Creating a News Data Application

Published 2014-10-27 | (Compatible with SDK 2.5,3.5,4.5,5.0,5.1 and 2011,2012,2013,2014 models)

> This tutorial describes creating a simple news data application under current service by Samsung Smart TV

Contents

The news application is made up of following parts:

List
Contents
Photo

The application receives data through XHR communication (from local directory) and displays a layout as displayed in the figure below. The remote control can be used to select the articles and change the layout.

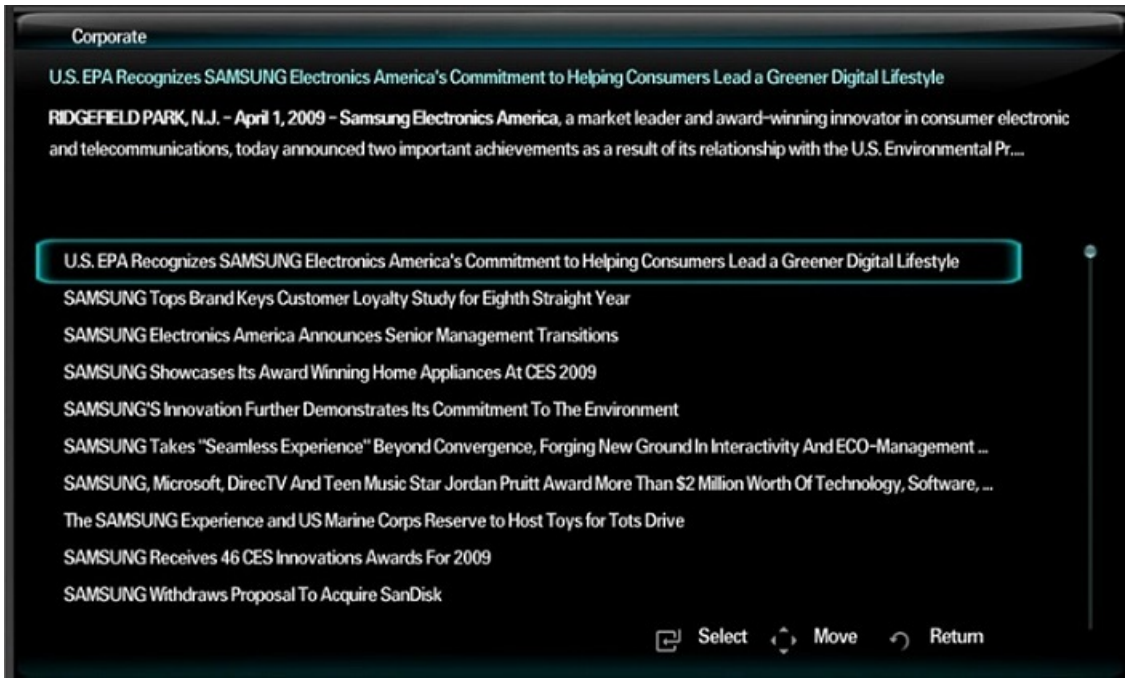

Figure. Layout

Figure. List area

Prerequisites

To create applications that run on a TV screen, you need:

Samsung TV connected to the Internet

SDK or a text editor for creating HTML, JavaScript and CSS files (using Samsung Smart TV SDK is recommended)

Development environment

Use Samsung Smart TV SDK to create the application. You can also use the emulator provided with the SDK to debug and test the application before uploading it in your TV. Later, you can run the application on a TV; see Testing Your Application on a TV. Note that applications may perform better on the TV than on the emulator.

Source Files

> Note
>
> The files needed for the sample application are **here**.

The directory structure for the tutorial application:

File/Directory	Description
Common/API	Contains the common modules provided by the Application Manager.
CSS	Contains the CSS files for the application.
Data	Contains the JavaScript files that receive, parse, and manage news data.
Language	Contains the JavaScript language files.
Main	Contains the JavaScript files for Main and Controller components.
Resource/image	Contains the image files for the application.
Resource/icon	Contains the image files for icons.
UI	Contains the JavaScript files for the view area.
Util	Contains the Util JavaScript files.

The news application design follows the model-view-controller (MVC) model. A news application is composed of three parts:
Model
receives news data
View
displays the received data
Controller
connects and controls the model and view and processes events from remote control buttons

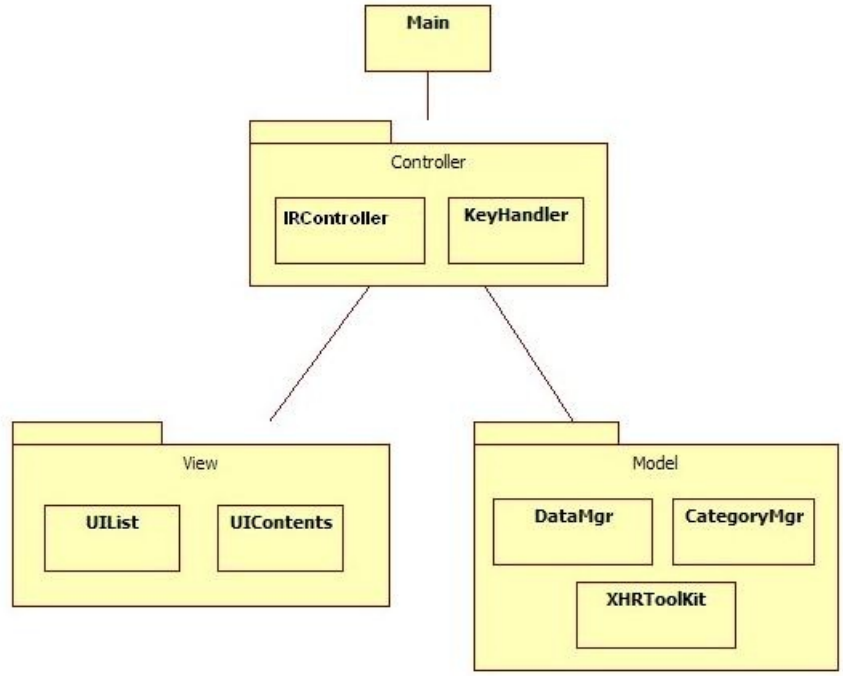

Figure. Package Diagram

Class Description

The classes and their description are as follows.

Class	Description
Main	Serves as the first access point of the news application.
NewsController	Provides overall control including handling events from keys on the remote control managing UI that is displayed on the TV screen, and making the model and view synchronized.

Class	Description
KeyHandler	Passes button inputs from the remote control to NewsController.
UIList	Fits data into the list area to create a proper layout.
UIContents	Fits data into the contents area to create a proper layout.
DataMgr	Manages the application data.
CategoryMgr	Manages data stored in individual categories.
XHRToolKit	Requests and receives data for a given URL through XHR communication.

Basic Behavior and Page Configuration

This task describes the steps to create a news application, access it, and use the remote control.

This tutorial task consists of the following parts:

Accessing the Application

1. Start the Samsung Smart TV SDK.
2. Click File > New project to create the project named ‘News’. A new project and config.xml file are created.
3. Add the following code to the config.xml file. For detailed information about individual tags, see Coding Your JavaScript Application.

```
<?xml version="1.0" encoding="UTF-8"?>
<widget>
  <previewjs>previewNews</previewjs>
  <cpname>User</cpname>
  <cplogo>Resource/image/news.png</cplogo>
  <prelcon>Resource/image/news.png</prelcon>
  <cpauthjs></cpauthjs>
  <ThumbIcon>Resource/image/news9.png</ThumbIcon>
  <BigThumbIcon>Resource/image/news9.png</BigThumbIcon>
  <ListIcon>Resource/image/news85x70.png</ListIcon>
  <BigListIcon>Resource/image/news95x78.png</BigListIcon>
  <category>Information</category>
  <autoUpdate>n</autoUpdate>
  <ver>0.930</ver>
  <mgrver>1.000</mgrver>
  <fullwidget>y</fullwidget>
  <srcctl>n</srcctl>
  <ticker>n</ticker>
  <childlock>n</childlock>
  <audiomute>n</audiomute>
  <videomute>n</videomute>
  <dcont>y</dcont>
  <type>user</type>
  <widgetname>News</widgetname>
  <description>News widget</description>
  <width>960</width>
  <height>540</height>
  <author>
    <name>Samsung SDS</name>
    <email></email>
    <link>http://acme-application.example.com</link>
    <organization>Acme Examples, Inc.</organization>
  </author>
</widget>
```

4. Create an index.html file.
5. Add the file to the SDK, and add the code given below. The common modules provided by the Application Manager are included as $MANAGER_WIDGET/Common/API/Plugin.js. For detailed information, see Using Common Modules.
6. Create the Main folder and create Main.js file in it.
7. Include the Main.js file. The HTML code includes the three common modules and the Main.js file.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>News widget Tutorial</title>
    <script type="text/javascript" language="javascript" src="$MANAGER_WIDGET/Common/API/Plugin.js"></scri
pt>
    <script type="text/javascript" language="javascript" src="$MANAGER_WIDGET/Common/API/Widget.js"></scr
ipt>
    <script type="text/javascript" language="javascript" src="$MANAGER_WIDGET/Common/API/TVKeyValue.js"
>
    </script>
    <script type="text/javascript" language="javascript" src="Main/Main.js"></script>
  </head>
  <body onload="Main.onLoad();" onunload="Main.onUnload();">
  </body>
</html>
```

8. The onload and onunload properties of the <body> tag are assigned to Main.onLoad() and Main.onUnload() functions, respectively. On accessing the application, the index.html file is loaded and the Main.onLoad() function assigned to onload is executed. Add code in the Main.js file as shown below.

```
var Main = {}

Main.onLoad = function () {
  alert("Main.onLoad()");
  this.analyzeVars();
  this.includeLanguage(function () {
    initNews();
  });
}

Main.onUnload = function () {
}
```

9. Start the SDK emulator. If the alert() : Main.onLoad() message is displayed in the log manager, the application has been launched.

Creating the Initial Page and CSS

1. Insert a div element in the index.html file and link it to CSS. If a ready message is sent to the Application Manager through the common module, the application is displayed on the screen.
2. Add code to use the common modules in Main, and create an NewsController object.
3. Create a UIList object to display the initial page.
4. Add the following code to the <body> tag in the index.html file.

```
<div id='UIList'></div>
```

5. Create the UIList.css file and add the following code.

```
#UIList {
  position: absolute;
  left: 0px;
  top: 0px;
  width: 960px;
  height: 540px;
  background-image: url(../Resource/image/sub2_bg_10.png);
  display: none;
}
```

6. Add the following code to include UIList.css file in <head> tag of index.html. CSS can be applied only after the appropriate files have been included in index.html file.

