
JAVA EE7
PRE
LAUNCH PARTY
eWEEK

Java EE 7
Wie zijn wij?

- Jouke Stoel
  - J-Technologies Innovation Board
  - Java ontwikkelaar
  - http://nl.linkedin.com/in/joukestoel

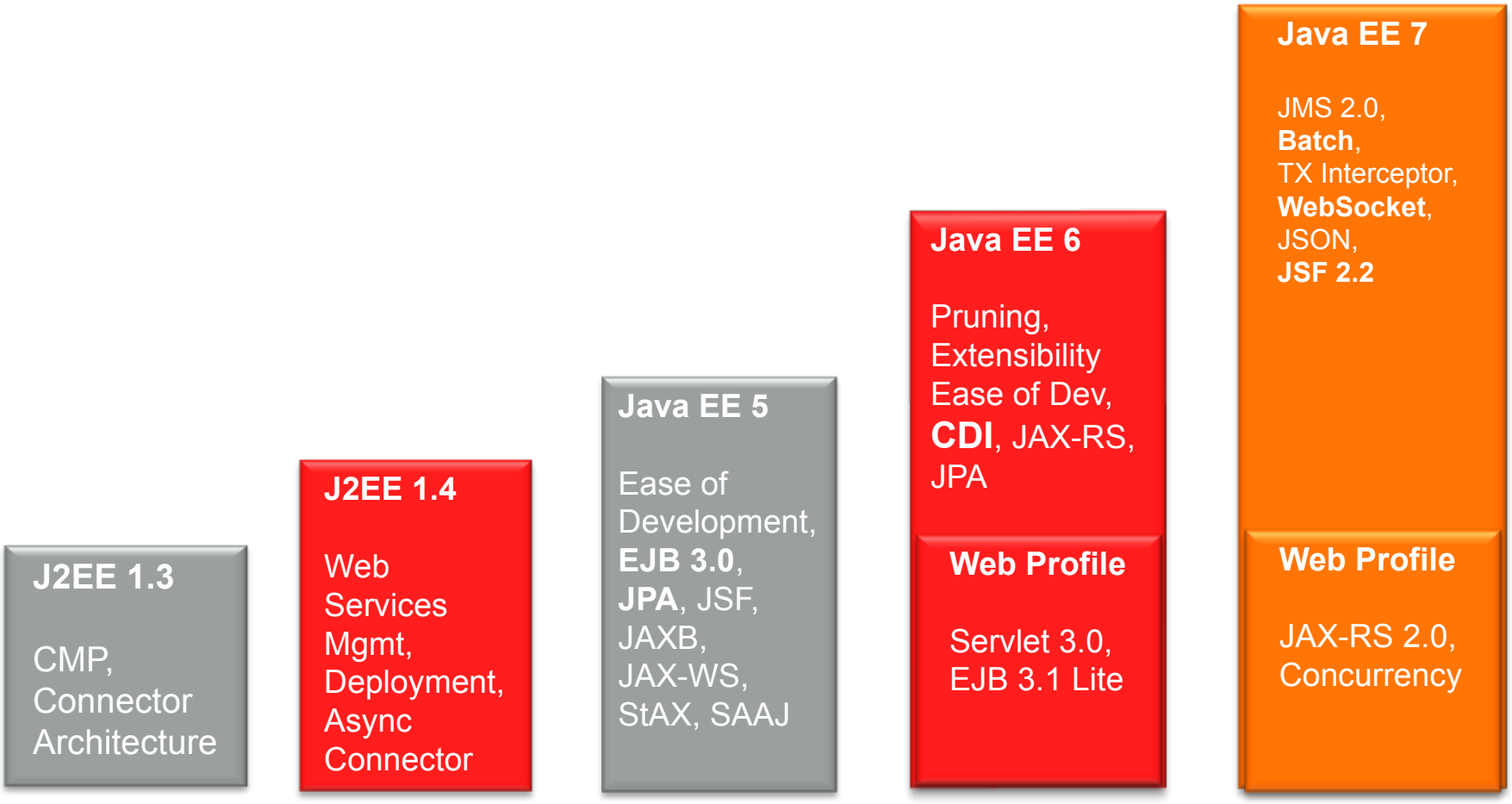
- Martijn Blankestijn
  - J-Technologies Innovation Board
  - Docent & Java ontwikkelaar
  - http://nl.linkedin.com/in/martijnblankestijn

Java EE 7
Agenda

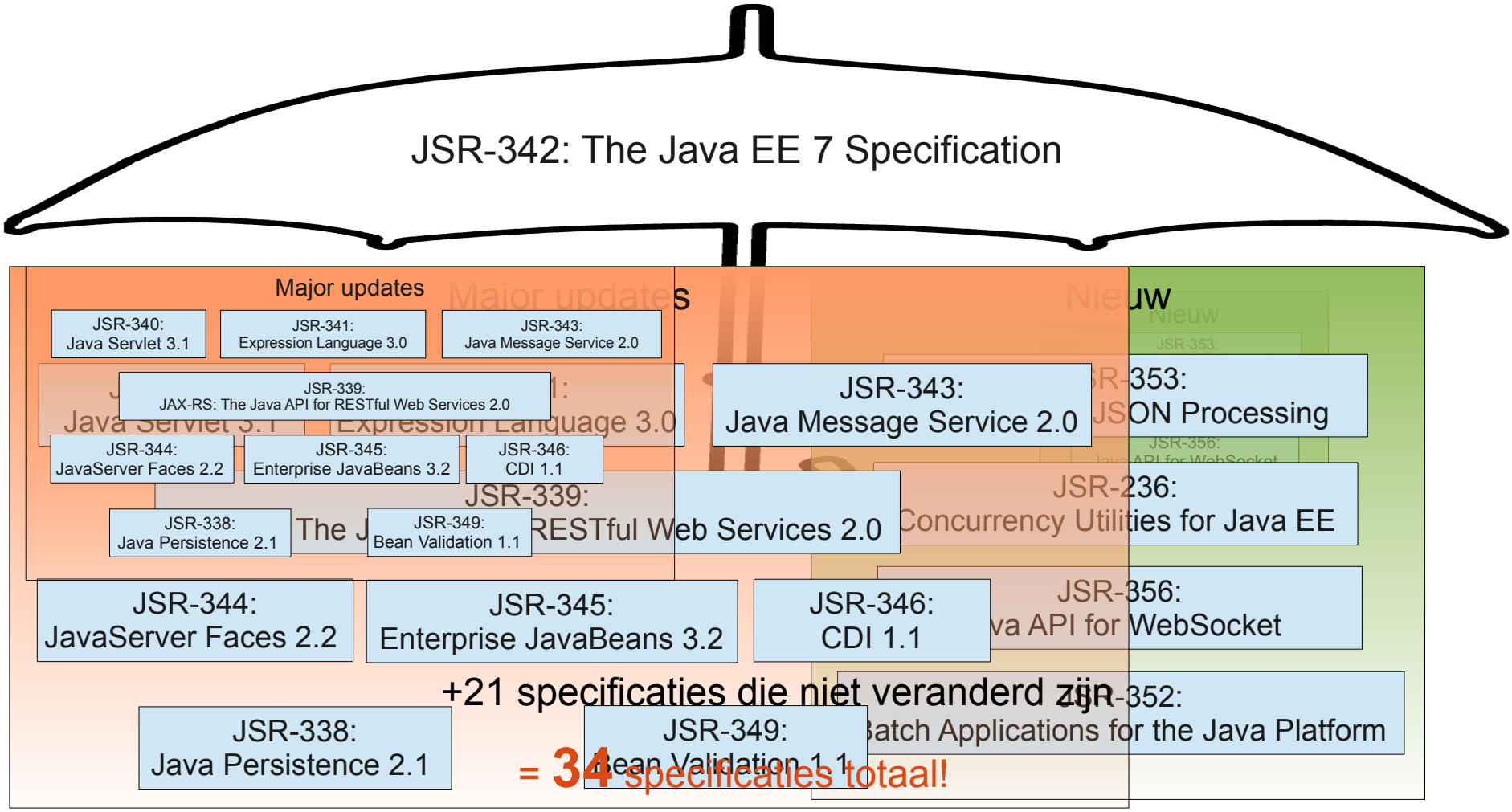
- Historie
- Totstandkoming
- JSON
- WebSockets
- Presentatietechnologie en HTML 5
- Batch verwerking
- REST
- Messaging
- Overige updates

