Examen Web/XML

Durée : 2 heures, Documents autorisés

Programmation Java avec le DOM (3 points)

Vous êtes le meilleur programmeur d'une entreprise qui vend des outils XML en Java.
Le projet actuel consiste à réaliser un moteur XPath qui utilise le DOM comme base de navigation.
Ecrivez un programme qui puisse parcourir l'axe `preceding-sibling`
Vous ne vous préoccuperez pas de l'analyse de l'expression XPath, mais seulement de son exécution.
Vous pourrez faire une classe abstraite qui serve de chapeau à tous les axes, et une autre qui se préoccupe de l'axe `preceding-sibling`
Faites un jeu de données XML, d'expression XPath à tester, et indiquez ce que vous obtenez avec votre programme.
Voir en annexes un extrait de l'API DOM.

Programmation XSLT (3 points)

Vous êtes le meilleur programmeur d'une entreprise qui vend des prestations XML.
Le client du moment vous demande de faire quelques feuilles de style XSLT.

1 – Mettre des données dans un tableau :

Fichier de test en entrée :

```
<?xml version="1.0"?>
<root>
  <data>abc</data>
  <data>def</data>
  <data>ghi</data>
  <data>jkl</data>
  <data>mno</data>
  <data>pqr</data>
  <data>stu</data>
  <data>vwx</data>
  <data>yz</data>
</root>
```

Sortie souhaitée en HTML :

abc	def	ghi
jkl	mno	pqr
stu	vwx	yz

2 – Changer d'espace de nommage :

Vous disposez de `copy.xslt` qui permet de faire une copie :

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="node(  ) | @*">
  <xsl:copy>
    <xsl:apply-templates select="@* | node(  )"/>
  </xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

Fichier en entrée :

```
<foo:someElement xmlns:foo="http://www.foo.com/namespaces/foo"
    xmlns:doc="http://www.doc.com/namespaces/doc">
    <foo:aChild>
        <foo:aGrandChild/>
        <foo:aGrandChild>
            <doc:doc>
                This documentation should not be removed
                or altered in any way.
            </doc:doc>
        </foo:aGrandChild>
    </foo:aChild>
</foo:someElement>
```

Sortie souhaitée en XML (les blancs peuvent être ignorés, seule la structure importe) :

```
<?xml version="1.0"?>
<bar:someElement
    xmlns:bar="http://www.bar.com/namespaces/bar">
    <bar:aChild>
```

Examen Web/XML

Durée : 2 heures, Documents autorisés

```
        <bar:aGrandChild>
        </bar:aGrandChild>
        <bar:aGrandChild>
            <doc:doc xmlns:doc="http://www.doc.com/namespaces/doc">
                This documentation should not be removed
                or altered in any way.
            </doc:doc>
        </bar:aGrandChild>
    </bar:aChild>
</bar:someElement>
```

XPath (0,5 point)

Vous êtes toujours le meilleur programmeur d'une entreprise qui vend des prestations XML.
Le client du moment vous demande de lui concocter une expression XPath qui sélectionne les catégories qui n'apparaissent qu'une fois :

```
<?xml version="1.0"?>
<dataset>
  <node>
    <category>C1</category>
  </node>
  <node>
    <category>C2</category>
  </node>
  <node>
    <category>C1</category>
  </node>
  <node>
    <category>C3</category>
  </node>
  <node>
    <category>C1</category>
  </node>
  <node>
    <category>C2</category>
  </node>
  <node>
    <category>C1</category>
  </node>
  <node>
    <category>C3</category>
  </node>
</dataset>
```

Problème (3,5 points)

Vous êtes le meilleur concepteur d'une entreprise qui vend des services et autres prestations du moment que le client paie.
Votre projet consiste à gérer le parc de machines de votre client qui exige que ses données soient en XML parce qu'il en a entendu parler mais ne sait pas trop ce que c'est.
1 – Identifiez ce que pourraient être les données à gérer, et faites une instance XML.
2 – Créez une DTD de sorte que votre instance soit valide vis à vis de la DTD.
3 – Faites une feuille de style XSLT pour visionner vos données en HTML.
4 – Dessinez le rendu HTML comme l'aurait fait un navigateur.

