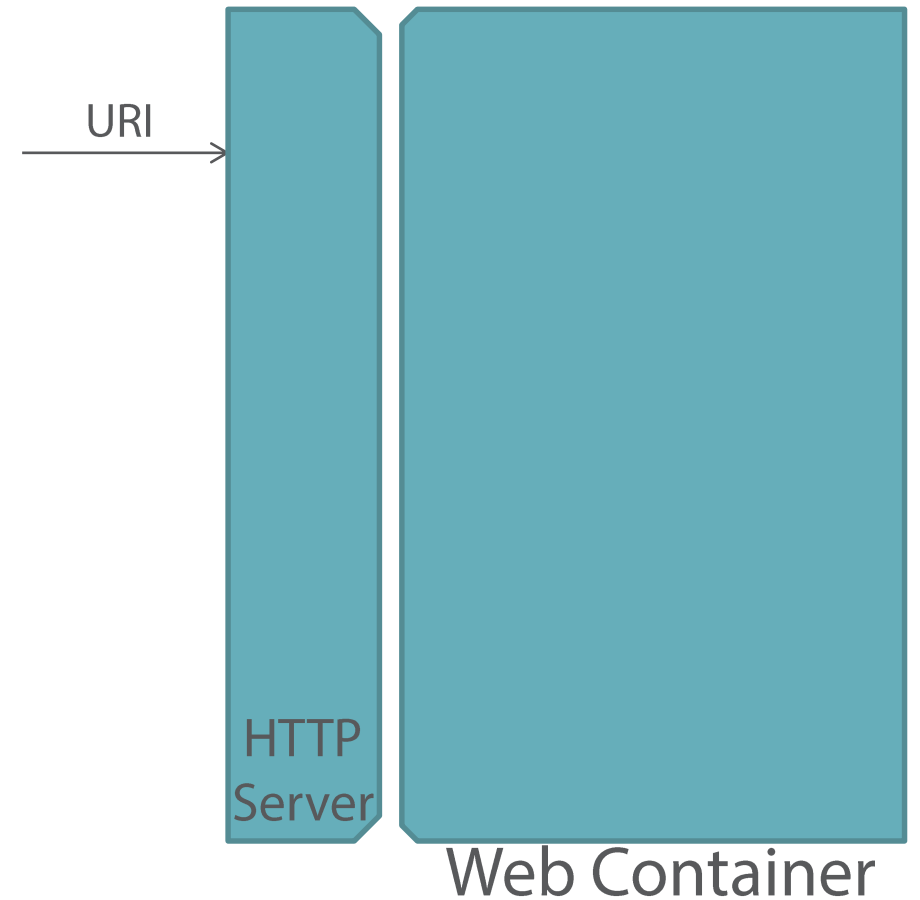
Understanding JAX-RS

- HTTP
- Web clients
- URI
- Several representations
- Status code



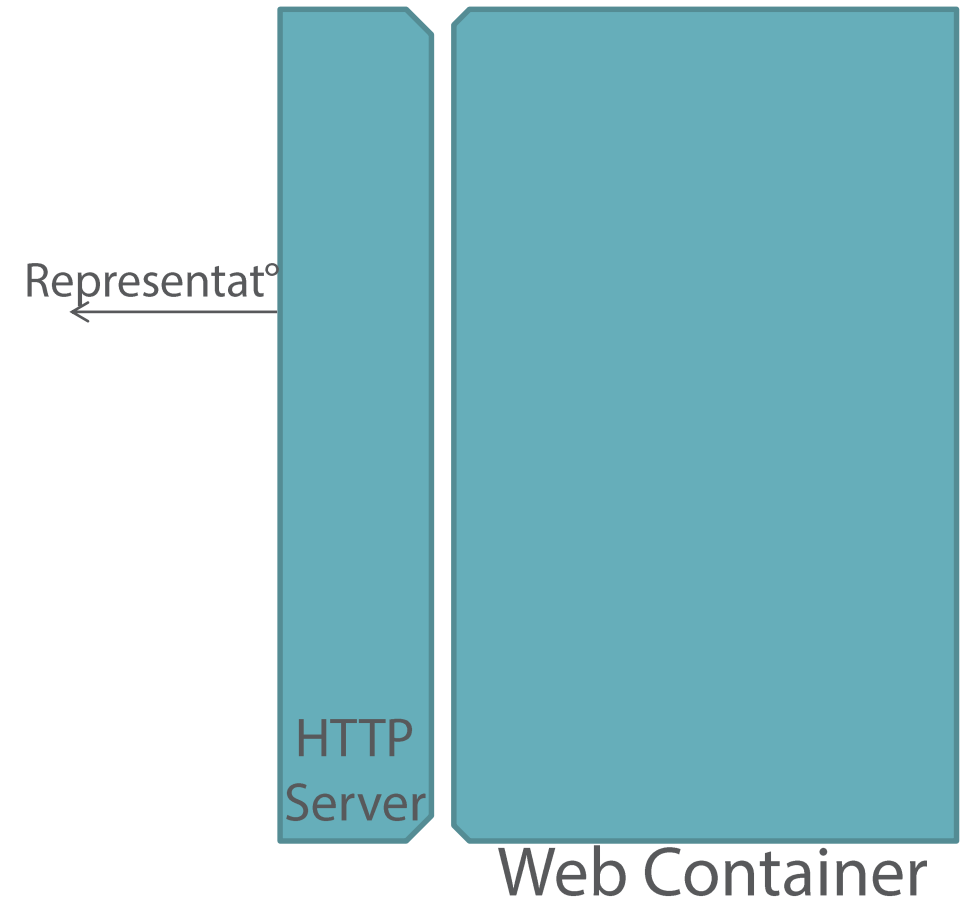
Resources and URIs

- Resource addressed with a URI
- Any piece of information
- Unique identifier
- `www.cdbookstore.com/books`
- `www.cdbookstore.com/books/1234`
- `www.cdbookstore.com/info/jobs`
- Descriptive as possible