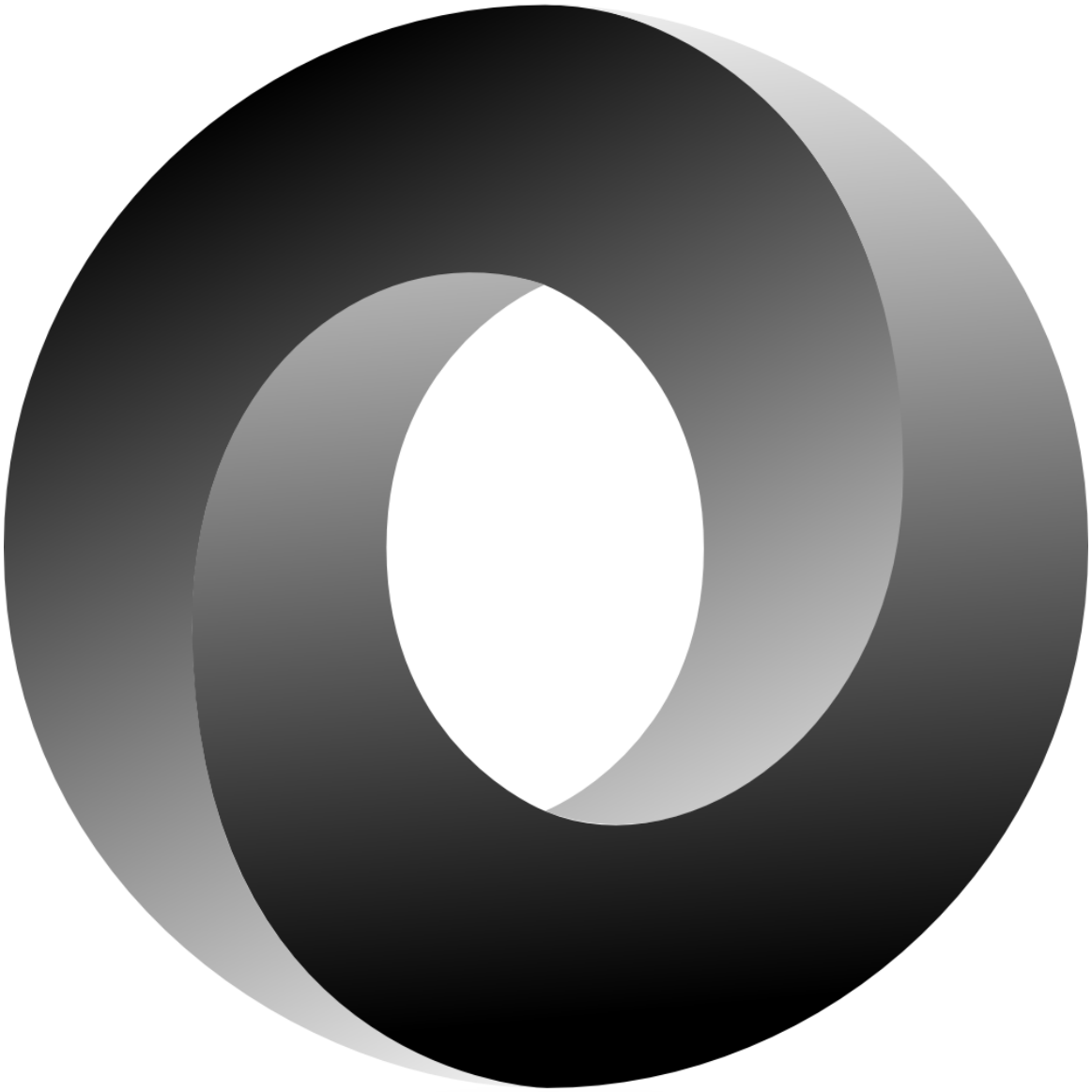
https://github.com/martijnblankestijn/javaee7

Java EE 7
Historie



Java EE 7
Totstandkoming



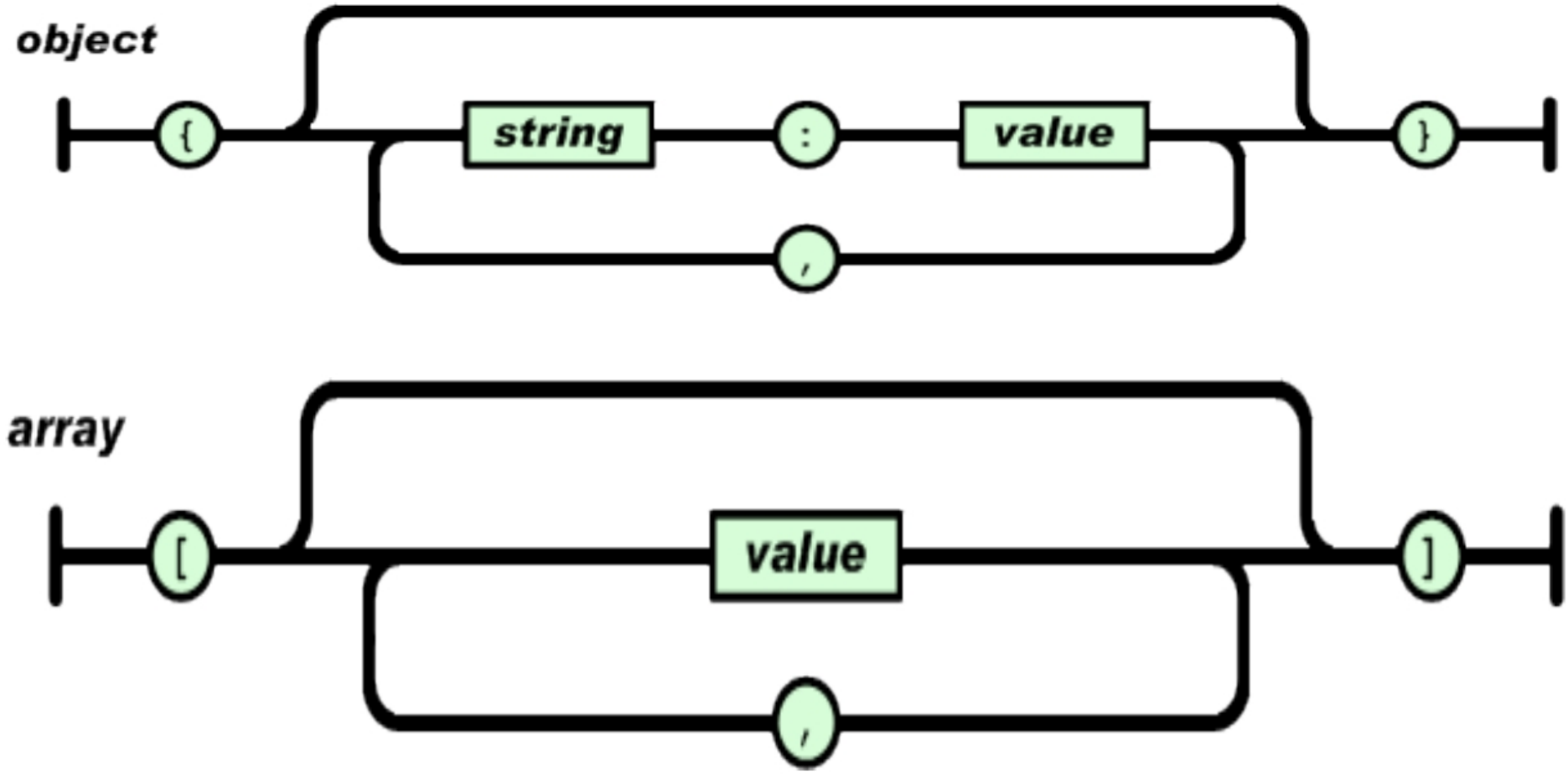


Java EE 7
JSON

- Tekst gebaseerd
- Leesbaar
- Lichtgewicht 'data exchange format'
- Veel gebruikt bij RESTful services
  - Facebook, twitter, …
- Gebaseerd op ECMA script

```
{
"previous_cursor": 0,
"ids": [143206502, 143201767, 777925],
"previous_cursor_str": "0",
...
}
```

JSON Grammar



JSON – Standaard API

- Parsen/verwerken JSON
- Data binding JSON text ↔ Java Objects
- Streaming API om JSON te produceren/consumeren
- Object Model API voor JSON (vergelijk DOM API)

```
public class Systeem {
  private long id;
  private String serialNumber;
  private long capacity;
```

Java EE 7
JSON – Streaming API – Writing


```
JsonGenerator writer = Json.createGenerator(System.out)
        .writeStartObject()
        .write("id", 5)
        .write("serialNumber", "ABC1223")
        .write("capacity", 125)
        .writeEnd();
writer.flush();
writer.close();
```

Java EE 7
JSON – Streaming API – Reading

JsonParser.Event

```
while (parser.hasNext()) {
  switch (parser.next()) {
    case START_OBJECT:   systeem = new Systeem(); break;
    case END_OBJECT:     System.out.println(systeem); return;
    case KEY_NAME:       key = parser.getString(); break;
    case VALUE_STRING:   systeem.setSerialNumber(parser.getString()); break;
    case VALUE_NUMBER:
      switch (key) {
        case "capacity": systeem.setCapacity(parser.getLong());              break;
        case "id":       systeem.setId(parser.getLong());                    break;
        default:         throw new IllegalArgumentException("Do not compute");
      } break;
  }}
```

JSON – Object Model API


```
JsonObject jsonObject = Json.createObjectBuilder()
        .add("id", 5)
        .add("serialNumber", "ABC1223")
        .add("capacity", 125)
        .build();

JsonWriter writer = Json.createWriter(System.out);
writer.writeObject(jsonObject);
writer.close();
```

JSON – Object Model API


```
JsonReader reader =
        Json.createReader(new StringReader(input));


Systeem systeem = new Systeem();


JsonObject object = reader.readObject();
systeem.setSerialNumber(object.getString("serialNumber"));
systeem.setId(          object.getInt("id"));
systeem.setCapacity(    object.getInt("capacity"));
```

JSON – Streaming versus Object Model API

STREAMING API

- Programmeur in control van parsen en/of generatie
- Event-gebaseerd
- Vergelijk StAX API for XML

- Pro
  Efficiënt
  Streaming (one event at the time)
- Con
  Low level

OBJECT MODEL API

- Random-access tree like structure
- Navigatie en query mogelijkheden
- Vergelijk DOM API for XML
- *Builder pattern*

- Pro
  random-access hele inhoud
- Con
  minder effiënt, meer geheugen dan streaming

