Master 2 LITL - Thématiques actuelles de la recherche en TAL (U.E. SLT0906V)
Programmation événementielle et parsing SAX

Franck Sajous/CLLE-ERSS

http://fsajous.free.fr/

Exploiter des données XML dans un programme

Exemples

- Dans Morphalou, lister toutes les formes fléchies et les lemmes correspondant des entrées homonymiques ?
- Dans un corpus oral de dialogues, afficher les tours de parole imbriqués.

Exploiter des données XML dans un programme

Exemples

- Dans Morphalou, lister toutes les formes fléchies et les lemmes correspondant des entrées homonymiques ?
- Dans un corpus oral de dialogues, afficher les tours de parole imbriqués.

Comment ?

Une première idée

- Ouvrir le fichier, le lire ligne à ligne
- Détecter les balises XML avec des expressions régulières

Exploiter des données XML dans un programme

Exemples

- Dans Morphalou, lister toutes les formes fléchies et les lemmes correspondant des entrées homonymiques ?
- Dans un corpus oral de dialogues, afficher les tours de parole imbriqués.

Comment ?

Une première idée

- Ouvrir le fichier, le lire ligne à ligne
- Détecter les balises XML avec des expressions régulières

Essayez (avec un *vrai* fichier tout-venant) . . . Bon courage !

Parseurs prêts à l'emploi

Utiliser l'API : plusieurs familles de parseurs

- Les parseurs de type DOM (Document Object Model)
- Les parseurs *« à la SAX »*

DOM

- DOM = représentation standardisée des documents XML/XHTML
- document = arbre où chaque balise et chaque élément de texte sont considérés comme des nœuds
- pour un nœud (élément) donné, possibilité d'accèder aux attributs de l'élément, à d'autres nœuds en exprimant des relations de parenté (e.g. filiation dans l'arborescence), etc.

Exemple : DOM en Perl

Accèder à un/des noeuds

```
my $parser = new XML::DOM::Parser;
my $doc = $parseur->parsefile(‘‘monfichier.xml’’);
my @listeEntrees = @{$doc->getElementByTagName(‘‘lexicalEntry’’)};
foreach my $noeud (@listeEntrees)
{
  ...
}
```

Depuis un noeud

Accès aux autres nœuds :

- `$noeud->getAttribute("id")`
- `$noeud->getParentNode()`
- `$noeud->getFirstChild()`
- `$noeud->getChildNodes()`

Modification :

- `$noeud->appendChild($autreNoeud)`
- `$noeud->removeChild($fils)`
- `$noeud->setAttribute ("id", $nouvelId)`

DOM, suite

+ / -

- Pratique, mais...
- Nécessite de stocker tout l'arbre en mémoire
- Impossible pour les documents volumineux (e.g. GLAWI)

DOM, un standard

"The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents"
`https://www.w3.org/DOM/`

- API similaire en Java, Python, Perl, Javascript, etc
- E.g. Firefox, Chrome, InternetExplorer, etc. → implémentations différentes de javascript, mais même interface DOM pour l'accès aux éléments du document

SAX (Simple API for XML)

Principes

- Traitement par flux
- Déclenchement d'« événements » à chaque :
  - ouverture de balise
  - fermeture de balise
  - élément de texte
- Le programmeur (utilisateur du module SAX) instancie un parseur (fourni par l'API), écrit des méthodes qui réagissent à ces événements lorsque le parseur les déclenche
- Ces méthodes sont spécifiées par une interface ou une classe abstraite
- Elles sont invoquées par le parseur

$\rightarrow$ on parle de « *handler* » (gestionnaire) d'événements.

SAX (Simple API for XML)

