|  |
| --- |
| [키워드] |
| **GAUCE 트리뷰 이미지 적용 가이드**  GAUCE for ActiveEnterprise |
| 영문세로흑백(소)  © SHIFT Information & Communication Co., Ltd.  Phone 02 464.7676 • Fax 02 464 7234 • www.shift.co.kr |
| 쉬프트정보통신㈜의 사전 승인 없이 본 내용의 전부 또는 일부에 대한 복사, 배포, 사용을 금합니다. |

|  |
| --- |
| Copyright Information  ©Copyright 2005 SHIFT Information & Communication Co. 모든 권리는 저작권자의 소유입니다.  저자: 쉬프트정보통신㈜  이 프로그램의 모든 정보에 대해서는 SHIFT Information & Communication Co.에서 소유권을 가집니다.  이 정보는 사용제한 및 기밀유지규정에 포함하는 사용자계약에 따라 제공되며 저작권법에 의해 보호됩니다.  이 소프트웨어를 역 엔지니어링하는 것은 금지 되어 있습니다.  이 설명서의 내용은 사전공지 없이 변경될 수 있습니다.  이 설명서에는 오류가 있을 수도 있으므로 잘못된 내용이 발견되면 서면으로 알려주시기 바랍니다.  SHIFT Information & Communication Co.에서는 오류에 대한 책임을 지지 않습니다.  기타 모든 제품들과 회사이름은 각각 해당 소유주의 상표로서 참조용으로만 사용됩니다. |

목 차

[**1. GAUCE 트리뷰 이미지 적용 가이드** 4](#_Toc259722338)

[**1.1 개요** 4](#_Toc259722339)

[**1.2 GAUCE 데이터셋과 이미지 데이터셋 연동 아키텍쳐** 4](#_Toc259722340)

[**1.3 화면 디자인** 5](#_Toc259722341)

[**1.4 이미지 데이터셋을 만들기 위한 서비스 작성** 6](#_Toc259722342)

[**1.5 기타 참고 사항** 9](#_Toc259722343)

**1. GAUCE 트리뷰 이미지 적용 가이드**

**1.1 개요**

- 본 가이드 문서는 데이터셋 컴포넌트와 이미지 데이터셋을 연동하기 위한 가이드 문서이며, 본 문서의 샘플을 동작하기 위해서는 로컬 혹은 서버쪽 GAUCE가 정상적으로 동작 하여야 합니다.

**1.2 GAUCE 데이터셋과 이미지 데이터셋 연동 아키텍쳐**

* GAUCE컴포넌트중에 트리뷰, 그리드, 멀티라인그리드, 메뉴 컴포넌트에 이미지 데이터셋을 사용하여 특정 부분에 이미지를 표시 할 수 있습니다.
* 이미지 데이터셋과 데이터셋의 차이점
* 일반 데이터셋은 사용자가 DB등에서 조회한 데이터를 UI컴포넌트를 화면에 표시해 주게 됩니다. 이미지 데이터셋은 직접적으로 UI컴포넌트를 통해 화면에 나타내는 방식이 아닌 데이터셋에 특정 컬럼의 데이터 값에 따라 해당 이미지가 표시되는 구조 입니다.
* 이미지 데이터셋의 컬럼 정의시에는 아래와 같은 형태로 컬럼을 정의하게 되며, 데이터타입이 다른 경우 올바르게 동작하지 않습니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 컬럼ID | 컬럼(데이터 타입) | 컬럼(길이) | 비고 |
| IMG\_ID | String | 255 | 컬럼ID는 달라도 무관 |
| IMG | BLOB | 255 |
| IMG\_SIZE | Integer | 6 |

* 이미지 데이터셋은 IMG\_ID라고 정의된 컬럼의 값과 실제 이미지 정보가 들어있는 BLOB(Stream)컬럼, 그리고 파일의 크기를 담고 있는 3의 컬럼으로 구성이 됩니다. 컴포넌트에 이미지가 표시되는 형태는 이미지 데이터셋을 만들 때, 해당 이미지 파일의 스트림정보와 매칭되는 IMG\_ID의 값과 실제 연동되는 데이터셋의 컬럼의 값과 일치되면, 데이터셋의 값이 화면에 표시되는 것이 아닌 이미지 데이터셋의 이미지가 해당 컴포넌트에 표시가 되는 구조 입니다.

이미지 데이터셋

|  |  |
| --- | --- |
| IMG\_ID | IMG |
| **ID1** | Stream |
| **ID2** | Stream |
| … | |

UI표시용 컴포넌트 (Grid, TreeView, Menu, MGrid)

|  |  |  |
| --- | --- | --- |
| 컬럼1 | 컬럼2 | 이미지(컬럼2) |
| 삼송 | **ID1** | 이미지 표시됨 |
| 엘쥐 | **ID3** | 이미지 표시안됨 |
|  |  |  |
|  |  |  |