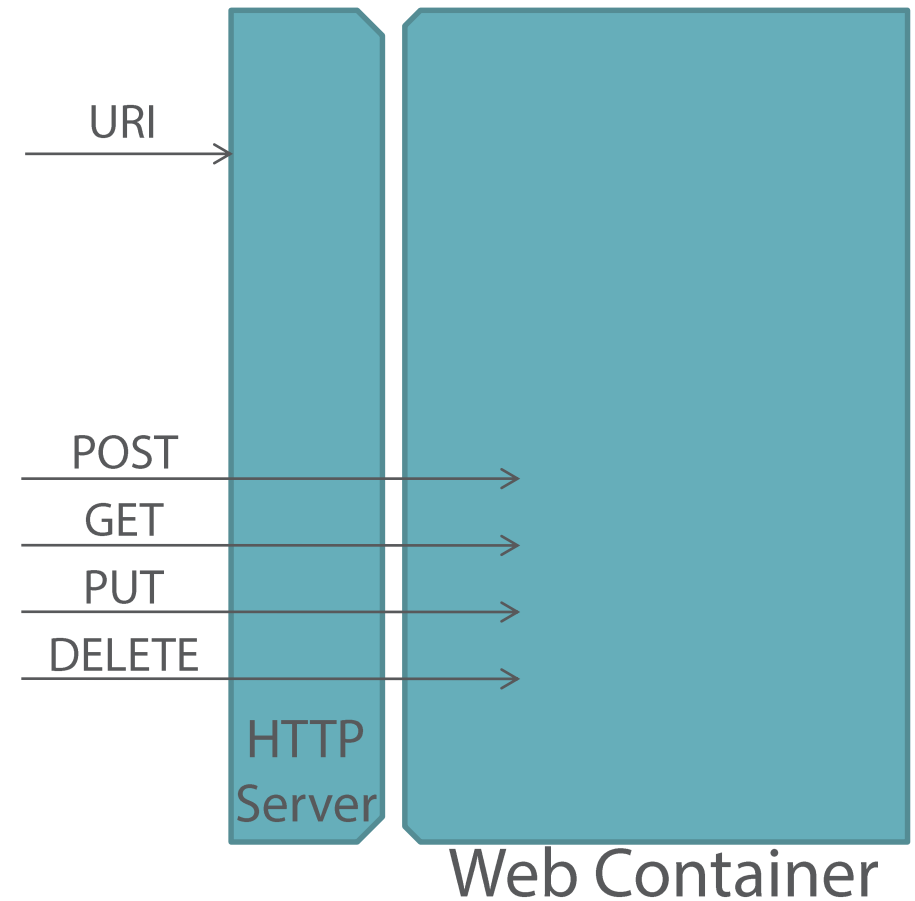
Representations

- Many representations for one resource
- Text, JSON, XML, PDF, image...
- Several URIs
 - `www.cdbookstore.com/books/xml`
 - `www.cdbookstore.com/books/csv`
- One single URI
 - `www.cdbookstore.com/books`
 - content negotiation
 - HTTP headers



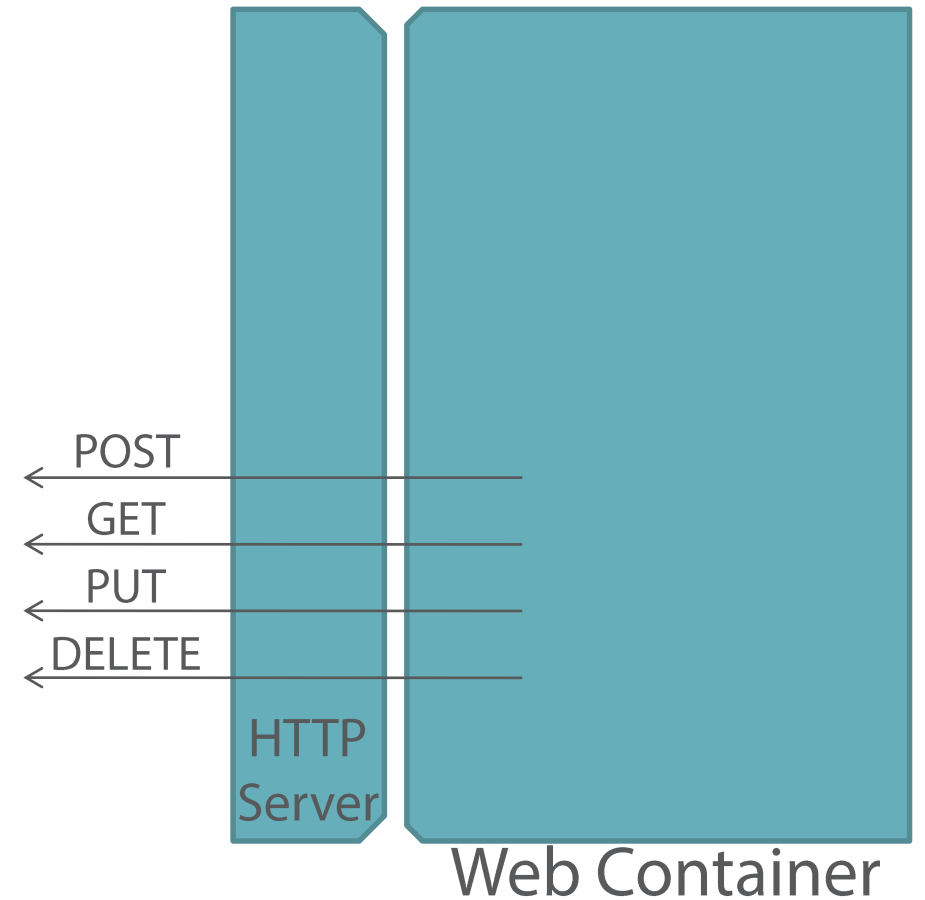
Uniform Interface

- Architecture is simplified
- Interaction is improved
- HTTP semantic
- POST (C)reates
- GET (R)eads
- PUT (U)pdates
- DELETE (D)eletes



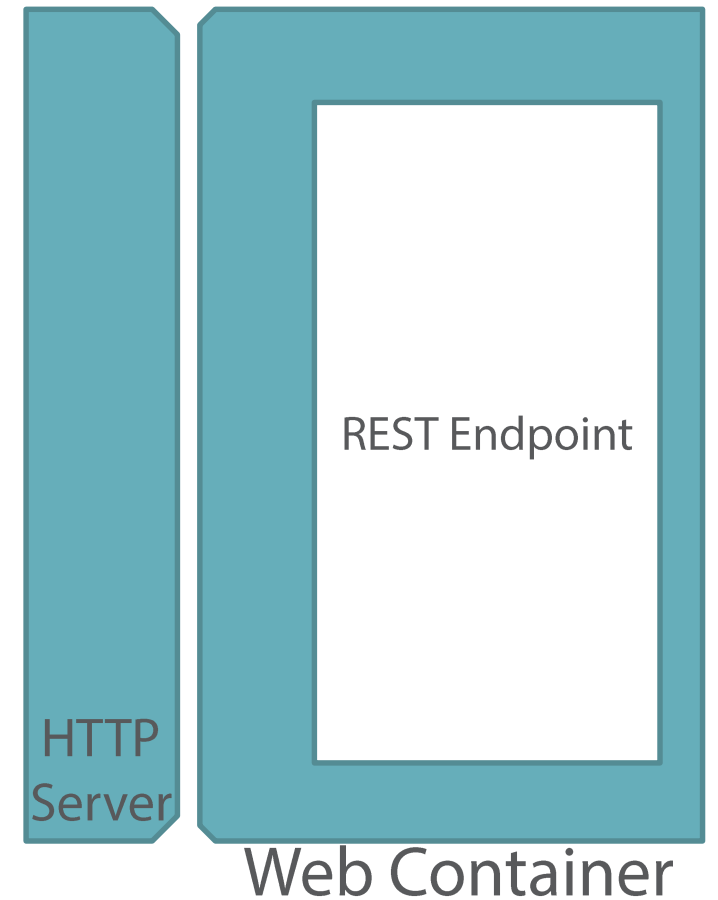
HTTP Status Code

- 1xx Informational
- 2xx Success
- 3xx Redirection
- 4xx Client Error
- 5xx Server Error
- 200 OK
- 401 Unauthorized
- 404 Not Found