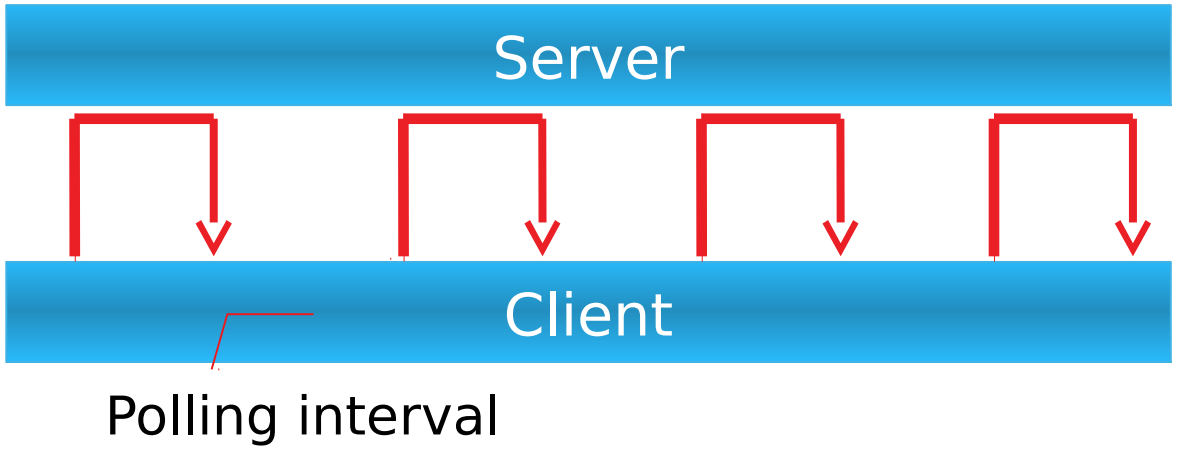
Java EE 7

The concept of “Real Time Web” is awesome.

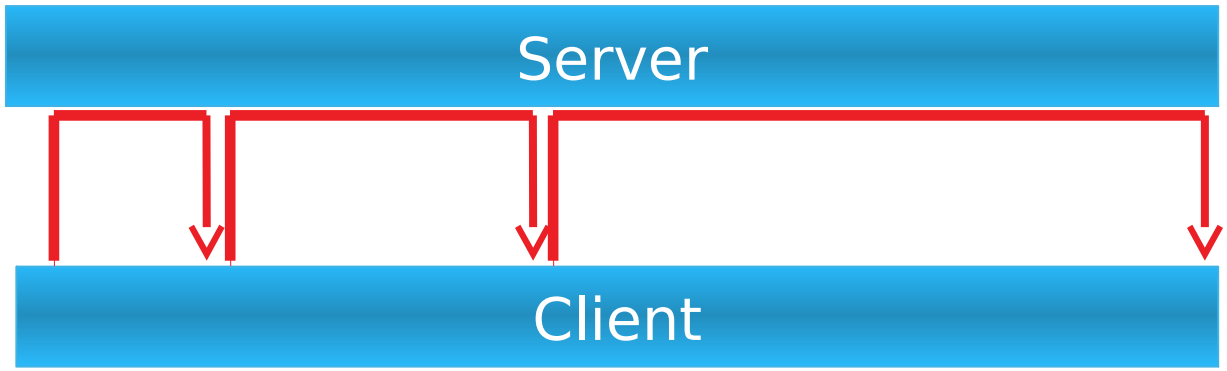
- HTTP is half-duplex
- HTTP is verbose
- Push vanuit de server
  - Polling
  - Long polling
  - Comet/Ajax
- Complex, inefficiënt

They can do better!!!

Polling

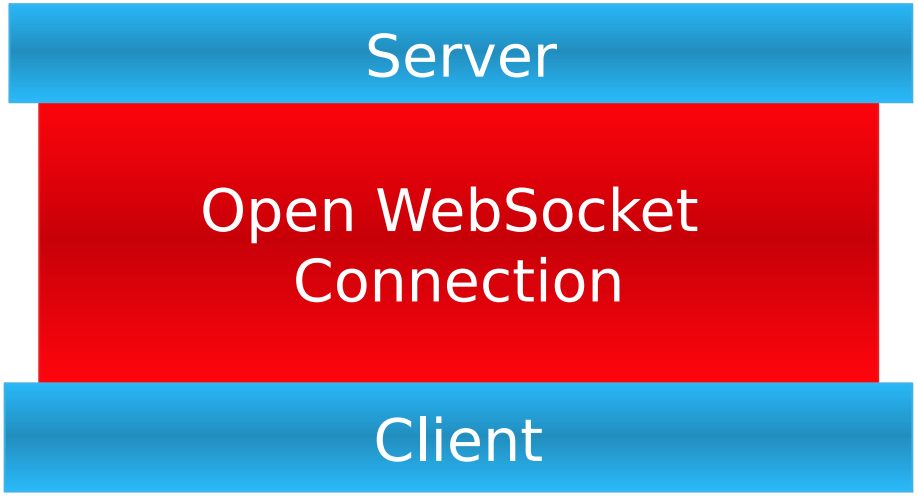


Long Polling



Wat is het?

- Full-duplex single socket connection
- TCP gebaseerd



Java EE 7
Tadááá …. Web Sockets

- JavaScript API — The WebSocket API

- Protocol — rfc 6455
  - Handshake
  - Data Transfer

The JavaScript part


```
Part of WebSocket Interface:
attribute Function onopen:
attribute Function onmessage:
attribute Function onerror:
attribute Function onclose:
```

Java EE 7
JavaScript Client Example


```
<script language="javascript" type="text/javascript">

 websocket =  new WebSocket("ws://localhost:8080/websocketsample/simple");

 websocket.onopen = function (evt)     { write("CONNECTED"); };
 websocket.onmessage = function (evt) { write("RECEIVED: " + evt.data); };
 websocket.onerror = function (evt)    {
     write('<span style="color: red;">ERROR:</span> ' + evt.data);
 };
 function send_echo() {
   websocket.send(textID.value);     write("SENT: " + textID.value);
 }
```

Java EE 7
JSR 356 – Server Endpoint


```
@ServerEndpoint(value = "/simple")
public class SimpleEndpoint {
  @OnOpen public void onOpen(Session client) { log("OPEN", client); }
  @OnClose public void onClose(Session client) { log("CLOSE", client); }
  @OnError public void onError(Session client, Throwable t) throws Exception { log(client, t); }

  @OnMessage  public void message(String message, Session client) {
    for (Session peer : client.getOpenSessions()) {
      if(peer.isOpen()) {
        try {
          peer.getBasicRemote().sendText(message);
        } catch (IOException e) {
          close(peer);
        }
      } } }
```

Java EE 7
Websocket Security

- Sluit aan bij servlet-security
- Ten tijde van de HTTP-UPGRADE

Bron: https://twitter.com/nonstopdemo

Highcharts JS

Interactive JavaScript charts for your web projects

Clustering overwegingen

Performance

Schaalbaarheid

JavaServer™ Faces

JSF

JSR 344: Java Server Faces 2.2

JSR 344: Java Server Faces 2.2
Veranderingen

Belangrijkste veranderingen

- HTML5 support
- Faces Flows
- Stateless views
- Resource Library Contracts

Voor een volledig overzicht van de veranderingen in JSF 2.2

http://jdevelopment.nl/jsf-22/

HTML5 support
Pass-through attributes

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:p="http://java.sun.com/jsf/passthrough">

    ...

    <h:form>
        <h:inputText value="#{bean.value}" p:placeholder="Enter text"/>
    </h:form>

    ...

