Java Message Service

JMS 2.0

Outline

- Introduction to JMS
- Basic JMS API concepts
- JMS Programming Model

Messaging

- Method of communication
- Peer to peer
- Loosely coupled
- Email → communication between people
- Messaging → communication between software

JMS API

- API to
  - Create messages
  - Send messages
  - Receive messages
  - Read messages

JMS API

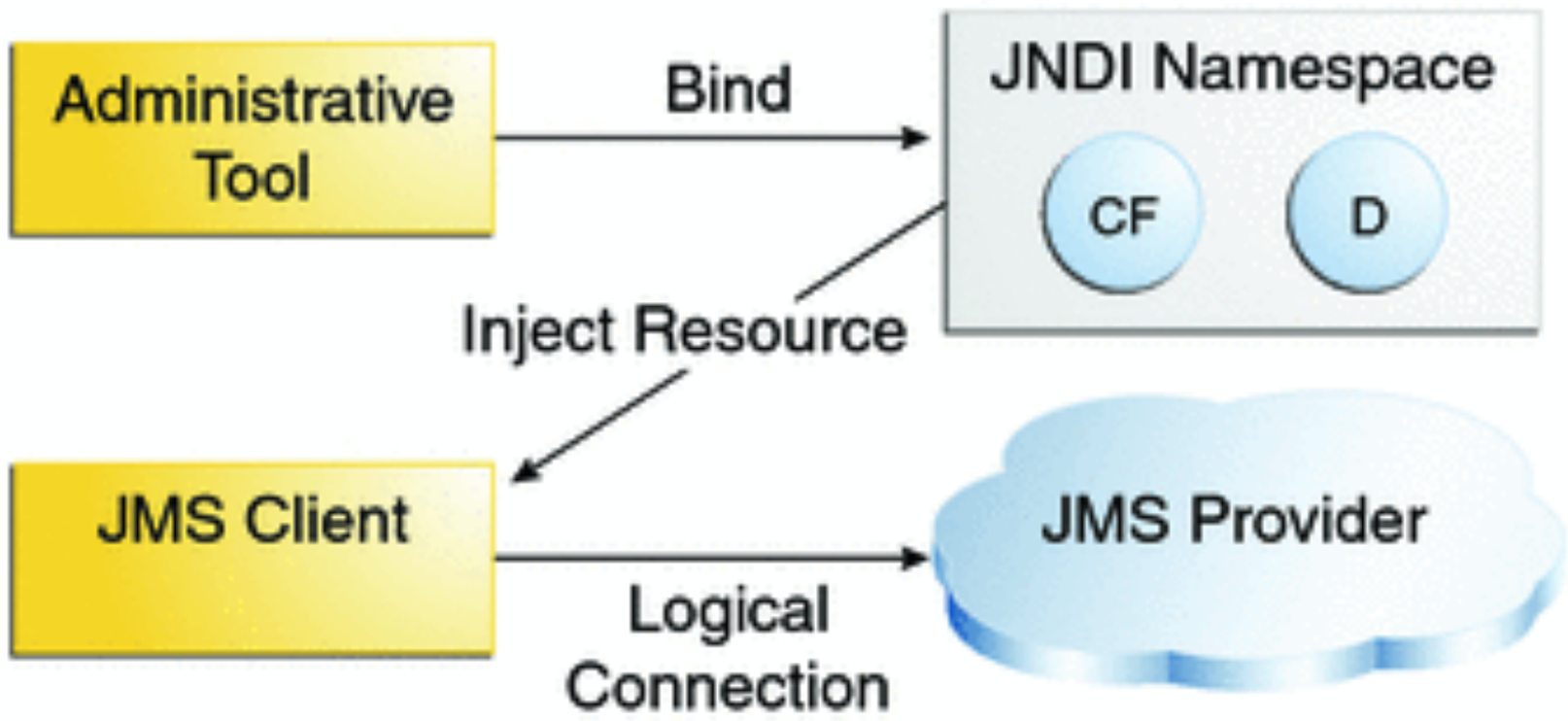
- Loosely coupled
- Asynchronous
- Reliable

Reasons to use JMS

- Not depended on interfaces
- Run applications whether or not all components are up and running simultaneously.
- Send information and continue to operate without receiving an immediate response