REST Endpoint

- HTTP is low-level
- Higher-level of abstraction
- Don't write plumbing code
- Describe REST Web service
- With few annotations



REST Endpoint

```
@Path("/books")
public class BookEndpoint {

    @Inject
    private EntityManager em;

    @GET
    @Produces({"application/xml"})
    public List<Book> listAll() {
        TypedQuery<Book> query = em.createQuery("SELECT b FROM Book b", Book.class);
        return query.getResultList();
    }
}
```

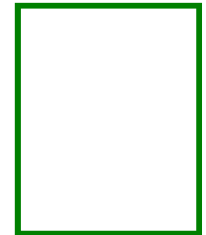
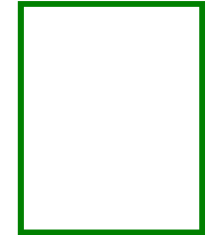
<http://www.cdbookstore.com/books>

JAX-RS Packages

➡	Package	Description
➡	javax.ws.rs	Core JAX-RS
➡	javax.ws.rs.client	Client API
➡	javax.ws.rs.container	Container-specific JAX-RS API
➡	javax.ws.rs.core	Low-level interfaces and annotations
➡	javax.ws.rs.ext	Extensions

Exposing REST Services

- Programmable Web
- Applying REST to services
- Focus on resources
- URIs
- Representations
- Uniform interface



URIs

```
@Path("/books")  
public class BookEndpoint {  
  
    @GET  
    public List<Book> listAll() { // ... }  
  
}
```

<http://www.cdbookstore.com>

URIs

```
@Path("/books")
public class BookEndpoint {

    @GET
    public List<Book> listAll() { // ... }

    @GET
    @Path("/paper")
    public List<Book> listPaperBooks() { // ... }

}
```

<http://www.cdbookstore.com/books>

URIs

```
@Path("/books")
public class BookEndpoint {

    @GET
    public List<Book> listAll() { // ... }

    @GET
    @Path("/paper")
    public List<Book> listPaperBooks() { // ... }

    @GET
    @Path("/paper/old")
    public List<Book> listOldPaperBooks() { // ... }

}
```

<http://www.cdbookstore.com/books/paper>

Extracting Parameters

```
@Path("/books")
public class BookEndpoint {

    @GET
    public List<Book> listAll() { // ... }

    @GET
    @Path("/{id}")
    public Book findById(@PathParam("id") long id) { // ... }

}
```

<http://www.cdbookstore.com/books>

Extracting Parameters

```
@Path("/books")
public class BookEndpoint {

    @GET
    public List<Book> listAll() { // ... }

    @GET
    @Path("/{id}")
    public Book findById(@PathParam("id") Long id) { // ... }

    @GET
    @Path("/search/{text}")
    public List<Book> search(@PathParam("text") Long searchText) { // ... }

}
```

http://www.cdbookstore.com/books

Representations

```
@Path("/books")
public class BookEndpoint {

    @Produces({"application/xml"})
    @GET
    public List<Book> listAll() { // ... }

    @Produces({"application/json"})
    @GET
    public List<Book> listAllinJson() { // ... }

}
```

<http://www.cdbookstore.com/books>

Representations

```
@Path("/books")
public class BookEndpoint {

    @Produces({"application/xml"})
    @GET
    public List<Book> listAll() { // ... }

    @Produces({"application/json"})
    @GET
    public List<Book> listAllinJson() { // ... }

    @Produces({"application/xml", "application/json"})
    @GET
    @Path("/{id}")
    public Book findById(@PathParam("id") Long id) { // ... }

}
```

<http://www.cdbookstore.com/books/1234>

CRUD Operations

```
@Path("/books")  
public class BookEndpoint {  
    @GET  
    @Produces({"application/xml"})  
    @Path("/{id}")  
    public Book findById(@PathParam("id") Long id) { // ... }  
}
```

 <http://www.cdbookstore.com/books/1234>

CRUD Operations

```
@Path("/books")
public class BookEndpoint {

    @GET
    @Produces({"application/xml"})
    @Path("/{id}")
    public Book findById(@PathParam("id") Long id) { // ... }

    @DELETE
    @Path("/{id}")
    public void deleteById(@PathParam("id") Long id) { // ... }

}
```

 <http://www.cdbookstore.com/books/1234>

CRUD Operations

```
@Path("/books")
public class BookEndpoint {

    @GET
    @Produces({"application/xml"})
    @Path("/{id}")
    public Book findById(@PathParam("id") Long id) { // ... }

    @DELETE
    @Path("/{id}")
    public void deleteById(@PathParam("id") Long id) { // ... }

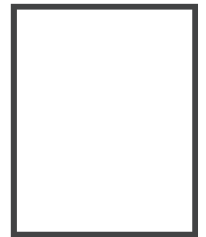
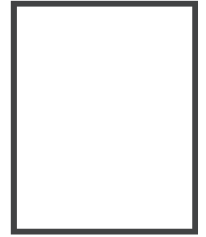
    @POST
    @Consumes({"application/xml"})
    public Response create(Book book) { // ... }

}
```

<http://www.cdbookstore.com/books>

Consuming REST Services

- Make an HTTP request on a URI
- Web browser
- Extensions to add more flexibility
- JavaScript
- cUrl
- Client API
 - Java SE
 - Android
 - Java EE



Client API

```
Client client = ClientBuilder.newClient();  
WebTarget target = client.target("http://www.cdbookstore.com/books/1234");  
Invocation invocation = target.request(MediaType.TEXT_PLAIN).buildGet();  
Response response = invocation.invoke();
```

GET

http://www.cdbookstore.com/books/1234

Client API

```
Response response = ClientBuilder.newClient()  
    .target("http://www.cdbookstore.com/books/1234")  
    .request(MediaType.TEXT_PLAIN)  
    .get();
```

