DCL HtmlTemplate

(from DCLNet)

2005.06.08

김대중<daejung@sysdeveloper.net>
http://www.sysdeveloper.net/daejung

개요

웹 어플리케이션 개발에 있어서 서버에서의 동적 HTML 생성은 매우 중요한 위치를 차지하고 있다. 동적 HTML 생성을 위한 개발 도구는 대체적으로 ASP, PHP, JSP 등과 같은 스크립트 언어를 사용하고 있는 것이 현실이다. 이러한 스크립트 언어는 그 사용법이 쉽기 때문에 자칫 HTML과 스크립트의 무분별한 사용으로 인하여 개발 과정 및 유지 보수에 있어서 생산성을 떨어뜨릴 뿐만 아니라 개발된 어플리케이션의 품질에도 나쁜 영향을 준다.

본 문서는 동적인 HTML 생성을 위하여 HtmlTemplate라는 클래스를 구현하여 보다 구조적인 방법을 제안한다.

문서의 내용은 HtmlTemplate 클래스의 동작 원리와 참조 매뉴얼을 포함하고 있고, ASP(Active Server Page)를 위한 HtmlTemplate의 ActiveX 버전의 설치와 사용에 대한 내용을 포함하고 있다.

- 목 차 -

제1장 개발의 배경

1.1 동적 HTML 생성의 세 가지 방법

웹 서버의 어플리케이션 개발에 있어서 동적인 HTML을 생성하는 어플리케이션은 가장 중요한 위치를 차지하고 있다. 동적인 HTML 생성의 방법은 그 언어가 스크립트를 번역하는 인터프리터 언어이든 혹은 컴파일러 언어이든, 다음과 같이 세 가지로 분류할 수 있다.

첫 번째는, HTML에 스크립트를 삽입하는 형태이다. 일반적으로 HTML 내부에 있는 스크립트는 “<%”와 “%>”, 또는 “<?” 와 “?>” 사이에 위치한다.

두 번째는, 스크립트의 출력 함수에 모든 HTML을 기술하는 것이다. 이 방법은 스크립트 언어에서는 선택적으로 사용할 수 있고, 컴파일러 언어는 필수 사항이다.

세 번째는, HTML 부분과 프로그램을 분리하는 방법이다. 이 방법은 HTML 부분을 별도의 파일로 작성하여 저장하여, 이것을 프로그래밍 언어에서 불러 들여 적절하게 수정 한 후 출력 함수를 통해 출력한다.

이 세가지 방법 중에서 첫 번째와 두 번째는 간단한 동적 페이지를 작성하는데 있어서 매우 손쉽게 사용할 수 있지만, HTML과 프로그래밍 언어의 코드가 섞여 있기 때문에 가독력이 떨어질 뿐만 아니라 더욱 심각한 것은 대규모 프로젝트에 있어서 인력배치와 관련하여 매우 치명적인 단점을 내포하고 있다. 그것은 개발 인력이 프로그래밍 언어와 디자인 중심의 HTML을 모두 잘 알아야만 한다는 점이다.

1.2 디자인과 비즈니스 로직의 분리

디자인과 비즈니스 로직의 분리는 단일 페이지를 생성하는 어플리케이션을 개발하는데 있어서 디자인 작업과 비즈니스 로직의 개발의 상호 의존성을 최소화 하는데 그 목적이 있다. 이렇게 할 경우 각각의 코드는 해당 전문가가 집중하여 작업할 수 있기 때문에 생산성 뿐만 아니라 결과물의 품질까지 제고할 수 있다.

디자인과 프로그램 코드를 분리하는 장점에 대한 내용은 “스크립트 언어를 사용한 웹 페이지 개발의 세 가지 방법”[1]에서 자세히 언급하고 있다.

제2장 class HtmlTemplate

2.1 DCLNet

HtmlTemplate 클래스는 DCL의 패키지 중 DCLNet에 포함되어 있는 클래스 라이브러리이다. HtmlTemplate 클래스는 HTML 혹은 텍스트로 작성된 파일을 읽어 들여 템플릿 객체를 구성한다. 템플릿에서의 중요한 두 가지 개념은 매크로와 서브 템플릿 이다.

2.2 매크로

매크로는 템플릿 객체에서 향후 다른 문자열이나 템플릿으로 치환되거나 추가될 수 있다. 이것은 템플릿 파일에서 '{' 와 '}' 사이의 문자열로 '_'과 '-' 을 포함한 영문 대소문자로 이루어 져야 한다. <그림 1>의 orderlist.html에서 {NAME}, {NO}, {PRODUCT_NAME} 등이 여기에 해당된다.

