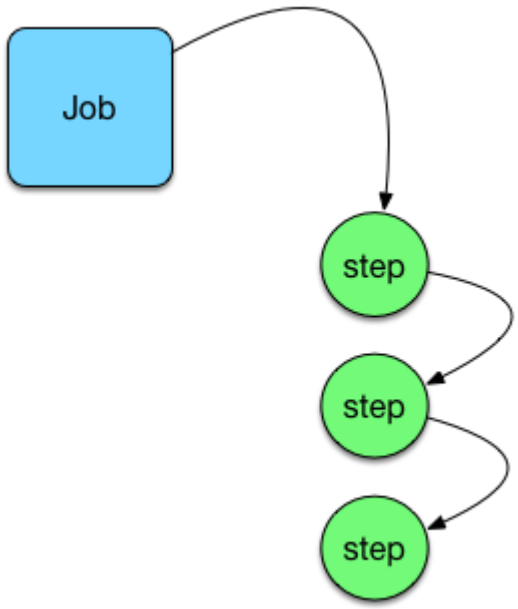
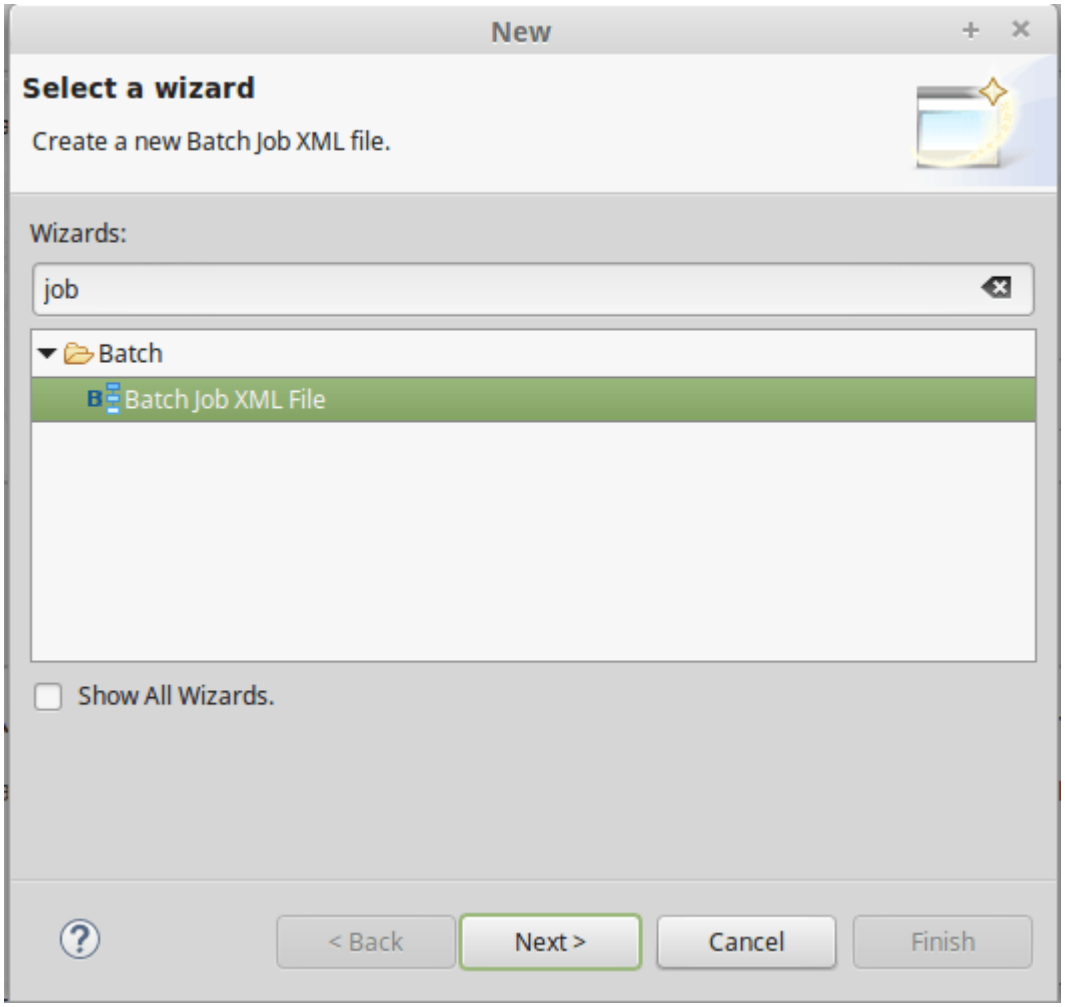
Java EE Batch es una de las especificaciones JSR 352 añadidas por la plataforma Java EE 7 . Esta especificación esta orientada a la creación de procesos batch dentro del mundo Java. Vamos a un una pequeña introducción a su funcionamiento.

Java EE Batch y Jobs

Los procesos Batch tienen varios conceptos importantes en cuanto a la especificación se refiere. Uno de los mas importantes es el de Job o trabajo , que se encarga de definir la estructura de un trabajo concreto.