GET

http://www.cdbookstore.com/books/1234

Client API

```
Response response = ClientBuilder.newClient()  
    .target("http://www.cdbookstore.com/books/1234")  
    .request(MediaType.TEXT_PLAIN)  
    .get();
```

DELETE

<http://www.cdbookstore.com/books/1234>

Client API

```
Response response = ClientBuilder.newClient()  
    .target("http://www.cdbookstore.com/books/1234")  
    .request(MediaType.TEXT_PLAIN)  
    .delete();
```

DELETE

<http://www.cdbookstore.com/books/1234>

Response

```
Response response = ClientBuilder.newClient()  
    .target("http://www.cdbookstore.com/books/1234")  
    .request(MediaType.TEXT_PLAIN)  
    .delete();  
  
assertTrue(response.getStatusInfo() == Response.Status.OK);  
assertTrue(response.getLength() == 4);  
assertTrue(response.getDate() != null);  
assertTrue(response.getHeaderString("Content-type").equals("text/plain"));
```

DELETE <http://www.cdbookstore.com/books/1234>

Response

```
Response response = ClientBuilder.newClient()  
    .target("http://www.cdbookstore.com/books/1234")  
    .request(MediaType.TEXT_PLAIN)  
    .get();  
  
assertTrue(response.getStatusInfo() == Response.Status.OK);  
assertTrue(response.getLength() == 4);  
assertTrue(response.getDate() != null);  
assertTrue(response.getHeaderString("Content-type").equals("text/plain"));  
  
String body = response.readEntity(String.class);  
Book book = response.readEntity(Book.class);
```

GET

http://www.cdbookstore.com/books/1234

REST Services

Exposing REST services

CRUD operations

Client API

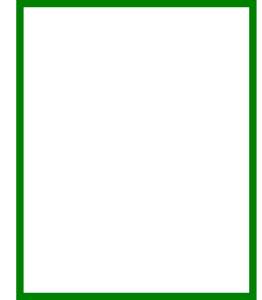


Messaging

Java Messaging Service (JMS) 2.0

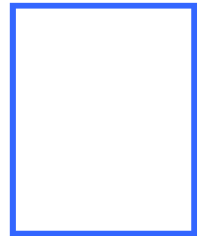
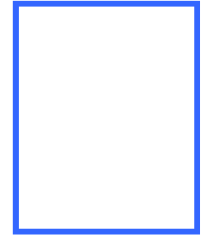
What Is Messaging?

- Message-oriented middleware (MOM)
- Broker
- Exchange messages asynchronously
- Between heterogeneous systems
- At their own pace
- Loosely coupled
- Not available at the same time



When to Use Messaging

- Many applications
- Written in different languages
- Different operating systems
- Asynchronous interaction
- Work independently
- Loosely coupled
- Producer and consumer agree on the message
- Local or distributed



JMS Specification

- Java Message Service 2.0
- JSR 343
- <http://jcp.org/en/jsr/detail?id=343>



Java
Community
Process

The screenshot shows the Java Community Process website for JSR 343. The page includes a navigation bar with tabs for JSR, Community, and Expert Group. The main content area displays the title "JSRs: Java Specification Requests" and "JSR 343: Java™ Message Service 2.0". Below this is a table with columns for Stage, Access, Start, and Finish. The table lists various stages of the specification process, including Maintenance Release, Maintenance Review Ballot, Maintenance Draft Review, Final Release, Final Approval Ballot, Proposed Final Draft, Public Review Ballot, Public Review, Early Draft Review, Expert Group Formation, and JSR Review Ballot. The table also includes links for downloading pages and viewing results. Below the table, there is a section for "Status: Maintenance" and "JCP version in use: 2.9", followed by a description of the specification and a link for "Expert Group Transparency: Public Communications".

Stage	Access	Start	Finish
Maintenance Release	Download page	16 Mar, 2015	
Maintenance Review Ballot	View results	03 Mar, 2015	09 Mar, 2015
Maintenance Draft Review	Download page	26 Jan, 2015	25 Feb, 2015
Final Release	Download page	21 May, 2013	
Final Approval Ballot	View results	26 Mar, 2013	08 Apr, 2013
Proposed Final Draft	Download page	26 Feb, 2013	
Public Review Ballot	View results	05 Feb, 2013	18 Feb, 2013
Public Review	Download page	02 Jan, 2013	04 Feb, 2013
Early Draft Review	Download page	28 Feb, 2012	29 Mar, 2012
Expert Group Formation		15 Mar, 2011	08 Sep, 2011
JSR Review Ballot	View results	01 Mar, 2011	14 Mar, 2011

Status: Maintenance
JCP version in use: 2.9
Java Specification Participation Agreement version in use: 2.0

Description:
This is an update to the Java Message Service API, an existing API for accessing enterprise messaging systems from Java programs.

Expert Group Transparency:
[Public Communications](#)