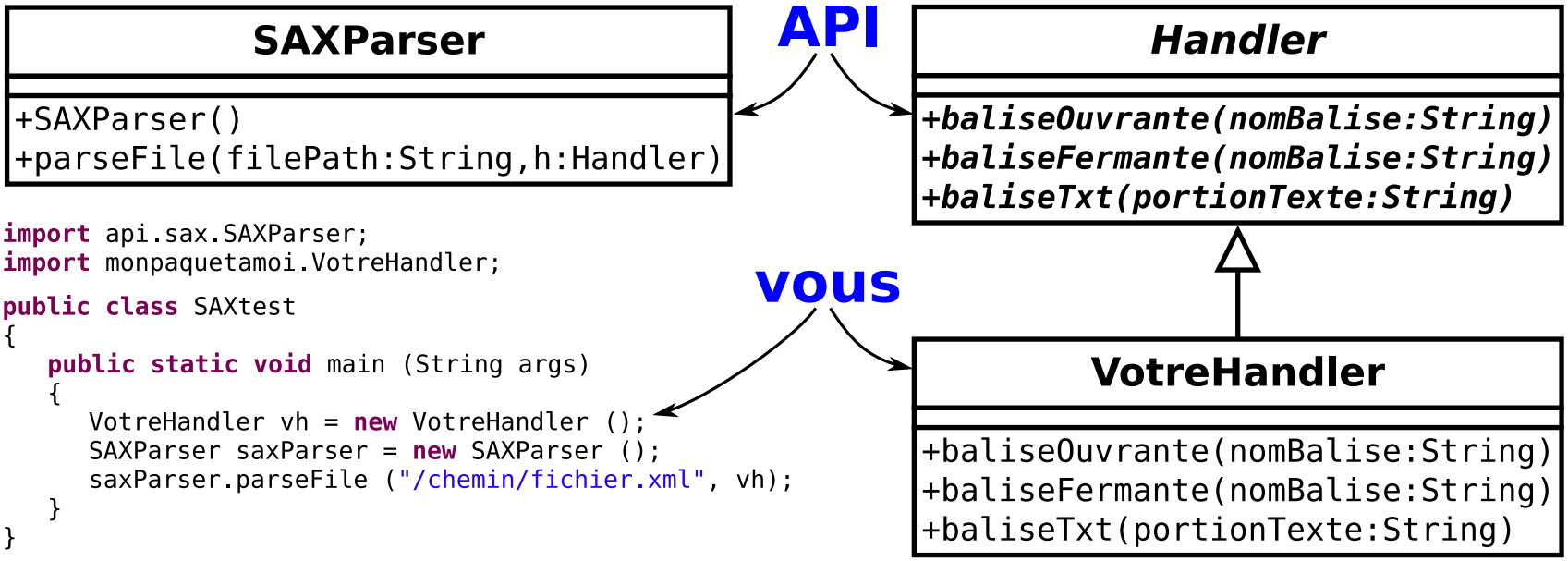
Principes

- Traitement par flux
- Déclenchement d'« événements » à chaque :
  - ouverture de balise
  - fermeture de balise
  - élément de texte
- Le programmeur (utilisateur du module SAX) instancie un parseur (fourni par l'API), écrit des méthodes qui réagissent à ces événements lorsque le parseur les déclenche
- Ces méthodes sont spécifiées par une interface ou une classe abstraite
- Elles sont invoquées par le parseur

$\rightarrow$ on parle de « *handler* » (gestionnaire) d'événements.

« *handler* » $\approx$ « *listener* » (écouteur) dans la terminologie des interface graphiques

