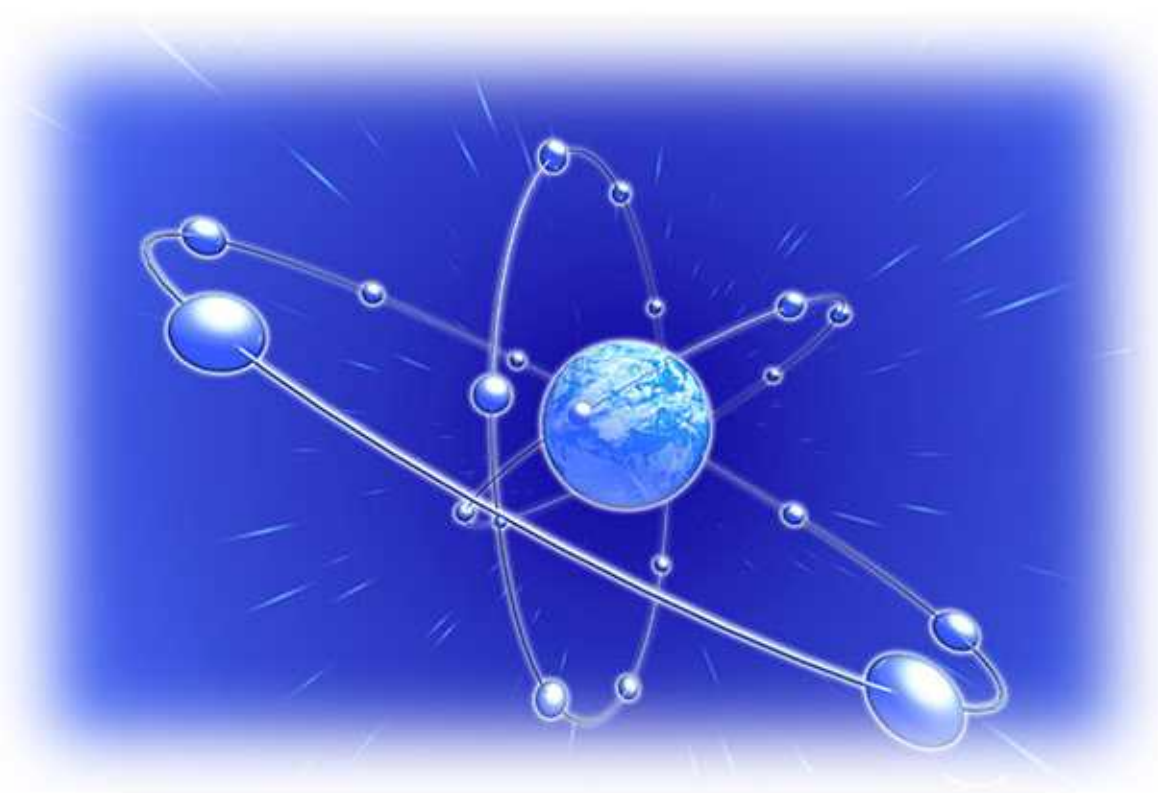


# GKit.NET Spreadsheet for WebForm

## アップデート 説明書

Version 1.2.20.3



2010年7月  
株式会社富士通アドバンストエンジニアリング

本説明書は、GKit.NET Spreadsheet for WebFormにおいて、アップデートで新しく機能追加した内容について記載しています。

仕様の変更、機能の追加については説明していますが、バグフィックスについては説明していません。

### バージョン「1.0.10.6」で追加された機能

1. 特定のセルの背景色を設定
2. ヘッダセルに入力部品を表示
3. ハイパーリンク型
4. テキストの複数行表示
5. 選択なしモード
6. タブの移動方式
7. ラベル、ボタン等のテキストの永続化の無効
8. カスタム型

