대학교 3-1/웹서

HW #3 (DOM Programming)

Launa 2025. 5. 1. 12:28

제출방법:
- 직접 구현한 메소드들의 소스 코드와 프로그램 실행 결과를 과제 게시글 안에 직접 copy & paste 하여 제출함 (주의: 탭 문자를 공백 문자들로 변환한 후 제출 -> 공지사항 게시글 참조)
- 실행 결과의 양이 많을 경우 중간 부분을 생략함

DOM API와 mondial.xml 문서를 이용하여 다음과 같은 기능을 구현하시오. (DOMLab 예제 참조)

1. 모든 국가(country)들에 대해 이름, 국가코드(car_code), 인구, 수도(capital city) 이름을 찾아 출력한다.
예: Albania
- 국가코드: AL
- 인구: 3249136
- 수도: Tirane …

2. 모든 국가들에 대해 이름, 면적, 인구, 수도에 관한 정보를 찾아 다음과 같이 요약된 문서를 생성한다. (기존 문서의 DOM 트리를 수정하여 새로운 문서로 저장함; result-sample.xml 참조)

```
   <mondial>
<country>                              <!-- <country>의 모든 속성들을 삭제함 -->
        <name>Albania</name>
        <area>28750</area>          <!-- 면적을 나타내는 <area>를 <name> 다음에 추가함 -->
        <population>3249136</population>
        <city country="AL" id="…" is_country_cap="yes" >  <!-- 수도에 해당하는 엘리먼트(속성 포함) -->
          <name>Tirane</name>                <!-- 수도의 자손 엘리먼트들을 모두 포함 -->
          <longitude>19.8</longitude>
```

```
        <latitude>41.3</latitude>
        <population year="87">192000</population>
      </city>      <!-- <country>에 관한 다른 정보(child element)들은 모두 삭제함 -->
    </country>
    …          <!-- <country>들 외의 다른 정보(sibling element)들은 모두 삭제함 -->
</mondial>
```

3. 각 국가의 인구와 종교 정보를 이용하여 전세계의 종교별 신자 수를 계산 및 출력한다.
각 국가의 종교들은 엘리먼트들을 통해 이름(content 값)과 인구 비율(percentage 속성)을 알 수 있다. 특정 국가의 인구 값(A)과 그 국가의 종교별 인구 비율(B)을 이용하여 그 국가의 종교별 신자 수 (AxB/100)를 계산할 수 있다. 각 종교별로 모든 국가에 속한 신자 수를 합산한다.
종교명이나 개수는 미리 가정하지 말고 mondial 문서에서 발견되는 대로 처리해야 한다. 종교별 신자 수를 저장하기 위해 배열이나 List 보다는 HashMap을 이용해서 (종교명, 신자 수) 쌍을 저장 및 검색하는 것이 효율적이다.
인구나 신자 수는 큰 값이 될 수 있으므로 long 타입 변수를 이용하여 저장한다. 산술 연산을 위 해 문자열 타입의 값을 long 타입의 값으로 바꾸려면 Long.parseLong(String s) 메소드를 이용한다.
Map에 저장된 모든 entry들을 출력하려면 Map#entrySet()을 통해 Map.Entry 객체들의 Set을 생성 한 후 iterator를 통해 하나씩 접근할 수 있다.
long 타입 값을 출력할 때 세 자리 마다 comma를 찍으려면 다음과 같이 java.text.NumberFormat 클래스의 format() 메소드를 이용하여 문자열로 변환한다. NumberFormat formatter = NumberFormat.getInstance(); formatter.format(longNumber); (선택 사항) 신자 수가 가장 많은 종교부터 내림차순으로 정렬하여 결과를 출력한다.
— Hint: Map.Entry들의 Set으로부터 Map.Entry들의 리스트(ArrayList)를 생성할 수 있다. 그 원소(Map.Entry)들을 value 값을 기준으로 정렬하기 위한 Comparator 클래스를 익명 클래스로 정 의한 후 Collections.sort(...) 메소드를 이용해서 리스트를 정렬한다.