Examen Web/XML

Durée : 2 heures, Documents autorisés

Annexes

org.w3c.dom

Interface Node

All Known Subinterfaces:

Attr, CDATASection, CharacterData, Comment, Document, DocumentFragment, DocumentType, Element, Entity, EntityReference, Notation, ProcessingInstruction, Text

All Known Implementing Classes:

IIOMetadataNode

---

public interface **Node**

The `Node` interface is the primary datatype for the entire Document Object Model. It represents a single node in the document tree. While all objects implementing the `Node` interface expose methods for dealing with children, not all objects implementing the `Node` interface may have children. For example, `Text` nodes may not have children, and adding children to such nodes results in a `DOMException` being raised.
The attributes `nodeName`, `nodeValue` and `attributes` are included as a mechanism to get at node information without casting down to the specific derived interface. In cases where there is no obvious mapping of these attributes for a specific `nodeType` (e.g., `nodeValue` for an `Element` or `attributes` for a `Comment`), this returns `null`. Note that the specialized interfaces may contain additional and more convenient mechanisms to get and set the relevant information.
The values of `nodeName`, `nodeValue`, and `attributes` vary according to the node type as follows:

Interface	nodeName	nodeValue	attributes
Attr	name of attribute	value of attribute	null
CDATASection	`"#cdata-section"`	content of the CDATA Section	null
Comment	`"#comment"`	content of the comment	null
Document	`"#document"`	null	null
DocumentFragment	`"#document-fragment"`	null	null
DocumentType	document type name	null	null
Element	tag name	null	NamedNodeMap
Entity	entity name	null	null
EntityReference	name of entity referenced	null	null
Notation	notation name	null	null
ProcessingInstruction	target	entire content excluding the target	null
Text	`"#text"`	content of the text node	null

See also the Document Object Model (DOM) Level 2 Core Specification.

Examen Web/XML

Durée : 2 heures, Documents autorisés

Field Summary

static short	**ATTRIBUTE_NODE** The node is an Attr.
static short	**CDATA_SECTION_NODE** The node is a CDATASection.
static short	**COMMENT_NODE** The node is a Comment.
static short	**DOCUMENT_FRAGMENT_NODE** The node is a DocumentFragment.
static short	**DOCUMENT_NODE** The node is a Document.
static short	**DOCUMENT_TYPE_NODE** The node is a DocumentType.
static short	**ELEMENT_NODE** The node is an Element.
static short	**ENTITY_NODE** The node is an Entity.
static short	**ENTITY_REFERENCE_NODE** The node is an EntityReference.
static short	**NOTATION_NODE** The node is a Notation.
static short	**PROCESSING_INSTRUCTION_NODE** The node is a ProcessingInstruction.
static short	**TEXT_NODE** The node is a Text node.

Method Summary

Node	**appendChild**(Node newChild) Adds the node newChild to the end of the list of children of this node.
Node	**cloneNode**(boolean deep) Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.
NamedNodeMap	**getAttributes**() A NamedNodeMap containing the attributes of this node (if it is an Element) or null otherwise.
NodeList	**getChildNodes**() A NodeList that contains all children of this node.
Node	**getFirstChild**() The first child of this node.
Node	**getLastChild**() The last child of this node.
String	**getLocalName**() Returns the local part of the qualified name of this node.
String	**getNamespaceURI**() The namespace URI of this node, or null if it is unspecified.
Node	**getNextSibling**() The node immediately following this node.
String	**getNodeName**() The name of this node, depending on its type; see the table above.
shor	**getNodeType**()

