Modélisation et interopérabilité : Semaine 49, cours 11

Benoît Valiron <benoit.valiron@monoidal.net>

http://inf356.monoidal.net/

Simple API for XML (SAX)

Lecture d'un document XML

- On a vu : DOM
  - Document : Structure d'arbre
  - Problème : tout le document est chargé en mémoire
  - Avantage : accès aléatoire simple
- Aujourd'hui : traitement évènementiel.
  - Document : lu par un automate
  - Génération d'un flux d'évènements

Évènements ?

- Rapport des choses rencontrées au cours de la lecture.
- Chaque événement peut donner certaines infos.
  - startDocument
  - endDocument
  - startElement :
    - String namespaceURI, localName, qualifiedName;
    - Attributes atts;
  - endElement
    - String namespaceURI, localName, qualifiedName;
  - characters et ignorableWhitespaces
    - char[] text;
    - int start, length;
  - (5 autres évènements)

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Exemple

```
<livre lang=”fr”>
   <titre>XML</titre>
   <auteur>Chagnon</auteur>
</livre>
```

Évenement : **startDocument**
Attributs :
(aucun)

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **startElement**
Attributs :
namespaceURI : ''
localName : livre
qualifiedName: livre
atts :
{'lang' → 'fr'}

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **ignorableWhitespace**
Attributs :
text : '\n '
start : 0
length : 3

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **startElement**
Attributs :
namespaceURI : ''
localName : titre
qualifiedName: titre
atts :
{}

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **characters**
Attributs :
text : '<titre>XML</titre>'
start : 7
length : 3

Note : le texte peut être n'importe quel sous-ensemble, pour autant que `start` et `length` correspondent

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **endElement**
Attributs :
namespaceURI : ''
localName : titre
qualifiedName: titre

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **ignorableWhitespace**
Attributs :
text : '>\n <'
start : 1
length : 3

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **startElement**
Attributs :
namespaceURI : ''
localName : auteur
qualifiedName: auteur
atts :
{}

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **characters**
Attributs :
text : 'Chagnon</au'
start : 0
length : 7

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **endElement**
Attributs :
namespaceURI : ''
localName : auteur
qualifiedName: auteur

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **ignorableWhitespace**
Attributs :
text : '\n <'
start : 0
length : 3

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **endElement**
Attributs :
namespaceURI : ''
localName : livre
qualifiedName: livre

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **endDocument**
Attributs :
(aucun)

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Résumé :

```
startDocument
startElement
ignorableWhitespace
startElement
characters
endElement
ignorableWhitespace
startElement
characters
endElement
ignorableWhitespace
endElement
endDocument
```

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Exemple avec espace de nom

```
<bio lang="fr" xmlns="http://ma-biographie"
               xmlns:h="http://www.w3.org/1999/xhtml">
  <nom né="1912" mort="1954">Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **startDocument**
Attributs :
(aucun)

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **startElement**
Attributs :
namespaceURI : 'http://ma-biographie'
localName : 'bio'
qualifiedName: 'bio'
atts :
{'lang' → 'fr'}

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **ignorableWhitespace**
Attributs :
text : '\n <'
start : 0
length : 3

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **startElement**
Attributs :
namespaceURI : 'http://ma-biographie'
localName : 'nom'
qualifiedName: 'nom'
atts :
{'né' → '1912' ; 'mort' → '1954' }

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **characters**
Attributs :
text : '>Turing</nom'
start : 1
length : 6

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **endElement**
Attributs :
namespaceURI : 'http://ma-biographie'
localName : 'nom'
qualifiedName: 'nom'

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **characters**
Attributs :
text : 'nom>\n est un <'
start : 4
length : 10

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **startElement**
Attributs :
namespaceURI : 'http://www.w3.org/1999/xhtml'
localName : 'i'
qualifiedName: 'h:i'
atts :
{}

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **characters**
Attributs :
text : '<h:i>informaticien'
start : 5
length : 13

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **endElement**
Attributs :
namespaceURI : 'http://www.w3.org/1999/xhtml'
localName : 'i'
qualifiedName: 'h:i'

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **characters**
Attributs :
text : '.\n'
start : 0
length : 2

Exemple avec espace de nom

```
<bio lang=”fr” xmlns=”http://ma-biographie”
               xmlns:h=”http://www.w3.org/1999/xhtml”>
  <nom né=”1912” mort=”1954”>Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **endElement**
Attributs :
namespaceURI : 'http://ma-biographie'
localName : 'bio'
qualifiedName: 'bio'