JMS Implementations

- OpenMQ
- ActiveMQ
- HornetQ
- RabbitMQ
- WebsphereMQ



ActiveMQ



HornetQ



MQ

 **RabbitMQ**

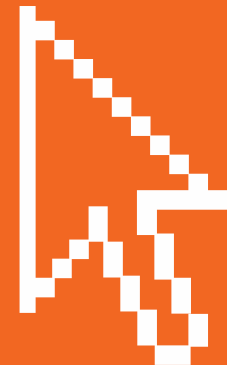
Messaging

CD-BookStore 24/7

Sends messages

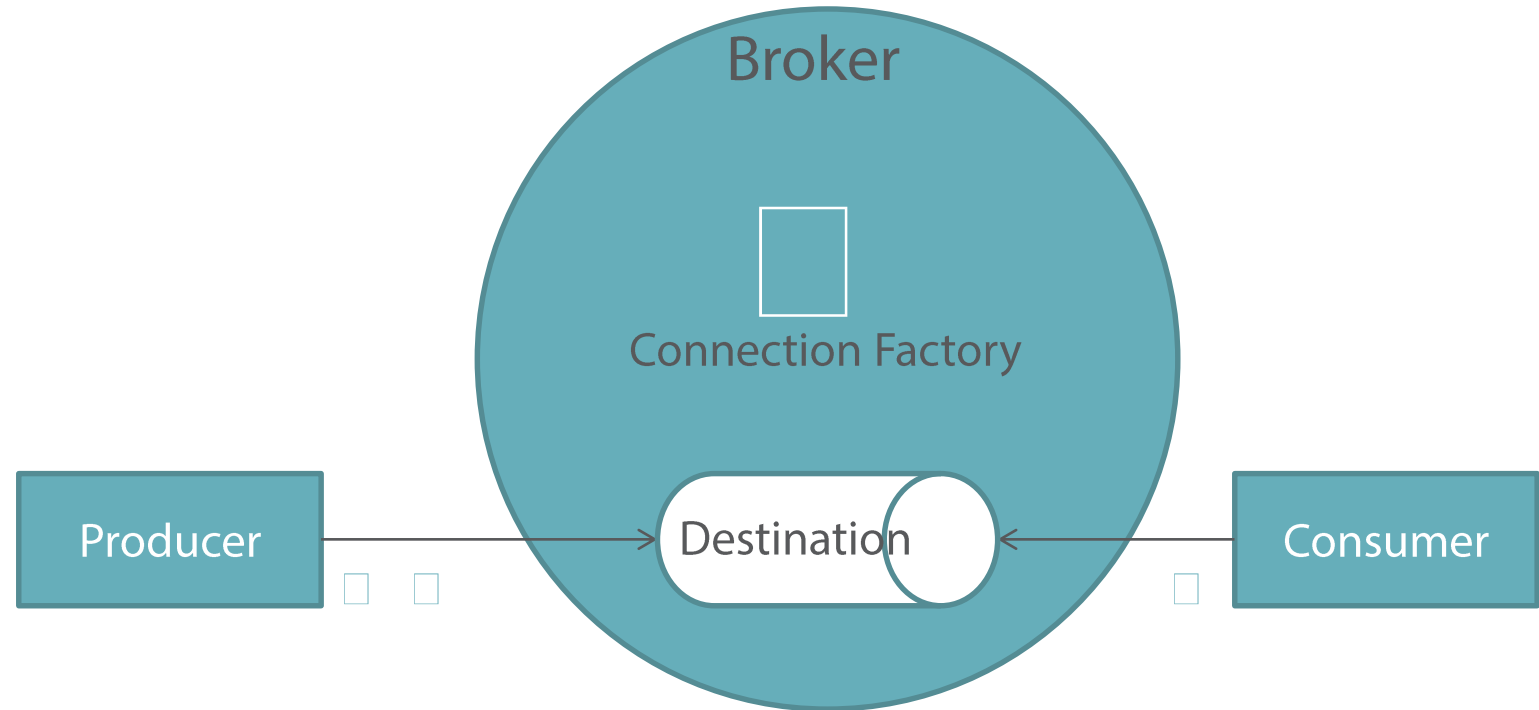
Invoice application

Even if down



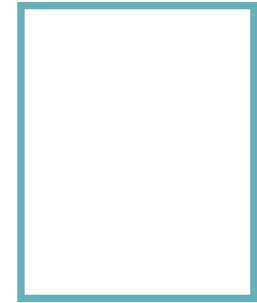
Understanding JMS

- Message broker
- Producer
- Destination
- Consumer
- Connection factory
- Messages



Messages

- Encapsulate information
- Header
 - Identifying and routing
- Properties
 - Name-value pairs
- Body
 - Text, bytes, object, map or a stream

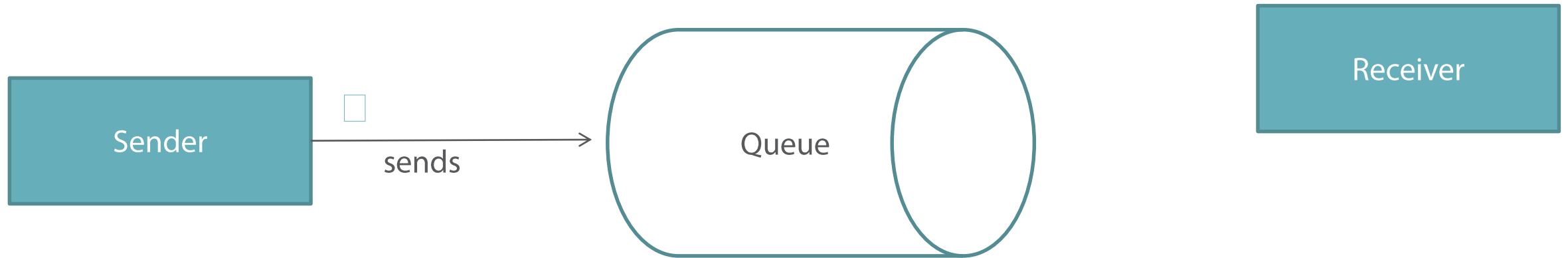


Destinations

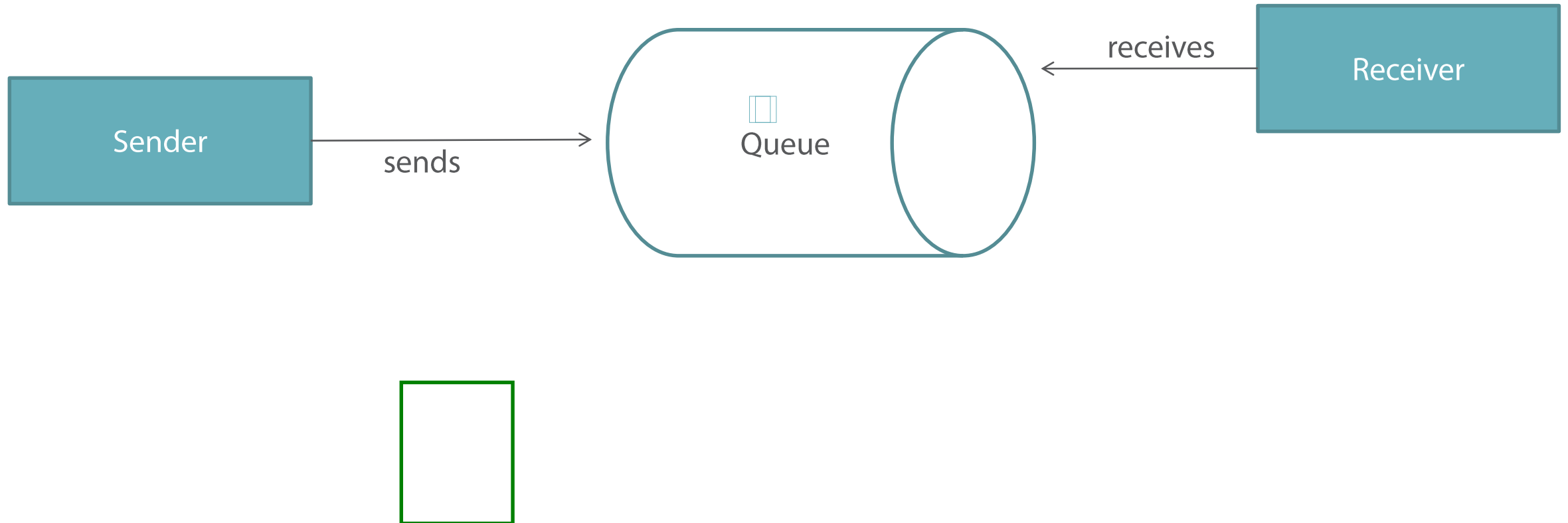
- Administered object
 - JNDI name
 - JNDI lookup
 - Injection
- Queue
 - Point-to-point
- Topic
 - Publish-subscribe