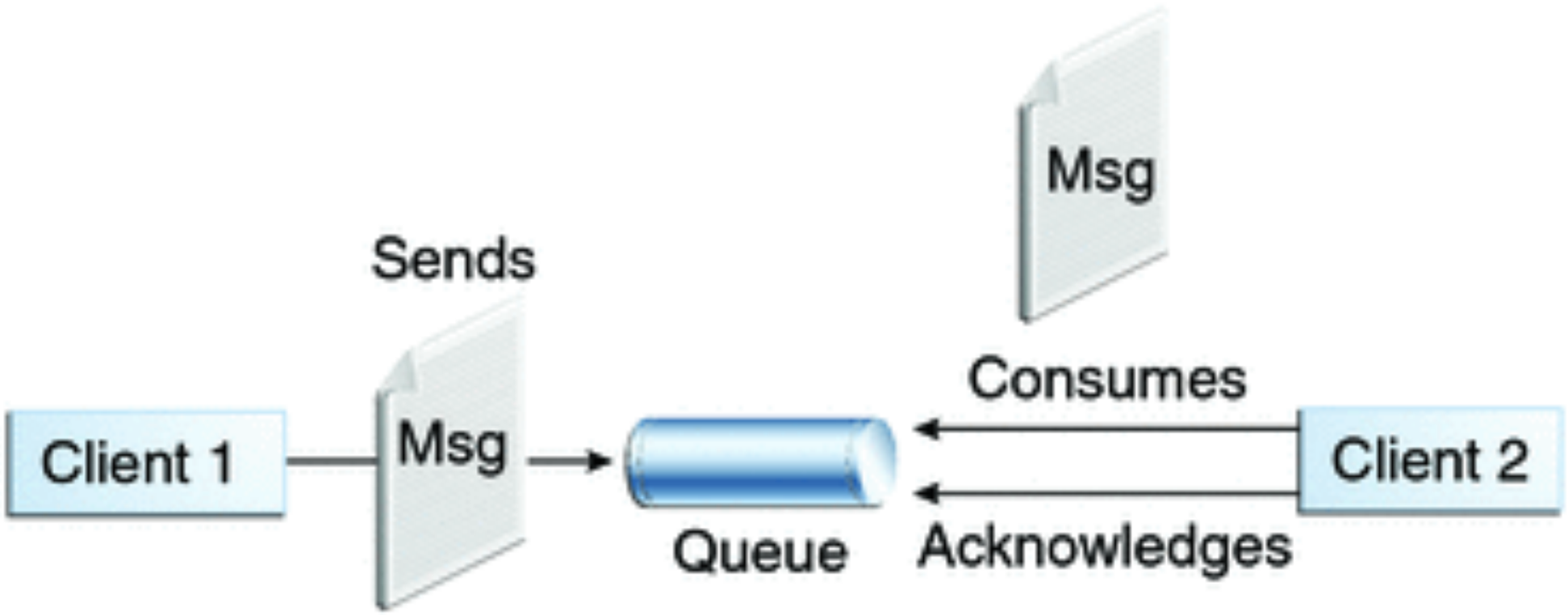
Basic JMS API concepts

- JMS architecture
  - ***JMS Provider*** → implements JMS interfaces
  - ***JMS clients*** → can produce or consume messages
  - ***Messages*** → communication objects
  - ***Administered objects*** → preconfigured JMS objects