실행 예:

(mondial-sample.xml 이용 시)

```
Populations of the Religions
----------------------------
Roman Catholic 신자: 82,081,850명
Christian Orthodox 신자: 20,659,908명
…
Jewish 신자: 583,174명
Chondogyo 신자: 90,964명
----------------------------
```

합계: 355,135,260명

(mondial.xml 이용 시)
Populations of the Religions

Roman Catholic 신자: 884,650,177명
Druze 신자: 108,439명
…
Church Christ 신자: 6,745명
Armenian Apostolic 신자: 417,584명

합계: 3,759,248,552명

4. (선택 문제) 각 국가의 인구 정보를 이용하여 대륙별 인구를 계산 및 출력한다.
각 국가가 속한 대륙은 엘리먼트의 continent 속성을 참조한다.
여러 대륙에 걸쳐 있는 국가(예: 터키, 러시아 등)의 경우, 소속 비율(percentage 속성)을 통해 가장 많이 속한 대륙에 그 나라의 전체 인구가 속한다고 가정한다 (예: 터키는 아시아에 95% 속해 있으 므로 터키 인구는 모두 아시아에 속하는 것으로 계산함)

실행 예:
(mondial-sample.xml 이용 시)
Populations of the Continents

Africa의 인구: 57,171,662명(14.476%)
America의 인구: 24,523,408명(6.209%)
Asia의 인구: …
Australia의 인구: …
Europe의 인구: …

합계: 394,938,991명

(mondial.xml 이용 시)
Populations of the Religions

Africa의 인구: 731,161,250명(12.662%)
America의 인구: 784,188,664명(13.58%)
Asia의 인구: …

Australia의 인구: …
Europe의 인구: …

합계: 5,774,449,258명

주의 사항
- 구현 과정에서는 mondial-sample.xml 파일을 이용해서 테스트하고, 프로그램 완성 후에 mondial.xml을 이용하여 결과를 생성 및 제출함
- 소스코드 내에 주석을 달아 간략히 설명할 것

제출방법: 직접 구현한 메소드들의 소스 코드와 프로그램 실행 결과를 과제 게시글 안에 직접 copy & paste 하여 제출함 (실행 결과의 양이 많을 경우 중간 부분 생략 가능)

result_sample. xml

```
<mondial>
<country>
<name>Albania</name>
<area>28750</area>
<population>3249136</population>
<city country="AL" id="cty-cid-cia-Albania-Tirane" is_country_cap="yes">
<name>Tirane</name>
<longitude>19.8</longitude>
<latitude>41.3</latitude>
<population year="87">192000</population>
</city>
</country>
<country>
<name>Turkey</name>
<area>780580</area>
<population>62484478</population>
<city country="TR" id="cty-Turkey-
Ankara" is_country_cap="yes" is_state_cap="yes" province="prov-cid-cia-Turkey-8">
<name>Ankara</name>
<longitude>32.8833</longitude>
<latitude>39.95</latitude>
<population year="94">2782200</population>
```

```
</city>
</country>
<country>
<name>France</name>
<area>547030</area>
<population>58317450</population>
<city country="F" id="cty-France-
Paris" is_country_cap="yes" is_state_cap="yes" province="prov-cid-cia-France-53">
<name>Paris</name>
<longitude>2.48333</longitude>
<latitude>48.8167</latitude>
<population year="90">2152423</population>
<located_at river="river-Seine" watertype="river"/>
</city>
</country>
<country>
<name>South Korea</name>
<area>98480</area>
<population>45482291</population>
<city country="ROK" id="cty-cid-cia-South-Korea-2" is_country_cap="yes">
<name>Seoul</name>
<longitude>126.967</longitude>
<latitude>37.5667</latitude>
<population year="95">10229262</population>
</city>
</country>
<country>
<name>Japan</name>
<area>377835</area>
<population>125449703</population>
<city country="J" id="cty-Japan-
Tokyo" is_country_cap="yes" is_state_cap="yes" province="prov-cid-cia-Japan-17">
<name>Tokyo</name>
<longitude>139.767</longitude>
<latitude>35.6833</latitude>
<population year="95">7843000</population>
<located_at sea="sea-Pacific" watertype="sea"/>
```