매크로는 템플릿 사이에서 중복되어 사용할 수 있다. 예를 들면 <그림 1>에서 메인 템플릿에 {NAME}이 있어도 서브 템플릿인 ROW에도 {NAME}을 사용할 수 있다.

2.3 서브 템플릿

템플릿은 한 개 이상의 서브 템플릿을 포함할 수 있다. 서브 템플릿은 "<!-- BEGIN macro -->" 와 "<!-- END macro -->" 사이의 블록을 말한다. <그림 1>에서 "<!-- BEGIN ROW -->"와 "<!-- END ROW -->"는 "ROW"로 이름 지어진 서브 템플릿 이다.

서브 템플릿을 구별하기 위해 "<!--"와 "-->"을 사용한 이유는 HTML에서 이것이 주석으로 처리되기 때문이다. 서브 템플릿을 구별하는 것은 HTML 자체를 문서화 하는 데에도 도움을 준다. 서브 템플릿의 이름은 매크로와 동일하지만 템플릿 파일 전체에서 유일해야 한다.

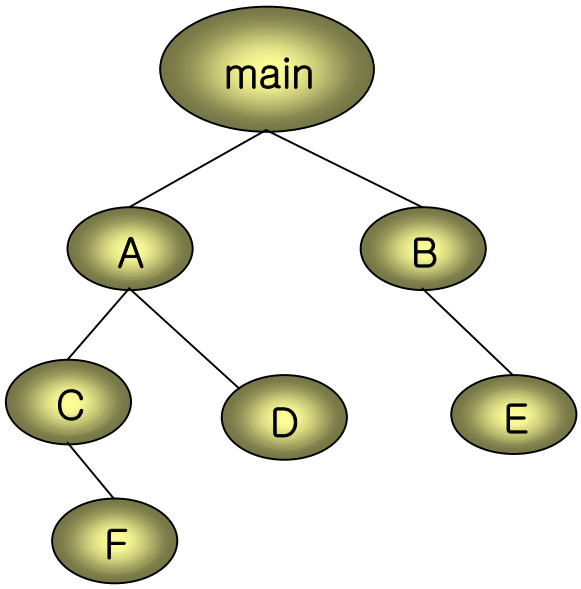
```
 1  <html>
 2  <head>
 3  <title>주문목록</title>
 4  </head>
 5  <body>
 6  <p>{NAME}님께서 주문하신 목록입니다..</p>
 7  <table>
 8  <tr>
 9      <th>순번</th>
10      <th>상품명</th>
11      <th>단가</th>
12      <th>수량</th>
13      <th>금액</th>
14  </tr>
15  <!-- BEGIN ROW -->
16  <tr>
17      <td>{NO}</td>
18      <td>{PRODUCT_NAME}</td>
19      <td>{PRICE}</td>
20      <td>{QUANTITY}</td>
21      <td>{AMOUNT}</td>
22  </tr>
23  <!-- END ROW -->
24  </table>
25  </body>
26  </html>
```

<그림 1> orderlist.html

서브 템플릿 역시 한 개 이상의 서브 템플릿을 포함 할 수 있으며, 이 경우 HtmlTemplate 객체는 서브 템플릿을 재귀적(recursive)으로 처리한다. 이때 서브 템플릿의 이름은 부모 템플릿에서 매크로로 바뀐다. <그림 1>에서 서브 템플릿 ROW는 부모 템플릿 에서는 {ROW}라는 매크로로 되며 서브 템플릿 블록은 서브 템플릿 객체를 통해서만 수정이 된다.

```
<!-- main -->

<!-- BEGIN A -->

    <!-- BEGIN C -->

        <!-- BEGIN F -->

        <!-- END F -->

    <!-- END C -->

    <!-- BEGIN D -->

    <!-- END D -->

<!-- END A -->

<!-- BEGIN B -->

    <!-- BEGIN E -->

    <!-- END E -->

<!-- END B -->
```

템플릿 파일



템플릿 객체의 구성

<그림 2> 서브 템플릿의 구성

구성된 템플릿 객체에서 서브 템플릿의 식별은 자신의 바로 하위 객체만 참조할 수 있다. HtmlTemplate의 메소드인 HtmlTemplate& HtmlTemplate::operator [] (const String& strName)를 통하여 이들을 참조할 수 있으며 ActiveX 버전에서는 객체의 GetSubTemplate 메소드를 사용한다.