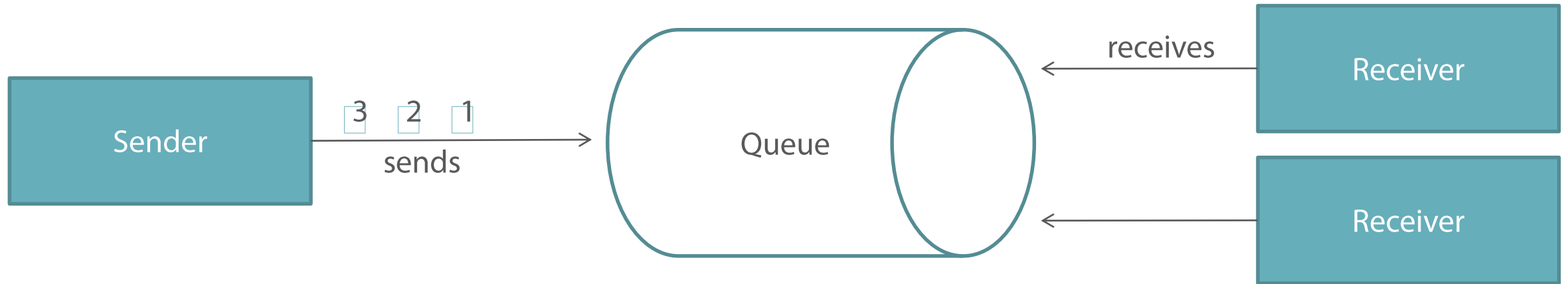
Queue and Point-to-Point



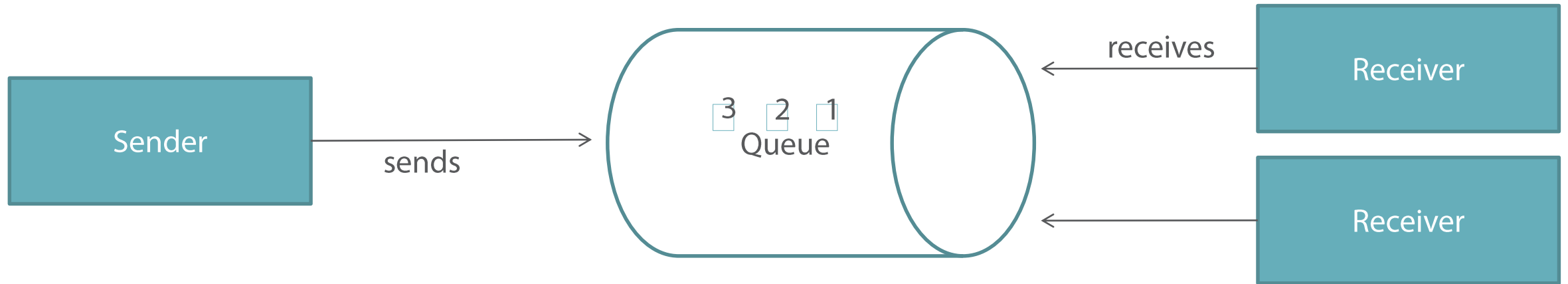
Queue and Point-to-Point



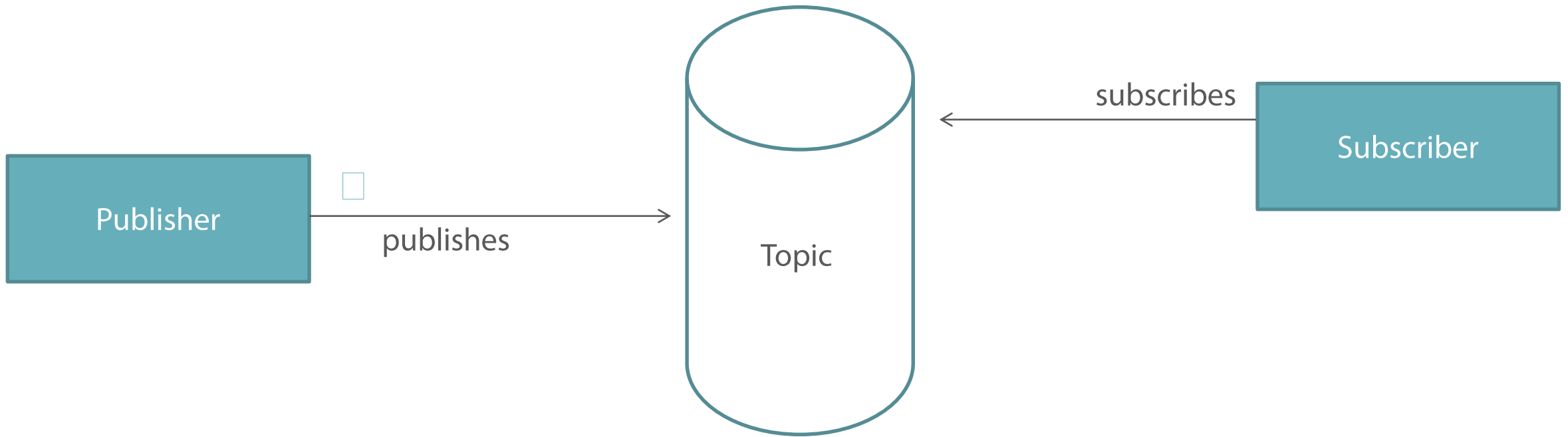
Queue and Point-to-Point



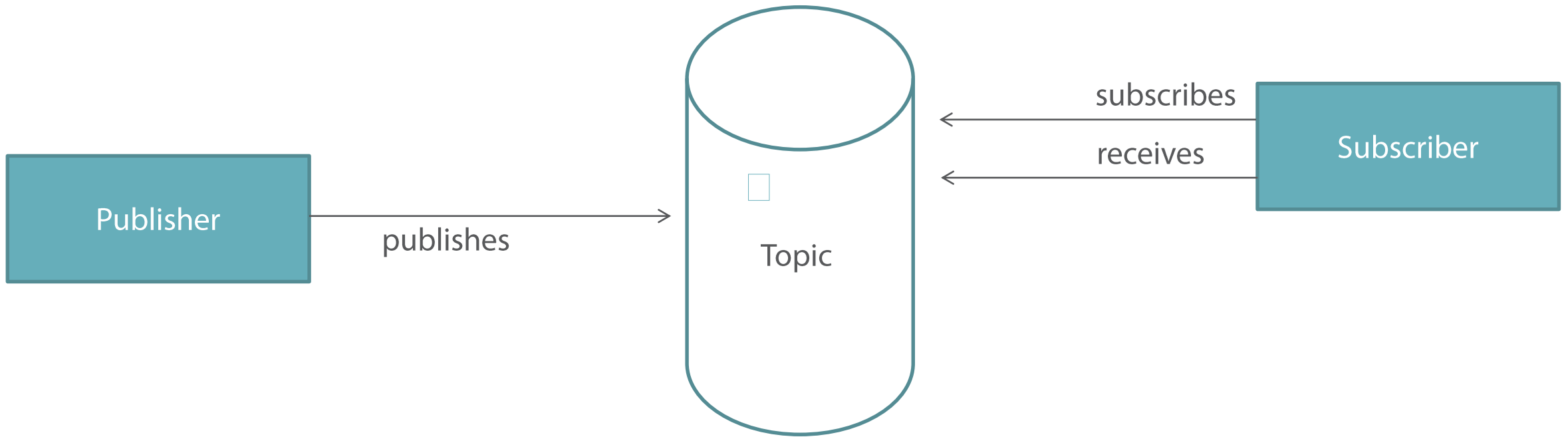
Queue and Point-to-Point



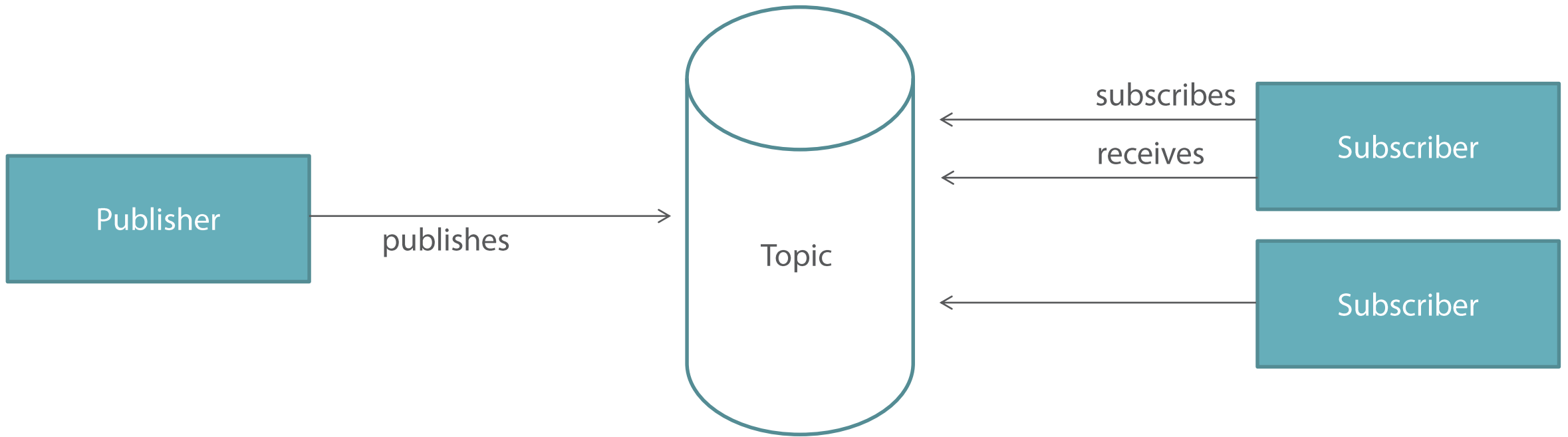
Topic and Publish-Subscribe



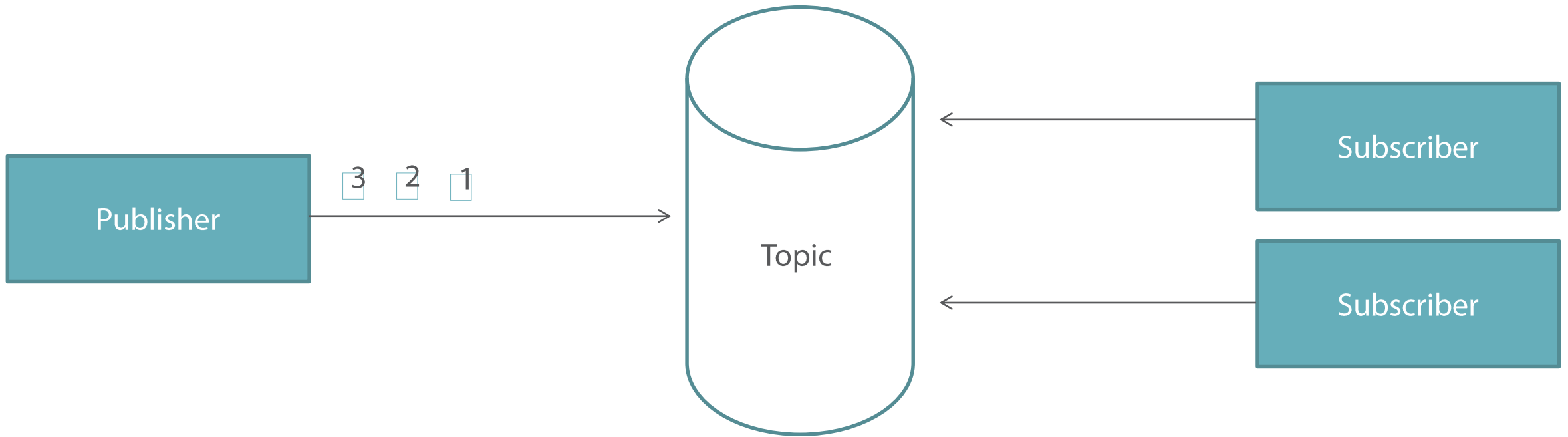
Topic and Publish-Subscribe



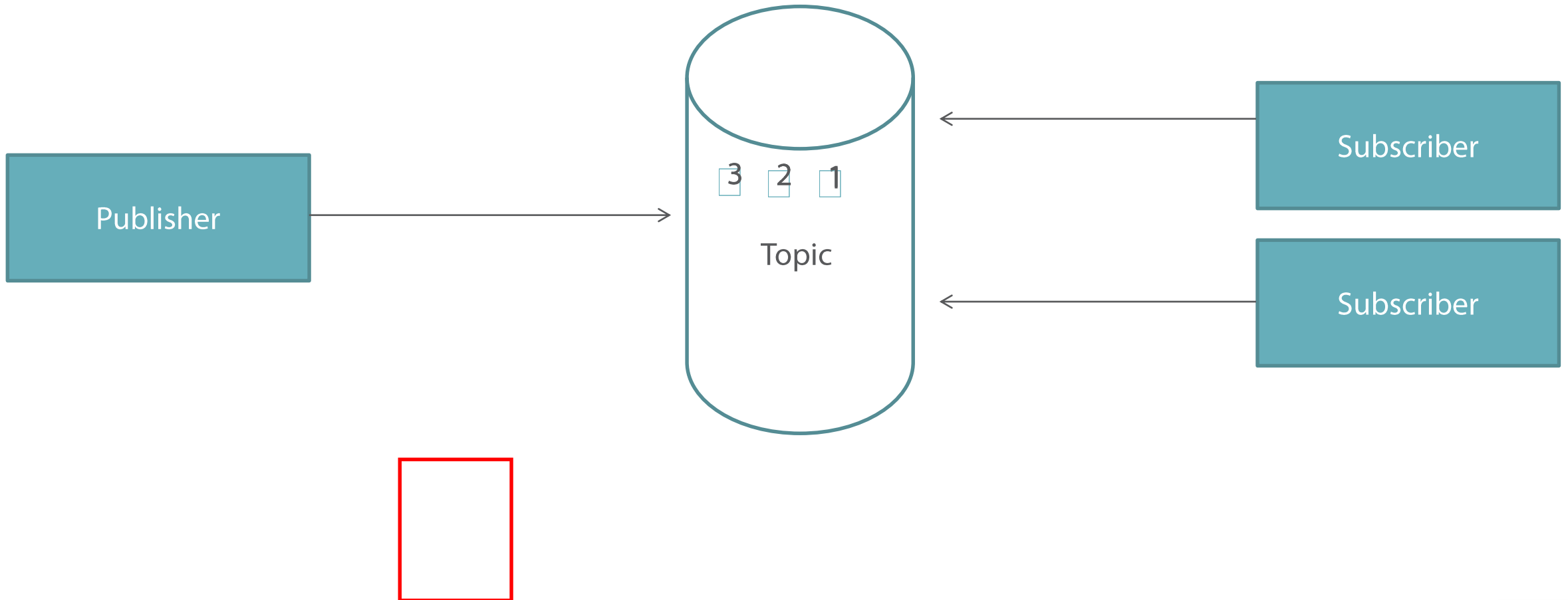
Topic and Publish-Subscribe



Topic and Publish-Subscribe

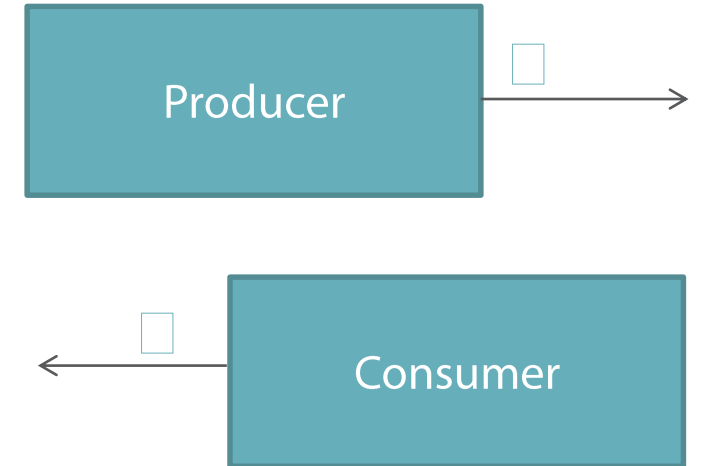


Topic and Publish-Subscribe



Producers and Consumers

- Producer
 - Sender
 - Publisher
- Consumer
 - Receiver
 - Subscriber
- Any Java EE component



Producing a Message

```
public class Sender {  
  
    @Resource(lookup = "jms/queue/invoiceQueue")  
    private Queue destination;  
  
    @Inject  
    private JMSContext jmsContext;  
  
    public void sendMessage() {  
        jmsContext.createProducer().send(destination, "Message sent at " + new Date());  
    }  
}
```

Producing a Message

```
public class Sender {  
  