</html>
```

```
<h:inputText value="#{bean.value}">
    <f:passThroughAttribute name="placeholder" value="Enter text" />
</h:inputText>
```

HTML5 support
Friendly markup

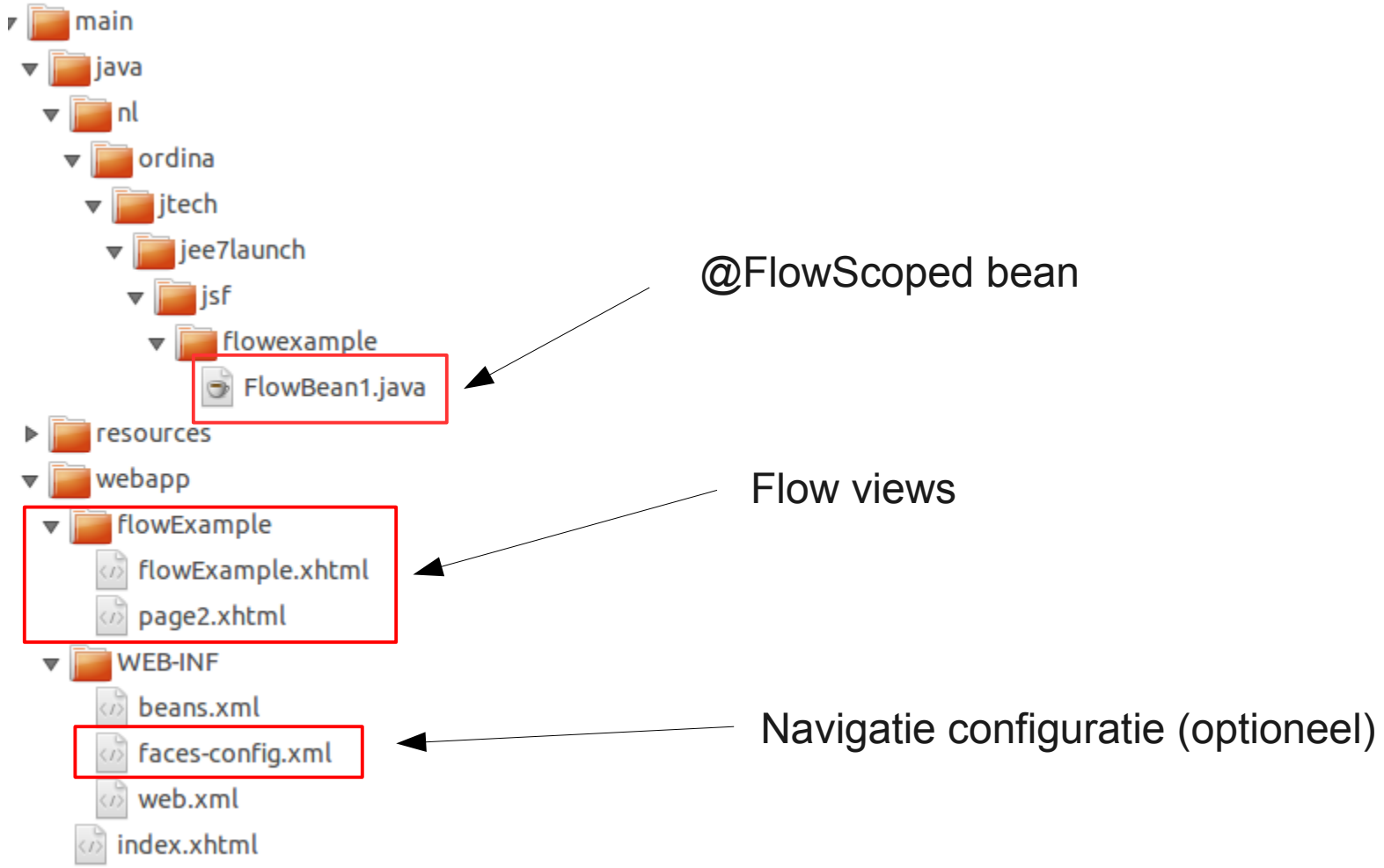
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:jsf="http://java.sun.com/jsf">

    ...

    <form jsf:id="form">
        <input jsf:id=”name” type=”text” jsf:value="#{bean.value}"
            placeholder="Enter text"/>
    </form>

    ...

</html>
```

Faces Flow
The 'missing' scope

- Lijkt op **Sping Web Flow** of **ADF Task Flows**
- Maakt het mogelijk om *Wizards* te implementeren
- Nesting is mogelijk (sub-flows)

DEMO

Faces Flow
Wat heb je nodig?



Configuratie
Mogelijkheid 1: Faces-config.xml

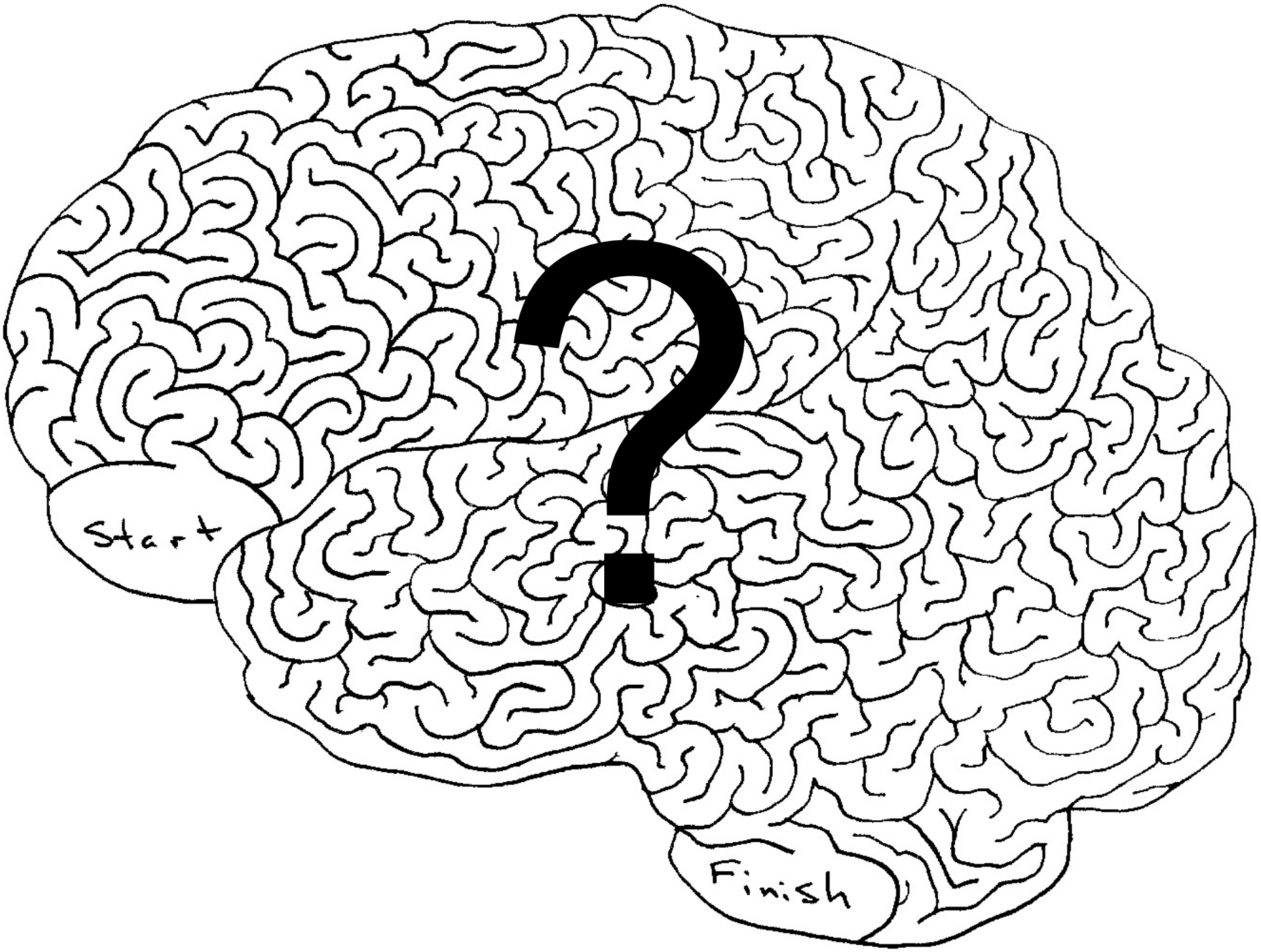
```
<flow-definition id="flowExample">
    <view id="start">
        <vdl-document>flowExample/flowExample.xhtml</vdl-document>
    </view>

    <flow-return id="returnHome">
        <from-outcome>/index.xhtml</from-outcome>
    </flow-return>
</flow-definition>
```

Configuratie
Mogelijkheid 2: [flowId]-flow.xml

- Zelfde type configuratie mogelijkheden als in faces-config maar dan in de flow directory zelf

Configuratie
Mogelijkheid 3: @FlowDefinition

```
public class FlowExample {

    @Produces @FlowDefinition
    public Flow defineFlow(@FlowBuilderParameter FlowBuilder flowBuilder) {
        String flowId = "flowExample";

        flowBuilder.id("", flowId);
        flowBuilder.viewNode(flowId, "/" + flowId + "/" + flowId +
            ".xhtml").markAsStartNode();
        flowBuilder.returnNode("returnHome").fromOutcome("/index.xhtml");

        return flowBuilder.getFlow();
    }
```


Start
Finish

Stateless Views
Waarom willen we dit?

- Performance
- Memory gebruik
- Niet 'sticky-sessie' gebonden
- Geen 'refresh' probleem
- De 'hipness' factor

Stateless Views
Hoe werkt het?

- Mogelijkheid bestond eigenlijk al
- Gebruik maken van de `transient` optie

Stateless Views
De view zelf

```
<f:view xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    transient="true">

   <html>
        <h:head/>
        <h:body>
             <h:form>
                 <!-- Input components here -->
             </h:form>
        </h:body>
    </html>

</f:view>
```