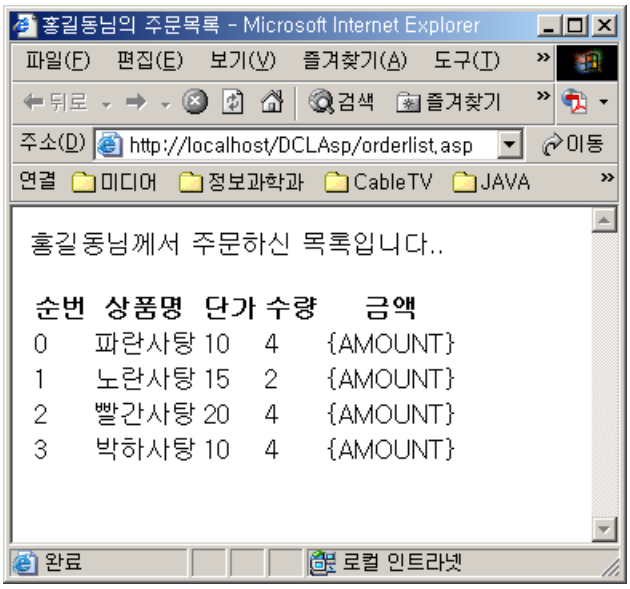
```
Set main = Server.CreateObject("DCL.HtmlTemplate")
main.Open "D:\TEMP\DCLAsp\orderlist.html"
Set row = main.GetSubTemplate("ROW")

main.Assign "NAME", "홍길동"
For I = 0 To 3
    row.Assign "NO", CStr(I)
    row.Assign "PRODUCT_NAME", orders(I).strProductName
    row.Assign "PRICE", CStr(orders(I).nPrice)
    row.Assign "QUANTITY", CStr(orders(I).nQuantity)
    row.Assign "AMOUNT", CStr(orders(I).nPrice * orders(I).nQuantity)

    main.Append "ROW", row
Next

main.WriteToResponse

%>
```

<그림 3> orderlist.asp

2.4 템플릿의 출력

구성된 HtmlTemplate 객체는 void HtmlTemplate::printTo(OutputStream& out) 메소드(ActiveX 버전은 writeToResponse) 를 사용하여 스트림에 출력한다. 이때 매크로에 어떠한 값도 할당되지 않은 매크로는 '{' 와 '}' 를 포함하여 매크로 이름 자체를 출력한다.

다음은 <그림 1>의 orderlist.html 템플릿을 <그림 3>의 orderlist.asp 에서 불러들여 처리 한 후의 결과이다. 이 결과는 <그림 3>의 44번 라인을 주석으로 처리하여 {AMOUNT}에 값을 할당하지 않은 결과이다.



<그림 4> orderlist.asp 의 결과

제3장 참조 매뉴얼

3.1 HtmlTemplate::HtmlTemplate

C++

```
HtmlTemplate();
HtmlTemplate(const HtmlTemplate& src);
HtmlTemplate(const String& strFileName) __DCL_THROWS1(IOException*);
```

ASP

```
Set obj = Server.CreateObject("DCL.HtmlTemplate")
```

설명

HtmlTemplate 객체를 생성한다. C++ 버전에서 HtmlTemplate(const HtmlTemplate& src)는 src의 복사본으로 초기화 된다. HtmlTemplate(const String& strFileName)은 템플릿 파일을 읽어 들여 템플릿 객체를 초기화 하며 파일을 읽어 들이는 중에 예외가 발생할 수 있다.

ASP 에서는 HtmlTemplate 객체의 인터페이스를 얻어온다. 템플릿 객체를 구성하려면 Open 메소드

를 사용하여야 한다.

3.2 HtmlTemplate::operator =

C++

```
HtmlTemplate& operator = (const HtmlTemplate& src);
```

설명

src의 구성을 복사한다.

3.3 HtmlTemplate::open

C++

```
void open(const String& strFileName) __DCL_THROWS1(IOException*);
void open(const String& strName, const String& strFileName)
```

ASP

```
obj.Open strFileName
```

설명

템플릿 파일을 읽어들인다. 두 번째의 open 메소드는 strFileName을 읽어들여 strName의 이름의 서브 템플릿을 구성한다. 만약 이전에 strName의 서브 템플릿이 있으면 이것으로 교체된다.

3.4 HtmlTemplate::clear

C++

```
void clear();
```

ASP

```
obj.Clear
```

설명

객체 내부에 구성된 템플릿을 지운다. 이 메소드를 호출하면 템플릿 객체는 비어 있게 된다. 객체를 다시 사용하려면 open 메소드를 사용하여야 한다.

3.5 HtmlTemplate::reset

C++

```
void reset();
```

ASP

```
obj.Reset
```

설명

객체 생성 후 템플릿 파일을 읽어 들인 직후의 상태로 되돌린다.

3.6 HtmlTemplate::erase

C++

```
void erase(const char* pszMacro);
```

ASP

```
obj.Erase strMacro
```

설명

매크로를 지운다. 이 매크로에 값이 할당 되어 있다면 할당된 값은 출력되지 않는다.