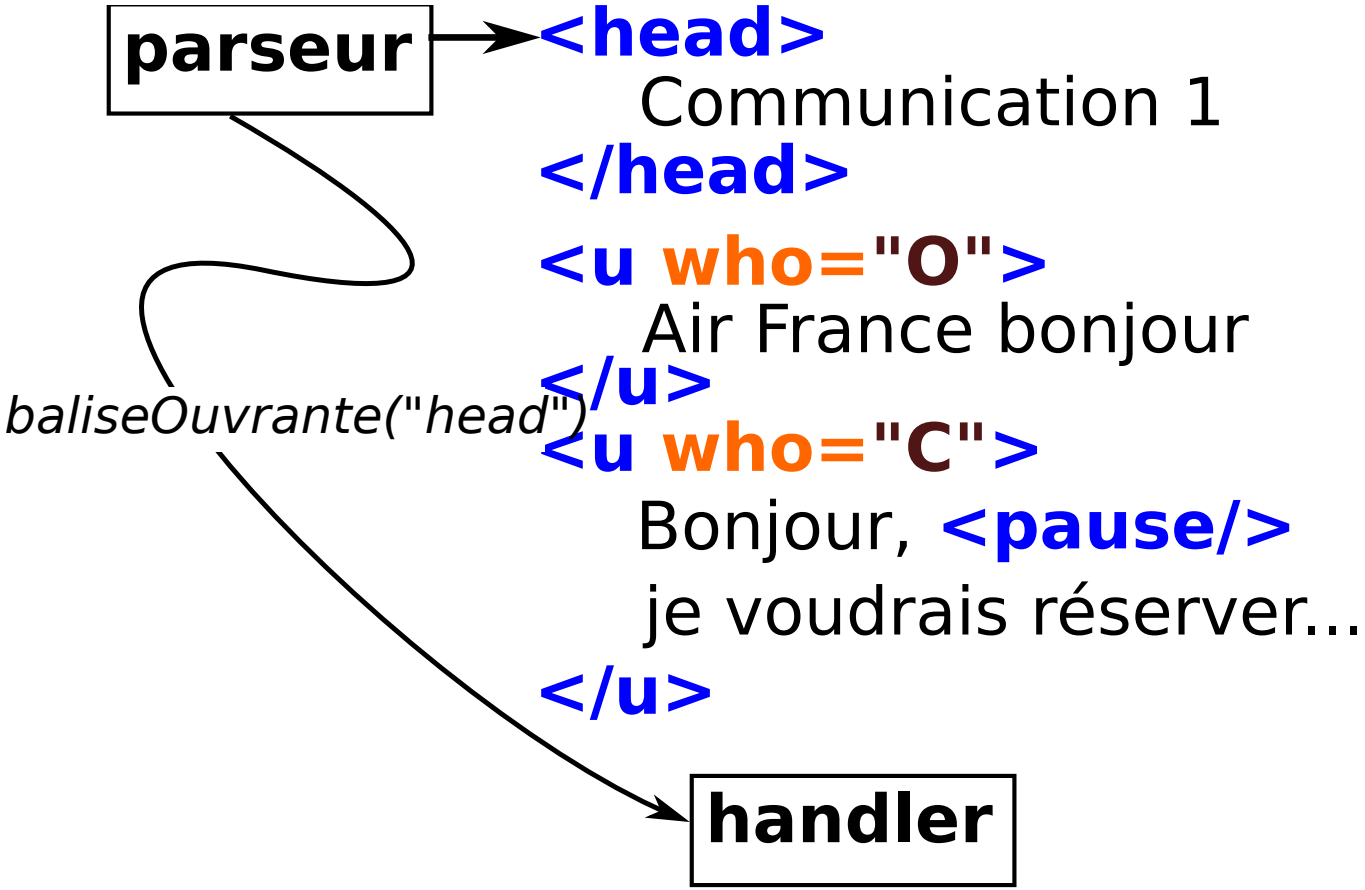
SAX : exemple d'implémentation



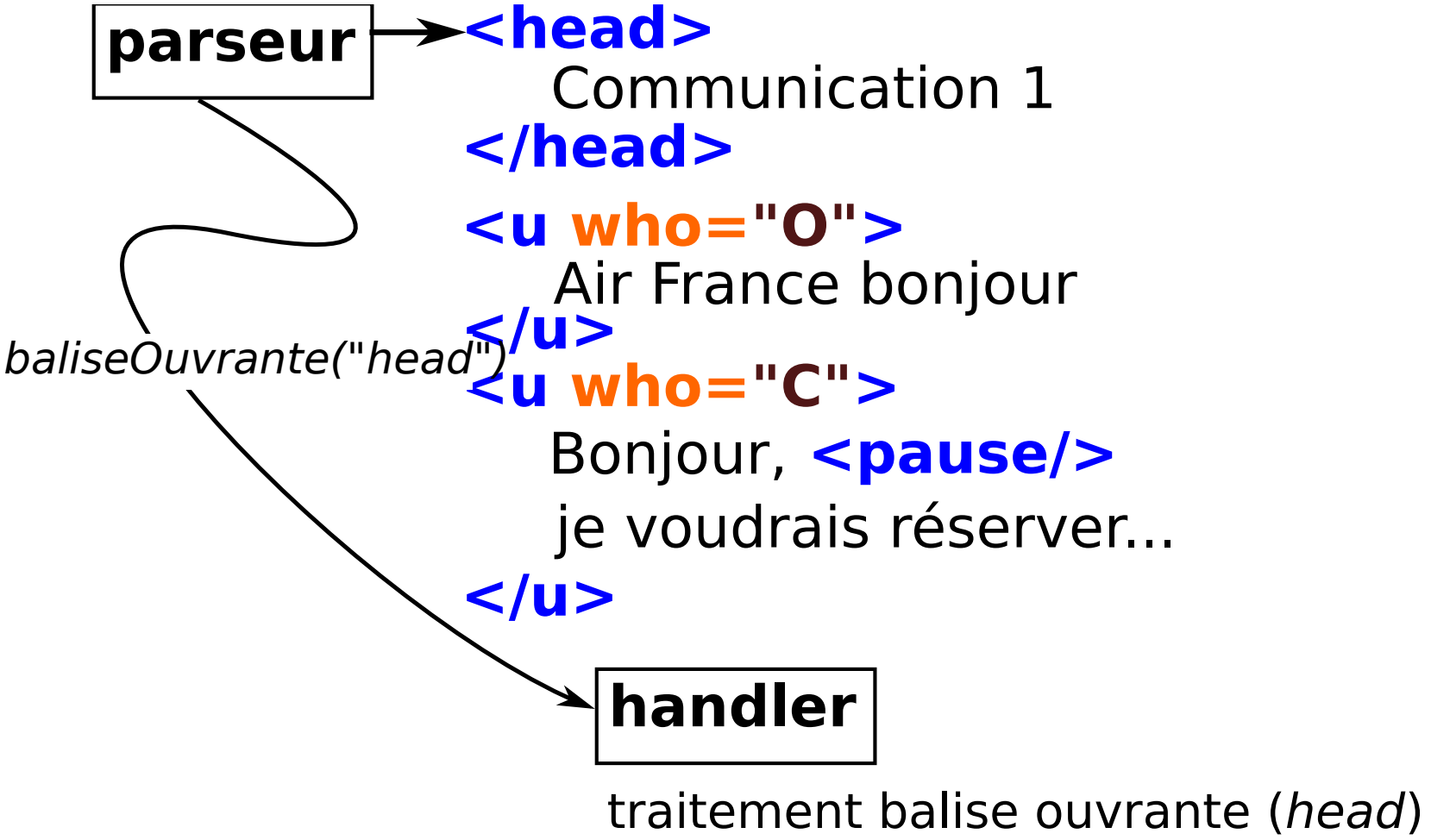

```
import api.sax.SAXParser;
import monpaquetamoi.VotreHandler;

public class SAXtest
{
   public static void main (String args)
   {
      VotreHandler vh = new VotreHandler ();
      SAXParser saxParser = new SAXParser ();
      saxParser.parseFile ("/chemin/fichier.xml", vh);
   }
}
```