### バージョン「1.0.10.7」で追加された機能

9. クライアント制御関数
10. 行と列のサイズ変更禁止
11. 自動編集モードの変更
12. アクションボタン
13. 入力部品のスタイルシート
14. 1レコードテーブル
15. コンテキストメニュー
16. Copy、Paste、Cut、Delete、Editの制限
17. 状態によるコピーアンドペースト
18. ペーストのリPEAT機能
19. 高度な書式指定
20. カスタムフィルタリング
21. 入力エラー時の動作
22. 編集開始、編集終了時の動作
23. リストアイテムの設定方法の拡張
24. カスタムソート
25. Home、Endキーの割り当て
26. テーブルのショートカット移動
27. 枠線表示の拡張

### バージョン「1.0.10.11」で追加された機能

28. レコード単位のプロパティ設定
29. ボタンタイプの拡張

### バージョン「1.0.10.12」で追加された機能

30. セル部品の無効化

### バージョン「1.0.10.13」で追加された機能

31. 特定行の高さ設定
32. クライアント制御関数の拡張

### バージョン「1.0.10.15」で追加された機能

33. Visual Studio 2005対応
34. 行の高さのパーセント指定

## **バージョン「1.0.10.16」で追加された機能**

35 . 行の高さの自動調整

## **バージョン「1.0.10.18」で追加された機能**

36 . コンテキストメニューの非表示  
37 . ソートによる背景色の入れ替え

## **バージョン「1.0.10.19」で追加された機能**

38 . ラジオボタンのグループ化

## **バージョン「1.2.20.1」で追加された機能**

39 . Visual Studio 2008対応

## **バージョン「1.2.20.2」で追加された機能**

40 . ポストバック時の動作を修正

## **バージョン「1.2.20.3」で追加された機能**

41 . Visual Studio 2010対応

## 1. 特定のセルの背景色を設定

### レコード部の背景色について

従来の仕様では、設定した背景色はレコード共通でした。

```
GKitSpread1.Record.Rows[0].EvenBackColor = Color.Yellow;  
GKitSpread1.Record.Rows[0].OddBackColor = Color.Green;  
GKitSpread1.Record.Rows[0].Cells[0].BackColor = Color.Gray;
```

上記のプログラムの場合、背景色は右図のようになります。  
色はレコードに設定するので、「3レコード目だけ色を変える」ということは  
できませんでした。

0レコード目			
1レコード目			
2レコード目			
3レコード目			
4レコード目			
5レコード目			

### アップデート内容

アップデート版では、任意のレコードに対して、行の背景色、セルの背景色を設定できるようになりました。

#### 行の背景色の設定

行に対して任意の位置の背景色を設定するには、以下のプロパティ(RecordRowクラス)を使用します。

##### [C#の宣言]

```
public System.Collections.IDictionary BackColorItems{get;}
```

##### [プログラム例]

```
GKitSpread1.Record.Rows[0].BackColorItems[2] = Color.Red;  
GKitSpread1.Record.Rows[0].BackColorItems[5] = Color.Blue;
```

上記のプログラムの場合、背景色は右図ようになります。  
BackColorItemsプロパティは、スプレッドデザイナーには表示されません。



#### セルの背景色の設定

セルに対して任意の位置の背景色を設定するには、以下のプロパティを(RecordCellクラス)を使用します。

##### [C#の宣言]

```
public System.Collections.IDictionary BackColorItems{get;}
```

##### [プログラム例]

```
GKitSpread1.Record.Rows[0].Cells[0].BackColorItems[2] = Color.Red;  
GKitSpread1.Record.Rows[0].Cells[2].BackColorItems[5] = Color.Blue;
```

上記のプログラムの場合、背景色は右図のようになります。  
BackColorItemsプロパティは、スプレッドデザイナーには表示されません。



BackColorItemsのインデクサのキーにはInt32型を、値にはColor型を指定してください。

0行、1列、2レコード目

0行、2列、5レコード目

## 背景色の優先度

背景色を設定できるプロパティはたくさんありますが、優先度は以下ようになります。

高い  
↑  
優先度  
低い

- RecordCellクラスのBackColorItem
- RecordCellクラスのOddBackColorとEvenBackColor
- RecordCellクラスのBackColor
- RecordRowクラスのBackColorItem
- RecordRowクラスのOddBackColorとEvenBackColor
- RecordRowクラスのBackColor