```
<link rel="stylesheet" type="text/css" href="CSS/UIList.css"/>
```

7. To display the image, create a UIList object that is in charge of the List page and display it on the screen using the NewsController object. Create the UIList.js file. Add the code given below. Include UIList.js in index.html.

```
var UIList = {
    listArea: null,          // UIList Div
    previewTitle: null,      // Preview title Div
    previewDescription: null,    // Preview description Div
    scrollBar: null,         // Scroll bar Div
    scrollBead: null,          // Scroll bead Div
    arrTitles: new Array(),    // Array of Title Divs
    SCROLLBAR_SIZE: 300,       // Scroll bar length
    titleIdx: 0,             // title index being highlighted
    TITLE_MAX_NUM: 10,         // max title number
    dataObj: null              // Data object
}
UIList.create = function () {
    var i = 0;
    this.listArea = document.getElementById("UIList");
    this.previewTitle = document.getElementById("UIListPreviewTitle");
    this.previewDescription = document.getElementById("UIListPreviewDescription");
    this.errorMessage = document.getElementById("UIListErrorMessage");
    this.scrollBar = document.getElementById("UIListScrollBar");
    this.scrollBead = document.getElementById("UIListScrollBead");

    for (i = 0; i < this.TITLE_MAX_NUM; i++) {
        this.arrTitles[i] = document.getElementById("UIListTitle" + i);
    }
}

UIList.show = function () {
    // alert("UIList.show()");
    this.listArea.style.display = "block";
    this.fillPreviewTitle();
    this.fillPreviewDescription();
    this.fillTitles();
    this.highlightTitle(this.titleIdx);
    KeyHandler.focusToList();
    // alert("UIList.show() End");
}
```

8. Create the NewsController.js file.

```
var NewsController = {
    // UI state define
    UILIST: 1,
    UICONTENTS: 2,
    UIPHOTOLIST: 3,
    UIPHOTOCONTENTS: 4
}

NewsController.create = function () {
    DataMgr.create();
    KeyHandler.create();
    UIList.create();
    UIContents.create();
}

NewsController.start = function (categoryID) {
    alert("NewsController.start(" + categoryID + ")");
    this.currentCategoryID = categoryID;
    DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
        NewsController.showList();
    });
}

NewsController.showList = function () {
    //  alert("NewsController.showList()");
    /*  the following are for help buttons*/
    //  hiding buttons in UI contents
    document.getElementById("BACK1").style.visibility = "hidden";
    document.getElementById("MOVE1").style.visibility = "hidden";
    // hiding text of UI contents
    document.getElementById("BACK1_Txt").style.visibility = "hidden";
    document.getElementById("MOVE1_Txt").style.visibility = "hidden";
    // show buttons in UI list
    document.getElementById("BACK").style.visibility = "visible";
    document.getElementById("MOVE").style.visibility = "visible";
    document.getElementById("SELECT").style.visibility = "visible";
    document.getElementById("BACK_Txt").innerHTML = BACK;
    document.getElementById("BACK_Txt").style.visibility = "visible";
    document.getElementById("MOVE_Txt").innerHTML = MOVE;
    document.getElementById("MOVE_Txt").style.visibility = "visible";
    document.getElementById("SELECT_Txt").innerHTML = NSELECT;
    document.getElementById("SELECT_Txt").style.visibility = "visible";
    this.currentState = this.UILIST;              // set current UI state
    UIList.setDataObj(DataMgr.getListData(this.currentCategoryID, this.currentArticleIdx));    // set data object
    UIList.setTitleIdx(this.currentArticleIdx%UIList.TITLE_MAX_NUM);    // set title index to be highlight
    UIList.adjustScrollBar(this.currentArticleIdx,
        DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length);
    UIList.show();
    //    alert("NewsController.showList() End");
}
```

The NewsController object calls the create() method of the UIList object to initialize UIList, and calls the show() method to display the UIList div on the screen.

9. Modify the Main object code to call the NewsController object.

```
var Main = {
    startCategoryID: 0,
    countryCode: -1,
    languageCode: -1,
},

widgetAPI = new Common.API.Widget(),
objMsgFromManager = {},
arrSupportLanguageList = ['en', 'ko', 'en-GB', 'bg', 'hr', 'cs', 'da', 'nl', 'et', 'fi', 'fr', 'de', 'el', 'hu',
                          'it', 'lv', 'lt', 'no', 'pl', 'pt-US', 'pt', 'ro', 'ru', 'sr', 'sk', 'es', 'sv', 'tr',
                          'zh-CN', 'zh-HK', 'zh-TW', 'th'];
Main.onLoad = function ()  {
    //alert("Main.onLoad()");
    this.analyzeVars();
    this.includeLanguage(function () {
        initNews();
    });
}

Main.onUnload = function () {
}
```

Main initializes the NewsController object, creates the application, and calls the sendReadyEvent() method to pass an event to the Application Manager. It calls start() to start the UI. A semi-transparent image is displayed on the left side of the screen.

Registering Key Handlers

This section explains making the application respond to events from the remote control. Pressing **Enter** on the List page, takes you to the Contents page. Pressing **Return** on the Contents page, takes you to the List page. When any button on the remote control is pressed, a keydown event occurs. If focus is given to <a> tag and a remote control button is pressed, <a> tag receives and handles the event.

To make the application respond to events from the remote control:

Add <a> tag to index.html and register key handler methods with the onkeydown property.

```
<a href="javascript:void(0);" id="List_Anchor" onkeydown="KeyHandler.mmKeyDown()"></a>
<a href="javascript:void(0);" id="Contents_Anchor" onkeydown="KeyHandler.contentsKeyDown()"></a>
<a href="javascript:void(0);" id="KeyBlocker_Anchor" onkeydown="KeyHandler.keyBlocker()"></a>
```

Add 3 anchors to <body> tag

- to respond to the key events when you are on the List page
- to respond to key events when you are on the Contents page
- to prevent key events

If a keydown event occurs, each anchor executes the KeyHandler method registered with the onkeydown property.

Create a KeyHandler object. The KeyHandler object sets anchors for each create() method and uses a switch statement to process the key events that occur. The unique value of each key is obtained from the TVKeyValue common module. Methods to give focus to each anchor are created.

> Note
>
> The applicationAPI.blockNavigation(event.keyCode) method prevents automatically returning to the Application Manager when the **Return** button is pressed. For more information about applicationAPI.blockNavigation(event.keyCode), see blockNavigation in Widget Object.

```
var KeyHandler = {
```

```
    tvKey: null,
    listAnchor: null,
    contentsAnchor: null,
    keyBlockerAnchor: null,
}

KeyHandler.create = function () {
    this.listAnchor = document.getElementById("List_Anchor");
    this.contentsAnchor = document.getElementById("Contents_Anchor");
    this.keyBlockerAnchor = document.getElementById("KeyBlocker_Anchor");
    this.tvKey = new Common.API.TVKeyValue();
}

KeyHandler.listKeyDown = function () {
    var listKeyCode = event.keyCode;
    alert("UIList KeyCode = " + listKeyCode);

    switch(listKeyCode) {
        case this.tvKey.KEY_LEFT:
            break;
        case this.tvKey.KEY_RIGHT:
            break;
        case this.tvKey.KEY_UP:
            break;
        case this.tvKey.KEY_DOWN:
            break;
        case this.tvKey.KEY_ENTER:
            NewsController.request(NewsController.LIST_START_CONTENTS);
            break;
        case this.tvKey.KEY_RETURN:
            break;
    }
}

KeyHandler.contentsKeyDown = function () {
    var contentsKeyCode = event.keyCode;
    alert("UIContents KeyCode = " + contentsKeyCode);
    switch(contentsKeyCode) {
        case this.tvKey.KEY_LEFT:
            break;
        case this.tvKey.KEY_RIGHT:
            break;
        case this.tvKey.KEY_UP:
            break;
        case this.tvKey.KEY_DOWN:
            break;
        case this.tvKey.KEY_GREEN:
            break;
        case this.tvKey.KEY_ENTER:
            break;
        case this.tvKey.KEY_RETURN:
            widgetAPI.blockNavigation(event);
            NewsController.request(NewsController.CONTENTS_RETURN);
            break;
    }
```

```
    }

    KeyHandler.keyBlocker = function () {
        var keyBlockerKeyCode = event.keyCode;
        alert("keyBlockerKeyCode = " + keyBlockerKeyCode);
    }

    KeyHandler.focusToList = function () {
        this.listAnchor.focus();
    }

    KeyHandler.focusToContents = function () {
        this.contentsAnchor.focus();
    }

    KeyHandler.focusToKeyBlocker = function () {
        this.keyBlockerAnchor.focus();
    }
```

Add Contents div, create the CSS and include it in the index.html file. The process is similar to that of adding a List div.

```
    <link rel="stylesheet" type="text/css" href="CSS/UIContents.css"/><div id="UIContents"></div>
    #UIContents {
        position: absolute;
        left: 0px;
        top: 0px;
        width: 641px;
        height: 540px;
        background-image: url(../Resource/image/sub2_bg_10.png);
        display: none;
    }
```

Create a UIContents object for displaying content on the Contents page. Before the UIContents object is used, call the create() method. To display the content on the screen, the object calls the show() method. To handle key events, it calls the focusToContents() method of KeyHandler. To hide content from the screen, it calls focusToKeyBlocker() that prevents receiving other key events.

```
var UIContents = {
    contentsArea: null    // UIContents Div
}

UIContents.create = function () {
    this.contentsArea = document.getElementById("UIContents");
    this.categoryTitle = document.getElementById("UIContentsCategoryTitle");
    this.articleTitle = document.getElementById("UIContentsTitle");
    this.articleDescription = document.getElementById("UIContentsDescription");
    this.scrollBar = document.getElementById("UIContentsScrollBar");
    this.scrollBead = document.getElementById("UIContentsScrollBead");
}

UIContents.show = function () {
    // alert("UIContents.show()");
    this.contentsArea.style.display = "block";
    this.fillCategoryTitle();
    this.fillArticleTitle();
    this.fillDescription();
    KeyHandler.focusToContents();
    // alert("UIContents.show() End");
}

UIContents.hide = function () {
    this.contentsArea.style.display = "none";
    KeyHandler.focusToKeyBlocker();
}
```

Add the following code to the UIList object to receive key events through the KeyHandler object.