SAX : fonctionnement

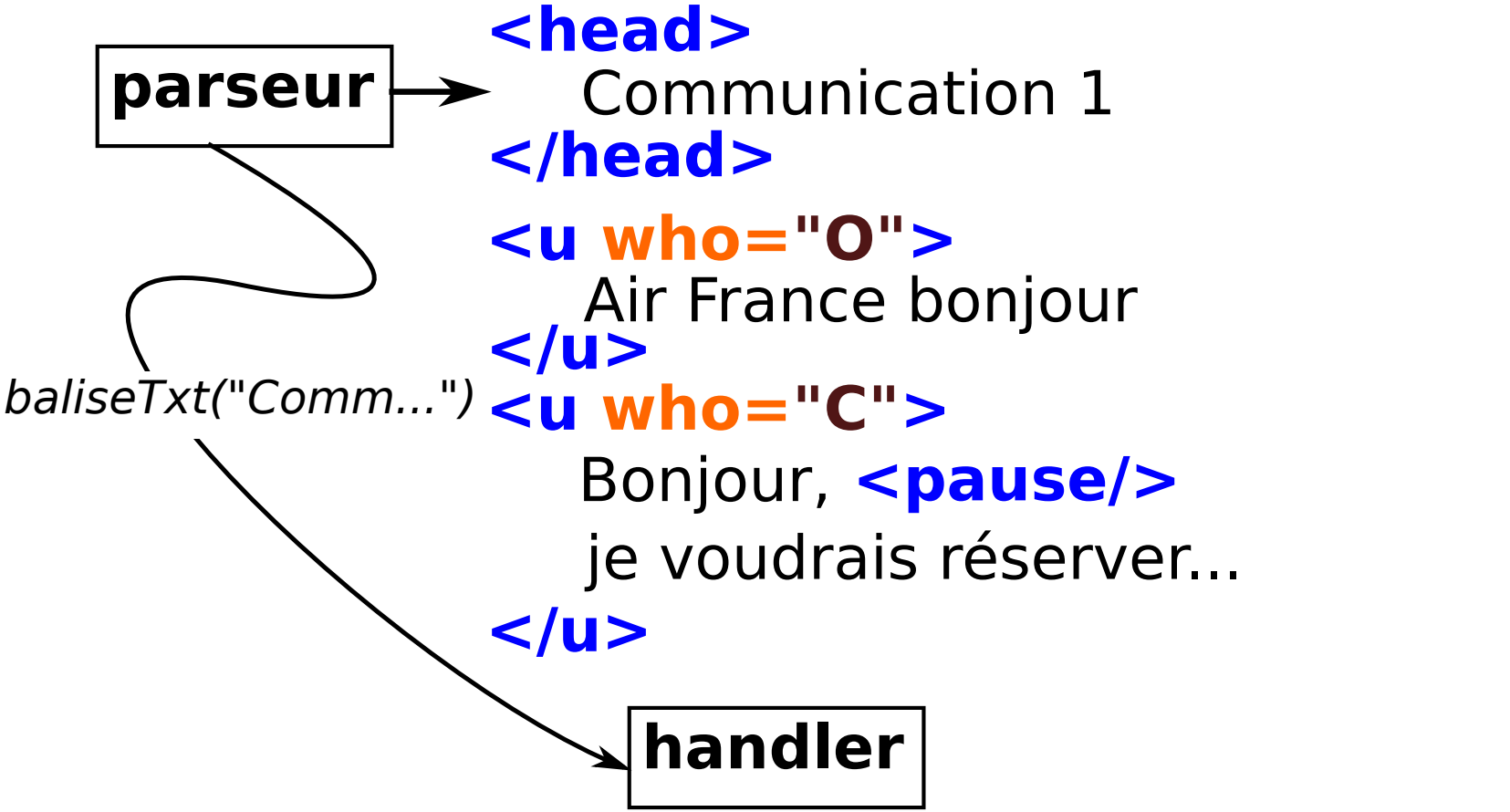
SAX : fonctionnement



SAX : fonctionnement

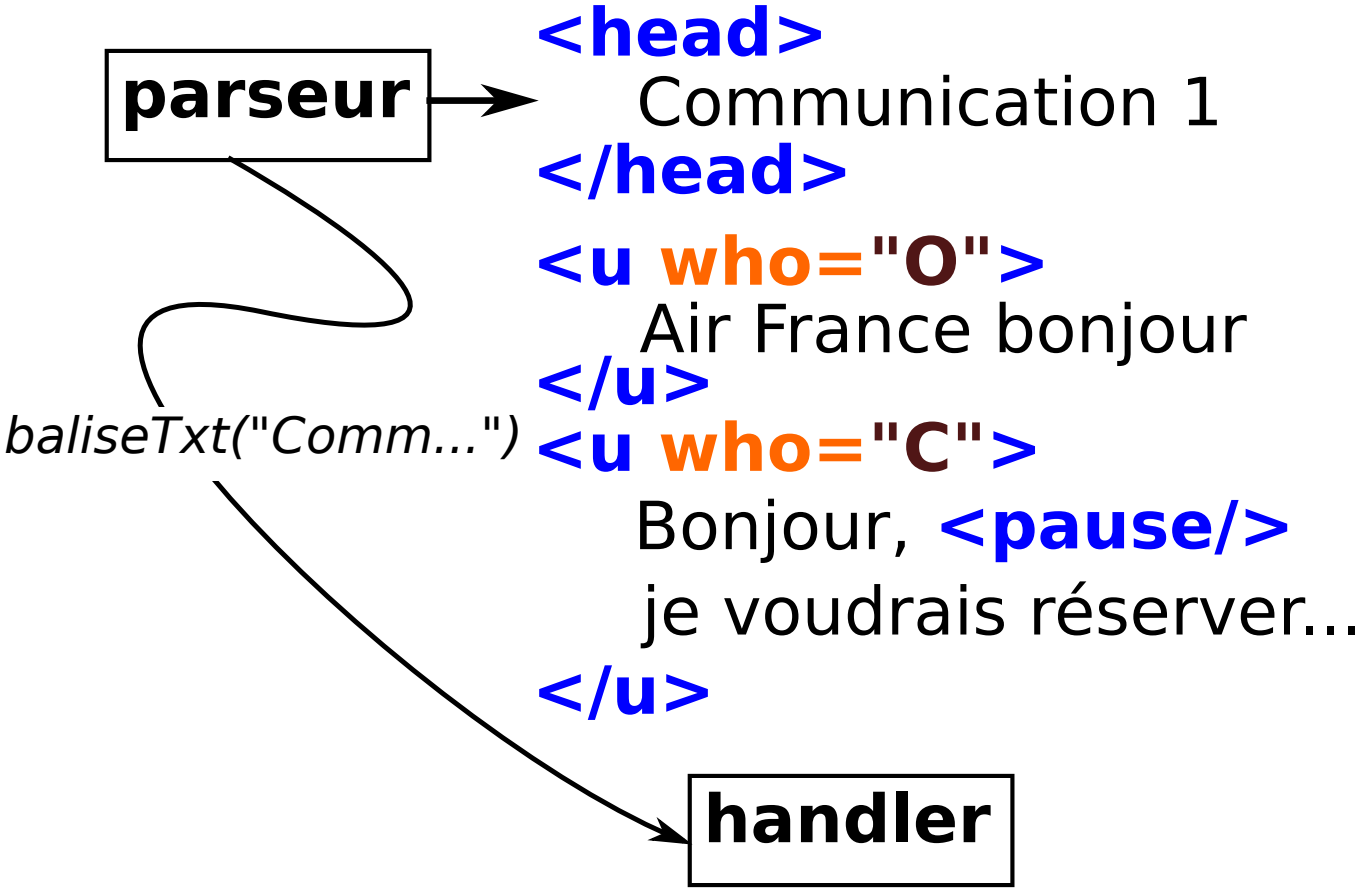


SAX : fonctionnement



traitement balise ouvrante (*head*)

SAX : fonctionnement



traitement balise ouvrante (*head*)
traitement texte (*Communication1*)