`GKitSpread1.Record.Rows[0].Cells[2].BackColorItems[1] = Color.Blue;`

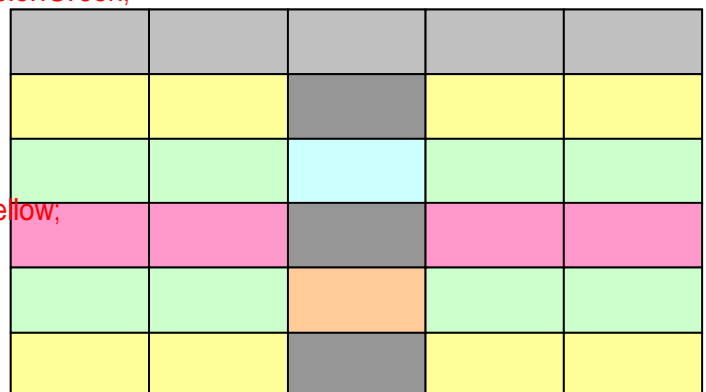
`GKitSpread1.Record.Rows[0].Cells[2].OddBackColor = Color.Orange;`

`GKitSpread1.Record.Rows[0].Cells[2].BackColor = Color.Gray;`

`GKitSpread1.Record.Rows[0].BackColorItems[2] = Color.Red;`

`GKitSpread1.Record.Rows[0].OddBackColor = Color.Green;`

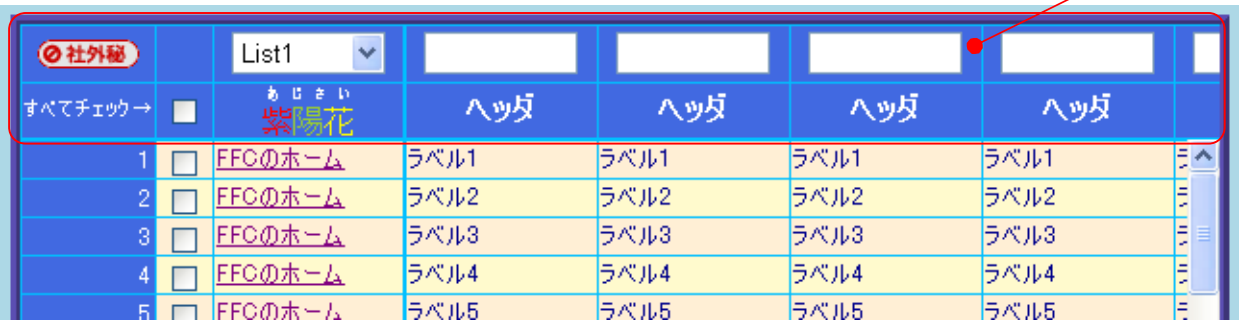
`GKitSpread1.Record.Rows[0].BackColor = Color.Yellow;`