Exemple avec espace de nom

```
<bio lang="fr" xmlns="http://ma-biographie"
               xmlns:h="http://www.w3.org/1999/xhtml">
  <nom né="1912" mort="1954">Turing</nom>
  est un <h:i>informaticien</h:i>.
</bio>
```

Évenement : **endDocument**
Attributs :
(aucun)

Push (SAX) versus Pull (StAX)

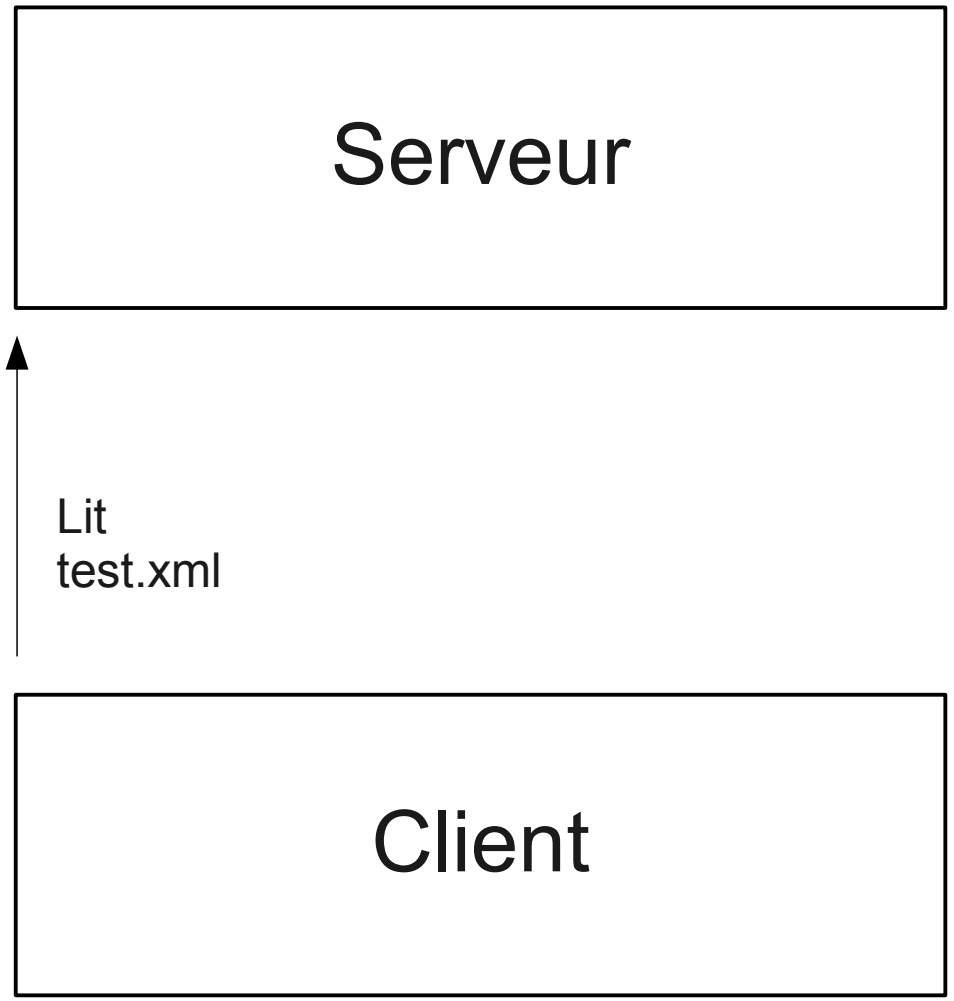
Serveur : Génère des évènements

Client : Traite les évènements

test.xml :