Basic JMS API concepts

- Point – to – point Messaging



Basic JMS API concepts

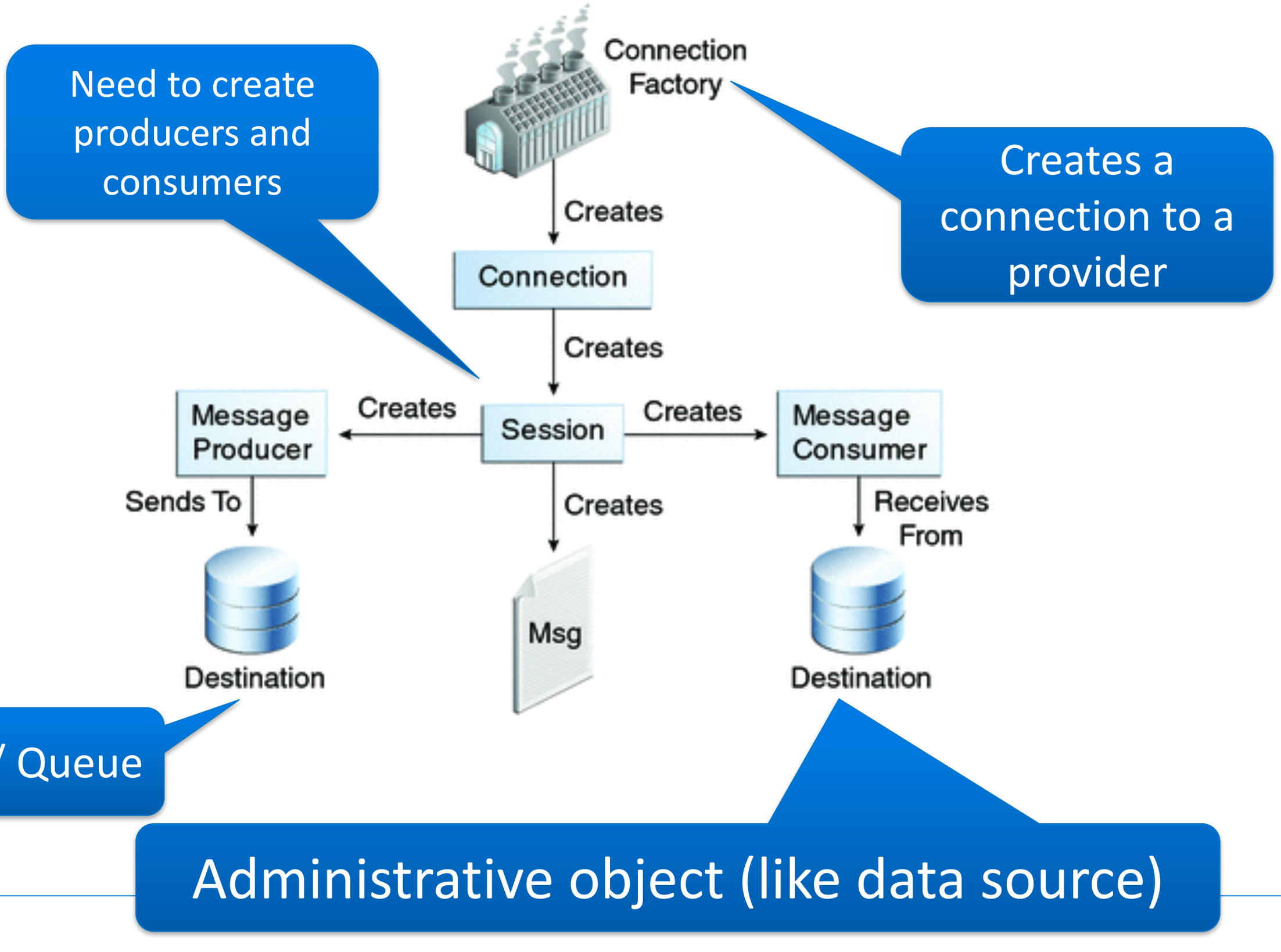
- Publish / Subscribe Messaging

Basic JMS API concepts

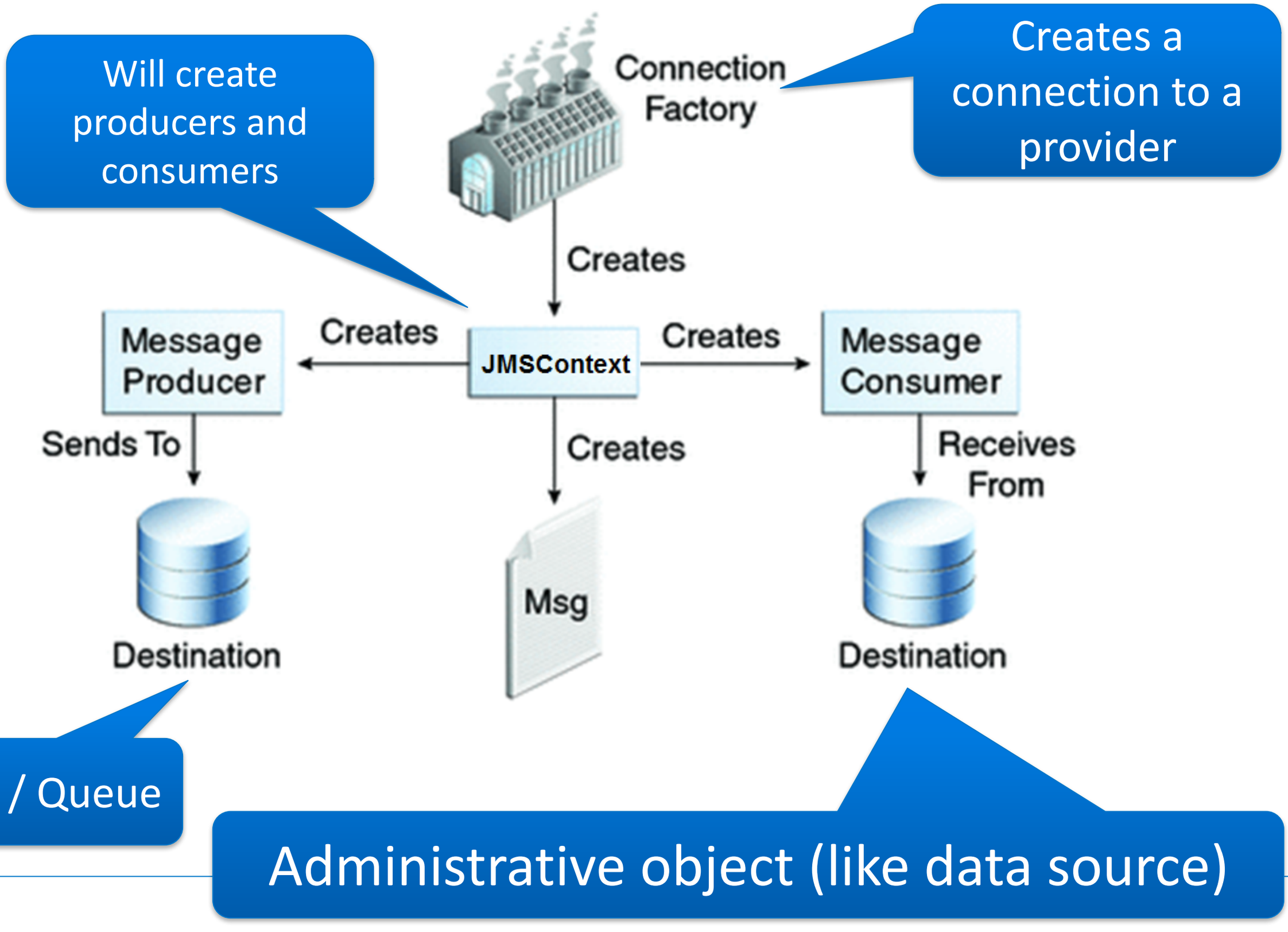
- Message Consumption

- Two ways of consumption:
  - Synchronously
    - *receiveBody()* method can block or can time out
  - Asynchronously
    - *onMessage()*

Original JMS API Programming Model



Simplified JMS API Programming Model



JMS Context

- Encapsulates a
  - Connection
  - Session
  - MessageProducer
- Call close() or use in a try-with-resource block
- Can be injected
- Used to create a JMSProducer
- Used to create a JMSConsumer

Steps to send a message

- Inject a ConnectionFactory
- Create an JMXContext with the factory
- Create a producer with the context
- Send message

Lookup resources

```
@Resource(lookup = “jms/myQueue”)
private static Queue queue;

@Resource(lookup = “jms/myConFactory”)
private static ConnectionFactory factory;
```

or lookup:

```
ctx = new InitialContext();
ctx.lookup(“jms/myConFactory”);
```

Create producer

```
JMSContext context =
    factory.createConnext();


JMSProducer producer =
    context.createProducer();


Producer.send(queue, “Hello world”);
```

Consume messages

```
JMSConsumer consumer =
    context.createConsumer(queue);

// synchronous
String message =
    consumer.receiveBody(String.class);

// asynchronous
Listener myListener = new Listener();
consumer.setMessageListener(myListener);
```