## 2. ヘッダセルに入力部品を表示

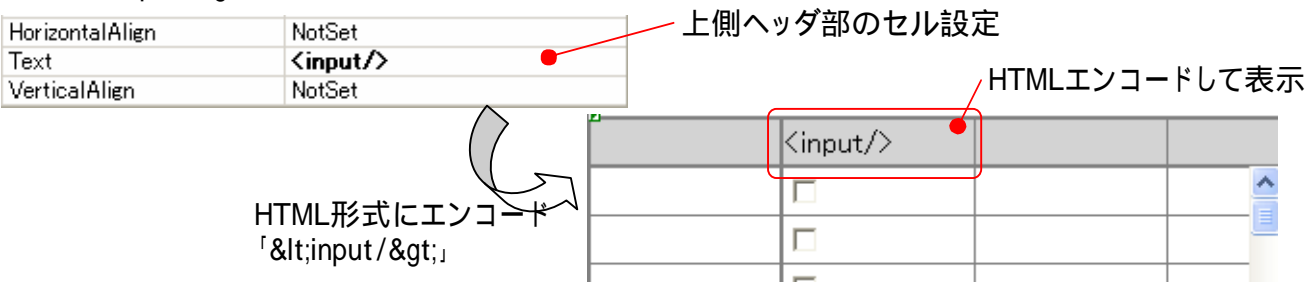
### 概要

従来の仕様では、ヘッダ部にはラベルしか表示できませんでした。  
アップデート版では、好きな入力部品を上側ヘッダ部に表示できるようになりました。



### 修正内容

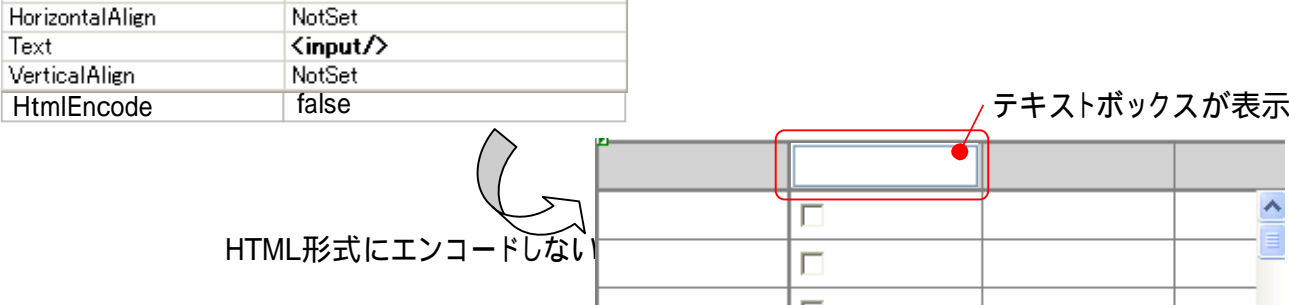
従来の仕様では、HeaderCellクラスのTextプロパティに設定した文字列は自動的にHTML形式にエンコードして表示ようになっていました。  
例えば、Textプロパティに<input/>と設定すると、テーブルには「<input/>」と表示されます。  
理由は「&lt;input/&gt;」に自動的にエンコードされたからです。



アップデート版ではHtmlEncodeプロパティを利用して、HTMLエンコードするか？しないか？を選択できます。  
HtmlEncodeプロパティはHeaderCellクラスで宣言されています。

### [C#の宣言]

```
public bool HtmlEncode {get; set;}
HtmlEncodeプロパティの初期値はtrueです。
HtmlEncodeプロパティは、スプレッドデザイナーにも表示されます。
trueのとき:エンコードする。falseのとき:エンコードしない。
```



以下のように、HeaderCellクラスのTextプロパティに直接、HTMLのタグを直接設定してください。



`<image src="secret.gif" />`

`<input style="width:80px" />`

`<select style="width:80px">  
<option>List1</option><option>List2</option><option>List3</option>  
</select>`

`<ruby><rb><font color="red">紫</font><font color="yellowgreen">陽</font>  
<font color="yellow">花</font></rb><rp></rp><rt>あじさい</rt><rp></rp></ruby>`

`<input type="checkbox" onclick="checkAll()" onkeydown="checkAll()" id="checkbox" />`

#### コード記述例(C#)

```
GKitSpread1.Header.Rows[0].Cells[0].HtmlEncode = false;  
GKitSpread1.Header.Rows[0].Cells[0].Text = "<image src=¥\"secret.gif¥\" />";
```