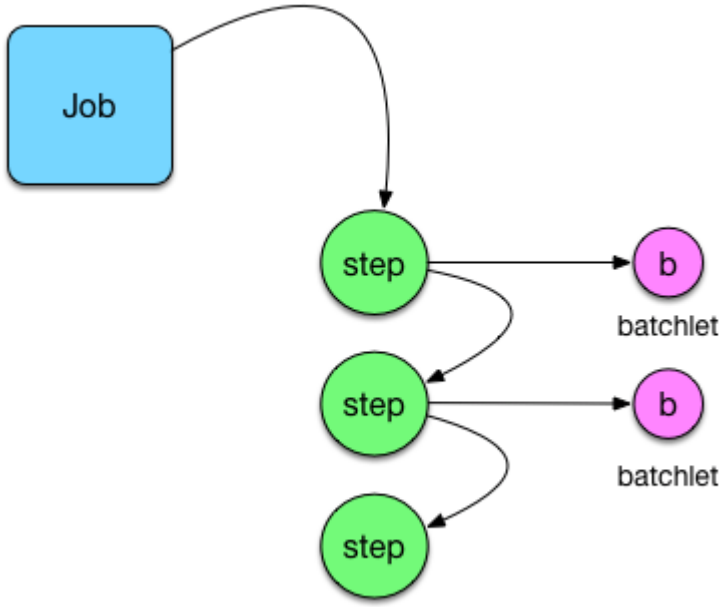
Un trabajo puede estar compuesto de varios pasos o steps . Vamos a crear un nuevo proyecto Web y crear un trabajo. Para ello me voy a apoyar en Jboss Developer Studio.Esta herramienta permite la creación de un trabajo a través de un asistente .



El asistente nos creará en fichero denominado job.xml en el cual podemos definir los diferentes pasos de nuestro trabajo.



Cada uno de los pasos puede estar ligado a varios elementos que gestionan nuestros procesos batch .



En este caso vamos a ligarlos a 2 Batchlet que son las unidades más básicas a nivel de procesos Batch. Vamos a ver las modificaciones que tenemos que hacer al fichero XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<job id="job" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/jobXML_1_0.xsd"
version="1.0">
<step id="paso1" next="paso2">
<batchlet ref="batchLet"></batchlet>
</step>
<step id="paso2">
<batchlet ref="batchLet2"></batchlet>
</step>
```

```
</job>
```

Acabamos de ver como enlazar dos Batchlet en nuestro proceso , es momento de construir su código.

```
package com.arquitecturajava.batch;

import javax.batch.api.AbstractBatchlet;
import javax.enterprise.context.Dependent;
import javax.inject.Named;

@Dependent
@Named
public class BatchLet extends AbstractBatchlet{

@Override
public String process() throws Exception {
System.out.println("paso uno del proceso batch");

return null;
}

}
```

```
package com.arquitecturajava.batch;

import javax.batch.api.AbstractBatchlet;
import javax.enterprise.context.Dependent;
import javax.inject.Named;

@Dependent
@Named
public class BatchLet2 extends AbstractBatchlet{

@Override
public String process() throws Exception {
System.out.println("paso dos del proceso batch");

return null;
}

}
```

Los Batchlet que acabamos de crear imprimen un mensaje por la consola del servidor. Vamos a invocar nuestro job desde un Servlet para ver como este trabajo se ejecuta.

```
import java.util.Properties;

import javax.batch.operations.JobOperator;
import javax.batch.runtime.BatchRuntime;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

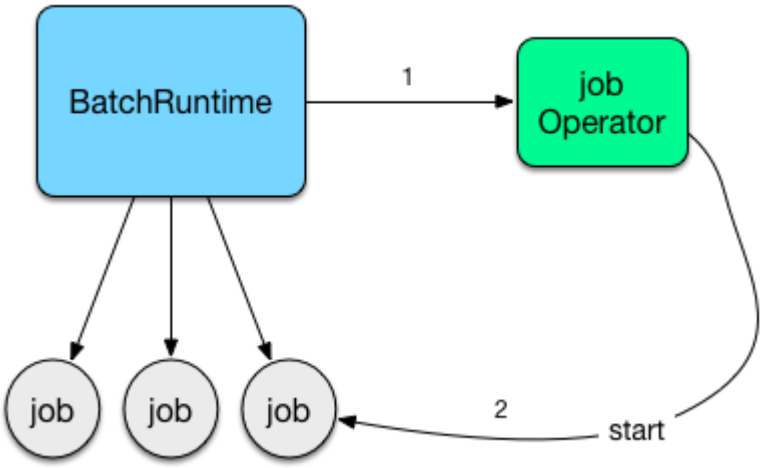

@WebServlet("/ServletBatch")
public class ServletBatch extends HttpServlet {
private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
JobOperator operador=BatchRuntime.getJobOperator();
operador.start("job", new Properties());

response.getWriter().append("proceso batch
").append(request.getContextPath());
}


}
```

En este caso hemos usado el BatchRunTime que es el encargado de gestionar todos los procesos batch y le hemos solicitado un operador. El operador será el encargado de que lancemos el trabajo.



Podremos ver el resultado por la consola de JBoss acabamos de ejecutar nuestro primer ejemplo de Java EE Batch

```
16:15,313 INFO  [stdout] (Batch Thread - 1) paso uno del proceso batch
16:15,320 INFO  [stdout] (Batch Thread - 1) paso dos del proceso batch
```

Otros artículos relacionados: Java EE EJB , EJB remoto