데이터셋

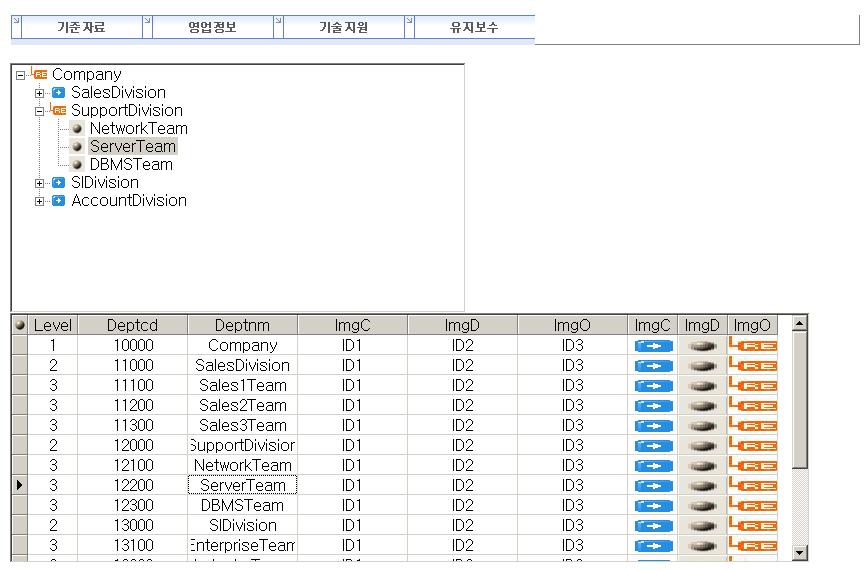
|  |  |
| --- | --- |
| 컬럼1 | 컬럼2 |
| 삼송 | ID1 |
| 엘쥐 | ID2 |

ID3값이 존재하지 않아 이미지가 표시되지 않음

* 위의 그림에서 이미지 데이터셋에 “ID1, ID2”라는 값으로 이미지가 저장 되어 있다면, 해당 이미지를 표시하기 위해서는 데이터셋에 이미지를 표시하기 위해 “컬럼2”라는 컬럼을 정의한 다음 해당 컬럼에 값을 “ID2, ID1”와 같이 넣어 주어야 화면에 이미지가 표시되게 됩니다. 위의 그림에서는 “삼송”에 해당하는 컬럼2의 값이 “ID1”이기 때문에 이미지 데이터셋의 “ID1”에 해당하는 이미지가 정상적으로 표시 되게 되며, “엘쥐”에 해당하는 “ID3”값이 이미지 데이터셋에 존재하지 않기 때문에 이미지가 나오지 않고 공백으로 나오게 됩니다.
* 이미지 사이즈는 17 \* 17 Pixel 로 맞춰주어야 합니다.

**1.3 화면 디자인**

* 첨부된 샘플중 Tree\_image\_4api를 기준으로 설명 하겠습니다.
* 첨부된 샘플과 같이 디자인을 해 줍니다. 사용한 컴포넌트는 메뉴, 트리뷰, 그리드, 데이터셋, 이미지 데이터셋 컴포넌트를 사용하였으며, 만들어진 샘플 화면은 아래와 같습니다.



* 트리에서 설정은 아래와 같이 데이터셋과 이미지 데이터셋을 연동하였으며, 이미지를 표시하기 아래와 같이 설정하고, “UseImage”속성의 값을 “true”로 설정을 하였습니다.
* 샘플의 데이터셋의 데이터는 CSV파일을 연동하였으며, 실제 업무에서는 DB에서 데이터를 조회하여 뿌려주는 서비스를 호출하는 형태로 구성하면 됩니다.
* 트리, 그리드에 표현될 이미지 데이터셋은 “/selectImage.jsp” 서비스를 호출하고 있으며, 메뉴 에 표현될 이미지 데이터셋은 “/menuImage.jsp” 서비스를 호출하고 있습니다. 세부적인 코딩은 아래 서비스 코드의 내용을 참고하시길 바랍니다.

**1.4 이미지 데이터셋을 만들기 위한 서비스 작성**

* 이미지 데이터셋에서 사용할 서비스를 아래와 같이 작성해 줍니다.
* 위의 화면에서 컬럼ID는 달라도 되지만, Type이 String, BLOB, Integer 3개는 반드시 존재하여야 합니다.