<located_on island="island-Honshu"/>
</city>
</country>
<country>
<name>Australia</name>
<area>7686850</area>
<population>18260863</population>
<city country="AUS" id="cty-cid-cia-Australia-
8" is_country_cap="yes" is_state_cap="yes" province="prov-cid-cia-Australia-9">
<name>Canberra</name>
<longitude>149.08</longitude>
<latitude>-35.1</latitude>
<population year="90">310000</population>
<located_at lake="lake-LakeBurleyGriffin" watertype="lake"/>
<located_at river="river-MurrumbidgeeRiver" watertype="river"/>
</city>
</country>
<country>
<name>Peru</name>
<area>1285220</area>
<population>24523408</population>
<city country="PE" id="cty-Peru-
Lima" is_country_cap="yes" is_state_cap="yes" province="prov-cid-cia-Peru-16">
<name>Lima</name>
<longitude>-77.05</longitude>
<latitude>-12.0833</latitude>
<population year="93">6321173</population>
</city>
</country>
<country>
<name>Ethiopia</name>
<area>1127127</area>
<population>57171662</population>
<city country="ETH" id="cty-Ethiopia-Addis-Ababa" is_country_cap="yes">
<name>Addis Ababa</name>
<longitude>38.75</longitude>
<latitude>9</latitude>

```
<population year="94">2316400</population>
</city>
</country>
</mondial>
```