3.7 HtmlTemplate::assign

C++

```
int assign(const char* pszMacro, const String& strValue);
int assign(const char* pszMacro, const HtmlTemplate& t);
int assign(const char* pszMacro);
int assign(const StrStrAssocVector& vStrToStr);
int assign(_CONST SQLFields& fields, const String& strFieldIsNull);
```

ASP

```
obj.Assign strMacro, variant
```

설명

매크로에 문자열을 할당한다. 이전에 할당된 문자열이 있으면 덮어 쓴다.

C++ 버전에서 assign(const char* pszMacro)는 매크로 이름과 동일한 서브 템플릿이 있으면 이것을 사용한다. fields는 DCL의 DBField의 필드 이름과 매크로를 대응시킨다. strFieldIsNull은 필드의 값이 NULL일 경우 사용된다.

ASP 버전에서 variant는 문자열이나 HtmlTemplate 객체가 올 수 있다.

3.8 HtmlTemplate::append

C++

```
int append(const char* pszMacro, const String& strValue);
int append(const char* pszMacro, const HtmlTemplate& t);
int append(const char* pszMacro);
int append(const StrStrAssocVector& vStrToStr);
int append(_CONST SQLFields& fields, const String& strFieldIsNull);
```

ASP

```
obj.Append strMacro, variant
```

설명

assign과 동일하지만 이전에 할당된 문자열 뒤에 문자열을 붙인다. HtmlTemplate 클래스 내부에서는 이것을 StringList로 구현하고 있다.

3.9 HtmlTemplate::operator[]

C++

```
HtmlTemplate& operator [] (const String& strName);
```

ASP

```
Set subObj = obj.GetSubTemplate strName
```

설명

객체의 서브 템플릿 객체의 참조를 얻는다.

3.10 HtmlTemplate::printTo

C++

```
void printTo(OutputStream& out) const   __DCL_THROWS1(IOException*);
```

ASP

```
obj.WriteToResponse
```

설명

스트림에 완성된 템플릿 데이터를 출력한다. ASP 버전의 경우 Response 객체에 출력한다.

3.11 HtmlTemplate::onSQLField

C++

```
virtual String onSQLFieldValue(_CONST SQLField& field, const String& strFieldIsNull);
```

설명

field의 데이터를 포멧팅할 필요가 있을 때 이 멤버를 오버라이드(override)한다. 기본 구현은 field의 값이 널일 경우 strFieldIsNull을 리턴하고 그렇지 않으면 SQLField::toStringF의 기본 포멧 결과를 리턴한다.

3.12 OutputStream& operator << (OutputStream& out, HtmlTemplate& t)

C++

```
DCLNAPI inline OutputStream& operator << (OutputStream& out, HtmlTemplate& t)
```

설명

스트림 out에 템플릿 t의 내용을 출력한다.

제4장 어플리케이션의 개발 및 설치

4.1 C++ 버전

C++ 버전에서의 HtmlTemplate을 사용하고자 하면 기본적으로 dcl/net/HtmlTemplate.h을 인크루드하고 DCLNet 라이브러리를 링크 시키면 된다. 다음은 <그림 3>의 ASP 버전을 C++ 버전으로 작성한 예이다.

```
23      HtmlTemplate tplMain("orderlist.html");
24      HtmlTemplate& tplRow = tplMain["ROW"];
25
26      tplMain.assign("NAME", "홍길동");
27      for(int i = 0; i < 3; i++)
28      {
29          tplRow.assign("NO", Integer::toString(i));
30          tplRow.assign("PRODUCT_NAME", orders[i].pszProductName);
31          tplRow.assign("PRICE", Integer::toString(orders[i].nPrice));
32          tplRow.assign("QUANTITY", Integer::toString(orders[i].nQuantity));
33      //  tplRow.assign("AMOUNT", Integer::toString(orders[i].nPrice * order
34
35          tplMain.append("ROW", tplRow);
36      }
37
38      ctx.getOutputStream() << tplMain;
```

<그림 5> orderlist.cpp

4.2 ASP 버전

DCLCore.dll, DCLNet.dll

ActiveX로 개발된 DCLAsp.dll은 내부에서 DCLCore.dll, DCLNet.dll을 사용하고 있기 때문에 DCLAsp.dll을 사용하기 위해서는 DCL의 두 라이브러리 파일을 설치해야 한다. 이 파일은 윈도우의 System32 아래에 복사하거나 IIS서버가 알 수 있는 %PATH% 경로에 있어야 한다

DCLAsp.dll

HtmlTemplate 컴포넌트 ActiveX 이다. 이 파일을 regsvr32.exe를 사용하여 시스템 레지스트리에 등록한다.

관련문서

[1] 스크립트 언어를 사용한 웹 페이지 개발의 세 가지 방법.pdf