Comparing the old and the simplified API

	Old	Simplified

Producer	`Connection connection = factory.createConnection();`
	`Session session = connection.createSession(…);`
	`MessageProducer producer = session.createProducer(queue);`
	`TextMessage message = session.createTextMessage();`
	
	`message.setText("Hello World");`
	
	`producer.send(queue, message);`
		`JMSContext context = factory.createConnext();`
		
		`JMSProducer producer = context.createProducer();`
		
		`Producer.send(queue, "Hello world");`

Consumer	`MessageConsumer consumer = session.createConsumer(queue);`
	
	`// synchronous`
	`Message message = consumer.receive();`
	
	`// asynchronous`
	`Listener myListener = new Listener();`
	`consumer.setMessageListener(myListener);`
		`JMSConsumer consumer = context.createConsumer(queue);`
		
		`// synchronous`
		`String message = consumer.receiveBody(String.class);`
		
		`// asynchronous`
		`Listener myListener = new Listener();`
		`consumer.setMessageListener(myListener);`

JMS Messages

- Message header fields
- Message bodies
- Several message types
- Message properties

JMS Header Fields

- JMS Destination
- JMS Delivery Mode
- JMS Expiration
- JMS Priority
- JMS Message ID
- JMS Timestamp
- JMS CorrelationID
- JMS Reply To
- JMS type
- JMS Redelivered

JMS Message Types

Message Type	Contains
`TextMessage`	`java.lang.String`
`MapMessage`	Name value pairs (Key → String, Value → primitive)
`BytesMessage`	Stream of bytes
`ObjectMessage`	`Serializable` object
`Message`	No body