modial 과 modial_test는 너무 용량이 커서 첨부 실패

교수님 코드

```
public static void retrieveCountryInfo(Element mondial) {


for (Node ch = mondial.getFirstChild(); ch != null; ch = ch.getNextSibling()) {
if (ch.getNodeName().equals("country")) { // ch: <country>
Element country = (Element) ch; // 미리 Element 타입 변수로 참조


/* 또는
NodeList list = mondial.getElementsByTagName("country");
for (int i = 0; i < list.getLength(); i++) {
Element country = (Element) countryList.item(i);
*/


// 1-1 국가 이름
Node name = country.getFirstChild(); // <name>Albania</name>
Text txt = (Text) name.getFirstChild();
String countryName = txt.getData(); // "Albania"
// 또는 = name.getFirstChild().getNodeValue();
// 또는 = name.getTextContent();
System.out.println(countryName);
```

```
// 1-2 국가 면적

Node area = country.getAttributes().getNamedItem("area");

String areaValue = area.getNodeValue();

// 또는 String areaValue = country.getAttribute("area");

System.out.println("- 면적: " + areaValue);


// 1-3 인구

Node popNode = name.getNextSibling(); // <name>의 다음 형제 노드

if (popNode.getNodeName().equals("population")) { // <population>

System.out.println("- 인구: " + popNode.getTextContent());

} else { // <population>이 아닌 다른 엘리먼트

System.out.println("- 인구 정보 없음");

}


// 1-4 수도의 이름

Element capital = null;

String capitalId = country.getAttribute("capital");

// 주의: Gaza Strip은 capital 속성이 없기 때문에 빈 문자열("")이 반환됨

if (!capitalId.isEmpty()) {

capital = country.getOwnerDocument().getElementById(capitalId);

String capitalName = capital.getFirstChild().getTextContent();

System.out.println("- 수도: " + capitalName);

}


/*

* // getElementById()를 이용하지 않고 직접 수도를 찾을 경우 아래와 같이 작성
```

```
* NodeList cityList = ((Element)country).getElementsByTagName("city");
* for (int i = 0; i < cityList.getLength(); i++) {
* Node city = cityList.item(i);
* String cityId = ((Element)city).getAttribute("id");
* if (cityId.equals(capitalId)) {
* System.out.println("- 수도 : " + city.getFirstChild().getTextContent());
* break;
* }
* }
*/
System.out.println();
}
}
}

public static void summarizeCountryInfo(Element mondial) {
NodeList list = mondial.getElementsByTagName("country");
for (int i = 0; i < list.getLength(); i++) {
Node ch = list.item(i);
Element country = (Element) ch; // 미리 Element 타입 변수로 참조

// 2-1 <area>를 생성해서 <name> 다음에 추가
String areaValue = country.getAttribute("area"); // "28750"
Document doc = country.getOwnerDocument();
Element areaElm = doc.createElement("area"); // <area></area>
Text areaTxt = doc.createTextNode(areaValue);
```

```
areaElm.appendChild(areaTxt); // <area>28750</area>

Node name = country.getFirstChild(); // <name>Albania</name>

Node population = name.getNextSibling();

country.insertBefore(areaElm, population);


// 2-2 수도에 해당하는 <city>(및 그 subtree)를 DOM 트리에서 찾아 미리 제거해 둠

Element capital = null;

String capitalId = country.getAttribute("capital");

if (!capitalId.isEmpty()) { // 주의: Gaza Strip은 capital 속성이 없음

capital = country.getOwnerDocument().getElementById(capitalId);

Node parent = capital.getParentNode(); // 부모 노드를 구함

parent.removeChild(capital); // 부모 노드에서 자식 노드를 삭제

// 주의: capital이 가리키는 <city>는 <province>의 자식인 경우가 있으므로

// country.removeChild(capital)는 오류가 발생함

}


// 2-3 <population> 이후의 형제 노드들을 모두 삭제

Node child = population.getNextSibling();

while (child != null) {

Node nextChild = child.getNextSibling();

country.removeChild(child);

child = nextChild;

}

/*

* // 또는

* for (Node child = population.getNextSibling(); child != null; ) {
```

```
* Node nextChild = child.getNextSibling();
* country.removeChild(child);
* child = nextChild;
* }
*/

// 2-4 수도에 해당하는 <city>(및 그 subtree)를 <country>의 마지막 자식으로 추가
if (capital != null) { // 수도가 없는 경우 capital == null
country.appendChild(capital); // 주의: capital이 null인 경우 오류 발생!


// <city>의 id를 제외한 모든 속성들을 삭제
String id = capital.getAttribute("id");
NamedNodeMap attrs = capital.getAttributes();
while (attrs.getLength() > 0) { // 모든 속성 삭제
Attr attrNode = (Attr)attrs.item(0);
capital.removeAttributeNode(attrNode);
// 또는 country.removeAttribute(attrNode.getName());
// 또는 attrMap.removeNamedItem(attrNode.getName())
}
capital.setAttribute("id", id); // id 속성 추
}

// 2-5 <country>의 모든 속성들을 삭제
NamedNodeMap attrMap = country.getAttributes();
while (attrMap.getLength() > 0) {
```

```
Attr attrNode = (Attr)attrMap.item(0);

country.removeAttributeNode(attrNode);

}

}


// 2-6 <country> 외의 다른 노드들은 모두 삭제

for (Node ch = mondial.getFirstChild(); ch != null;) {

if (!ch.getNodeName().equals("country")) {

Node nextChild = ch.getNextSibling();

mondial.removeChild(ch);

ch = nextChild;

} else

ch = ch.getNextSibling();

}

// 변경된 DOM Tree를 XML file로 저장

saveDOMTreeToFile(mondial.getOwnerDocument());

}


public static void computeBelieversOfReligions(Element mondial) {

Map<String, Long> religions = new HashMap<String, Long>();


NodeList list = mondial.getElementsByTagName("country");


for (int i = 0; i < list.getLength(); i++) {

Element country = (Element) list.item(i);
```

```
Node popNode = country.getFirstChild().getNextSibling(); // <population>?

// if (!popNode.getNodeName().equals("population")) // <population>이 아닌 다른 엘리먼트

// return;


long population = Long.parseLong(popNode.getTextContent()); // population의 값을 구해서 long type으로 변환