```
var UIList = {
    listArea: null    // UIList Div
}

UIList.create = function () {
    var i;
    this.listArea = document.getElementById("UIList");
    this.previewTitle = document.getElementById("UIListPreviewTitle");
    this.previewDescription = document.getElementById("UIListPreviewDescription");
    this.errorMessage = document.getElementById("UIListErrorMessage");
    this.scrollBar = document.getElementById("UIListScrollBar");
    this.scrollBead = document.getElementById("UIListScrollBead");

    for (i = 0; i < this.TITLE_MAX_NUM; i++) {
        this.arrTitles[i] = document.getElementById("UIListTitle" + i);
    }
}

UIList.show = function () {
    //alert("UIList.show()");
    this.listArea.style.display = "block";
    this.fillPreviewTitle();
    this.fillPreviewDescription();
    this.fillTitles();
    this.highlightTitle(this.titleIdx);
    KeyHandler.focusToList();
    // alert("UIList.show() End");
}

UIList.hide = function () {
    this.listArea.style.display = "none";
    this.blurTitle(this.titleIdx);
    KeyHandler.focusToKeyBlocker();
    block remote controller key event
}

this.listArea.style.display = "none";
this.blurTitle(this.titleIdx);
```

The application is ready to make page transfer with button inputs. KeyHandler notifies NewsController of remote button inputs to handle those inputs.

To make the NewsController object handles page transfers, modify NewsController. To identify the UI object being displayed on the screen, define currentState variables and the state of each variable. Using the request() method, objects request NewsController to process the request. Add the task to be performed by NewsController to the work list. Add case statements for the request() method. Add the code needed to process the requests. If a remote key event occurs when the List page is being displayed, that is, when listAnchor of KeyHandler has focus, the listKeyDown() method of KeyHandler is executed. If **Enter** is pressed, KeyHandler executes the request method NewsController.LIST_START_CONTENTS of NewsController to make the page visible. The NewsController object calls the hide() method of the UIList object to hide the List page, and calls the show() method of the UIContents object to make the Contents page visible.

```
var NewsController = {
    currentState: 0,    // current UI state (List, Contents, PhotoList, PhotoContents)
    // UI state
    UILIST: 1,
    UICONTENTS: 2,
    UIPHOTOLIST: 3,
    UIPHOTOCONTENTS: 4
```

```
    UIPHOTOCONTENTS: 4,
    // Process
    LIST_START_CONTENTS: 109,
    CONTENTS_RETURN: 204,
}

NewsController.create = function () {
    DataMgr.create();
    KeyHandler.create();
    UIList.create();
    UIContents.create();
}

NewsController.start = function (categoryID) { //    alert("NewsController.start("+categoryID+")");
    this.currentCategoryID = categoryID;
    DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
        NewsController.showList();
    });
//    alert("NewsController.start(" + categoryID + ") End");
}

NewsController.showList = function () {
    //    alert("NewsController.showList()");
    /*  the following are for help buttons*/
    // hiding buttons in UI contents
    document.getElementById("BACK1").style.visibility = "hidden";
    document.getElementById("MOVE1").style.visibility = "hidden";
    // hiding text of UI contents
    document.getElementById("BACK1_Txt").style.visibility = "hidden";
    document.getElementById("MOVE1_Txt").style.visibility = "hidden";
    // show buttons in UI list
    document.getElementById("BACK").style.visibility = "visible";
    document.getElementById("MOVE").style.visibility = "visible";
    document.getElementById("SELECT").style.visibility = "visible";
    document.getElementById("BACK_Txt").innerHTML = BACK;
    document.getElementById("BACK_Txt").style.visibility = "visible";
    document.getElementById("MOVE_Txt").innerHTML = MOVE;
    document.getElementById("MOVE_Txt").style.visibility = "visible";
    document.getElementById("SELECT_Txt").innerHTML = NSELECT;
    document.getElementById("SELECT_Txt").style.visibility = "visible";
    this.currentState = this.UILIST;                // set current UI state
    UIList.setDataObj(DataMgr.getListData(this.currentCategoryID, this.currentArticleIdx));        // set data object
    UIList.setTitleIdx(this.currentArticleIdx%UIList.TITLE_MAX_NUM);    // set title index to be highlight
    UIList.adjustScrollBar(this.currentArticleIdx,
        DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length);
    UIList.show();
    //    alert("NewsController.showList() End");
}

NewsController.showContents = function () { //    alert("NewsController.showContents()");
    this.currentState = this.UICONTENTS;
    //hide buttons inUIList
    document.getElementById("BACK").style.visibility="hidden";
    document.getElementById("MOVE").style.visibility="hidden";
    document.getElementById("SELECT").style.visibility="hidden";
```

```
    //show buttons in UI contents
    document.getElementById("BACK1").style.visibility="visible";
    document.getElementById("MOVE1").style.visibility="visible";
    //hiding Text in UI list
    document.getElementById("BACK_Txt").style.visibility="hidden";
    document.getElementById("MOVE_Txt").style.visibility="hidden";
    document.getElementById("SELECT_Txt").style.visibility="hidden";
    //text with buttons inUI contents
    document.getElementById("BACK1").style.visibility="visible";
    document.getElementById("BACK1_Txt").innerHTML=BACK;
    document.getElementById("BACK1_Txt").style.visibility="visible";
    document.getElementById("MOVE1_Txt").innerHTML=MOVE;
    document.getElementById("MOVE1_Txt").style.visibility="visible";
    UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID, this.currentArticleIdx));
    setTimeout("UIContents.adjustScrollBar()", 0);
    UIContents.show();
    //    alert("NewsController.showContents() End");
}

NewsController.request = function (REQUEST, option) {
    var totalArticleCount = 0;
    //    alert("NewsController.request("+REQUEST+","+option+")");
    switch (REQUEST) {
        // UIList
        case this.LIST_CATEGORY_LEFT:
            this.currentArticleIdx = 0;
            if (--this.currentCategoryID < 0) {
                this.currentCategoryID = DataMgr.getCategoryList().length - 1;
            }
            if (DataMgr.getCategoryData(this.currentCategoryID)
            && DataMgr.getCategoryData(this.currentCategoryID).bReady) {
                this.changeCategory();
            } else {
                DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
                    NewsController.changeCategory();
                });
            }
            break;
        case this.LIST_CATEGORY_RIGHT:
            this.currentArticleIdx = 0;
            if (++this.currentCategoryID > DataMgr.getCategoryList().length - 1) {
                this.currentCategoryID = 0;
            }
            if (DataMgr.getCategoryData(this.currentCategoryID)
            && DataMgr.getCategoryData(this.currentCategoryID).bReady) {
                this.changeCategory();
            } else {
                DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
                    NewsController.changeCategory();
                });
            }
            break;
        case this.LIST_START_CONTENTS:
            UIList.hide();
            this.showContents();
```

```
        this.showContents();

        break;

    case this.LIST_ARTICLE_UP:
        totalArticleCount = DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length;
        if (--this.currentArticleIdx < 0) {
            this.currentArticleIdx = totalArticleCount - 1;
        }
        UIList.articleUp();
        break;
    case this.LIST_ARTICLE_DOWN :
        totalArticleCount = DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length;
        if (++this.currentArticleIdx > totalArticleCount - 1) {
            this.currentArticleIdx = 0;
        }
        UIList.articleDown();
        break;
    case this.LIST_FIRST_TITLE_UP:
        UIList.setDataObj(DataMgr.getListData(this.currentCategoryID, this.currentArticleIdx));
        UIList.setTitleIdx(UIList.getLastTitleIdx());
        UIList.adjustScrollBar(this.currentArticleIdx,
                DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length);
        UIList.refresh();
        break;
    case this.LIST_LAST_TITLE_DOWN:
        UIList.setDataObj(DataMgr.getListData(this.currentCategoryID, this.currentArticleIdx));
        UIList.setTitleIdx(this.currentArticleIdx%UIList.TITLE_MAX_NUM);
        UIList.adjustScrollBar(this.currentArticleIdx,
                DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length);
        UIList.refresh();
        break;
    // UIContents
    case this.CONTENTS_ARTICLE_LEFT:
        if (--this.currentArticleIdx < 0) {
            this.currentArticleIdx = DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length -1;
        }
        UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID, this.currentArticleIdx));
        setTimeout("UIContents.adjustScrollBar()", 0);
        UIContents.refresh();
        break;
    case this.CONTENTS_ARTICLE_RIGHT:
        ++this.currentArticleIdx;
        if (this.currentArticleIdx > DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length - 1) {
            this.currentArticleIdx = 0;
        }
        UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID, this.currentArticleIdx));
        setTimeout("UIContents.adjustScrollBar()", 0);
        UIContents.refresh();
        break;
    case this.CONTENTS_SCROLL_UP:
        setTimeout("UIContents.adjustScrollBar()", 0);
        break;
    case this.CONTENTS_SCROLL_DOWN:
        setTimeout("UIContents.adjustScrollBar()", 0);
```

```
            break;
        case this.CONTENTS_RETURN:
            UIContents.hide();
            this.showList();
            break;
        default:
            break;
    }
}
```

Start the application. Pressing **Enter** displays the Contents page. Pressing **Return** displays the List page.

Creating List Page Layout

The List page is divided into the following areas:

- Area for article preview
- Area with 10 news headlines
- Scrollbar area
- Navigation area for users
- Area for network errors

To design the List page layout:

1. Modify the UIList div tag as shown below.

```
<div id='UIList'>
    <div id='UIListPreview'>
        <div id='UIListPreviewTitle' class='title_style'></div>
        <div id='UIListPreviewDescription'></div>
    </div>
    <div id='UIListTitles'>
        <div id='UIListTitle0'></div>
        <div id='UIListTitle1'></div>
        <div id='UIListTitle2'></div>
        <div id='UIListTitle3'></div>
        <div id='UIListTitle4'></div>
        <div id='UIListTitle5'></div>
        <div id='UIListTitle6'></div>
        <div id='UIListTitle7'></div>
        <div id='UIListTitle8'></div>
        <div id='UIListTitle9'></div>
    </div>
    <div id='UIListErrorMessage'></div>
        <div id='UIListScrollBar'><div id="UIListScrollBead"></div>
    </div>
    <div id='UIListHelpArea' class='helpArea_style'></div>
</div>
```

2. Modify UIList.css as shown below.