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

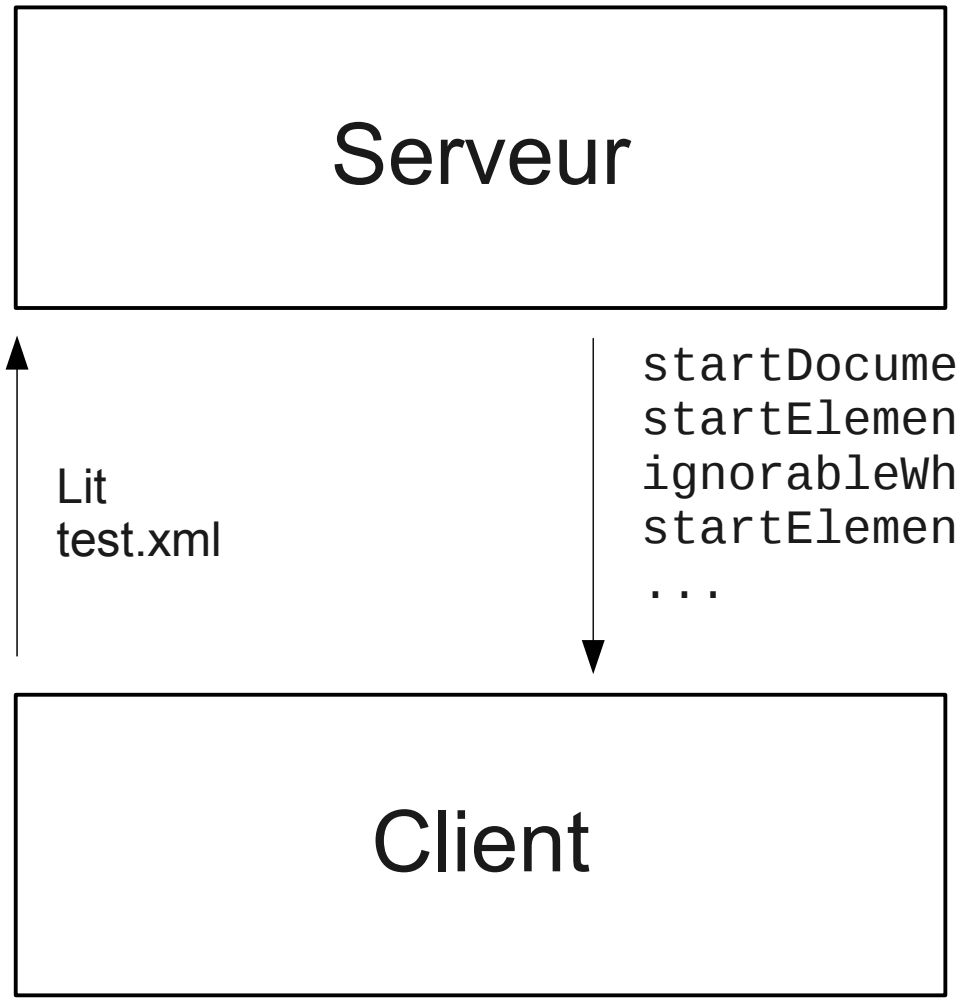
Push



test.xml :

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Push



test.xml :

