**DEV #7 – 2020.04.28**

>> XML을 조사하시오.

1. XML prolog ?
   1. Xml 형식의 버전,인코딩을 알리는 태그로써 최상위 요소(루트 요소)외에 추가정보를 담기위한 것(필수는 아님) –있다면 문서의 맨 앞부분에 있어야 하며 공백도 존재하면 안됨
2. XML 구현방식 [ DTD / XSD(스키마) ]
   1. DTD(문서 타입 정의)는 XML 문서 구조 및 해당 문서에서 사용 할 수 있는 적법한 요소와 속성을 정의함

DTD 사용시 새로운 XML 문서의 구조를 정의함으로써 새로운 문서 타입 생성가능

DTD 단점 1. 네임스페이스를 지원하지 않습니다.

2. XML 문법이 아닌 별도의 문법으로 작성해야 합니다.

3. 제한된 타입만을 지원합니다.

4. 재사용과 확장이 불가능합니다.

DTD 문법

<!DOCTYPE 루트요소 DTD식별자 [ 선언1 선언2 ... ]>

* 1. XSD는 스키마 정의를 의미함 XSD는 XML 문서의 구조 및 해당 문서가 포함할 수 있는 적법한 요소와 속성을 명시함. DTD 문법의 단점을 개선하기 위해서 만들어짐

XSD 특징 1. 네임스페이스를 지원합니다.

2. XML 문법으로 작성할 수 있어 재사용과 확장이 쉽습니다.

3. 정수, 문자열 등의 다양한 타입을 지원합니다.

XSD 문법

<?xml version="1.0" encoding="UTF-8" ?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

     targetNamespace="http://codingsam.com"

     xmlns="http://codingsam.com"

     elementFormDefault="qualified">

...

</xs:schema>

1. Maven pom.xml 최상단 마크업의 의미를 설명하시오.

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

>> 기존 Java Project에 web.xml을 사용하여 신규 서블릿을 하나 생성하시오.

>> 기존 Project에 com.gachidata.vo패키지를 만들고 Food class를 만드시오 (POJO객체)

* 메소드 없이 멤버변수 자유 추가 – 예) name, color, ....

>> 기존 Project에 com.gachidata.service 패키지를 만들고 FoodService class를 만드시오.

* cookedFoodLIst method를 만들고 3 ~ 5개의 Food 객체 리스트를 만들어 리턴.
* hint) List<Food> cookedFoolList() {

List<Food> foods = new ArrayList<Food>(); .

.......

return foods;

}

>> 기존 Project에 com.gachidata.controller 패키지를 만들고 FoodController class를 만드시오.

1. 서블릿을 doGet으로 구현하고 FoodService의 cookedFoodList()의 결과를 받아와 html table에 담아 웹으로 출력하세요.

package com.gachidata.controller;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.gachidata.service.FoodService;

*@WebServlet*(urlPatterns = { "/food" })

public class FoodController extends HttpServlet {

/\*\*

\*

\*/

private static final long ***serialVersionUID*** = 1L;

*@Override*

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

resp.setContentType("text/html");

/\* encoding 설정 \*/

resp.setCharacterEncoding("UTF-8");

FoodService flist = new FoodService();

PrintWriter out = resp.getWriter();

out.println("<html>");

out.println("<head><title>FoodController</title></head>");

out.println("<body>");

out.println("현재 시간은");

out.println(new Date());

out.println("입니다.");

out.println("<table>");

out.println("<tr><td>");

out.println("<h3>" + flist.cookedFoodList().get(0) + "<h3>");

out.println("</tr></td>");

out.println("<tr><td>");

out.println("<h3>" + flist.cookedFoodList().get(1) + "<h3>");

out.println("</tr></td>");

out.println("<tr><td>");

out.println("<h3>" + flist.cookedFoodList().get(2) + "<h3>");

out.println("</tr></td>");

out.println("<tr><td>");

out.println("<h3>" + flist.cookedFoodList().get(3) + "<h3>");

out.println("</tr></td>");

out.println("<tr><td>");

out.println("<h3>" + flist.cookedFoodList().get(4) + "<h3>");

out.println("</tr></td>");

out.println("</table>");

out.println("</body></html>");

}

}

>> forward / redirect를 조사하고 서블릿으로 forward를 구현하시오.

Forward 는 특정 서블릿에 대한 요청을 다른 서블릿이나 jsp로 넘겨준다 이때 forward는 redirect 와 다르게 파라미터를 넘겨 줄 수 있다. Forward 는 상대방에게 페이지 주소를 숨길 때 사용 할 수 있으며 URL이 바뀌지 않고, 내부에서만 접근이 가능하다.

ex)

RequestDispatcher rd = request.getRequestDispatcher("/result.jsp");

rd.forward(request, response);

// “/result.jsp” 으로 파라미터를 넘겨주고 /result.jsp로 넘겨줌

Redirect 는 다른테이지로 넘어가도록 하게 해주는 것으로써 직접 파라미터 값을 넘겨 줄 수 없다 요청을 받게되면 url을 클라이언트에게 전달하고, 클라이언트가 새로운 url을 요청하고 그에 따른 응답을 한다. //URL 값이 넘어가기 때문에 길이에 제한이 있음

ex)

response.sendredirect("URL");

1. controller에 등록된 서블릿에서 response결과를 jsp로 forward처리 구현

ServletContext context = getServletContext();

RequestDispatcher dispatcher = context.getRequestDispatcher("/view/index.jsp"); // index.jsp 로 넘겨줌

req.setAttribute("FoodList", flist.cookedFoodList().toString()); // FoodList 라는 태그로 flist 객체를 넘겨줌

dispatcher.forward(req, resp);

1. webroot에 view 디렉토리를 만들고 위에 forward되는 jsp파일 생성 구현

1. jsp 파일에서 foodList를 JSTL EL문법을 사용하여 html table 생성 구현

<%@ **page** language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ **taglib** prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<!**DOCTYPE** html>

<**html**>

<**head**>

<**meta** charset=*"UTF-8"*>

<**title**>index.jsp</**title**>

</**head**>

<**body**>

<**br**>

<**c:forEach** items=*"*${requestScope.FoodList}*"* var=*"Food"*>

<**br**>

<**tr**>

<**td**>"${Food}"</**td**>

</**tr**>

</**c:forEach**>

<**table**>

</**table**>

</**body**>

</**html**>

1. 웹상에서 구동여부 확인