```
#UIList {
    position: absolute;
    left: 0px;
    top: 0px;
    width: 960px;
    height: 540px;
    background-image: url(../Resource/image/sub2_bg_10.png);
    display: none;
}
```

```
#BACK {
    position: absolute;
    top: 495px;
    left:  750px;
    width: 28px;
    height: 28px;
    visibility: visible;
    z-index: 2;
}

#BACK_Txt{
    position: absolute;
    top: 495 px;
    left: 790 px;
    width: 50 px;
    color: #fff;
    height: 20px;
    font-size: 13pt;
    visibility: visible;
    z-index: 2;
}

#MOVE{
    position: absolute;
    top: 495px;
    left: 650px;
    width: 28px;
    height: 28px;
    visibility: visible;
    z-index: 2;
}

#MOVE_Txt{
    position: absolute;
    top: 495px;
    left: 690px;
    width: 50px;
    color: #fff;
    height: 20px;
    font-size: 13pt;
    visibility: visible;
    z-index: 2;
}

#SELECT{
    position: absolute;
    top: 495px;
    left: 550px;
    height: 28px;
    width: 28px;
    visibility: visible;
    z-index: 2;
```

```
    z-index: 2;
}

#SELECT_Txt{
    position: absolute;
    top: 495px;
    left: 590px;
    width: 50px;
    color: #fff;
    height: 20px;
    font-size: 13pt;
    visibility: visible;
    z-index: 2;
}

#UIListLogo {
    position: absolute;
    left: 229px;
    top: 2px;
    z-index: 2;
}

#UIListPreview {
    position: absolute;
    left: 0px;
    top: 6px;
    width: 960px;
    height: 168px;
    background-image: url(../Resource/image/nnavi_preview_s_bg.png);
}

#UIListPreviewTitle {
    position: absolute;
    left: 50px;
    top: 11px;
}

#UIListPreviewDescription {
    position: absolute;
    left: 30px;
    top: 42px;
    width: 880px;
    height: 110px;
    overflow: hidden;
    color: #fff;
    font-size: 16px;
    text-align: left;
    line-height: 22px;
}

#UIListTitles {
    position: absolute;
    left: 0px;
```

```
    top: 175px;
    width: 960px;
    height: 337px;
    background-image: url(../Resource/image/sub2_list_bg.png);
}

#UIListTitle0 {
    position: absolute;
    left: 0px;
    top: 9px;
    width: 960px;
    height: 46px;
}

#UIListTitle1 {
    position: absolute;
    left: 0px;
    top: 39px;
    width: 960px;
    height: 46px;
}

#UIListTitle2 {
    position: absolute;
    left: 0px;
    top: 69px;
    width: 960px;
    height: 46px;
}

#UIListTitle3 {
    position: absolute;
    left: 0px;
    top: 99px;
    width: 960px;
    height: 46px;
}

#UIListTitle4 {
    position: absolute;
    left: 0px;
    top: 129px;
    width: 960px;
    height: 46px;
}

#UIListTitle5 {
    position: absolute;
    left: 0px;
    top: 159px;
    width: 960px;
    height: 46px;
```

```
}

#UlListTitle6 {
    position: absolute;
    left: 0px;
    top: 189px;
    width: 960px;
    height: 46px;
}

#UlListTitle7 {
    position: absolute;
    left: 0px;
    top: 219px;
    width: 960px;
    height: 46px;
}

#UlListTitle8 {
    position: absolute;
    left: 0px;
    top: 249px;
    width: 960px;
    height: 46px;
}

#UlListTitle9 {
    position: absolute;
    left: 0px;
    top: 279px;
    width: 960px;
    height: 46px;
}

.UlListTitle_style {
    position: absolute;
    left: 34px;
    top: 0px;
    width: 815px;
    height: 46px;
    font-size: 16px;
    color: #fff;
    text-align: left;
    vertical-align:middle;
    overflow: hidden;
    text-overflow: ellipsis;
    white-space: nowrap;
}

#UlListErrorMessage {
    position: absolute;
    left:
```

```
    top: 275px;
    width: 258px;
    height: 100px;
    font-size: 16px;
    line-height: 22px;
    text-align: center;
    color: #fff;
}

#UIListScrollBar {
    position: absolute;
    left: 920px;
    top: 185px;
    width: 17px;
    height: 317px;
    background-image: url(../Resource/image/scroll_bg_sub2.png);
}

#UIListScrollBead {
    position: absolute;
    left: 0px;
    top: 0px;
    width: 17px;
    height: 17px;
    background-image: url(../Resource/image/scroll_sub2.png);
}

#UIListHelpArea {
    position: absolute;
    right: 15px;
    bottom: 3px;
    width: 310px;
    height: 28px;
}
```

Creating Contents Page Layout

The Contents page is divided into the following areas:

- Area displaying the content title
- Area displaying the article
- Scrollbar area
- Navigation area

To design the Contents page layout:

1. Modify the UIContents div and CSS as shown below.

```
<div id='UIContents'>
    <div id='UIContentsBG'></div>
    <div id='UIContentsCategoryTitle' class='title_style'></div>
    <div id='UIContentsTitle'></div>
    <div id='UIContentsTitleLeft'></div>
    <div id='UIContentsTitleRight'></div>
    <div id='UIContentsDescription' onkeydown='KeyHandler.contentsKeyDown()'></div>
    <div id='UIContentsScrollBar'>
        <div id="UIContentsScrollBead"></div>
    </div>
    <div id='UIContentsHelpArea' class='helpArea_style'></div>
</div>
#UIContents {
    position: absolute;
    left: 0px;
    top: 0px;
    width: 960px;
    height: 540px;
    background-image: url(../Resource/image/contents_bg.png);
    display: none;
}

#BACK1{
    position: absolute;
    top: 495px;
    left:  750px;
    width: 28px;
    height: 28px;
    visibility: visible;
    z-index: 2;
}

#BACK1_Txt{
    position: absolute;
    top: 495px;
    left: 790px;
    width: 50px;
    color: #fff;
    height: 20px;
    font-size: 13pt;
    visibility: visible;
    z-index: 2;
}

#MOVE1{
    position: absolute;
    top: 495px;
    left: 650px;
    width :28px;
    height :28px;
    visibility: visible;
    z-index: 2;
`
```

```
}

#MOVE1_Txt{
    position: absolute;
    top: 495px;
    left: 690px;
    width: 50px;
    color: #fff;
    height: 20px;
    font-size: 13pt;
    visibility: visible;
    z-index: 2;
}

#UIContentsBG{
    position: absolute;
    left: 0px;
    top: 0px;
    width: 960px;
    height: 540px;
    background-image: url(../Resource/image/contentImg.png);
}

#UIContentsCategoryTitle {
    position: absolute;
    left: 24px;
    top: 10px;
    z-index: 2;
}

#UIContentsLogo {
    position: absolute;
    left: 525px;
    top: 2px;
    z-index: 2;
}

#UIContentsTitle {
    position: absolute;
    left: 28px;
    top: 48px;
    width: 585px;
    height: 41px;
    background-image: url(../Resource/image/sub3_box_1.png);
}

.UIConentsTitle_style {
    position: absolute; left: 62px;
    top: 0px;
    width: 461px;
    height: 39px;
    font-size: 21px;
    color: #9ff;
```

```
    color: #fff;
    font-weight: bolder;
    text-align: center;
    vertical-align: middle;
    overflow: hidden;
    text-overflow: ellipsis;
    white-space: nowrap;
}

#UlContentsTitleLeft {
    position: absolute;
    left: 28px;
    top: 48px;
    width: 35px;
    height: 41px;
    background-image: url(../Resource/image/sub3_box_1_l.png);
}

#UlContentsTitleRight {
    position: absolute;
    left: 578px;
    top: 48px;
    width:35px;
    height: 41px;
    background-image: url(../Resource/image/sub3_box_1_r.png);
}

#UlContentsDescription {
    position: absolute;
    left: 31px;
    top: 101px;
    width: 569px;
    height: 390px;
    overflow: auto;
    text-align: left;
    font-size: 16px;
    color: #fff;
    line-height: 180%;
}

#UlContentsDescription a {
    text-decoration: none;
    overflow: hidden;
    text-align: left;
    font-size: 16px;
    color: #fff;
    line-height: 22px;
}

#UlContentsScrollBar {
    position: absolute;
    left: 612px;
    top: 96px;
```

```
    width: 17px;
    height: 401px;
    background-image: url(../Resource/image/scroll_bg_sub3.png);
}

UIContentsScrollBead {
    position: absolute;
    left: 0px;
    top: 0px;
    width: 17px;
    height: 17px;
    background-image: url(../Resource/image/scroll_sub2.png);
}

#UIContentsHelpArea {
    position: absolute;
    right: 15px;
    bottom: 3px;
    width: 500px;
    height: 28px;
}
```

Data Management

In this task, the tutorial examines how to receive, parse and store data to be displayed by the News application using XHR communication. Because of copyright issues, News data is provided in the form of XML files. The XML files are stored on a local drive and the News application accesses them through XHR communication. The process is the same as receiving data from the Internet using URLs, except that a local drive is used.

News data is provided in the form of XML files. Through XHR communication, the News application requests and receives data in a URL. It then parses the received data to put them in a structure to display on the screen using UI objects.

The application uses the following classes or objects to request, receive, parse and store News data.
The DataMgr object manages all data transmission procedures.
The DataMgr has multiple CategoryMgrs, and each CategoryMgr manages data in a category.
The XHRToolKit class creates an XHR object, and requests and receives data from URLs.

This tutorial task consists of the following parts:

Setting Up XHR Communication

1. Create a XHRToolKit class to perform communication for News data. The XHRToolKit class requests and receives News data, and passes a Boolean value indicating the success to the registered callback. Objects using an instance of this class get the received data using the getResponseXML() method.

   When a class instance is created, the XHRToolKit class receives a URL to be used to request data, and a callback function to pass the communication result.

   It requests data from the URL using the sendXHRRequest() method, and passes a Boolean value indicating the communication success status to the callback function.

   The sample source code given below calls the destroy() method of the XHR object. This code solves the problem of memory shortage in TV environments. For detailed information, see Example for XHR communication.
2. Create an XHRToolKit.js file and include it in the index.html file.

```
var XHRToolKit = function (pURL, pSendResultFunc) {
    var URL = pURL,
        sendResultFunc = pSendResultFunc,
        XHRObj = null;

        this.sendXHRRequest = function () {
            if (XHRObj) {
                XHRObj.destroy();
            }
            XHRObj = new XMLHttpRequest();
            if (XHRObj) {
                XHRObj.onreadystatechange = function () {
                    if (XHRObj.readyState == 4) {
                        receiveXHRResponse();
                    }
                };
                XHRObj.open("GET", URL, true);
                XHRObj.send(null);
            } else {
                alert("XHR Object is NULL");
            }
        }
        var receiveXHRResponse = function () {
            if (XHRObj.status == 200) {
                alert("Good XHR response"); sendResultFunc(true);
            } else {
                alert("Bad XHR response."); sendResultFunc(false);
            }

            this.getResponseXML = function () {
                return XHRObj.responseXML;
            }

            this.getXHRObj = function () {
                return XHRObj;
            }

            this.abortXHRObj = function () {
                if (XHRObj) {
                    XHRObj.abort();
                }
            }
        }
```

3. Add the code below in KEY_RIGHT case of the KeyHandler.listKeyDown() method. This makes communication happen on pressing the **right** button on the remote control in the List page.

