# XML/HTML Parsing

Devdatta Kulkarni

# What is XML? What is HTML?

- XML:
  - A “generic” markup language
  - Data is defined within <u>matching tags/nodes</u>

```
<cs378>
    <assignments>
        <assignment1>Servlets</assignment1>
    </assignments
</cs378>
```

  - XML document needs to be “well-formed” (every opening tag should have a closing tag)
- HTML:
  - A markup language for representing data/pages for browser display
  - Pre-defined set of tags (such as <html>, <title>, <head>, <a>, etc.)
  - HTML document *need not be well-formed*

## Problem

- Given an XML/HTML document, output specific nodes from it that match a *Query* criteria

Example:

***Input:***

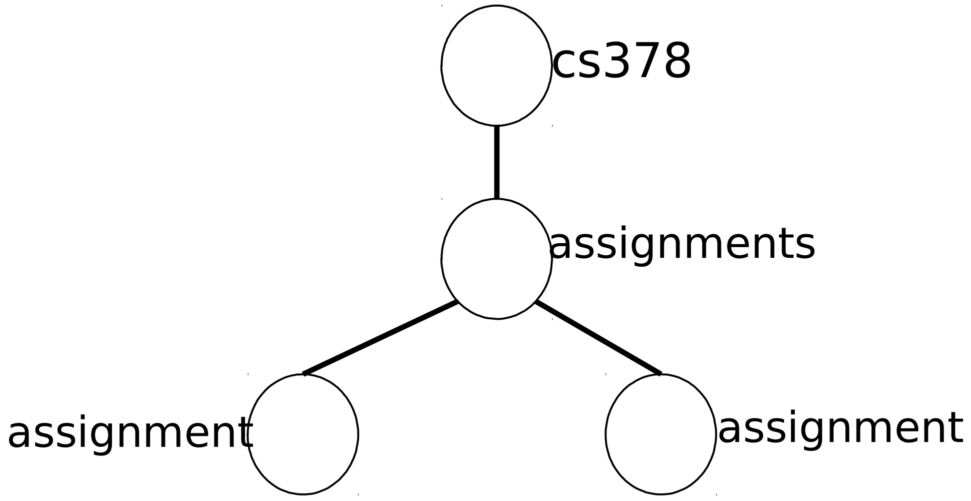
```
<cs378><assignments>
      <assignment>Caching Proxy</assignment>
      <assignment>Servlets</assignment>
</assignment></cs378>
```

***Query Criteria:*** ‘assignment’ node

***Output:***

```
<assignment>Caching Proxy</assignment>
<assignment>Servlets</assignment>
```

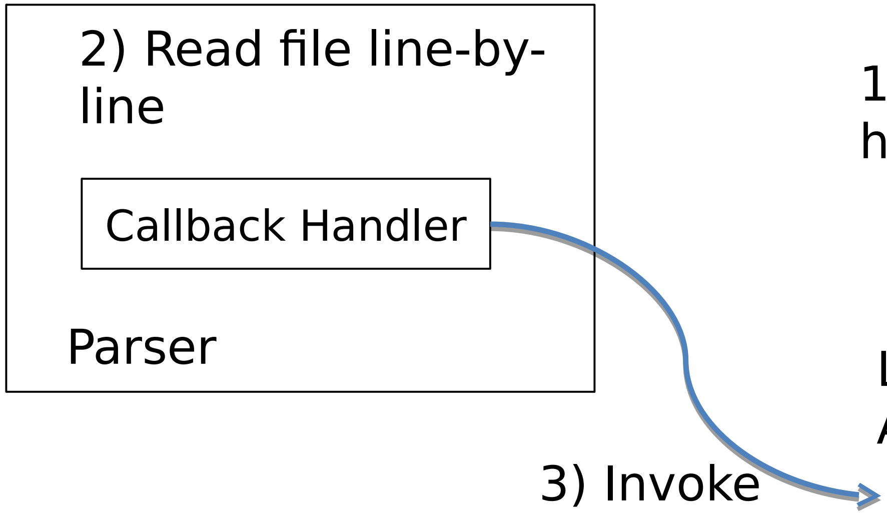
# XML Parsing: Option 1: Tree parsing



```
List assignmentNodes = new
ArrayList();
Node head = cs378;
Queue.enqueue(head);
while(!Queue.isEmpty()) {
  Node n = Queue.dequeue();
  if (n.getNodeName().equals
    ("assignment")) {
     assignmentNodes.add(n);
  } // end of if
 Queue.enqueue(n.getChildren());
}// end of while
return assignmentNodes;
```

This approach is called *DOM* parsing
DOM stands for Document Object Model (DOM)

# XML Parsing: Option 2: Parsing using callbacks



1) Register callback handler with the Parser

3) Invoke Callback Handler

```
List assignmentNodes = new
ArrayList();

public nodeHandler(Node n) {
     if (n.getNodeName().equals
          (“assignment”)) {
          assignmentNodes.add(n);
  }
}
```

This approach is called *SAX* parsing
SAX stands for Simple API for XML

# DOM vs SAX Comparison

- DOM
  - Advantage:
    - Provides fine-grained control over parsing
  - Disadvantage:
    - Entire tree is built in memory before parsing can begin
      - Memory intensive
- SAX
  - Advantage:
    - Does not build entire tree; so memory is not an issue
  - Disadvantage:
    - State between callback invocations needs to be maintained by the program

# Parsing XML: Current way

- XPath
  - Declarative model for querying XML documents
    - ``Queries’’ are specified using ``path expressions’’
      - Example Query: /cs378/assignments/assignment
      - Read:
        - » http://docs.oracle.com/javase/7/docs/api/javax/xml/xpath/package-summary.html

- Example: XPathParser.java

# Examples

- Examples
  - DOMParser
  - SAXParser

- What about parsing HTML documents?
  - Can we use XML parsing techniques?
    - Use DOMParser and SAXParser with cs378.html
    - Use DOMParser and SAXParser with cs378.not_well_formed.html

## Parsing HTML

- Parsing using Java regular expressions
  - Example: RegexParser
- Parsing using a library such as jsoup
  - http://jsoup.org/
  - Example: JSoupParser

- Parsing HTML disadvantages:
  - Parsing presentation logic instead of working with the domain objects
  - Very brittle; will break if the HTML page is changed
  - No formal contract defined; so cannot validate the HTML document

# Reading

- XML Parsing
  - http://docs.oracle.com/javase/7/docs/api/javax/xml/xpath/package-summary.html
  - http://docs.oracle.com/javase/tutorial/essential/regex/test_harness.html
  - http://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html
  - http://jsoup.org/