    @Resource(lookup = "jms/queue/invoiceQueue")  
    private Topic destination;  
  
    @Inject  
    private JMSContext jmsContext;  
  
    public void sendMessage() {  
        jmsContext.createProducer().send(destination, "Message sent at " + new Date());  
    }  
}
```

Producing a Message

```
public class Sender {  
  
    @Resource(lookup = "jms/queue/invoiceTopic")  
    private Topic destination,  
  
    @Inject  
    private JMSContext jmsContext;  
  
    public void sendMessage() {  
        jmsContext.createProducer().send(destination, "Message sent at " + new Date());  
    }  
}
```

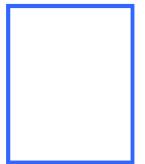
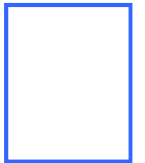
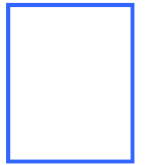
Consuming a Message

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationType",
                                propertyValue = "javax.jms.Queue"),
    @ActivationConfigProperty(propertyName = "destination",
                                propertyValue = "jms/queue/invoiceQueue")
})
public class Receiver {

    @Override
    public void onMessage(Message message) {
        // Process message
    }
}
```

Reliability Mechanisms

- Ensure messages are delivered
- Broker crashes or is under load
- Filtering messages
- Setting Time-to-live
- Providing message persistence
- Controlling acknowledgment
- Setting priorities
- Creating durable subscribers



Reliability Mechanisms

```
public class Sender {  
  
    @Resource(lookup = "jms/queue/invoiceQueue")  
    private Queue destination;  
  
    @Inject  
    private JMSContext jmsContext;  
  
    public void sendMessage() {  
  
        jmsContext.createProducer()  
            .setPriority(2)  
            .setTimeToLive(1000)  
            .setDeliveryMode(DeliveryMode.NON_PERSISTENT)  
            .send(queue, "Text message sent at " + new Date());  
    }  
}
```

JMS Packages



Package	Description
<code>javax.jms</code>	Core JMS API

Messaging

Send invoice message

Receive message

Queue



Summary



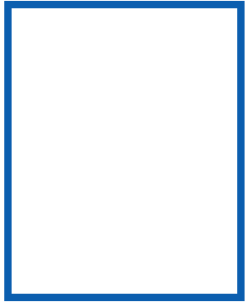
Interoperability tier

XML and JSON

RESTful Web Services

Java Message Service

What's Next



All together

Flexibility

Modular