웹서비스 실습 #5

Ajax Programming

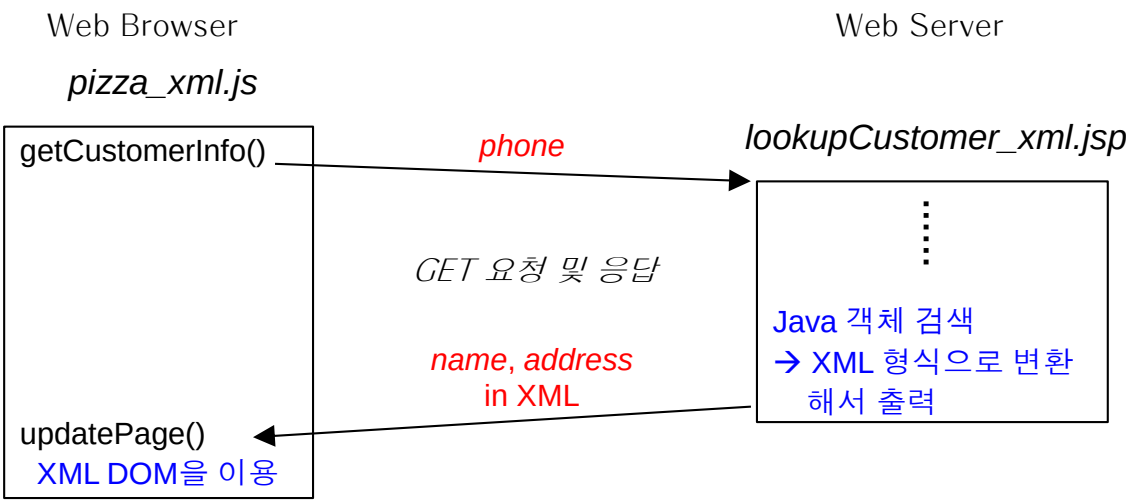
AjaxLab의 pizza 예제 프로그램을 다음과 같이 수정하시오.

1. 고객의 주소 정보를 나타내는 Address 클래스를 정의해서 사용하고, lookupCustomer.jsp에서는 전송된 전화번호에 해당하는 고객 정보를 검색하여 그 결과를 반환하도록 수정한다. (lookupCustomer2.jsp 참조)
   (1) src 폴더 아래의 example.ajax.pizza 패키지에 Address 클래스를 생성한다. Customer 클래스에 있는 주소에 관한 네 가지 필드들을 옮기고 생성자와 get method들을 정의함
   (2) Customer 클래스에서 주소 필드들을 삭제하고 Address 객체에 대한 참조 변수를 추가함. 전화번호를 저장하기 위한 필드도 추가하고, 추가된 필드에 대한 get method들을 정의함
   (3) lookupCustomer.jsp에서는 수정된 Customer와 Address 클래스를 이용하여 Customer 객체들을 생성하고 map에 저장함. 전송된 phone 값을 갖고 있는 Customer 객체를 찾아 이름과 주소 정보를 결과로 출력(반환)함 (객체를 찾지 못할 경우 400 status code를 갖는 응답 메시지를 반환함)

2. 서버 프로그램은 검색한 고객 이름 및 주소 정보를 XML 형식의 데이터로 표현하여 전송하고, 클라이언트에서는 DOM API를 이용하여 데이터를 추출 및 출력한다(아래 그림 참조). pizza_xml 폴더 밑에 pizza_xml.html, pizza_xml.js, lookupCustomer_xml.jsp를 각각 작성한다.
   (1) lookupCustomer_xml.jsp에서는 검색된 고객 정보를 XML로 출력하기 위해 다음과 같은 코드를 이용함

```
<result>
  <name><%=c.getName()%></name>
  <address>
    <street><%=c.getAddress().getStreet()%></street>
    …
  </address>
</result>
```

   - 주의: 이 응답 메시지가 XML 형식의 데이터임을 나타내기 위해 @page 지시자의 contentType 속성의 값을 "text/xml; charset=utf-8"로 지정해야 함
   (2) pizza_xml.js에서는 updatePage()에서 request.responseXML 속성과 DOM API를 이용하여 전송된 XML 데이터로부터 필요한 값들을 구해 기존과 같이 아래쪽 창에 출력함
   (3) pizza_xml.html에서는 pizza_xml.js를 이용함



3. 위 2번의 결과를 수정하여, 고객들의 최근 주문 정보를 저장 관리하고 고객 정보 요청 시 검색 및 출력한다.
   (1) Customer 클래스에 가장 최근에 주문한 내용을 나타내는 필드(예: recentOrder)를 추가함
   (2) 서버 프로그램은 각 고객 객체 생성 시 그 필드에 대해 “no recent order”를 저장함
   (3) 고객 정보 요청(GET 요청) 시 최근 주문 정보도 검색하여 XML 데이터에 추가 및 반환함
   (4) 클라이언트에서는 전송받은 데이터에서 최근 주문 정보를 추출하여 주문 입력 창에 초기값으로 출력함. 또 주문 입력 창 위에 고객의 이름을 포함하는 인사말을 출력함 (예: “Hi, Mary Jenkins! Type your order in here:”)
   (5) 서버 프로그램에서 주문 요청(POST 요청) 처리 시 해당 고객의 최근 주문 정보를 갱신함

4. 위 3번과 같은 프로그램을 XML 대신 JSON을 이용하여 구현한다(아래 그림 참조). pizza_json 폴더 밑에 pizza_json.html, pizza_json.js, lookupCustomer_json.jsp, placeOrder_json.jsp를 각각 작성한다.

   (1) lookupCustomer_json.jsp에서는 검색된 고객 정보를 JSON 형식으로 나타내기 위해 JSON-simple 또는 Jackson library를 이용함.
   - JSON-simple 이용 시 Address와 Customer에 대한 JSONObject를 생성 및 연동함
   - 주의: 응답 메시지가 JSON 형식임을 나타내기 위해 @page 지시자의 contentType 속성 값을 "application/json; charset=utf-8"로 지정함

   (2) pizza_json.js의 updatePage()에서는 request.responseText 속성과 JSON.parse()를 이용하여 전송된 JSON 데이터를 JavaScript 객체로 변환한 후, 객체에 저장된 값들을 이용한다.

   (3) pizza_json.js의 submitOrder()에서는 form에 입력된 값들을 하나의 객체에 저장한 후 JSON.stringify()를 이용하여 객체를 JSON 텍스트로 변환한다. Ajax 요청의 url과 content type은 각각 placeOrder_json.jsp와 application/json으로 설정한다.

   (4) placeOrder_json.jsp에서는 JSON-simple 또는 Jackson library를 이용하여 전송된 JSON 텍스트를 parsing 및 객체로 변환한다. 요청 메시지에서 JSON 텍스트를 읽기 위해 request.getReader() 메소드를 이용한다.

   (5) pizza_json.html에서는 pizza_json.js를 이용하여 Ajax 호출 및 화면 갱신을 실행한다.
   - updatePage()와 submitOrder()에서는 전송받거나 전송할 데이터가 올바른 JSON 형식의 텍스트인지 확인하기 위해 먼저 alert()이나 console.log()를 이용해서 출력해 볼 것