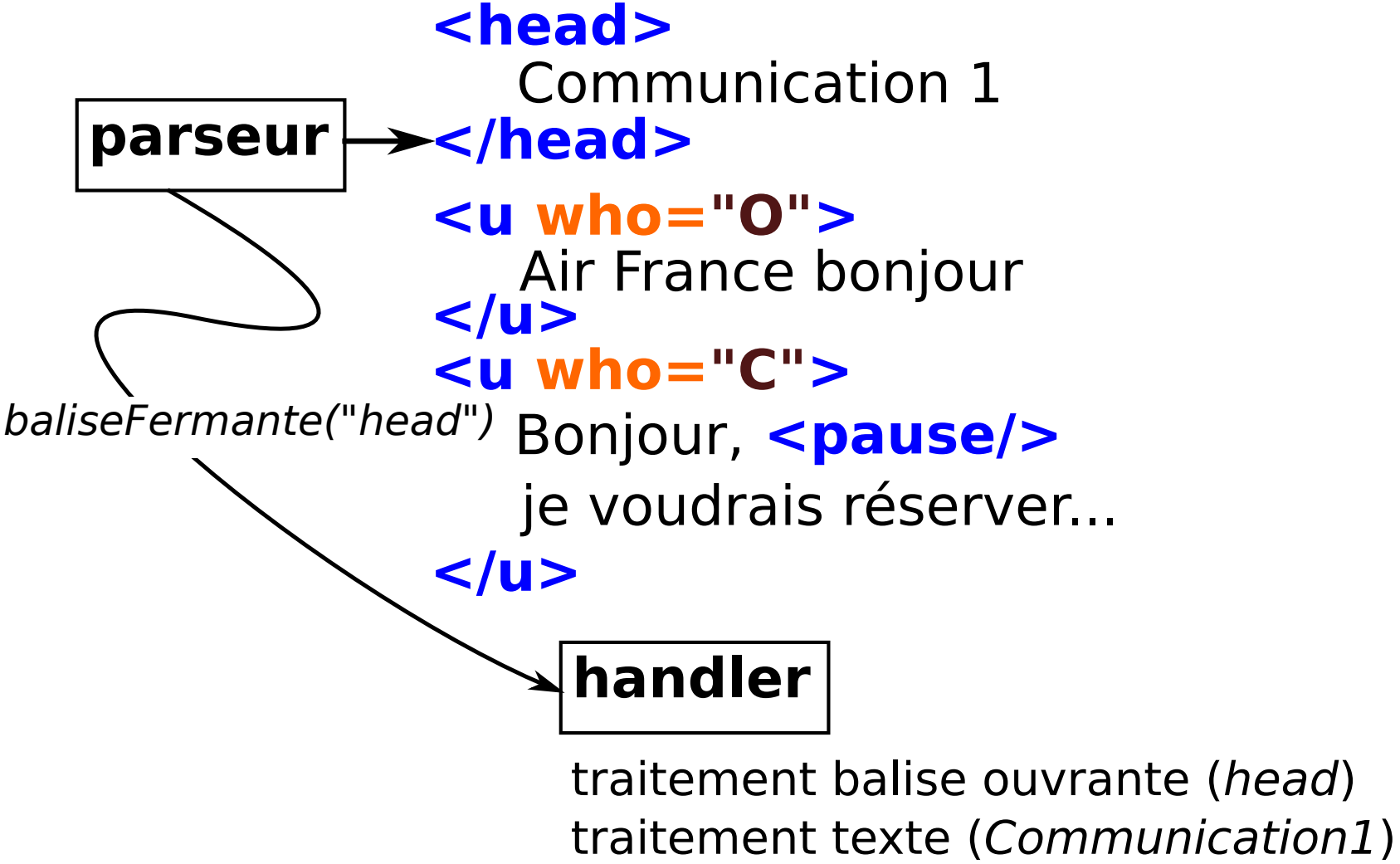
SAX : fonctionnement


```
<head>
    Communication 1
</head>
<u who="O">
    Air France bonjour
</u>
<u who="C">
    Bonjour, <pause/>
    je voudrais réserver...
</u>
```