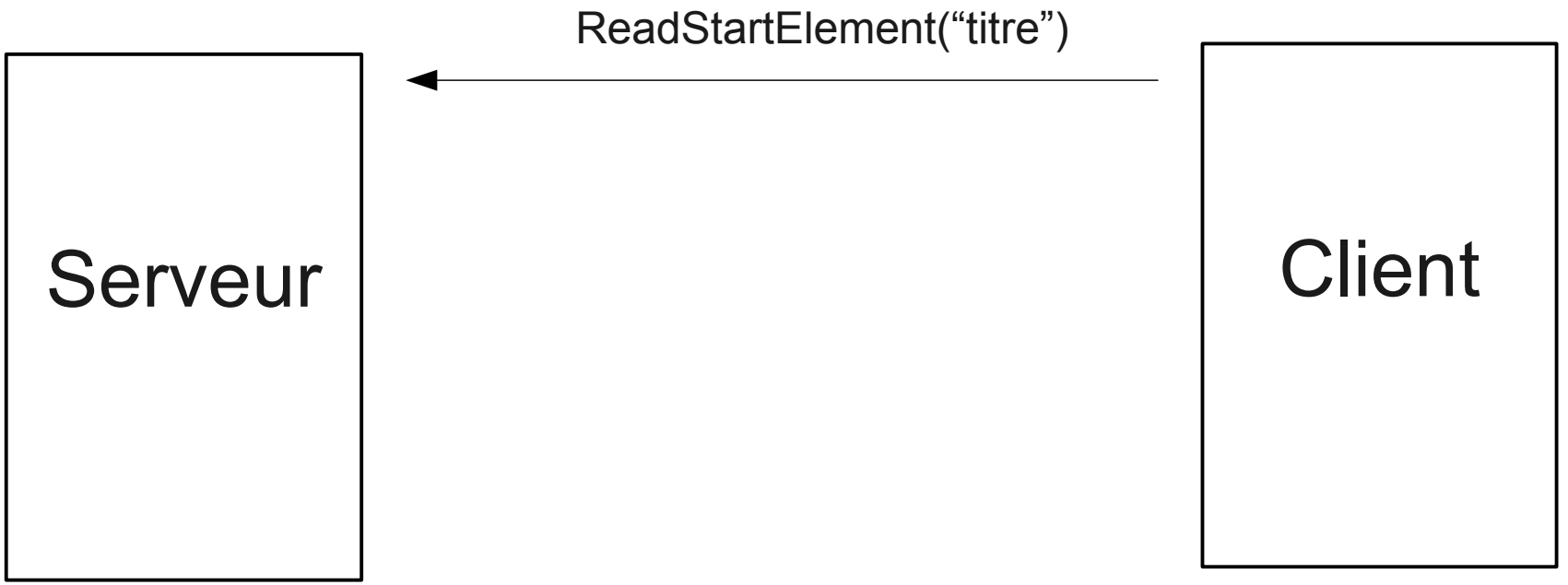
```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Le flux de données est géré par le client

Pull

test.xml :

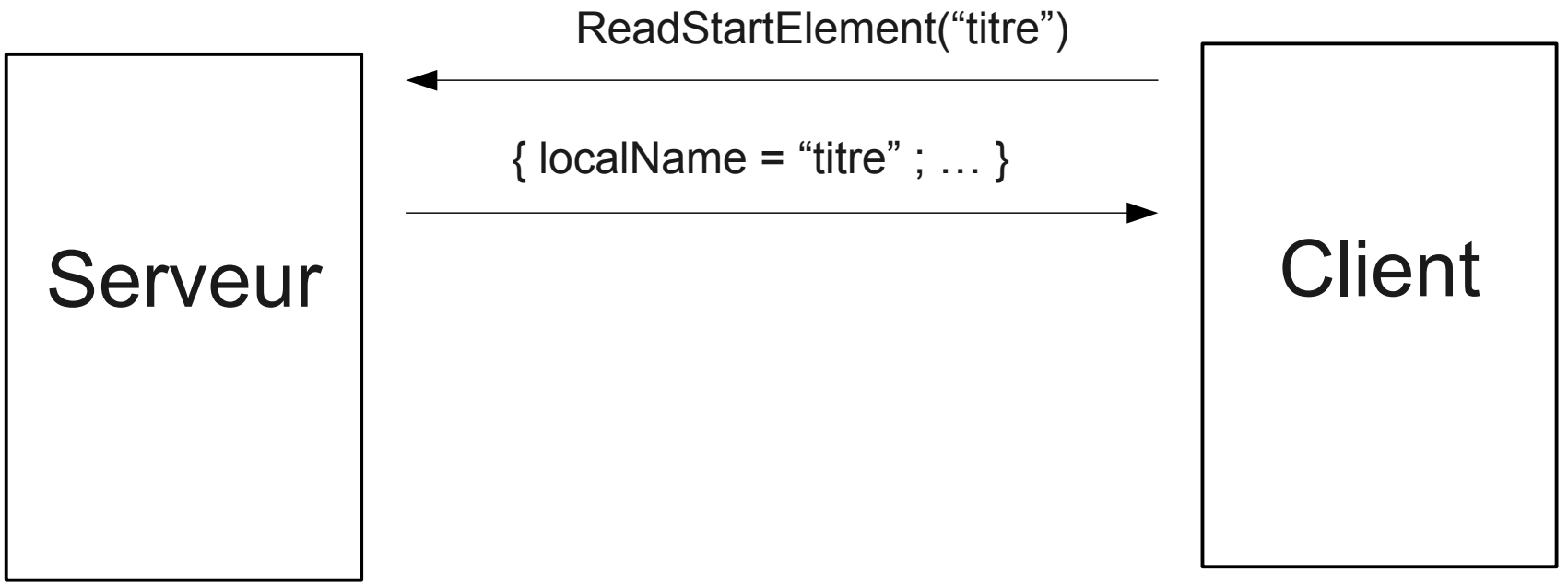
```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```



Pull

test.xml :