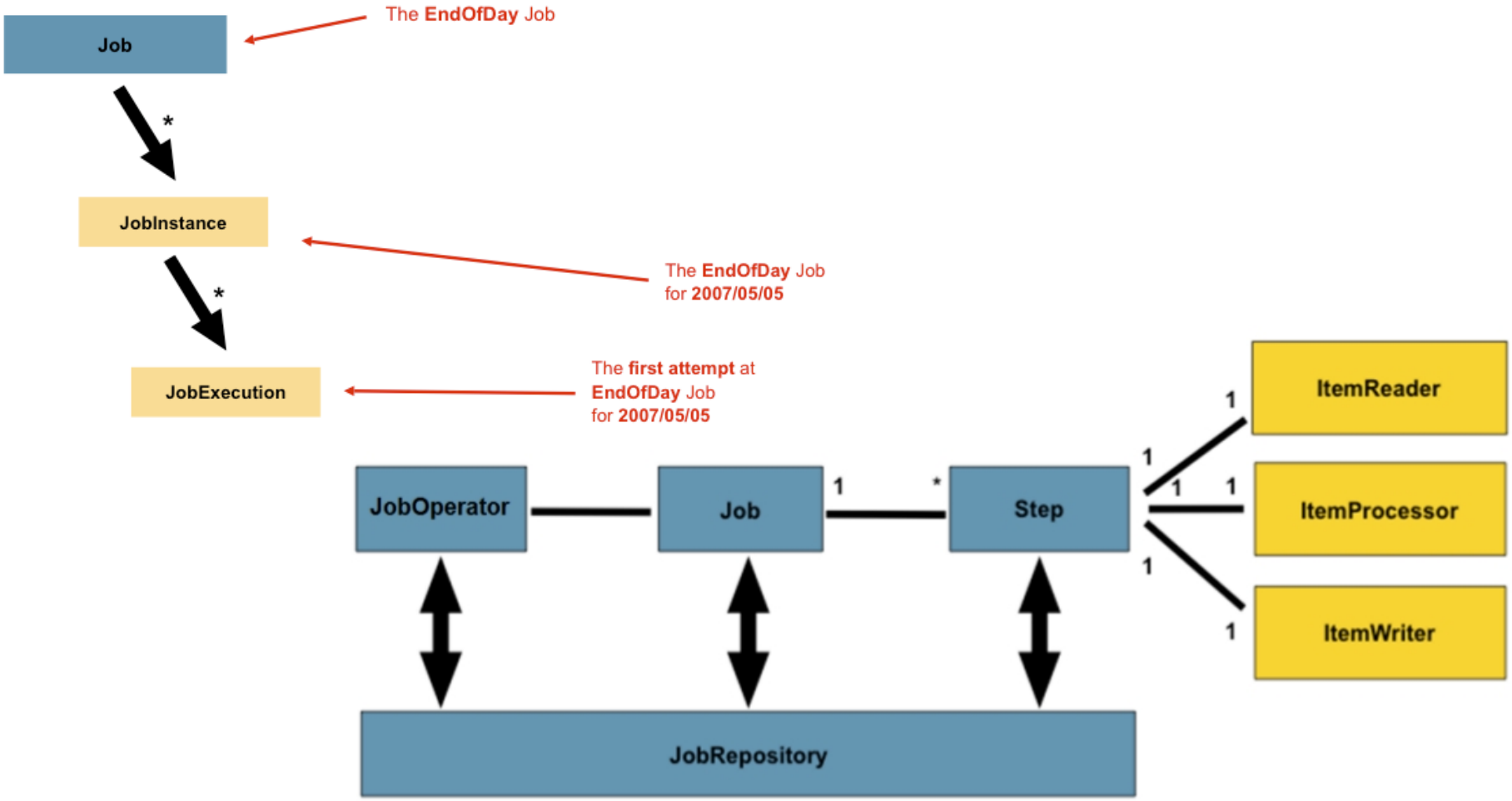
Stateless Views
Aandachtspunten

- Bestaande componenten kunnen het doen … of niet
- Geen mogelijkheid om de hele applicatie in 1x in stateless 'mode' te zetten

Source: http://thedeadhub.com/2011/07/04/at-what-temperature-should-beer-be-served/

Source: http://ifmycoastercouldtalk.bangordailynews.com/files/2013/03/P1080749.jpg

Java EE 7
Batch Application for the Java Platform – JSR 352



Java EE 7 – JSR 352
Job Configuration


```
<job id="inverterBatch" version="1.0" restartable="true" parent="false"
xmlns="http://xmlns.jcp.org/xml/ns/javaee" >
    <step id="readFiles">
        <listeners><listener ref="SkipInvalidRecordProcessListener"/> </listeners>
        <chunk checkpoint-policy="item" item-count="1000" skip-limit="25">
            <reader ref="InverterDataReader">
                <properties>
                  <property name="file" value="#{jobParameters['csv-file']}"/>
                 </properties>
            </reader>
            <processor ref="InverterItemProcessor"/>
            <writer ref="InverterDataWriter"/>
```

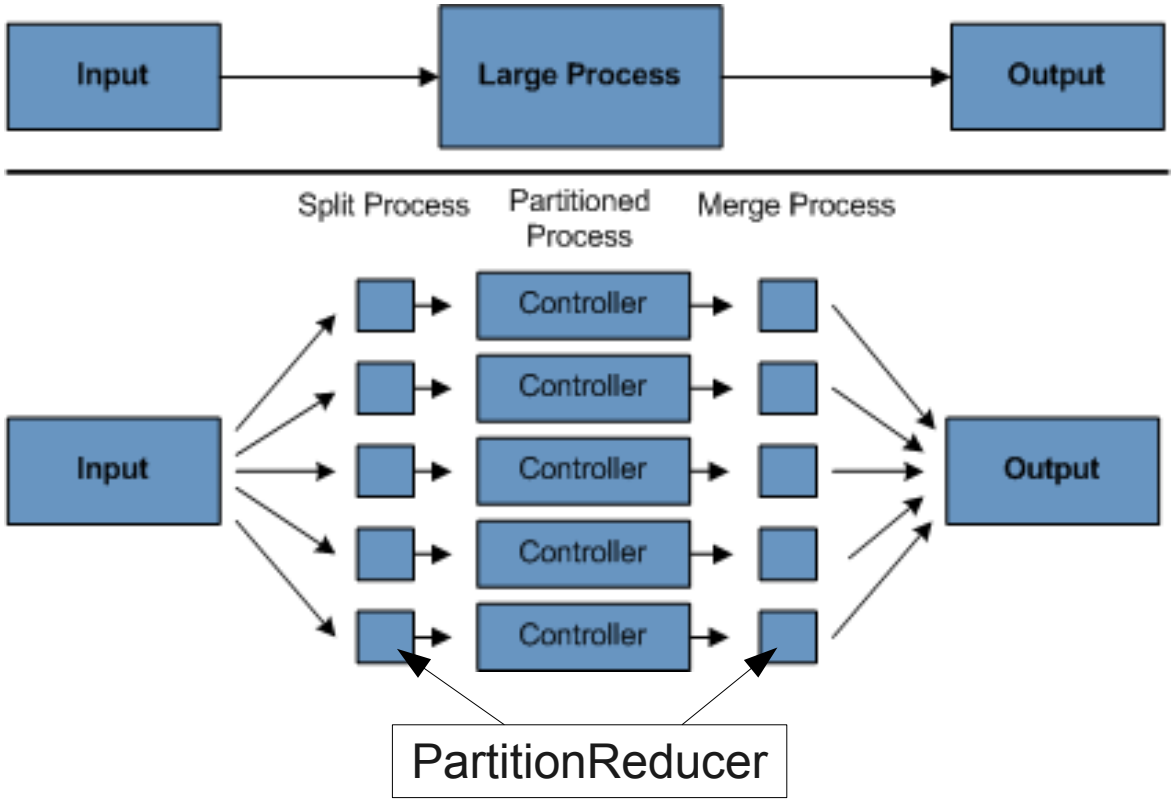
Batch Artifacts


```
<?xml version="1.0" encoding="UTF-8"?>
<batch-artifacts xmlns="http://xmlns.jcp.org/xml/ns/javaee" version="1.0">

    <ref id="InverterDataReader"
         class="nl.ordina.javaee7.batch.InverterDataReader" />

    <ref id="InverterDataWriter"
         class="nl.ordina.javaee7.batch.InverterDataWriter" />

    <ref id="InverterItemProcessor"
         class="nl.ordina.javaee7.batch.InverterDataItemProcessor" />
```

```
public class InverterDataReader implements ItemReader {
  @BatchProperty String inverterCsvFile;
  private Scanner scanner;

  @Override public void open(Serializable cp) throws Exception {
    scanner = new Scanner(Paths.get(inverterCsvFile));
  }

  @Override public void close() throws Exception {
    scanner.close();
  }
```

```
@Override public String readItem() throws Exception {
  if (isHeaderline()) {
    LOG.log(FINEST, "Headerline : {0} (ignored)", scanner.nextLine());
  }
  return determineOutput();
}

@Override public Externalizable checkpointInfo() throws Exception {
  LOG.log(FINEST, "Checkpoint info {0} ", checkpoint);
  return new ExternalizableAtomicInteger(checkpoint);
}
```

Partitioning



Source: http://static.springsource.org/spring-batch/batch-processing-strategies.html