```
test_URL = " XML/category1.xml";
test_XHRTool = new XHRToolKit(test_URL, function (result) {
    test_XHRToolKit(result);
});
test_XHRTool.sendXHRRequest();
```

4. Add the following code to the KeyHandler.listKeyDown() method. This displays the initial 300 words of the data received using a XHR object of test_XHRTool, using the alert() function.

```
function test_XHRToolKit(result) {
    alert("result : " + result);
    alert(test_XHRTool.getXHRObj().responseText.substr(0,300));
}
```

Parsing News data

1. Create a CategoryMgr class for parsing the received data and storing it in a structure.
2. Create a Structure.js file and add the code below.
3. Include it in the index.html file.

```
var Article = function (title, description) {
    this.title = title;
    this.description = description;
}
```

4. The CategoryMgr class has an XHRToolKit class instance as a member variable to receive News data. The received data is parsed in the onReceiveData() method and stored the arrArticles structure array. To create an instance, it receives as an argument an object with ID, title, URL and callback as properties. After parsing and storing the data, it notifies the appropriate category of the completion of data request and receipt by passing the ID to the callback function received as an argument.

```
var CategoryMgr = function (paramObject) {
    var ID = paramObject.ID    // Category ID
        title = paramObject.title,    // Category title
        URL = paramObject.URL,    // Category URL
        noticeCallback = paramObject.callback,    // Callback

        XHRTool = new XHRToolKit(URL, function (result) {
            onReceiveData(result);
        }),

        arrArticles = null,    // Article structure array
        bDataReady = false, // Data ready flag
        // Initializing
        initData = function () {
            arrArticles = new Array();
            bDataReady = false;
        },

        // Parses the data, saves it to a structure, and executes callback function
        onReceiveData = function (bSuccess) {
            var responseXML,
                responseDoc,
                itemElements,
                i,
                tTitle,
                tDescription

            alert("CategoryMgr.onReceiveData("+bSuccess+")");
            alert("Category ID : " + ID);
            if (bSuccess) {
                responseXML = XHRTool.getResponseXML();
                responseDoc = responseXML.documentElement;
                if (responseDoc) {
                    itemElements = responseDoc.getElementsByTagName("item");
                    if (itemElements && itemElements.length > 0) {
```

```
            // More than 1 article
            for (i = 0; i < itemElements.length; i++) {
              if (isValidNormalArticle(itemElements[i])) {
                tTitle = getElementData(itemElements[i], "title");
                tDescription = getElementData(itemElements[i], "description");
                arrArticles[arrArticles.length] = new Article(tTitle, tDescription);
              }
            }
            if (arrArticles.length > 0) {
              // If a valid article exists, set the flag 'true'
              bDataReady = true;
            }
          }
        }
      }
      noticeCallback(ID);
      alert("CategoryMgr.onReceiveData() End");
    },
    // Valid article format
    isValidNormalArticle = function (itemElement) {
      if (itemElement.getElementsByTagName("title")[0]
          && itemElement.getElementsByTagName("description")[0]) {
        return true;
      }
      return false;
    },
    getElementData = function (itemElement, element) {
      return itemElement.getElementsByTagName(element)[0].firstChild.data;
    };
    // Sends the XHR request
this.sendXHRRequest = function () {
    alert("Category " + ID + " XHR requested."); initData();
    XHRTool.sendXHRRequest();
}

// Gets the category ID
this.getID = function () {
    return ID;
}
// Gets the category title
this.getTitle = function () {
    return title;
}

// Gets the category URL
this.getURL = function () {
    return URL;
}

// Gets the category data
this.getArticles = function () {
    return arrArticles;
```

```
    }

    // Gets the category ready state
    this.isReady = function () {
        return bDataReady;
    }
}
```

The goal of this section is to create a CategoryMgr instance and make communication happen when the **left** button on the remote control is pressed on the List page.

5. Add the code below in the KEY_LEFT case of the KeyHandler.listKeyDown() method.

```
test_URL = "XML/category1.xml";
test_CategoryMgr = new CategoryMgr ({
    ID: 0,
    title: "Corporate",
    URL: test_URL,
    callback: test_CategoryMgr_Fn
})

test_CategoryMgr.sendXHRRequest();
```

6. Add the code shown below to display the category of the received News data, the number of news articles, and the title of the first article.

```
function test_CategoryMgr_Fn() {
    alert(test_CategoryMgr.getTitle());
    alert(test_CategoryMgr.getArticles().length);
    alert(test_CategoryMgr.getArticles()[0].title);
}
```

If the log manager is displayed when the **left** button on the remote control is pressed on the List page, the implementation of the function is successful.

Creating the Data Manager Object

1. Create a DataMgr object to manage all News data. This object synthesizes data in many categories and provides the data to other objects when needed.
2. The create() method must be called before using this object.
3. If there is a category for which a data request is to be made, call the sendRequest() method of this object.
4. The sendRequest() method receives category ID and a callback function that is executed after data receipt and processing.
5. Add the code below to the Structure.js file to store information on individual categories in structures.

```
var Category = function (title, URL) {
    this.title = title;
    this.URL = URL;
}
```

6. Create a Define.js file.
7. Add the code below to set information on an individual category.
8. Include the Define.js file in the index.html file.

```
var NEWS_CATEGORY1 = 0;
   NEWS_CATEGORY2 = 1,
   NEWS_CATEGORY3 = 2,

   URL_CATEGORY1 = "XML/category1.xml",
   URL_CATEGORY2 = "XML/category2.xml",
   URL_CATEGORY3 = "XML/category3.xml",

   LANG_NEWS_CATEGORY1 = "Corporate ",
   LANG_NEWS_CATEGORY2 = "Exhibition ",
   LANG_NEWS_CATEGORY3 = "Product ";
```

9. Create a DataMgr.js file and include it in the index.html file.

```
var DataMgr = {
   arrCategoryList: null,    // Category list
   arrCategoryManagers: null,    // Category manager
   arrCategoryData: null,    // All category data array
   noticeCallback: null
}

DataMgr.create = function () {
   var i = 0;
   alert("DataMgr.create()");

   this.arrCategoryList = new Array();
   this.arrCategoryList[0] = new Category(CORPORATE,URL_CATEGORY1);
   this.arrCategoryList[1] = new Category(EXHIBITION, URL_CATEGORY2);
   this.arrCategoryList[2] = new Category(PRODUCT,URL_CATEGORY3);
   this.arrCategoryManagers = new Array();
   this.arrCategoryData = new Array();

   for (i = 0; i < this.arrCategoryList.length; i++) {
      this.arrCategoryManagers[this.arrCategoryManagers.length] = new CategoryMgr({
         ID : i,
         title : this.arrCategoryList[i].title,
         URL : this.arrCategoryList[i].URL,
         callback : function (categoryID) {
            DataMgr.receiveHandler(categoryID);
         }
      });
   }

   alert(this.arrCategoryManagers.length + " Category manager created.");
   alert("DataMgr.create() End");
}

DataMgr.sendRequest = function (categoryID, callback) {
   alert("DataMgr.sendRequest("+categoryID+")");
   this.arrCategoryManagers[categoryID].sendXHRRequest();
   this.noticeCallback = callback;
   alert("DataMgr.sendRequest("+categoryID+") End");
}
```

```
DataMgr.receiveHandler = function (categoryID) {
    alert("DataMgr.receiveHandler("+categoryID+")");
    // Save data recieved from CategoryMgr and execute callback function.

    if (this.arrCategoryData[categoryID] == null) {
        this.arrCategoryData[categoryID] = [];
    }

    this.arrCategoryData[categoryID].title = this.arrCategoryManagers[categoryID].getTitle();
    this.arrCategoryData[categoryID].bReady = this.arrCategoryManagers[categoryID].isReady();
    this.arrCategoryData[categoryID].arrArticles = this.arrCategoryManagers[categoryID].getArticles();

    if (this.noticeCallback) {
        this.noticeCallback(categoryID);
    }
    alert("DataMgr.receiveHandler() End");
}

DataMgr.getCategoryData = function(categoryID) {
    alert("DataMgr.getCategoryData("+categoryID+")");
    return this.arrCategoryData[categoryID];
}
```

10. Add the code below in the KEY_UP case of the KeyHandler.listKeyDown() method.

```
DataMgr.create();
DataMgr.sendRequest(NEWS_ CATEGORY1, test_DataMgr_Fn);
```

11. The function below displays the category of the received News data, the number of articles and the title of the first article. Add this function.

```
function test_DataMgr_Fn(categoryID) {
    alert("categoryID: " + categoryID);
    alert(DataMgr.getCategoryData(NEWS_CATEGORY1).title);
    alert(DataMgr.getCategoryData(NEWS_CATEGORY1).bReady);
    alert(DataMgr.getCategoryData(NEWS_CATEGORY1).arrArticles.length);
}
```

If the log manager appears when the **up** button on the remote control on the List page, the function implementation is successful.

Data Display and Remote Control Keys

This section looks at displaying news data received from the Internet on the List and Content pages, and making the application respond to other key events.

This tutorial task consists of the following parts:

- Displaying Data on the List Page
- Giving a Highlight and Key Inputs on the List Page
- Displaying Data on the Contents Page

Displaying Data on the List Page

The data managed by the DataMgr object is displayed on the screen using the UIList object. The data needed for the List page are the category name and 10 news articles. The Category name, article title and some of the articles are displayed in the preview area of the List page. Below the preview area, 10 article titles are displayed.

To display the data:

1. Add the following method to the DataMgr object to request the data to be displayed.

```
DataMgr.getListData = function (categoryID, articleIdx) {
    var listDataObj = {},
        arrArticles,
        from,
        to;

    listDataObj.categoryTitle = this.arrCategoryData[categoryID].title;
    listDataObj.bReady = this.arrCategoryData[categoryID].bReady;
    if (listDataObj.bReady) {
        arrArticles = this.arrCategoryData[categoryID].arrArticles;
        from = this.getFirstArticleIdx(UIList.TITLE_MAX_NUM, articleIdx);
        to = parseInt(from) + parseInt(UIList.TITLE_MAX_NUM);
        listDataObj.arrArticles = arrArticles.slice(from, to);
    } else {
        listDataObj.arrArticles = [];
    }
    return listDataObj;
}

DataMgr.getPageNum = function (MAX_LIST_NUM, articleIdx) {
    var retValue = Math.floor(articleIdx / MAX_LIST_NUM);
    alert("DataMgr.getPageNum() returns " + retValue);
    return retValue;
}

DataMgr.getFirstArticleIdx = function (MAX_LIST_NUM, articleIdx) {
    var retValue = MAX_LIST_NUM * this.getPageNum(MAX_LIST_NUM, articleIdx);
    alert("DataMgr.getFirstArticleIdx() returns " + retValue);
    return retValue;
}
```

2. Modify the UIList object and display the selected data.