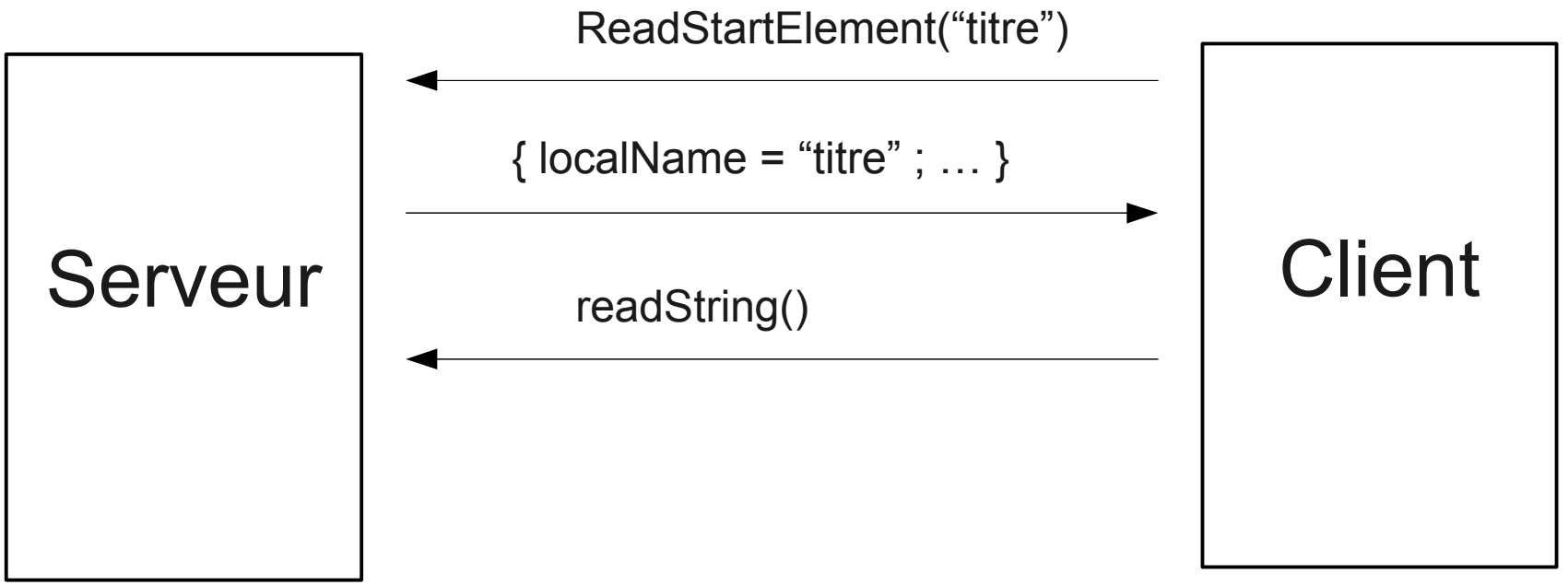
```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```



Pull

test.xml :

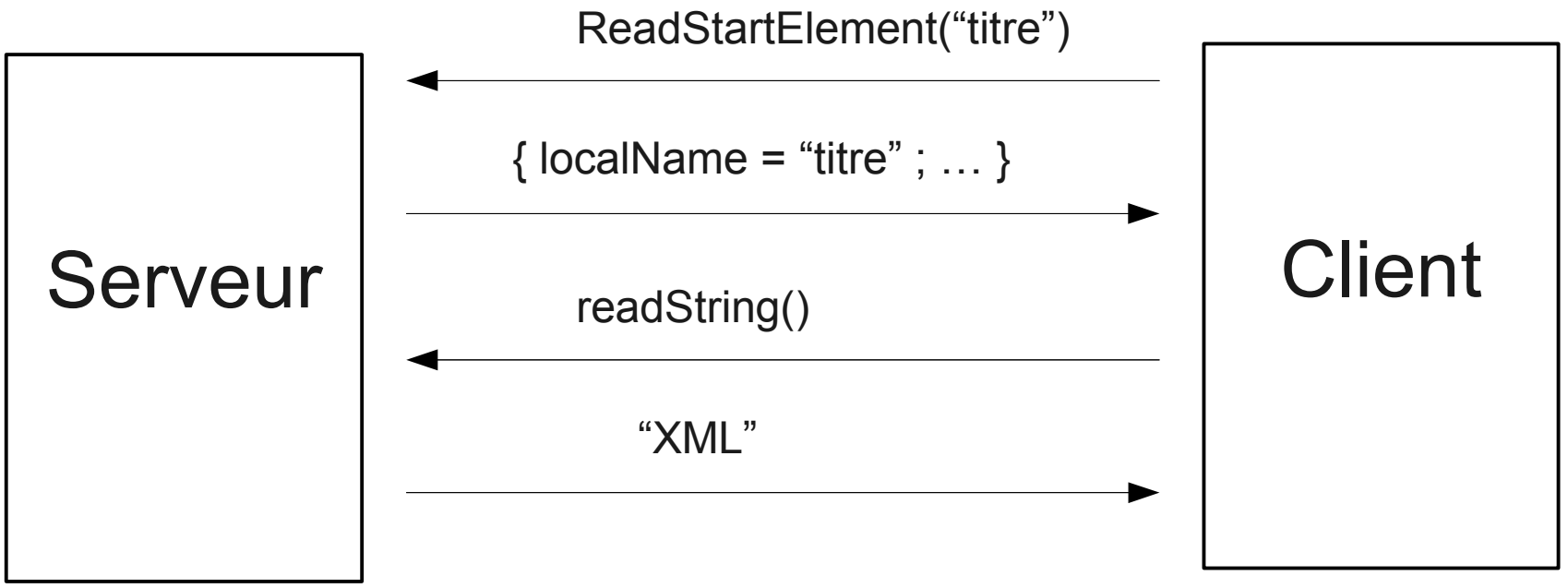
```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```