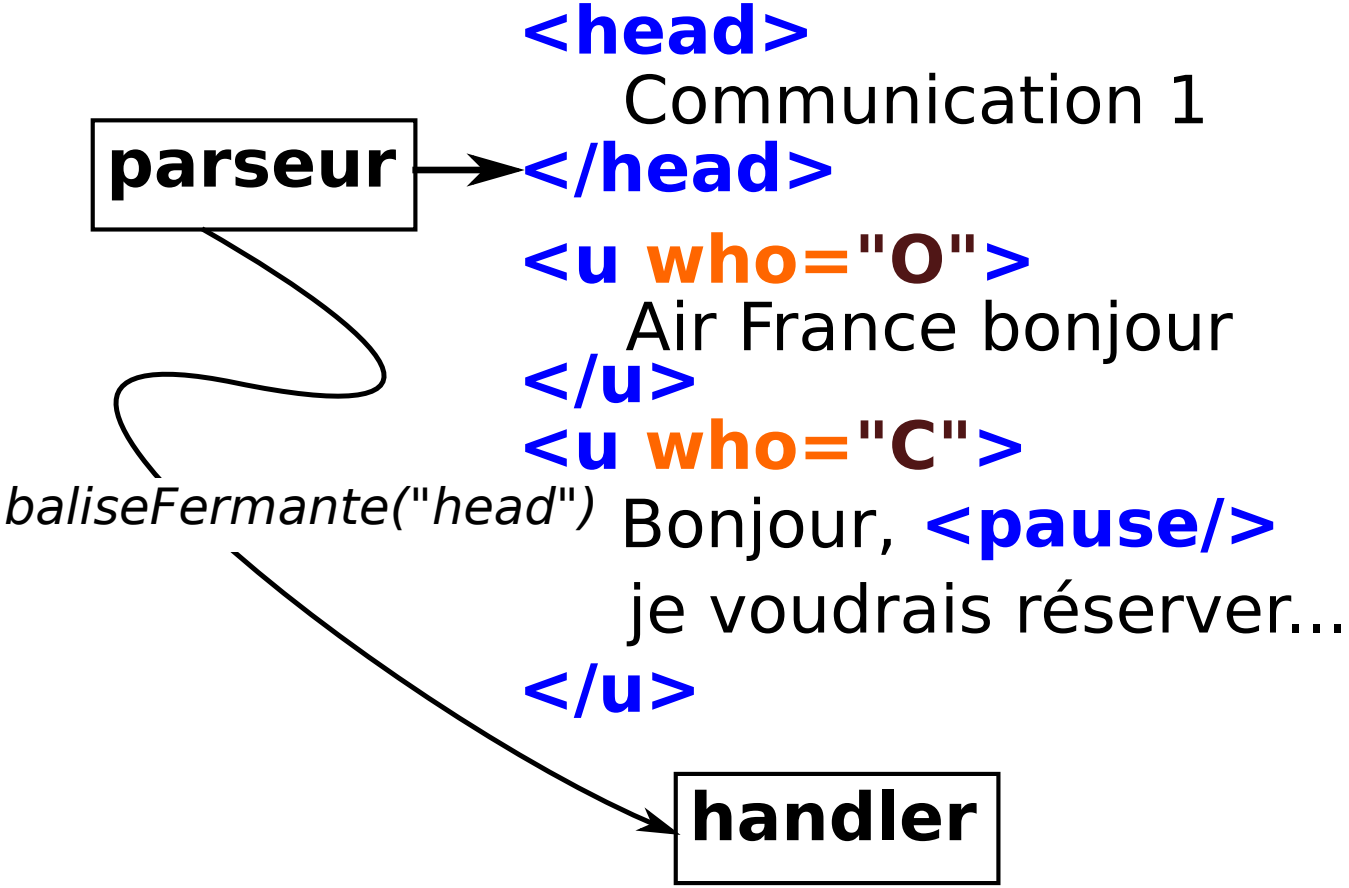
handler

traitement balise ouvrante (*head*)
traitement texte (*Communication1*)

SAX : fonctionnement



SAX : fonctionnement



traitement balise ouvrante (*head*)
traitement texte (*Communication1*)