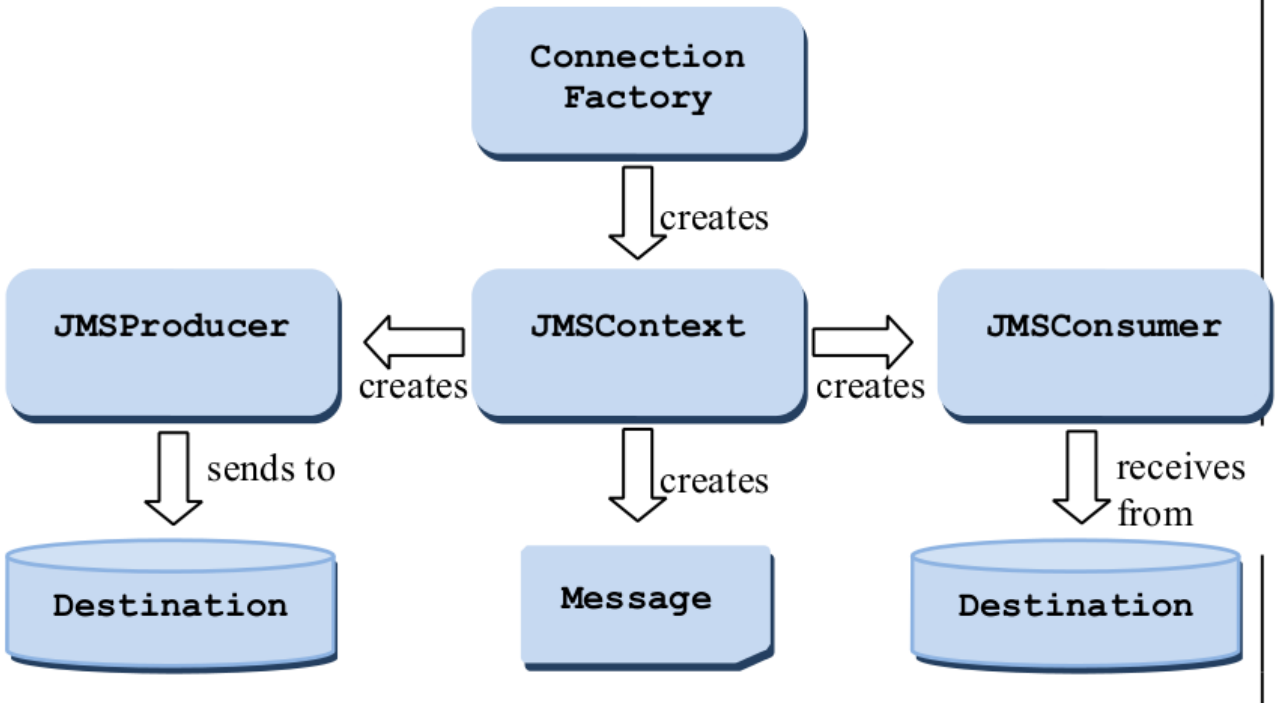

The
Old School
NEW & USED
ALL STYLES & FORMATS
708-366-7588
THEOLDSCHOOLRECORDS.COM
WE BUY
AND SELL
RECORDS
{ est. 1992 }
REVERSE
START/STOP
SPEED
33
45
PITCH ADJ

Java EE 7
JMS 1.1 + EJB 3.1


```
@Resource(lookup = "jms/connFactory") ConnectionFactory cf;
@Resource(lookup="jms/inboundQueue") Destination dest;

public void sendMessage (String payload) throws JMSException {
  Connection conn = cf.createConnection();
  Session sess = conn.createSession(false,Session.AUTO_ACKNOWLEDGE);
  MessageProducer producer = sess.createProducer(dest);
  TextMessage textMessage = sess.createTextMessage(payload);
  messageProducer.send(textMessage);
}
```

- Ease-of-development
- Nieuwe (verplichte) API

Figure 2-3 Overview of simplified API



JMS 2.0 – New Features

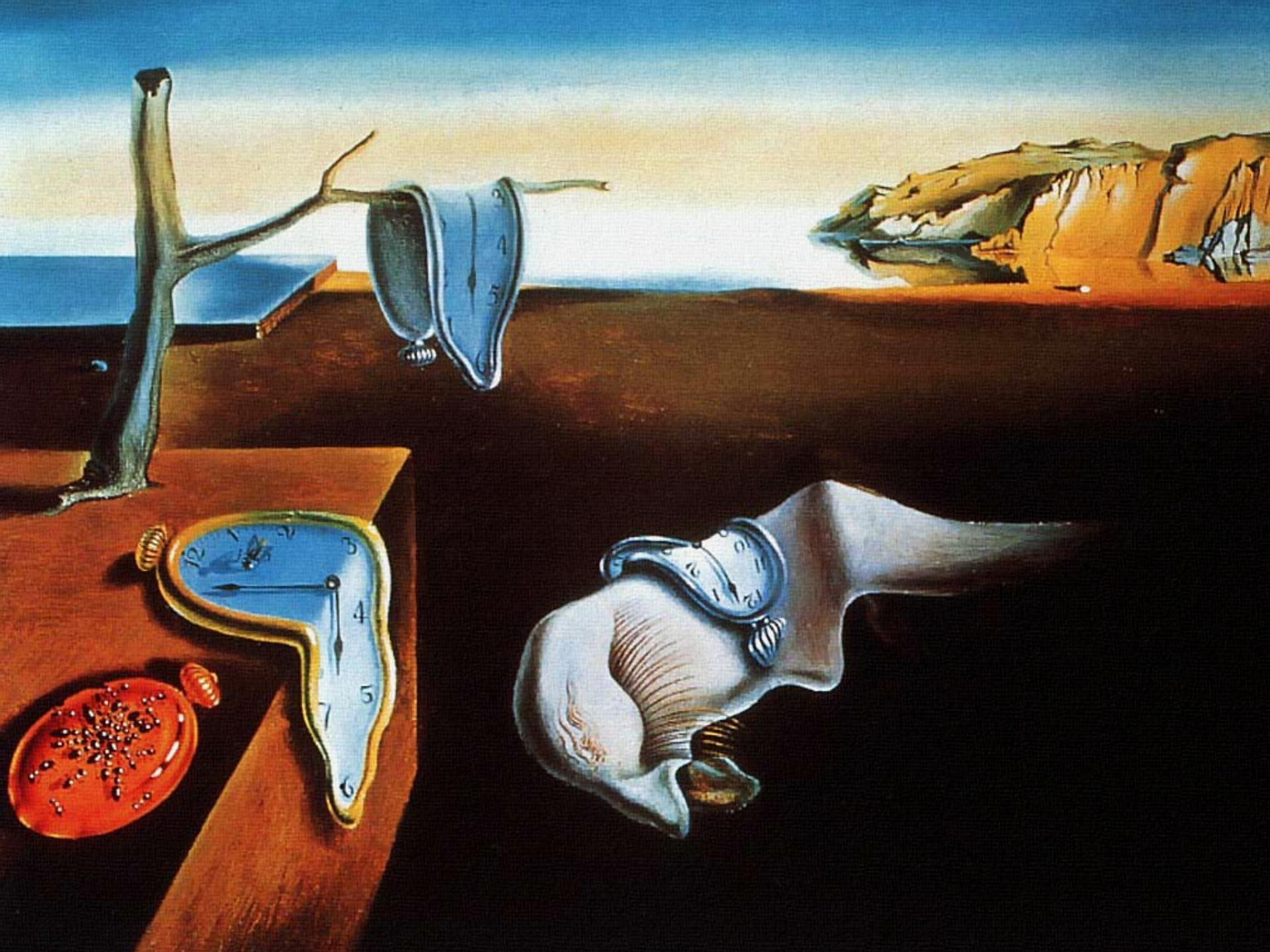
Nieuw

- Delivery delay
- Send asynchronous
- JMXDeliveryCount
- java.lang.AutoClosable

Java EE 7
JMS 2.0 – Voorbeeld


```
@Stateless public class FireAndForgetSender {

  @Inject JMSContext context;

  @Resource(mappedName = "jms/FireAndForgetQueue") Queue
queue;


  public void sendMessage(String content) {

    context.createProducer().send(queue, content);

  }
}
```

```
try {
    context.createProducer().send(queue, content);
} catch (JMSRuntimeException e) {}
```



Java EE 7 – Java Persistence API 2.1
Nieuw

- Conversie van attributen
- Aanroepen van bestaande database functies of door gebruiker gedefinieerde
- Criteria Update en Delete
- CDI in EntityListener
- Unsynchronized PersistenceContext
- Schema Generation (Index
- Entity graphs

Java EE 7 – Java Persistence API 2.1
Type Conversion of Basic Attributes


```
@Converter(autoApply = true)
public class GeslachtConverter implements AttributeConverter<Geslacht, Integer> {
  @Override public Integer convertToDatabaseColumn(Geslacht geslacht) {
    switch (geslacht) {
      case VROUW: return 2;
      ...
  }

  @Override public Geslacht convertToEntityAttribute(Integer integer) {
    switch (integer) {
      case 2: return VROUW;
      ...
  }
```

Type Conversion of Basic Attributes


```
public class Cursist {
  @Id @GeneratedValue private Long id;

  private Geslacht geslacht;

  @Convert(converter = LandConverter.class)
  private Land landVanHerkomst;
```

Automatische Conversie voor type

Criteria Update en Delete


```
CriteriaDelete<Cursist> criteriaUpdate =
        em.getCriteriaBuilder()
          .createCriteriaDelete(Cursist.class);

int aantalGeupdate =
         em.createQuery(criteriaUpdate).executeUpdate();
```

Java Persistence API 2.1 – Stored Procedure


```
StoredProcedureQuery storedProcedureQuery =
        em
          .createStoredProcedureQuery("JAVAEE7.PLUS")
          .registerStoredProcedureParameter(1, Integer.class, IN)
          .registerStoredProcedureParameter(2, Integer.class, IN)
          .registerStoredProcedureParameter(3, Integer.class, OUT)
          .setParameter(1, 5)
          .setParameter(2, 8);
boolean resultaat = storedProcedureQuery.execute();
```

```
@NamedStoredProcedureQuery(name="plus",

 procedureName="JAVAEE7.PLUS" , parameters = {

  @StoredProcedureParameter(name = "EERSTE", type = Integer.class, mode
= IN),

  @StoredProcedureParameter(name = "TWEEDE", type = Integer.class, mode
= IN),

  @StoredProcedureParameter(name = "RESULT", type = Integer.class, mode
= OUT)
})
```

```
  StoredProcedureQuery sp = em
          .createNamedStoredProcedureQuery("plus")
          .setParameter("EERSTE", 5)
          .setParameter("TWEEDE", 8);
  boolean resultaat = sp.execute();

  return (Integer) sp.getOutputParameterValue("RESULT");
```

Database generation options


```
<property name="javax.persistence.schema-generation.database.action"
          value="drop-and-create" />
<property name="javax.persistence.schema-generation.scripts.action"
         value="drop-and-create" />
<property name="javax.persistence.schema-generation.create-source"
          value="metadata" />
<property name="javax.persistence.schema-generation.scripts.create-target"
         value="target/create_ddl.sql"/>
<property name="javax.persistence.schema-generation.scripts.drop-target"
         value="target/drop_ddl.sql"/>
```

Java EE 7 – Java Persistence API 2.1
Entity Graph

- Template voor het pad en de grenzen voor een operatie of query
- Gebruikt voor 'fetch plans' voor query of find-operaties

```
@NamedEntityGraphs({
  @NamedEntityGraph(name = "beperkt",
      attributeNodes =
          {@NamedAttributeNode(value = "landVanHerkomst")}),
  @NamedEntityGraph(name = "volledig",
       includeAllAttributes = true)
})
```

Java EE 7 – Java Persistence API 2.1
Entity Graph gebruik


```
public List<Cursist> getLand(Land land) {
  return em
          .createQuery("...where c.landVanHerkomst = :land",
                       Cursist.class)
          .setParameter("land", land)
          .setHint("javax.persistence.loadgraph", "volledig")
          .getResultList();
}
```

REST TIME®

Wijzigingen:

- Filters and Interceptors
- JAX-RS en CDI integration
- Server-side Asynchronous HTTP
- Client API

Java EE 7
Just a regular JAX-RS 1.1 (Java EE 6) resource ...