Pull

test.xml :

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```



Le flux de données est gérée par le serveur

Modèle Push (SAX)

SAX
Utilisation d'un flot d'évènements

- On peut reconstruire un arbre...
  - Mais peu utile ! On ne veut justement plus d'arbre.
- Avec un automate :
  - Un ensemble de variables
    - Appelé environnement
  - Initialisation lors de startDocument
  - Modification à chaque événement rencontré
  - Résultat lors de endDocument

Exemple : Nombre d'éléments

- Environnement:

  int number;
- Événements :
  - startDocument : number = 0;
  - endDocument : affiche number;
  - startElement : ++number;
  - Tous les autres : rien.

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Environnement :

```
int number;
```

Exemple

```
<livre lang=”fr”>
   <titre>XML</titre>
   <auteur>Chagnon</auteur>
</livre>
```

Évenement : **startDocument**

```
number = 0;
```

Environnement :

```
int number = 0;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **startElement**

```
++number;
```

Environnement :

```
int number = 1;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **ignorableWhitespace**

(rien)

Environnement :

```
int number = 1;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **startElement**

```
++number;
```

Environnement :

```
int number = 2;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **characters**

(rien)

Environnement :

```
int number = 2;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **endElement**

(rien)

Environnement :

```
int number = 2;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **ignorableWhitespace**

(rien)

Environnement :

```
int number = 2;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **startElement**

```
++number;
```

Environnement :

```
int number = 3;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **characters**

(rien)

Environnement :

```
int number = 3;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **endElement**

(rien)

Environnement :

```
int number = 3;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **ignorableWhitespace**

(rien)

Environnement :

```
int number = 3;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **endElement**

(rien)

Environnement :

```
int number = 3;
```

Exemple

```
<livre lang=”fr”>
  <titre>XML</titre>
  <auteur>Chagnon</auteur>
</livre>
```

Évenement : **endDocument**

Affiche number

Sur la sortie standard : 3

Environnement :

int number = 3;

En java ?

```
package org.xml.sax;

public interface XMLReader {
  …
  public void setContentHandler(ContentHandler handler);
  public void parse (String filename);
  …
}

Public final static class XMLReaderFactory {
  public static XMLReader createXMLReader();
  public static XMLReader createXMLReader(String classname);
}

public interface ContentHandler {
  public void startDocument();
  public void endDocument();
  public void startElement(String nsURI, String localName, String qName,
                           Attributes atts);
  public void endElement(String nsURI, String localName, String qName);
  public void characters(char[] text, int start, int length);
  public void ignorableWhitespace(char[] text, int start, int length);

  public void setDocumentLocator(Locator l);
  public void startPrefixMapping(String prefix, String uri);
  public void endPrefixMapping(String prefix);
  public void processingInstruction(String target, String data);
  public void skippedEntity(String name);
}
```

En java ?

- XMLReaderFactory : pour créer un automate vide
- ContentHandler : pour décrire un automate
- Méthode setContentHandler de XMLReader : associe une description de l'automate à un automate
- Méthode parse de XMLReader : lance l'automate sur un document XML.

Exemple : Nombre d'éléments