#### [制約事項]

直接HTMLのタグを記述できるので自由度がありますが、不正なタグを記述した場合はテーブル全体に影響を与えます。

また、タグの種類によっては正常に表示、動作できないパターンもあります。

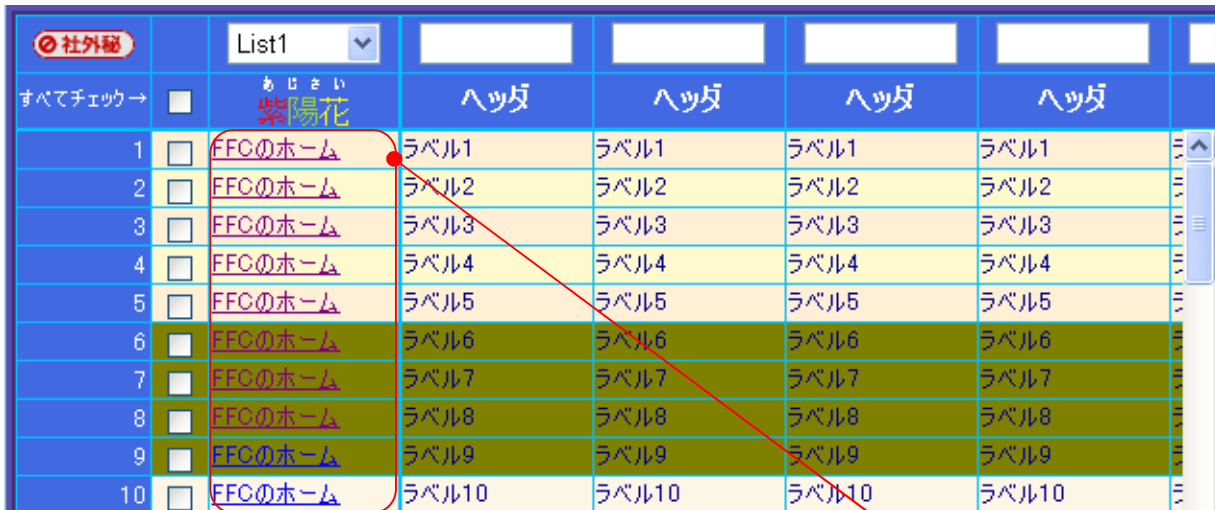
HtmlEncodeプロパティをfalseに設定する場合は、以下の制約事項をご考慮ください。

- ・ HTMLのタグは必ず正しく記述してください。  
例えば、「<input>>」のように、不正に記述すると、そのままページ出力されるのでページ全体に影響を及ぼします。
- ・ HTMLを記述するときのタグの階層は最大で3階層までです。
- ・ 動作検証済みのタグは、Input、Select、Imageです。  
その他のタグの使用については動作保証しておりません。
- ・ ヘッダに表示したテキストボックス、ドロップダウンリストなどのフォーカスの自動制御はおこないません。  
ヘッダに配置した部品のフォーカス制御はお客様が独自におこなう必要があります。

### 3. ハイパーリンク型

#### 概要

アップデート版ではハイパーリンクをセルに配置できるようになりました。



	List1					
すべてチェック→	<input type="checkbox"/>	あじさい 紫陽花	ヘッダ	ヘッダ	ヘッダ	ヘッダ
1	<input type="checkbox"/>	FFCのホーム	ラベル1	ラベル1	ラベル1	ラベル1
2	<input type="checkbox"/>	FFCのホーム	ラベル2	ラベル2	ラベル2	ラベル2
3	<input type="checkbox"/>	FFCのホーム	ラベル3	ラベル3	ラベル3	ラベル3
4	<input type="checkbox"/>	FFCのホーム	ラベル4	ラベル4	ラベル4	ラベル4
5	<input type="checkbox"/>	FFCのホーム	ラベル5	ラベル5	ラベル5	ラベル5
6	<input type="checkbox"/>	FFCのホーム	ラベル6	ラベル6	ラベル6	ラベル6
7	<input type="checkbox"/>	FFCのホーム	ラベル7	ラベル7	ラベル7	ラベル7
8	<input type="checkbox"/>	FFCのホーム	ラベル8	ラベル8	ラベル8	ラベル8
9	<input type="checkbox"/>	FFCのホーム	ラベル9	ラベル9	ラベル9	ラベル9
10	<input type="checkbox"/>	FFCのホーム	ラベル10	ラベル10	ラベル10	ラベル10