CDI Integration

```
@Path("systemen") @ApplicationScoped
public class SysteemResource {


  @GET @Path("{systeemId}") @Produces({APPLICATION_JSON,
APPLICATION_XML})
  public Systeem get(@PathParam("systeemId") long id) {
    Systeem systeem = systemen.get(id);
    if(systeem==null) {
      throw new WebApplicationException(NOT_FOUND);
    }
    return systeem;
  }
```

Java EE 7
Server-side en client-side filters


```
public class CachingFilter implements ContainerRequestFilter,ContainerResponseFilter {
  private ConcurrentHashMap<Object, Response> cache = new ConcurrentHashMap<>();

  @Override public void filter(ContainerRequestContext req) throws IOException {
    Response response = cache.get(req.getUriInfo().getAbsolutePath());
    if (response != null) req.abortWith(response);
  }

  @Override public void filter(ContainerRequestContext req,
                               ContainerResponseContext resp) throws IOException {
    URI uri = req.getUriInfo().getAbsolutePath();
    cache.putIfAbsent(uri, Response.ok(resp.getEntity()).build());
```

Java EE 7

Server-side en client-side filters: How to bind them


```
@Provider
public class CachingFilter implements ContainerRequestFilter {
```

```
@NameBinding
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE, ElementType.METHOD})
public @interface Cached {}
```

```
@Provider @Cached
public class CachingFilter implements ContainerRequestFilter {
```

```
@Path("systemen") @ApplicationScoped
@Cached
public class SysteemResource {
```

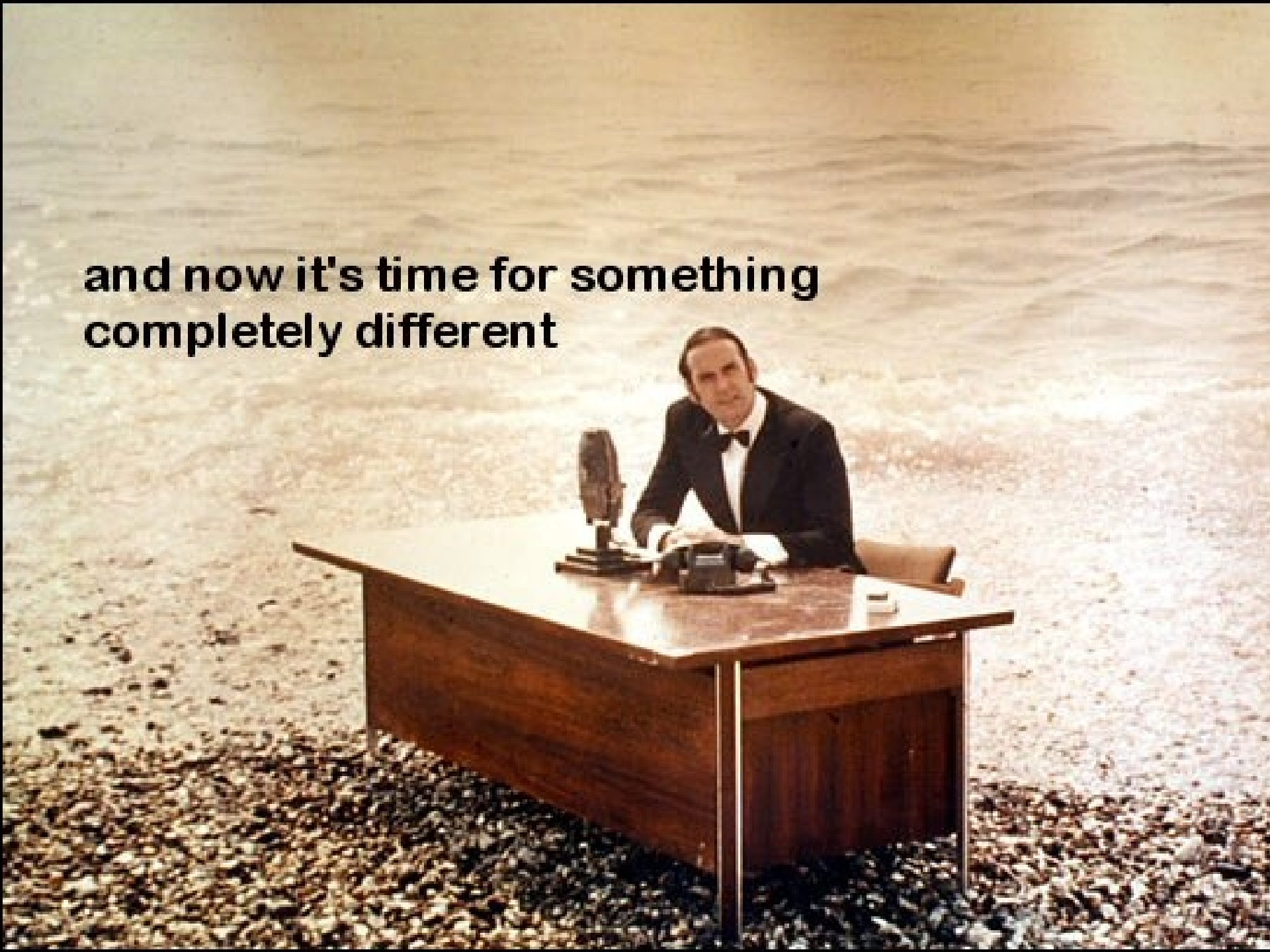
Java EE 7
Server-side en client-side filters: dynamic configuration


```
@Provider
public class CachingFeature implements DynamicFeature {
  @Override
  public void configure(ResourceInfo info, FeatureContext ctx) {
    if(info.getResourceMethod().isAnnotationPresent(GET.class))
{
      ctx.register(CachingFilter.class);
    }
  }
}
```

Bron: https://twitter.com/nonstopdemo

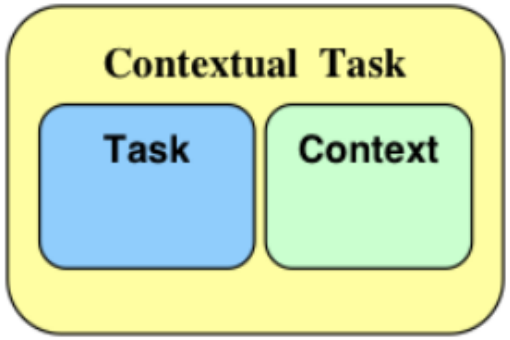
- Developer Account aangemaakt
- Applicatie geauthoriseerd
- En nu Jax-rs 2.0 client met Json