|  |
| --- |
| <%@ page import=*"java.io.\*,*  *java.util.\*,*  *java.text.\*,*  *com.gauce.\*,*  *com.gauce.io.\*,*  *com.gauce.http.\*"*  %><%  HttpGauceResponse gResponse = **null**;  GauceOutputStream gos = **null**;  File f = **null**;  FileInputStream fis = **null**;    **try** {  gResponse = ((HttpGauceResponse) response);  gos = gResponse.getGauceOutputStream();  GauceDataSet dSet = **new** GauceDataSet();  gos.fragment(dSet, 40);  // 이미지가 들어있는 경로(상대경로지정)  //String imgDir = request.getRealPath("/") + "/sample/tree\_img/img/";    // 이미지가 들어있는 경로(절대경로지정)  String imgDir = "D:\\workspace\\GAUCE40\\sample\\tree\_image\_4api\\image\\";    // 이미지 리스트 이미지파일명 지정  String[] imgList = **new** String[] {  "img1.gif",  "img2.gif",  "img3.gif",  };    // 이미지 ID(Key) 리스트 [외부 데이터셋의 컬럼값과 매칭되는 이미지 구분자]  ////////////////// 데이터셋 컴포넌트의 ID와 매칭되야 한다. /////////////////  String[] imgIdList = **new** String[] {  "ID1",  "ID2",  "ID3",  };    **for** ( **int** i=0; i<imgList.length; i++ ) {  f = **new** File(imgDir + imgList[i]);  fis = **new** FileInputStream(f);    dSet.put("img", fis, f.length()); // 이미지를 담을 컬럼에 FileInputStream을 넣어 준다.  dSet.put("img\_id", imgIdList[i], 100); // imgIdList에 넣은 image ID값을 데이터셋에 넣어준다.  dSet.put("img\_size", (**int**)f.length(), 100); // 이미지의 SIZE를 지정한다.  dSet.heap();  }  gos.write(dSet); //만든 데이터셋을 리턴한다.    } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {fis.close();} **catch**(Exception e) {}  **if** (gos != **null**) {  **try** {  gos.close();  } **catch**(IOException ioe) {  gos = **null**;  }  }  }  %> |

[트리, 그리드에 표현될 이미지 데이터셋을 구성하는 서비스 작성 예]

|  |
| --- |
| <%@ page import=*"java.io.\*,*  *java.util.\*,*  *java.text.\*,*  *com.gauce.\*,*  *com.gauce.io.\*,*  *com.gauce.http.\*"*  %><%  HttpGauceResponse gResponse = **null**;  GauceOutputStream gos = **null**;  File f = **null**;  FileInputStream fis = **null**;    **try** {  gResponse = ((HttpGauceResponse) response);  gos = gResponse.getGauceOutputStream();  GauceDataSet dSet = **new** GauceDataSet();  gos.fragment(dSet, 40);  // 이미지가 들어있는 경로(상대경로지정)  //String imgDir = request.getRealPath("/") + "/sample/tree\_img/img/";    // 이미지가 들어있는 경로(절대경로지정)  String imgDir = "D:\\workspace\\GAUCE40\\sample\\tree\_image\_4api\\image\\";    // 이미지 리스트 이미지파일명 지정  String[] imgList = **new** String[] {  "menuSecom.gif",  "menuSebus.gif",  "menuSetec.gif",  "menuSemntc.gif"  };    // 이미지 ID(Key) 리스트 [외부 데이터셋의 컬럼값과 매칭되는 이미지 구분자]  ////////////////// 데이터셋 컴포넌트의 ID와 매칭되야 한다. /////////////////  String[] imgIdList = **new** String[] {  "menuSecom",  "menuSebus",  "menuSetec",  "menuSemntc"  };    **for** ( **int** i=0; i<imgList.length; i++ ) {  f = **new** File(imgDir + imgList[i]);  fis = **new** FileInputStream(f);    dSet.put("img", fis, f.length()); // 이미지를 담을 컬럼에 FileInputStream을 넣어 준다.  dSet.put("img\_id", imgIdList[i], 100); // imgIdList에 넣은 image ID값을 데이터셋에 넣어준다.  dSet.put("img\_size", (**int**)f.length(), 100); // 이미지의 SIZE를 지정한다.  dSet.heap();  }  gos.write(dSet); //만든 데이터셋을 리턴한다.    } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {fis.close();} **catch**(Exception e) {}  **if** (gos != **null**) {  **try** {  gos.close();  } **catch**(IOException ioe) {  gos = **null**;  }  }  }  %> |
|  |

[메뉴에 표현될 이미지 데이터셋을 구성하는 서비스 작성 예]

**1.5 기타 참고 사항**

* 이미지 데이터셋은 VC++과 같은 언어에서 Resource Symbol과 유사하다고 보시면 됩니다. 이미지가 표시될 기준은 이미지 데이터셋이 아닌 일반 데이터가 들어있는 데이터셋의 특정 컬럼의 값과 이미지 데이터셋의 ID컬럼의 값이 매칭되어 이미지가 표시된다고 생각하시면 됩니다.
* 컴포넌트 매뉴얼에서 이미지 데이터셋 연동하는 소스를 살펴보면, CSV형태로 되어있는데 이것은 아래와 같은 과정을 거쳐 CSV형태로 만들게 됩니다.

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">  <HTML>  <HEAD>  <TITLE>이미지데이터셋의 내용을 CSV로 만들기</TITLE>  </HEAD>  <BODY>  <a href="http://localhost:8088/treeImage.bf?X-UIClient=G40">다운받기</a>  </BODY>  </HTML> |

* 위와 같이 HTML소스를 작성후에 해당 HTML을 브라우져에서 실행한 다음 해당 링크를 다른 이름으로 저장을 하여 csv로 저장을 하시면 됩니다.
* 이렇게 만들어진 CSV를 “/selectImage.jsp” 호출하던 부분에 사용하시면 되며, 이미지를 표시하기 위해 매번 서비스를 호출하는 것 보다 성능 향상에 도움이 됩니다.