```
var UIList = {
    listArea: null,            // UIList Div
    previewTitle: null,        // Preview title Div
    previewDescription: null,  // Preview description Div
    scrollBar: null,           // Scroll bar Div
    scrollBead: null,          // Scroll bead Div
    arrTitles: new Array(),    // Array of Title Divs
    SCROLLBAR_SIZE: 300,       // Scroll bar length
    titleIdx: 0,               // Title index being highlighted
    TITLE_MAX_NUM: 10,         // Max number of titles
    dataObj: null              // Data object
}

UIList.create = function () {
    var i = 0;
    this.listArea = document.getElementById("UIList");
    this.previewTitle = document.getElementById("UIListPreviewTitle");
    this.previewDescription = document.getElementById("UIListPreviewDescription");
    this.errorMessage = document.getElementById("UIListErrorMessage");
    this.scrollBar = document.getElementById("UIListScrollBar");
    this.scrollBead = document.getElementById("UIListScrollBead");
```

```
    for (i = 0; i < this.TITLE_MAX_NUM; i++) {
        this.arrTitles[i] = document.getElementById("UIListTitle"+i);
    }
}

UIList.show = function () {
    alert("UIList.show()");
    this.listArea.style.display = "block";
    this.fillPreviewTitle();
    this.fillPreviewDescription();
    this.fillTitles();
    KeyHandler.focusToList();
    alert("UIList.show() End");
}

UIList.hide = function () {
    this.listArea.style.display = "none";
    KeyHandler.focusToKeyBlocker();
}

UIList.setDataObj = function(pDataObj) {
    alert("UIList.setDataObj()");
    alert("CategoryTitle : " + pDataObj.categoryTitle);
    this.dataObj = pDataObj; alert("UIList.setDataObj() End");
}

UIList.fillPreviewTitle = function () {
    alert("UIList.fillPreviewTitle()");
    this.previewTitle.innerHTML = this.dataObj.categoryTitle;
    alert("UIList.fillPreviewTitle() End");
}

UIList.fillPreviewDescription = function () {
    alert("UIList.fillPreviewDescription()");
    var description = '<div style="width: 287px; height:32px; font-size: 16px; text-align: left; ' +
        'color: #99FFFF; overflow: hidden; white-space: nowrap; text-overflow: ellipsis;">';
    description += this.dataObj.arrArticles[this.titleIdx].title;
    description += '</div>';
    description += '<div style="width: 287px; height: 86px; font-size: 16px; text-align: left; ' +
        'line-height: 160%; color: #FFFFFF; overflow: hidden; text-overflow ellipsis;">';
    description += this.dataObj.arrArticles[this.titleIdx].description
                    .replace(/<img [^>]+>/g, "").replace(/<br>/g, " ").substr(0,100);
    description += "</div>";
    this.previewDescription.innerHTML = description;
    alert("UIList.fillPreviewDescription() End");
}

UIList.fillTitles = function () {
    alert("UIList.fillTitles()");
    var article = null,
        i;
    for (i = 0; i < this.TITLE_MAX_NUM; i++) {
```

```
        article = this.dataObj.arrArticles[i];
        if (article) {
          this.arrTitles[i].innerHTML = Util.wrapInTable(article.title, "", "UIListTitle_style");
        } else {
          this.arrTitles[i].innerHTML = "";
        }
      }
      alert("UIList.fillTitles() End");
    }
```

3. Use the wrapInTable() method to set data in ten title divs. This is to display multiple languages in appropriate ways. The vertical-align property of this displays words with different font sizes in the vertical center.
4. Add a Util object and create a method for it.

```
var Util = {}
Util.wrapInTable = function (pStrContents, pStyle, pClass) {
  var retValue = "",
    strStyle = "";
    strClass = "";

  if (pStyle) {
    strStyle = " style='" + pStyle + "' ";
  }
  if (pClass) {
    strClass = " class='" + pClass + "' ";
  }
  retValue += "<table cellpadding='0px' cellspacing='0px'>";
  retValue += "</tr>";

  retValue +="<td" + strStyle + strClass + ">";
  retValue += "<nobr>";
  retValue += pStrContents;
  retValue += "</nobr>";
  retValue += "</td>";
  retValue += "</tr>";
  retValue += "</table>";

  alert("Util.wrapInTable() returns [" + retValue + "]");
  return retValue;
}
```

5. Modify the NewsController object. The create() method of NewsController calls the create() method of DataMgr to initialize the DataMgr object before using it.
6. Add code for setting the data to the showList() method. The NewsController object receives data from the DataMgr object, and displays the List page displayed using the UIList object.

```
var NewsController  = {
  currentState: 0,    // current UI state (UILIST, UICONTENTS, UIPHOTOLIST, UIPHOTOCONTENTS)
  currentCategoryID: 0,
  currentArticleIdx: 0,

  // UI state
  UILIST: 1,
  UICONTENTS: 2,
  UIPHOTOLIST: 3,
  UIPHOTOCONTENTS: 4,
```

```
    // Process
    LIST_START_CONTENTS: 109,
    CONTENTS_RETURN: 204,
}

NewsController.create = function () {
    DataMgr.create();
    KeyHandler.create();
    UIList.create();
    UIContents.create();
}

NewsController.start = function(categoryID) {
    alert("NewsController.start("+categoryID+")");
    this.currentCategoryID = categoryID;
    DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
        NewsController.showList();
    });
    alert("NewsController.start(" + categoryID + ") End");
}

NewsController.showList = function () {
    alert("NewsController.showList()");

    /*  the following are for help buttons*/
    // hiding buttons in UI contents
    document.getElementById("BACK1").style.visibility = "hidden";
    document.getElementById("MOVE1").style.visibility = "hidden";
    // hiding text of UI contents
    document.getElementById("BACK1_Txt").style.visibility = "hidden";
    document.getElementById("MOVE1_Txt").style.visibility = "hidden";
    // show buttons in UI list
    document.getElementById("BACK").style.visibility = "visible";
    document.getElementById("MOVE").style.visibility = "visible";
    document.getElementById("SELECT").style.visibility = "visible";
    document.getElementById("BACK_Txt").innerHTML = BACK;
    document.getElementById("BACK_Txt").style.visibility = "visible";
    document.getElementById("MOVE_Txt").innerHTML = MOVE;
    document.getElementById("MOVE_Txt").style.visibility = "visible";
    document.getElementById("SELECT_Txt").innerHTML = NSELECT;
    document.getElementById("SELECT_Txt").style.visibility = "visible";
    this.currentState = this.UILIST;              // set current UI state
    UIList.setDataObj(DataMgr.getListData(this.currentCategoryID, this.currentArticleIdx));       // set data object
    UIList.setTitleIdx(this.currentArticleIdx%UIList.TITLE_MAX_NUM);    // set title index to be highlight
    UIList.adjustScrollBar(this.currentArticleIdx,
        DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length);
    UIList.show();
    //    alert("NewsController.showList() End");
}

NewsController.showContents = function () {
NewsController.showContents = function () {
```

```
    //    alert("NewsController.showContents()");
    this.currentState = this.UICONTENTS;
    //hide buttons inUIList
    document.getElementById("BACK").style.visibility = "hidden";
    document.getElementById("MOVE").style.visibility = "hidden";
    document.getElementById("SELECT").style.visibility = "hidden";
    //show buttons in UI contents
    document.getElementById("BACK1").style.visibility = "visible";
    document.getElementById("MOVE1").style.visibility = "visible";
     //hiding Text in UI list
    document.getElementById("BACK_Txt").style.visibility = "hidden";
    document.getElementById("MOVE_Txt").style.visibility = "hidden";
    document.getElementById("SELECT_Txt").style.visibility = "hidden";
    //text with buttons inUI contents
    document.getElementById("BACK1").style.visibility = "visible";
    document.getElementById("BACK1_Txt").innerHTML = BACK;
    document.getElementById("BACK1_Txt").style.visibility = "visible";
    document.getElementById("MOVE1_Txt").innerHTML = MOVE;
    document.getElementById("MOVE1_Txt").style.visibility = "visible";

    UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID, this.currentArticleIdx));
    setTimeout("UIContents.adjustScrollBar()", 0);
    UIContents.show();
    //    alert("NewsController.showContents() End");
}

NewsController.request = function (REQUEST, option) {
    alert("NewsController.request("+REQUEST+","+option+")");
    switch (REQUEST) {
    // UIList
        case this.LIST_START_CONTENTS: UIList.hide(); this.showContents();
            break;
        // UIContents
        case this.CONTENTS_RETURN: UIContents.hide(); this.showList();
            break;
        default:
            break;
    }
}
```

7. Start the application to see the data received from the Internet displayed on the screen.

Giving a Highlight and Key Inputs on the List Page

On the List page, 10 article titles are displayed. If a title is selected by the user, it gets surrounded by a green box, which means that the title is highlighted. The highlight moves to the previous or next article title when the **up** or **down** button is pressed on the remote control.

To highlight a title:

1. Create a titleIdx variable in which the location of the highlighted title is stored to identify the selected article.
2. Create the function to highlight a title. The methods below receive the location of a specific title as an argument and change the background image of Div of the title. If no arguments are received, the title indicated by the titleIdx variable of the UIList object is highlighted.

```
UIList.highlightTitle = function (pIndex) {
    alert("UIList.highlightTitle("+pIndex+")");
    var index = (pIndex != null ? pIndex : this.titleIdx);
    this.arrTitles[index].style.backgroundImage = "url(Resource/image/highlight_sub2_list.png)";

    alert("UIList.highlightTitle("+pIndex+") End");
}

UIList.blurTitle = function (pIndex) {
    alert("UIList.blurTitle("+pIndex+")");
    var index = (pIndex != null ? pIndex : this.titleIdx);
    this.arrTitles[index].style.backgroundImage = "url(none)";
    alert("UIList.blurTitle("+pIndex+") End");
}
```

3. Call the highlightTitle() method in the show() method of the UIList object.