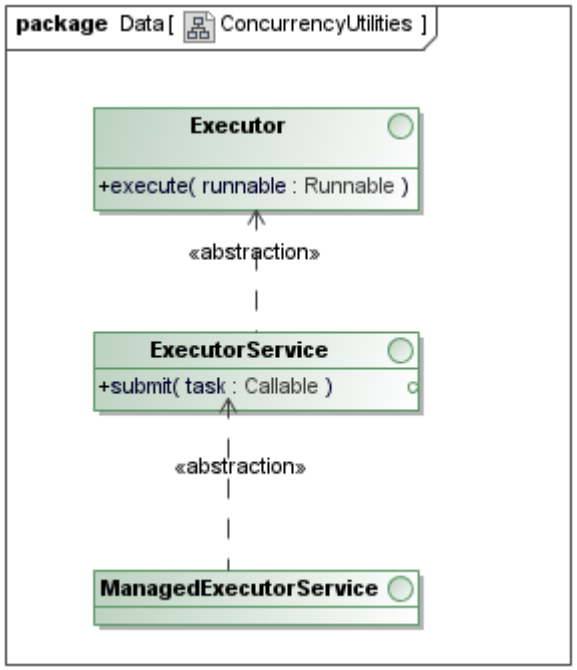
Bron: https://twitter.com/nonstopdemo


and now it's time for something
completely different

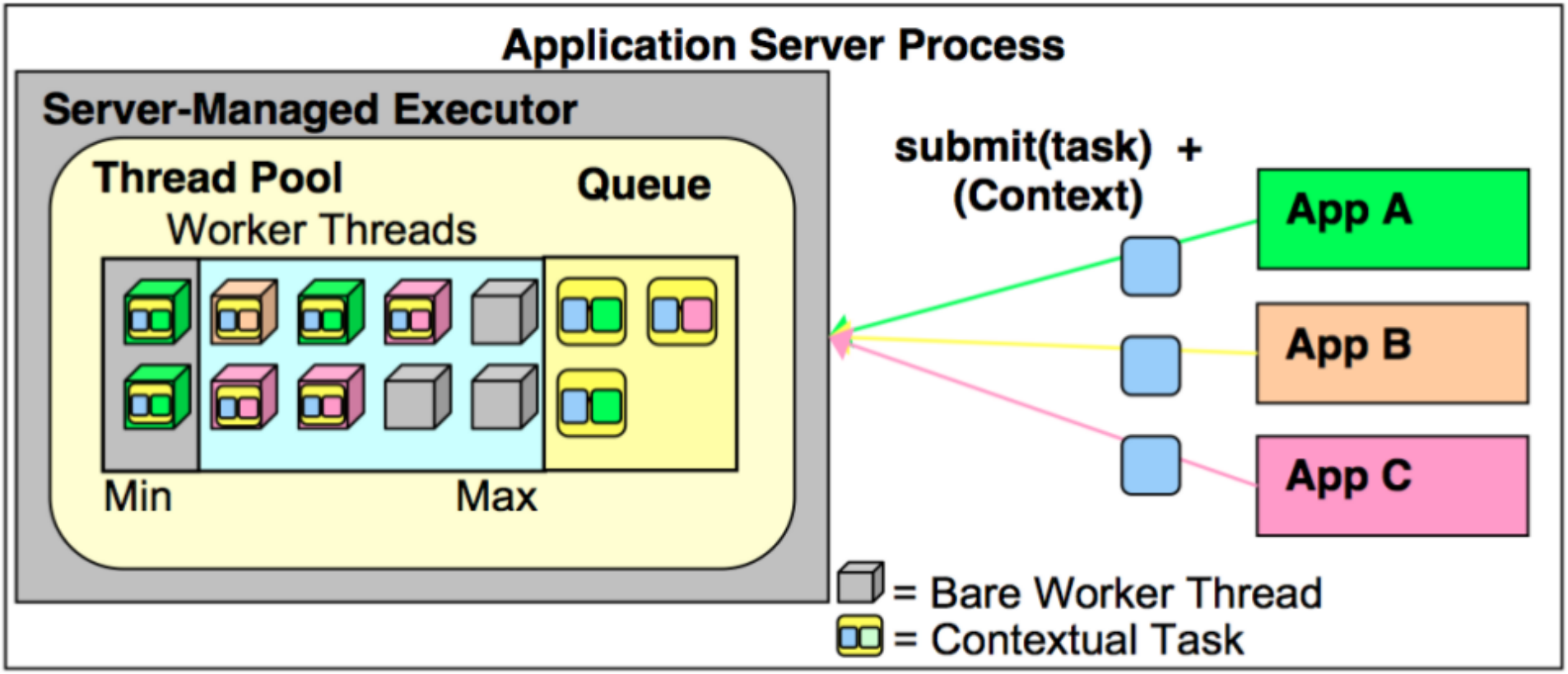
Java EE 7
Concurrency Utilities

- Veilige API voor concurrency binnen Java EE
- Bouwt op Java SE Concurrency
- Low-level API






```
public interface Callable<V> {
    V call() throws Exception;
}
```


Application Server Process
Server-Managed Executor
Thread Pool
Worker Threads
Queue
Min
Max
submit(task) +
(Context)
App A
App B
App C
= Bare Worker Thread
= Contextual Task

Java EE 7
Asynchroniciteit


```
@WebServlet("/report") public class ReportServlet extends HttpServlet {
  @Resource ManagedExecutorService mes;

  @Override protected void doPost(HttpServletRequest req,
      HttpServletResponse resp) throws ServletException, IOException {
    String reportName = req.getParameter("reportName");

    Future<String> reportFuture = mes.submit(new ReporterTask(reportName));

    req.getSession().setAttribute("report", reportFuture);
    resp.getWriter().write("<html><body><a href=\"report\">Check</a>");
    resp.getWriter().write("</body></html>");
  }
```

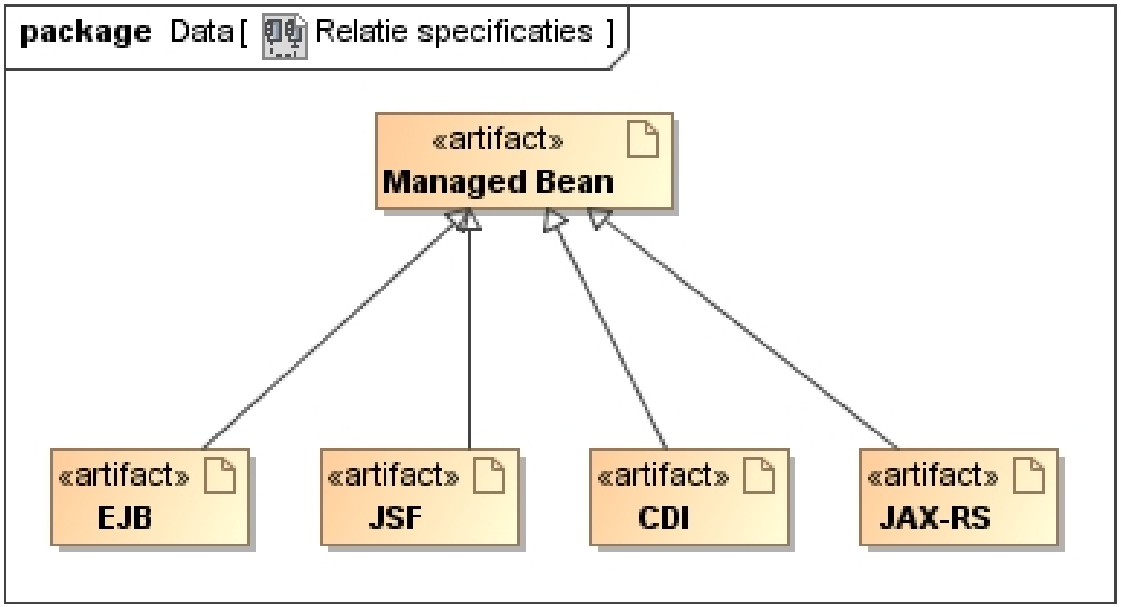
Java EE 7
Asynchroniciteit


```
@Override protected void doGet(HttpServletRequest req,
    HttpServletResponse resp) throws ServletException, IOException {

  Future<String> reportFuture =
          (Future<String>)_req.getSession().getAttribute("report");

  String resultaat = reportFuture.isDone() ? reportFuture.get():CHECK;
  resp.getWriter().write(resultaat);
```

```
public class ReporterTask implements Callable<String> {

  @Override public String call() throws Exception {
```

Java EE 7
Managed Bean Alignment

- JTA transactional interceptor



Java EE 7
JTA 1.2

- Declaratieve transacties voor CDI managed beans
  - @Transactional
  - TxType
- @TransactionScoped

```
public enum TxType {
    REQUIRED,
    REQUIRED_NEW,
    MANDATORY,
    SUPPORTS,
    NOT_SUPPORTED,
    NEVER
}
```

```
@Transactional(value = REQUIRED,
        rollbackOn = IllegalArgumentException.class,
        dontRollbackOn = EntityNotFoundException.class)
public class TransactionalCdiService {
  @Inject TransactionContextObject transactionContextObject;

  public void save(Object obj) {
    transactionContextObject.log("Just persisted it");
  }
}
```

```
@TransactionScoped
public class TransactionContextObject
                    implements Serializable {
```

Java EE 7
EJB 3.2

- New
  - EJB Lite: Local asynchronous method invocation & non-persistent Timer Service
  - getAllTimers
  - Disable passivation of stateful beans
- Pruning
  - EJB 2.1 Container Managed Persistence
  - EJB 2.1 Bean Managed Ppersistence
  - EJB Query Language
  - JAX-RPC

Expression Language – Stand Alone


```
@Test
public void testSetEl() {
  ELProcessor elProcessor = new ELProcessor();


  System.out.println(
          elProcessor.eval("[1,2,3.4].size()"));
}
```

Java EE 7
Expression Language – Closures

```
String expr = "persons.stream()" +
                      ".filter(p->p.age > 18)" +
                      ".map(p->p.age)" +
                      ".sum()";
assertEquals(75L, elProcessor.eval(expr));
```

Java EE 7
Validation 1.1

- Method validation
- Integratie CDI en Bean Validation
  Dependency injection in ConstraintValidators
- Message interpolation met EL Expressies

Validation – Method Validation

- “Programming by Contract" (PbC).
  - Pre-condities
  - Post-condities

Validation – Method Validation


```
@ApplicationScoped
public class MethodValidationS

  @NotNull @Future
  public Date estimateResignDate(
          @NotNull @Size(min=5, max=8) String key,
          @NotNull @Past Date employmentDate) {

    return estimateResignDate(employmentDate);
  }
```

Post-condition

Pre-conditions

Servlet 3.1

- Non-blocking I/O
- HTTP protocol upgrade mechanism
- Change Session ID on authentication

Server sent events

Context and Dependency Injection 1.1 – JSR 346

Changes

- Global enablement of decorators en interceptors
- @Vetoed
- CDI – eenvoudige (globale) toegang tot CDI container

```
<scan>
  <exclude name="nl.ordina.javaee7.tobeignored.**" />
</scan>
```


Cache

https://github.com/martijnblankestijn/javaee7

CONNECTIVATE

www.ordina.nl