```
import org.xml.sax.*;

public class CompteElement implements ContentHandler {
  int number;

  public void startDocument() {
     number = 0;
  }
  public void endDocument() {
     System.out.println(number)
  }
  public void startElement(String n, String l, String q, Attributes a) {
     ++number;
  }
  public void endElement(String n, String l, String q) {}
  public void characters(char[] text, int start, int length) {}
  public void ignorableWhitespace(char[] text, int start, int length) {}

  public void setDocumentLocator(Locator l) {}
  public void startPrefixMapping(String prefix, String uri) {}
  public void endPrefixMapping(String prefix) {}
  public void processingInstruction(String target, String data) {}
  public void skippedEntity(String name) {}
}
```

Exemple : Nombre d'éléments

```
… des imports …

public class Main {
  public static void main(String[] argv) {

    XMLReader parser;
    String filename = “test.xml”;

    parser = XMLReaderFactory.createXMLReader();

    parser.setContentHandler(new CompteElement());

    parser.parse(filename);

  }
}
```

Exercice 1 : Que cela fait-il ?

```
import org.xml.sax.*;
public class Exo1 implements ContentHandler {
  int number1;
  int number2;

  public void startDocument() {
      number1 = 0;
      number2 = 0;
  }
  public void endDocument() {
      System.out.println(number2);
  }
  public void startElement(String n, String l, String q, Attributes a) {
      ++number1;
      if ( number1 > number2 ) {
          number2 = number1;
      }
  }
  public void endElement(String n, String l, String q) {
      --number1;
  }
  public void characters(char[] text, int start, int length) {}
  public void ignorableWhitespace(char[] text, int start, int length) {}
  public void setDocumentLocator(Locator l) {}
  public void startPrefixMapping(String prefix, String uri) {}
  public void endPrefixMapping(String prefix, String uri) {}
  public void processingInstruction(String target, String data) {}
  public void skippedEntity(String name) {}
}
```

Exercice 2 : Que cela fait-il ?

```
import org.xml.sax.*;
public class Exo2 implements ContentHandler {
  boolean t;

  public void startDocument() {
      t = false;
  }
  public void endDocument() {}
  public void startElement(String n, String l, String q, Attributes a) {
      if (l.equals("titre"))
          t = true;
  }
  public void endElement(String n, String l, String q) {
      t = false;
  }
  public void characters(char[] text, int start, int length) {
      if (t) {
          String s = new String(text);
          System.out.println(s.substring(start,start+length));
      }
  }
  public void ignorableWhitespace(char[] text, int start, int length) {}
  public void setDocumentLocator(Locator l) {}
  public void startPrefixMapping(String prefix, String uri) {}
  public void endPrefixMapping(String prefix, String uri) {}
  public void processingInstruction(String target, String data) {}
  public void skippedEntity(String name) {}
}
```

Exercice 3 : Que cela fait-il ?

```
public class Exo3 implements ContentHandler {
  Document doc; boolean t; Element root;

  public void startDocument() {
      DocumentBuilderFactory dbfac = DocumentBuilderFactory.newInstance();
      DocumentBuilder docBuilder = dbfac.newDocumentBuilder();
      doc = docBuilder.newDocument();
      root = doc.createElement(“biblio”);
  }
  public void endDocument() {
      doc.appendChild(root);
      FONCTION_D_AFFICHAGE(doc);
  }
  public void startElement(String n, String l, String q, Attributes a) {
      if (l.equals("titre"))
          t = true;
  }
  public void endElement(String n, String l, String q) {
      t = false;
  }
  public void characters(char[] text, int start, int length) {
      if (t) {
          String s = new String(text);
          Element e = doc.createElement(“livre”);
          e.setAttribute(“titre“, s.substring(start,start+length));
          root.appendChild(e);
      }
  }
  public void ignorableWhitespace(char[] text, int start, int length) {}
  public void setDocumentLocator(Locator l) {}
  public void startPrefixMapping(String prefix, String uri) {}
  public void endPrefixMapping(String prefix, String uri) {}
  public void processingInstruction(String target, String data) {}
  public void skippedEntity(String name) {}
}
```