```
UIList.show = function () {
    alert("UIList.show()");
    this.listArea.style.display = "block";
    this.fillPreviewTitle();
    this.fillPreviewDescription();
    this.fillTitles();
    this.highlightTitle(this.titleIdx);
    KeyHandler.focusToList();
    alert("UIList.show() End");
}
```

4. When the application starts, the title number '0' designated as titleIdx is highlighted. Move the highlight by pressing remote control buttons. The KeyHandler object receives remote control key events, and notifies the NewsController object. The NewsController calls the appropriate functions. If the **down** button is pressed on the List page, the KeyHandler object requests NewsController to perform the LIST_ARTICLE_DOWN action. Add the code below in the KEY_DOWN case of the listKeyDown() method of KeyHandler.

```
NewsController.request(NewsController.LIST_ARTICLE_DOWN);
```

5. The NewsController object processes the KeyHandler object requests. Add LIST_ARTICLE_DOWN to the request list of NewsController, and add a case to the request() method.

```
var NewsController  = {
    currentState: 0, // current UI state (UILIST, UICONTENTS, UIPHOTOLIST, UIPHOTOCONTENTS)
    currentCategoryID: 0,
    currentArticleIdx: 0,
    // UI state
    UILIST: 1,
    UICONTENTS: 2,
    UIPHOTOLIST: 3,
    UIPHOTOCONTENTS: 4,
    // Process
    LIST_START_CONTENTS: 109,
    LIST_ARTICLE_DOWN: 103,
    CONTENTS_RETURN: 204,
}
NewsController.request = function (REQUEST, option) {
    //    alert("NewsController.request("+REQUEST+","+option+")");
    switch (REQUEST) {
        // UIList
        case this.LIST_CATEGORY_LEFT:
```

```
        this.currentArticleIdx = 0;
        if (--this.currentCategoryID < 0) {
            this.currentCategoryID = DataMgr.getCategoryList().length - 1;
        }
        if (DataMgr.getCategoryData(this.currentCategoryID)
        && DataMgr.getCategoryData(this.currentCategoryID).bReady) {
            this.changeCategory();
        }
        else {
            DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
                NewsController.changeCategory();
            });
        }
        break;
    case this.LIST_CATEGORY_RIGHT:
        this.currentArticleIdx = 0;
        if (++this.currentCategoryID > DataMgr.getCategoryList().length - 1) {
            this.currentCategoryID = 0;
        }
        if (DataMgr.getCategoryData(this.currentCategoryID)
        && DataMgr.getCategoryData(this.currentCategoryID).bReady) {
            this.changeCategory();
        }
        else {
            DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
                NewsController.changeCategory();
            });
        }
        break;
    case this.LIST_START_CONTENTS:
        UIList.hide();
        this.showContents();
        break;
    case this.LIST_ARTICLE_UP:
        var totalArticleCount = DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length;
        if (--this.currentArticleIdx < 0) {
            this.currentArticleIdx = totalArticleCount - 1;
        }
        UIList.articleUp();
        break;
    case this.LIST_ARTICLE_DOWN :
        var totalArticleCount = DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length;
        if (++this.currentArticleIdx > totalArticleCount - 1) {
            this.currentArticleIdx = 0;
        }
        UIList.articleDown();
        break;
    case this.LIST_FIRST_TITLE_UP:
        UIList.setDataObj(DataMgr.getListData(this.currentCategoryID, this.currentArticleIdx));
        UIList.setTitleIdx(UIList.getLastTitleIdx());
        UIList.adjustScrollBar(this.currentArticleIdx,
            DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length);
        UIList.refresh();
```

```
            UIList.refresh();
            break;
        case this.LIST_LAST_TITLE_DOWN:
            UIList.setDataObj(DataMgr.getListData(this.currentCategoryID, this.currentArticleIdx));
            UIList.setTitleIdx(this.currentArticleIdx%UIList.TITLE_MAX_NUM);
            UIList.adjustScrollBar(this.currentArticleIdx,
                DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length);
            UIList.refresh();
            break;
        // UIContents
        case this.CONTENTS_ARTICLE_LEFT:
            if (--this.currentArticleIdx < 0) {
                this.currentArticleIdx = DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length -1;
            }
            UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID, this.currentArticleIdx));
            setTimeout("UIContents.adjustScrollBar()", 0);
            UIContents.refresh();
            break;
        case this.CONTENTS_ARTICLE_RIGHT:
            if (++this.currentArticleIdx >
                DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length - 1) {
                this.currentArticleIdx = 0;
            }
            UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID, this.currentArticleIdx));
            setTimeout("UIContents.adjustScrollBar()", 0);
            UIContents.refresh();
            break;
        case this.CONTENTS_SCROLL_UP:
            setTimeout("UIContents.adjustScrollBar()", 0);
            break;
        case this.CONTENTS_SCROLL_DOWN:
            setTimeout("UIContents.adjustScrollBar()", 0);
            break;
        case this.CONTENTS_RETURN:
            UIContents.hide();
            this.showList();
            break;
        default:
            break;
    }
}
```

6. The UIList object removes the highlight from the currently highlighted news title and gives it to another title. Add a method to UIList, and call the method from the LIST_ARTICLE_DOWN case of the NewsController object.

```
UIList.articleDown = function () {
    alert("UIList.articleDown()");
    this.blurTitle(this.titleIdx);
    ++this.titleIdx;
    this.highlightTitle(this.titleIdx);
    alert("UIList.articleDown() End");
}
```

Pressing the **down** button on the remote control moves the highlight across other titles. The top of the List page displays some of the news articles being highlighted. When the highlight moves to other titles, the article currently displayed on the List page must be changed. The NewsController object is responsible for the response when the **down** key is pressed in

the last title, and for deciding the number of articles before the article with the highlighted title. Assume there are 15 news articles in a category. The UIList object shows articles 1 to 10. If the **down** is key pressed when article 10 is highlighted, the articles 11 to 15 are displayed. If the **down** key is pressed when article 15 is highlighted, the titles of article 1 to 10 are displayed on the screen. In order to change the titles currently displayed, dataObj used by the UIList object has to be modified by NewsController.

7. Add the code notifying NewsController to modify the dataObj when the **down** key is pressed.

```
UIList.articleDown = function () {
    alert("UIList.articleDown()");
    this.blurTitle(this.titleIdx);
    if (++this.titleIdx > this.TITLE_MAX_NUM - 1 || this.titleIdx > this.dataObj.arrArticles.length - 1) {
        // If this is the last title, request to update data object
        NewsController.request(NewsController.LIST_LAST_TITLE_DOWN);
    } else {
        this.highlightTitle(this.titleIdx);
        this.fillPreviewDescription();
    }
    alert("UIList.articleDown() End");
}
```

8. Add the code below in the LIST_ARTICLE_DOWN case for the NewsController object to identify the location of article currently displayed in the news data category.

```
case this.LIST_ARTICLE_DOWN :
    var totalArticleCount = DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length;
    if (++this.currentArticleIdx > totalArticleCount - 1) {
        this.currentArticleIdx = 0;
    }
    UIList.articleDown();
    break;
```

9. Add LIST_LAST_TITLE_DOWN to the NewsController list.
10. Add a case to the Request() method. The UIList object must replace the currently used data when receiving a request and give a highlight to the first article title.
11. Refresh all data on the screen.

```
case this.LIST_LAST_TITLE_DOWN:
    UIList.setDataObj(DataMgr.getListData(this.currentCategoryID, this.currentArticleIdx));
    UIList.setTitleIdx(this.currentArticleIdx%UIList.TITLE_MAX_NUM);
    UIList.refresh();
    break;
```

12. Add the following function to the UIList object.

```
UIList.refresh = function () {
    alert("UIList.refresh()");
    this.clearContents();    // Clears all the contents in Divs
    this.fillPreviewTitle();
    this.fillPreviewDescription();
    this.fillTitles();
    this.highlightTitle(this.titleIdx);
    alert("UIList.refresh() End");
}

UIList.setTitleIdx = function (titleIdx) {
    this.titleIdx = titleIdx;
}

UIList.clearContents = function () {
    alert("UIList.clearContents()");
    this.previewTitle.innerHTML = ""; this.previewDescription.innerHTML = "";
    for (var i=0; i < this.TITLE_MAX_NUM; i++) {
        this.arrTitles[i].innerHTML = "";
    }
    alert("UIList.clearContents() End");
}
```

The tasks needed for the **down** key are now completed. When the highlight is moved across the titles, the news article displayed at the top of the page changes. If the **down** key is pressed when the last title is highlighted, the next news article is displayed.

Displaying Data on the Contents Page

News article content is displayed using the UIList objects. Pressing the **Enter** key on the List page, displays the Contents page. The content and title of the News article receiving the highlight when the key is pressed are displayed on the screen.

To display data on the Contents page:

1. Add a method to the DataMgr object requesting the data used by the UIList object. This method receives the category ID and index of the article, and has category name and the article as properties of its return values.

```
DataMgr.getContentsData = function (categoryID, articleIdx) {
    var contentsDataObj = {};
    contentsDataObj.categoryTitle = this.arrCategoryData[categoryID].title;
    contentsDataObj.article = this.arrCategoryData[categoryID].arrArticles[articleIdx];
    return contentsDataObj;
}
```

2. Add the code below to the UIContents object to display data on the Contents page.
3. Create a method to set a data object to be used by UIContents, and display the set data on the screen.

```
var UIContents = {
    contentsArea: null,        // UIContents Div
    categoryTitle: null,       // Category title Div
    articleTitle: null,        // Article title Div
    articleDescription: null,  // Article description Div
    scrollBar: null,           // Scroll bar Div
    scrollBead: null,          // Scroll bead Div
    SCROLLBAR_SIZE: 384,       // scroll bar length
    dataObj: null,             // Data object
}
UIContents.create = function () {
    this.contentsArea = document.getElementById("UIContents");
```

```
    this.contentsArea = document.getElementById("UIContents");
    this.categoryTitle = document.getElementById("UIContentsCategoryTitle");
    this.articleTitle = document.getElementById("UIContentsTitle");
    this.articleDescription = document.getElementById("UIContentsDescription");
    this.scrollBar = document.getElementById("UIContentsScrollBar");
    this.scrollBead = document.getElementById("UIContentsScrollBead");
}
UIContents.show = function () {
    alert("UIContents.show()");
    this.contentsArea.style.display = "block";
    this.fillCategoryTitle();
    this.fillArticleTitle();
    this.fillDescription();
    KeyHandler.focusToContents();
    alert("UIContents.show() End");
}
UIContents.hide = function () {
    this.contentsArea.style.display = "none";
    KeyHandler.focusToKeyBlocker();
}
UIContents.setDataObj = function (pDataObj) {
    alert("UIContents.setDataObj()");
    this.dataObj = pDataObj;
    alert("UIContents.setDataObj() End");
}
UIContents.fillCategoryTitle = function () {
    alert("UIContents.fillCategoryTitle()");
    this.categoryTitle.innerHTML = this.dataObj.categoryTitle;
    alert("UIContents.fillCategoryTitle() End");
}
UIContents.fillArticleTitle = function () {
    alert("UIContents.fillArticleTitle()");
    this.articleTitle.innerHTML = Util.wrapInTable(this.dataObj.article.title, "", "UIConentsTitle_style");
    alert("UIContents.fillArticleTitle() End");
}
UIContents.fillDescription = function () {
    alert("UIContents.fillDescription()");
    this.articleDescription.innerHTML = this.dataObj.article.description +
        "<a id='UIContentsDescriptionAnchor' href=javascript:> </a>";
    KeyHandler.focusToContents();
    alert("UIContents.fillDescription() End");
}
```

4. Modify the showContents() method of NewsController. Add code that sets the data before the Contents page is displayed.