Node node = popNode.getNextSibling();

while (node != null && !node.getNodeName().equals("religions")) {

node = node.getNextSibling();// <religions>가 발견될 때까지 이동

}

if (node == null) break; // <religions>가 없는 국가


while (node.getNodeName().equals("religions")) { // 각 <religions>에 대해 처리

String percentage = ((Element) node).getAttribute("percentage");

double per = Double.parseDouble(percentage);

long numOfBelievers = (long) (population * per / 100);

String religionName = node.getTextContent();

System.out.println(religionName);


if (religions.containsKey(religionName)) { // religions map에 이미 같은 종교 정보가 존재

religions.put(religionName, religions.get(religionName) + numOfBelievers);

} else { // 새로운 종교

religions.put(religionName, numOfBelievers);

}


node = node.getNextSibling();
```

```
}

}


// 계산된 각 종교의 인구를 출력

NumberFormat formatter = NumberFormat.getInstance();

System.out.println("Populations of the Religions");

System.out.println("----------------------------");


// religions map의 각 entry의 key와 value를 출력 (iterator 이용)

long total = 0L;

Set<Map.Entry<String, Long>> religionSet = religions.entrySet(); // map entry들의 집합

Iterator<Map.Entry<String, Long>> iter = religionSet.iterator();

while (iter.hasNext()) {

Map.Entry<String, Long> religion = iter.next();

long NumOfBelievers = religion.getValue();

System.out.println(religion.getKey() + " 신자: " + formatter.format(NumOfBelievers) + "명");

total += NumOfBelievers;

}


System.out.println("----------------------------");

System.out.println("합계: " + formatter.format(total) + "명");

System.out.println();



// religions map에 저장된 entry들을 신자 수의 역순으로 정렬해서 출력하는 경우:
```

```
Set<Map.Entry<String, Long>> religionSet2 = religions.entrySet(); // map entry들의 집합

List<Map.Entry<String, Long>> religionList = new LinkedList<Map.Entry<String, Long>>
(religionSet2); // map entry들의 리스트



// map entry들의 비교를 위한 Comparator 정의 (anonymous class로 정의 및 객체 생성)

Comparator<Map.Entry<String, Long>> religionComparator = new
Comparator<Map.Entry<String, Long>>() {

@Override

public int compare(Entry<String, Long> rel1, Entry<String, Long> rel2) {

return (int) (rel2.getValue() - rel1.getValue()); // 내림차순을 위한 비교

}

};



// religionList의 entry들을 내림차순으로 정렬

Collections.sort(religionList, religionComparator);



// 정렬된 리스트의 entry들을 차례대로 출력

long total2 = 0L;

Iterator<Map.Entry<String, Long>> iter2 = religionList.iterator();

while (iter2.hasNext()) {

Map.Entry<String, Long> religion = iter2.next();

long NumOfBelievers = religion.getValue();

System.out.println(religion.getKey() + " 신자: " + formatter.format(NumOfBelievers) + "명");

total2 += NumOfBelievers;

}



System.out.println("----------------------------");
```

```
System.out.println("합계: " + formatter.format(total2) + "명");

System.out.println();

}


public static void computePopulationsOfContinents(Element mondial) {

// 대륙별 인구 계산 및 출력을 위해 대륙 이름들을 저장한 배열 정의

final String continent[] = new String[] { "Africa", "America", "Asia", "Australia", "Europe" };

// 각 대륙의 인구 값을 저장할 배열 선언(대륙의 순서는 위 배열과 동일)

long pop[] = new long[5];


NodeList list = mondial.getElementsByTagName("country");

for (int i = 0; i < list.getLength(); i++) {


Node country = list.item(i);


Node popNode = country.getFirstChild().getNextSibling(); // <population>?

// if (!popNode.getNodeName().equals("population")) // <population>이 아닌 다른 엘리먼트

// return;


String popStr = popNode.getTextContent(); // population의 값

long population = Long.parseLong(popStr); // long형 정수로 변환


Node node = popNode.getNextSibling();

while (node != null && !node.getNodeName().equals("encompassed"))

node = node.getNextSibling(); // <encompassed>가 발견될 때까지 이동
```

```
double maxPer = 0.0; // percentage의 max 값을 저장할 변수

String maxContinent = null; // percentage가 max일 때의 continent 값을 저장할 변수

while (node.getNodeName().equals("encompassed")) {

Element encompassed = (Element) node;

String percentage = encompassed.getAttribute("percentage");

double per = Double.parseDouble(percentage);


if (per > maxPer) { // 기존보다 더 큰 percentage 값 발견

maxPer = per; // max percentage 값 갱신

maxContinent = encompassed.getAttribute("continent"); // continent 값 갱신

}

node = node.getNextSibling();

}


switch (maxContinent) {

case "africa":

pop[0] += population;

break;

case "america":

pop[1] += population;

break;

case "asia":

pop[2] += population;

break;

case "australia":

pop[3] += population;
```

```
break;
case "europe":
pop[4] += population;
}
}

// 계산된 각 대륙의 인구를 출력
NumberFormat formatter = NumberFormat.getInstance();
System.out.println("Populations of the Continents");
System.out.println("----------------------------");
long total = 0L;
for (int i = 0; i < pop.length; i++) {
total += pop[i];
}
for (int i = 0; i < pop.length; i++) {
System.out.println(continent[i] + "의 인구: " + formatter.format(pop[i]) + "명("
+ formatter.format(pop[i] * 100.0 / total) + "%)");
}
System.out.println("----------------------------");
System.out.println("합계: " + formatter.format(total) + "명");
System.out.println();
}
```

공감

'대학교 3-1 > 웹서' 카테고리의 다른 글

웹서 기말고사 대비 (0) 2025.06.26

HW #4 (HTML DOM, jQuery Programming) (0) 2025.05.01

'대학교 3-1/웹서' Related Articles

NO IMAGE

웹서 기말고사 대비

NO IMAGE

HW #4 (HTML DOM, jQuery Programming)

댓글 0

Launa

내용을 입력하세요.

등록