ハイパーリンク型

#### 修正内容

CellType列挙体にHyperLink項目が追加されました。

```
public enum CellType
{
    Label,
    Text,
    Button,
    CheckBox,
    RadioButton,
    DropDownList,
    HyperLink
}
```

RecordCellクラスに「NavigateUrl」「NavigateUrlItems」「TargetFrame」の3つのプロパティが追加されました。

#### NavigateUrl

##### [C#の宣言]

```
public string NavigateUrl {get; set;}
```

##### [説明]

すべてのレコード要素に共通のリンク先のURLを指定します。  
スプレッドデザイナーでも設定が可能です。

#### NavigateUrlItems

##### [C#の宣言]

```
public System.Collections.IDictionary NavigateUrlItems{get;}
```

##### [説明]

特定のレコード要素にリンク先のURLを指定します。  
インデクサのキーにはInt32型を、値にはstring型を指定してください。  
スプレッドデザイナーでは設定できません。



## TargetFrame

### [C#の宣言]

```
public string TargetFrame{get; set;}
```

### [説明]

リンク先Webページを読み込む、表示先のウィンドウまたはフレームです。  
値は、次に示すアンダースコア(\_)から始まる特殊な値を除き、  
大文字小文字を区別しないaからzの範囲の文字で始まる必要があります。

例	説明
_blank	フレームなしの新しいウィンドウに内容を表示します。
_parent	直前のフレームセットの親に内容を表示します。
_self	フォーカスのあるフレームに内容を表示します。
_top	最大化されたフレームなしのウィンドウ

ターゲットフレームは、すべてのレコード要素に対して共通です。  
特定のレコードのみ設定することはできません。  
スプレッドデザイナーでも設定が可能です。

## プログラム例

以下のプログラムは、コードからハイパーリンクを設定する例です。

```
// セルタイプをハイパーリンク型に設定します。  
GKitSpread1.Record.Rows[0].Cells[3].Type = CellType.HyperLink;  
// 表示先のウィンドウを設定します。  
GKitSpread1.Record.Rows[0].Cells[3].TargetFrame = "_blank";  
  
// レコードごとにリンク先URLを設定します。  
GKitSpread1.Record.Rows[0].Cells[3].NavigateUrlItems[0] = "http://www.ffc.co.jp/?val=0";  
GKitSpread1.Record.Rows[0].Cells[3].NavigateUrlItems[1] = "http://www.ffc.co.jp/?val=1";  
GKitSpread1.Record.Rows[0].Cells[3].NavigateUrlItems[2] = "http://www.ffc.co.jp/?val=2";  
GKitSpread1.Record.Rows[0].Cells[3].NavigateUrlItems[3] = "http://www.ffc.co.jp/?val=3";  
GKitSpread1.Record.Rows[0].Cells[3].NavigateUrlItems[4] = "http://www.ffc.co.jp/?val=4";  
GKitSpread1.Record.Rows[0].Cells[3].NavigateUrlItems[5] = "http://www.ffc.co.jp/?val=5";
```

## 4. テキストの複数行表示

セルに表示するテキストを1行表示するか？複数行表示するかをNoBreakプロパティで設定できます。NoBreakプロパティはHeaderCellクラスとRecordCellクラスで定義されています。