traitement balise fermante (*head*)

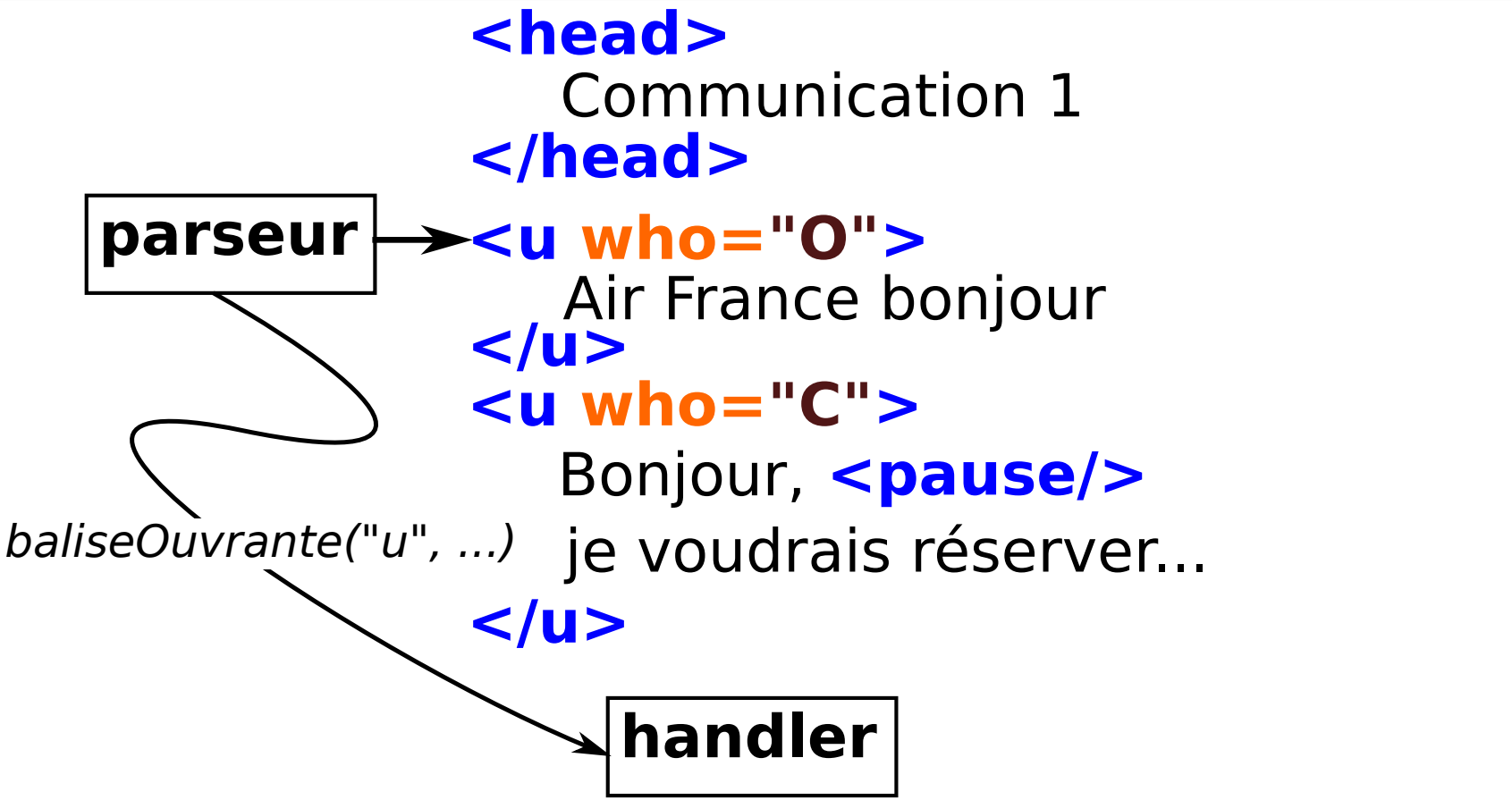
SAX : fonctionnement


```
<head>
    Communication 1
</head>
<u who="O">
    Air France bonjour
</u>
<u who="C">
    Bonjour, <pause/>
    je voudrais réserver...
</u>
```