Examen Web/XML

Durée : 2 heures, Documents autorisés

t	A code representing the type of the underlying object, as defined above.
String	**getNodeValue**() The value of this node, depending on its type; see the table above.
Document	**getOwnerDocument**() The Document object associated with this node.
Node	**getParentNode**() The parent of this node.
String	**getPrefix**() The namespace prefix of this node, or null if it is unspecified.
Node	**getPreviousSibling**() The node immediately preceding this node.
boolean	**hasAttributes**() Returns whether this node (if it is an element) has any attributes.
boolean	**hasChildNodes**() Returns whether this node has any children.
Node	**insertBefore**(Node newChild, Node refChild) Inserts the node newChild before the existing child node refChild.
boolean	**isSupported**(String feature, String version) Tests whether the DOM implementation implements a specific feature and that feature is supported by this node.
void	**normalize**() Puts all Text nodes in the full depth of the sub-tree underneath this Node, including attribute nodes, into a "normal" form where only structure (e.g., elements, comments, processing instructions, CDATA sections, and entity references) separates Text nodes, i.e., there are neither adjacent Text nodes nor empty Text nodes.
Node	**removeChild**(Node oldChild) Removes the child node indicated by oldChild from the list of children, and returns it.
Node	**replaceChild**(Node newChild, Node oldChild) Replaces the child node oldChild with newChild in the list of children, and returns the oldChild node.
void	**setNodeValue**(String nodeValue) The value of this node, depending on its type; see the table above.
void	**setPrefix**(String prefix) The namespace prefix of this node, or null if it is unspecified.

Interface Element

All Superinterfaces:

Node

All Known Implementing Classes:

IIOMetadataNode

---

public interface **Element**

extends Node

Examen Web/XML

Durée : 2 heures, Documents autorisés

The `Element` interface represents an element in an HTML or XML document. Elements may have attributes associated with them; since the `Element` interface inherits from `Node`, the generic `Node` interface attribute `attributes` may be used to retrieve the set of all attributes for an element. There are methods on the `Element` interface to retrieve either an `Attr` object by name or an attribute value by name. In XML, where an attribute value may contain entity references, an `Attr` object should be retrieved to examine the possibly fairly complex sub-tree representing the attribute value. On the other hand, in HTML, where all attributes have simple string values, methods to directly access an attribute value can safely be used as a convenience.In DOM Level 2, the method `normalize` is inherited from the `Node` interface where it was moved.
See also the Document Object Model (DOM) Level 2 Core Specification.

Field Summary

Fields inherited from interface org.w3c.dom.Node
ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Method Summary

String	**getAttribute**(String name) Retrieves an attribute value by name.
Attr	**getAttributeNode**(String name) Retrieves an attribute node by name.
Attr	**getAttributeNodeNS**(String namespaceURI, String localName) Retrieves an Attr node by local name and namespace URI.
String	**getAttributeNS**(String namespaceURI, String localName) Retrieves an attribute value by local name and namespace URI.
NodeList	**getElementsByTagName**(String name) Returns a NodeList of all descendant Elements with a given tag name, in the order in which they are encountered in a preorder traversal of this Element tree.
NodeList	**getElementsByTagNameNS**(String namespaceURI, String localName) Returns a NodeList of all the descendant Elements with a given local name and namespace URI in the order in which they are encountered in a preorder traversal of this Element tree.
String	**getTagName**() The name of the element.
boolean	**hasAttribute**(String name) Returns true when an attribute with a given name is specified on this element or has a default value, false otherwise.
boolean	**hasAttributeNS**(String namespaceURI, String localName) Returns true when an attribute with a given local name and namespace URI is specified on this element or has a default value, false otherwise.
void	**removeAttribute**(String name) Removes an attribute by name.
Attr	**removeAttributeNode**(Attr oldAttr) Removes the specified attribute node.

Examen Web/XML

Durée : 2 heures, Documents autorisés

void	**removeAttributeNS**(String namespaceURI, String localName) Removes an attribute by local name and namespace URI.
void	**setAttribute**(String name, String value) Adds a new attribute.
Attr	**setAttributeNode**(Attr newAttr) Adds a new attribute node.
Attr	**setAttributeNodeNS**(Attr newAttr) Adds a new attribute.
void	**setAttributeNS**(String namespaceURI, String qualifiedName, String value) Adds a new attribute.

Methods inherited from interface org.w3c.dom.Node
appendChild, cloneNode, getAttributes, getChildNodes, getFirstChild, getLastChild, getLocalName, getNamespaceURI, getNextSibling, getNodeName, getNodeType, getNodeValue, getOwnerDocument, getParentNode, getPrefix, getPreviousSibling, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild, setNodeValue, setPrefix