```
NewsController .showContents = function () {
    alert("NewsController .showContents()");
    this.currentState = this.UICONTENTS;
    UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID, this.currentArticleIdx));
    UIContents.show();
    alert("NewsController .showContents() End");
}
```

5. Start the application. Pressing the **Enter** key on the List page displays the article with the highlight on the Contents page. Scroll up/down to view all the News content. For this, there must be an object having focus and receiving key events for scrolling up/down.

6. Each article has an anchor tag. See the fillDescription() method created before. Modify the focusToContents() method to give focus to the added anchor. This makes the scrollbar move. For more information on scrolling up/down, see Adding a Scroll Bar.

```
KeyHandler.focusToContents = function () {
// this.contentsAnchor.focus();
    document.getElementById("UIContentsDescriptionAnchor").focus();
}
```

Extended Function

This task describes changing a category or news article currently displayed on the screen using remote control key events, and creating and using a scrollbar.

Changing a Category

The listKeyDown() method of the KeyHandler object, requests the NewsController to assign LIST_CATEGORY_LEFT and LIST_CATEGORY_RIGHT for KEY_LEFT and KEY_RIGHT events, respectively.

It adds the foregoing task to its list and adds a case to the request() method.

The NewsController checks whether data exists. If there is no data, it requests data using DataMgr. The changeCategory() method is called to refresh the List page.

```
case this.LIST_CATEGORY_LEFT:
    this.currentArticleIdx = 0;
    if (--this.currentCategoryID < 0) {
        this.currentCategoryID = DataMgr.getCategoryList().length - 1;
    }
    if (DataMgr.getCategoryData(this.currentCategoryID)
            && DataMgr.getCategoryData(this.currentCategoryID).bReady) {
        this.changeCategory();
    }
    else {
        DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
            NewsController .changeCategory();
        });
    }
    break;

case this.LIST_CATEGORY_RIGHT:
    this.currentArticleIdx = 0;
    if (++this.currentCategoryID > DataMgr.getCategoryList().length - 1) {
        this.currentCategoryID = 0;
    }
    if (DataMgr.getCategoryData(this.currentCategoryID)
            && DataMgr.getCategoryData(this.currentCategoryID).bReady) {
        this.changeCategory();
    } else {
        DataMgr.sendRequest(this.currentCategoryID, function (categoryID) {
            NewsController .changeCategory();
        });
    }
break;
```

Add a changeCategory() method.

```
NewsController .changeCategory = function () {
    alert("NewsController .changeCategory()");
    alert("Current category ID : " + this.currentCategoryID);
    alert("CurrentState : " + this.currentState);
    UIList.blurTitle(UIList.getTitleIdx());
    this.refreshList();
    alert("NewsController .changeCategory() End");
}
```

The changed categories respond to the **left** and **right** key events on the List page.

Changing News Articles

The news articles on the Contents page can be changed by pressing the **left** or **right** key on the remote control by adding the code given below. When the **left** key is pressed, the previous article is displayed, and when the **right** key is pressed, the next article is displayed. Pressing the **right** key when in the last article, takes you to the first article in the same category. Pressing the **left** key when in the first article, takes you to the last article.

When KEY_LEFT or KEY_RIGHT event occurs, the contentsKeyDown() method of the KeyHandler object requests the NewsController object to call the CONTENTS_ARTICLE_LEFT or CONTENTS_ARTICLE_RIGHT cases, respectively.

NewsController adds the foregoing task to its list and adds the code below to the request() method. It calculates the index of all the articles and replaces the data currently displayed by the UIContents with new ones.

```
case this.CONTENTS_ARTICLE_LEFT:
    if (--this.currentArticleIdx < 0) {
        this.currentArticleIdx=DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length - 1;
    }
    UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID,this.currentArticleIdx));
    UIContents.refresh();
    break;

case this.CONTENTS_ARTICLE_RIGHT:
    if(++this.currentArticleIdx>DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length - 1) {
        this.currentArticleIdx = 0;
    }
    UIContents.setDataObj(DataMgr.getContentsData(this.currentCategoryID,
    this.currentArticleIdx));
    UIContents.refresh();
    break;
```

Add the function below to UIContents.

```
UIContents.refresh = function () {
    alert("UIContents.refresh()");
    this.fillCategoryTitle();
    this.fillArticleTitle();
    this.fillDescription();
    alert("UIContents.refresh() End");
}
```

If the **left** or **right** key on the Contents page is pressed, the article currently displayed on the screen is replaced with another one.

Creating a Scrollbar

This section describes creating a scrollbar on the List and Contents pages.

Creating a Scrollbar on the List Page

On the List page, a scrollbar is created by receiving the index of the currently highlighted article and the number of the total articles as arguments. If the number of the total articles is less than 10, the scrollbar does not appear on the screen.

```
UIList.adjustScrollBar = function (articleIdx, totalArticle) {
    TRACE("UIList.adjustScrollBar("+articleIdx+","+totalArticle+")");
    var curPage = Math.floor(articleIdx/this.TITLE_MAX_NUM),
        totalPage = Math.floor((totalArticle-1)/this.TITLE_MAX_NUM),
        position;

    if (totalPage <= 0) {
        // if number of articles is less than 10,
        this.hideScrollBar();    // hide scroll bar.
    }
    else {
        position = Math.round((curPage*this.SCROLLBAR_SIZE)/totalPage);
        TRACE("Position : " + position);
        this.scrollBead.style.top = position + "px";
        this.showScrollBar();
    }
    TRACE("UIList.adjustScrollBar("+articleIdx+","+totalArticle+") End");
}

UIList.showScrollBar = function () {
    this.scrollBar.style.display = "block";
}
UIList.hideScrollBar = function () {
    this.scrollBar.style.display = "none";
}

UIList.adjustScrollBar(this.currentArticleIdx, DataMgr.getCategoryData(this.currentCategoryID).arrArticles.length);
```

Creating a Scrollbar on the Contents Page

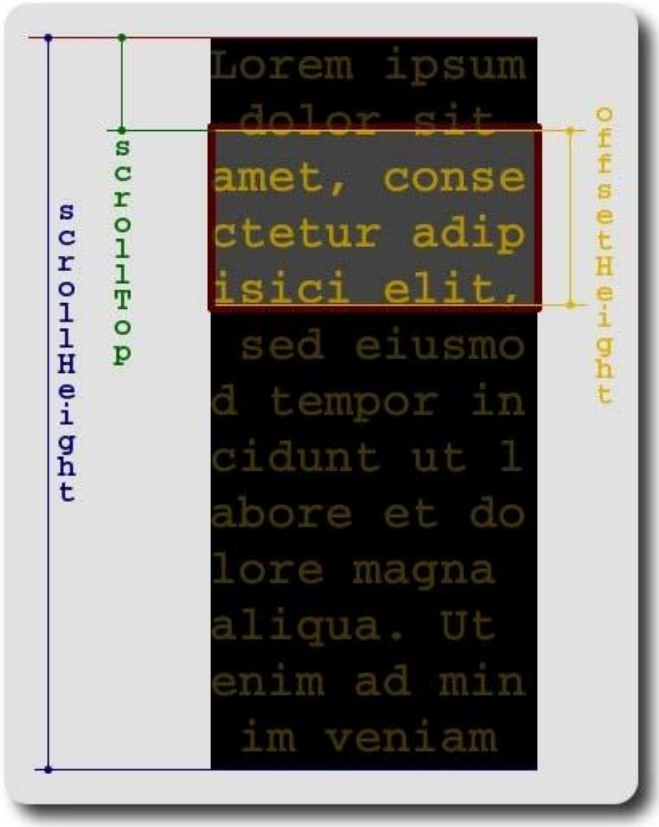

Figure: Creating a scrollbar on the contents page

The scrollbar appearing on the Contents page is more complicated than that on the List page. The Contents page scrollbar must change its length based on the entire length of the article being viewed.

Use the scrollTop , scrollHeight and offsetHeight values of the area where the article is displayed. The offsetHeight indicates the length of the scrollbar actually displayed on the screen, the scrollHeight indicates the entire length of the article, and the scrollTop indicates the part of the article that is being viewed. The application calculates the length based on these values to display a scrollbar.

> Note
>
> The scrollTop value cannot be changed after scrolling starts. The function executed as a callback function of the timer finds out the scrollTop value that has been changed by a key input.

To create the scrollbar:

1. Add the method below. This method works properly only when it is called using the timer.

```
UIContents.adjustScrollBar = function () {
    alert("UIContents.adjustScrollBar()");
    var scrollTop = this.articleDescription.scrollTop,
        scrollHeight = this.articleDescription.scrollHeight,
        offsetHeight = this.articleDescription.offsetHeight,
        position;
    alert("Description scrollTop : " + scrollTop);
    alert("Description scrollHeight : " + scrollHeight);
    alert("Description offsetHeight : " + offsetHeight);
    if (scrollHeight <= offsetHeight) {
        this.hideScrollBar();
    } else {
        position = parseInt((this.articleDescription.scrollTop)/(this.articleDescription.scrollHeight -
        this.articleDescription.offsetHeight) * this.SCROLLBAR_SIZE);
        alert("Position : " + position);
        this.scrollBead.style.top = position + "px";
        this.showScrollBar();

    }
    alert("UIContents.adjustScrollBar() End");
}

UIContents.showScrollBar = function () {
    this.scrollBar.style.display = "block";
}

UIContents.hideScrollBar = function () {
    this.scrollBar.style.display = "none";
}
```

   The contentsKeyDown() method of the KeyHandler object requests NewsController to make CONTENTS_SCROLL_UP and CONTENTS_SCROLL_DOWN respond to KEY_LEFT and KEY_RIGHT events, respectively. NewsController adds this request to its list and adds a case to the request() method to execute the code below.

```
setTimeout("UIContents.adjustScrollBar()", 0);
```

2. Press the **up** or **down** key, to make the scrollbar move across the screen.