handler

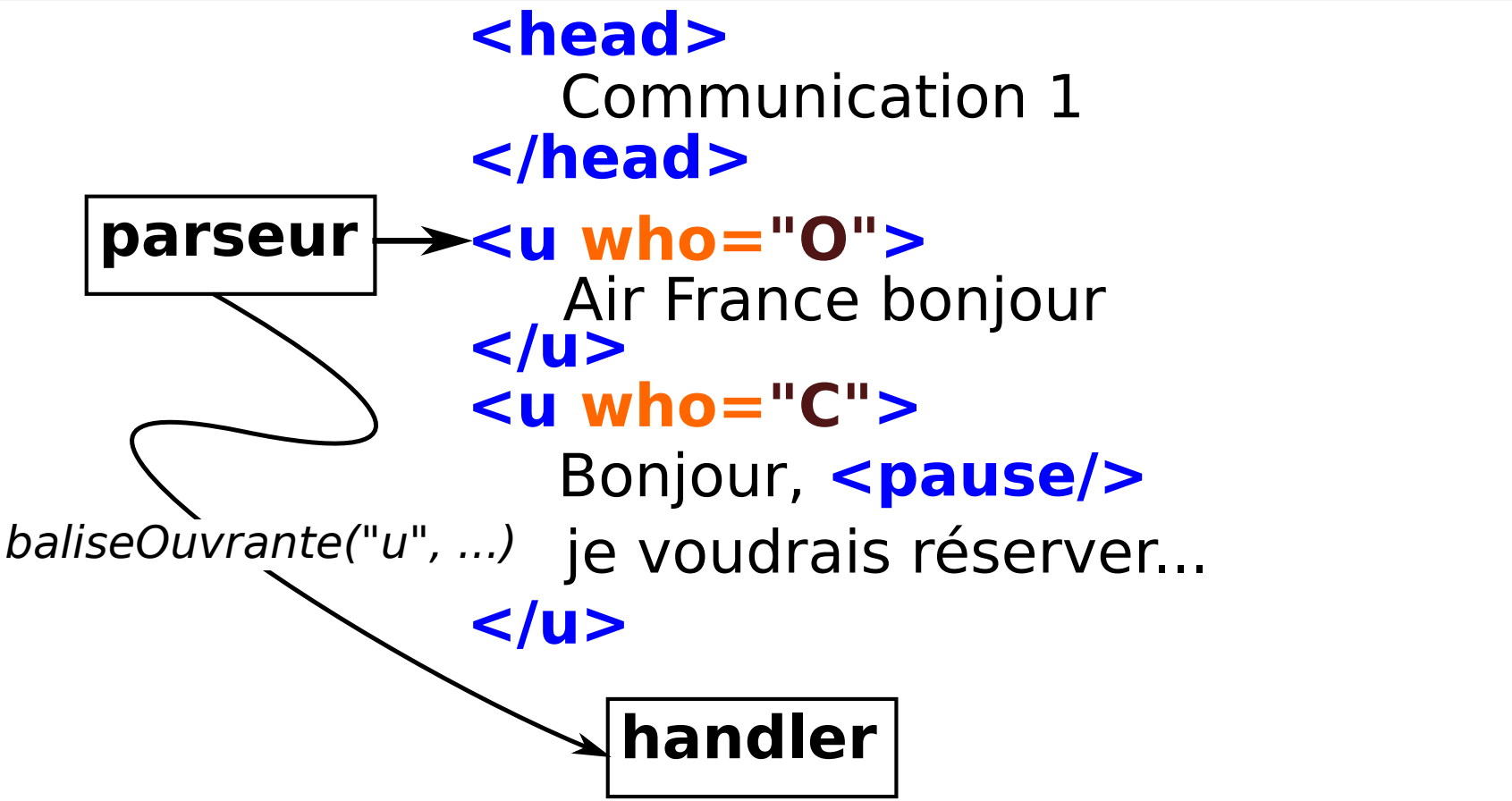
traitement balise ouvrante (*head*)
traitement texte (*Communication1*)
traitement balise fermante (*head*)

SAX : fonctionnement



traitement balise ouvrante (*head*)
traitement texte (*Communication1*)
traitement balise fermante (*head*)

SAX : fonctionnement



traitement balise ouvrante (*head*)
traitement texte (*Communication1*)
traitement balise fermante (*head*)
traitement balise ouvrante (*u*)