### NoBreak

#### [C#の宣言]

```
public bool NoBreak {get; set;}
```

#### [説明]

セル内のテキストが、trueのとき一行表示になります。  
falseのとき複数行表示になります。  
デフォルトはtrueです。



## 5. 選択なしモード

従来の仕様では、選択方式(SelectMode)はセルモード(CellMode)、複数レコードモード(MultipleRecordMode)単数レコードモード(SingleRecordMode)の3通りでした。これらの3通りの選択方式を選択モードと呼びます。選択モードでは、エクセルのような操作性でテーブルを編集することができました。アップデート版では新たに「選択なしモード(NoSelectMode)」を追加しました。「選択なしモード」とは、操作性は一般的なHTMLと同じで、スクロール機能を実現したモードです。選択モードの場合は、性能上、100レコード程度のテーブルが推奨サイズでしたが、「選択なしモード」の場合は、さらに大きなテーブルを扱うことができます。

以下は、選択方式を「選択なしモード」に設定するコード例です。

```
GKitSpread1.SelectMode = SelectMode.NoSelectMode;
```

## 6. タブ移動方式

従来の仕様では、フォーカスはテーブルの中に移れば、Tab移動でテーブルの外へ出ることはできませんでした。アップデート版では、Tab移動でフォーカスがテーブルの外へ出られるようになりました。また、TabIndexプロパティを設定して、Webページにおけるフォーカスの順序を設定することができます。

以下は、テーブルにタブインデックスを設定するコード例です。

```
GKitSpread1.TabIndex = 10;
```

## 7. ラベル、ボタン等のテキストの永続化の無効

GKitSpreadクラスで宣言されているDataSendプロパティを用いてラベル、ボタン等のテキストの永続化を無効にすることができます。

trueのとき、レコード部のラベル、ボタン、チェックボックス、ラジオボタンのテキストを隠しフィールドに格納します。ポストバック時に、CopyRequestToListItemCollectionメソッドを用いて、これらのデータをコピーすることができます。

falseのときは、データは送信されません。デフォルト値はtrueです。

従来の仕様では、trueが設定されていた状態です。

CopyRequestToListItemCollectionメソッドを使用しない場合や、ラベル、ボタン等のテキストを永続化しない場合は、falseに設定する選択肢も考えられます。

## 8. カスタム型

カスタム型は任意のHTMLタグを記述することができます。(現状は画像のみサポートしています)  
HeaderCellクラスのHtmlEncodeプロパティをfalseに設定した場合と類似しています。  
カスタム型はHeaderCellクラスではなく、RecordCellクラスに設定したテキストが有効になります。

CellType列挙体にCustomが追加されました。  
RecordCellクラスのTypeプロパティの値をCustomに設定することで、カスタム型が有効になります。  
仕組みはHtmlEncodeがfalseに設定した場合と同様です。  
Textに設定したテキストがエンコードされずに、そのままHTMLタグとして出力されます。



### [制約事項]

直接HTMLのタグを記述できるので自由度がありますが、不正なタグを記述した場合はテーブル全体に影響を与えます。

また、タグの種類によっては正常に表示、動作できないパターンもあります。

レコード部にカスタム型セルを設定する場合は、以下の制約事項をご考慮ください。

- ・ HTMLのタグは必ず正しく記述してください。  
例えば、「<input>>」のように、不正に記述すると、そのままページ出力されるのでページ全体に影響を及ぼします。
- ・ HTMLを記述するときのタグの階層は最大で3階層までです。
- ・ 動作検証済みのタグは、Imageのみです。  
画像を貼り付けるタグ以外の使用については動作保証しておりません。
- ・ タブキーでフォーカスが移動するタグは使用しないでください。  
(テキストボックス、チェックボックス、リンク、TabIndexを設定したタグなど)

## 9. クライアント制御関数

クライアント制御関数は、JavaScriptを用いてクライアントサイドでテーブルを操作することができます。クライアント制御関数は、「基本関数」「イベント」「フィルタ関数」より構成されます。関数を使用するためには、faceオブジェクトを取得する必要があります。下記のコード例のように、faceオブジェクトを取得して、関数を呼び出してください。

```
// faceオブジェクトの取得
var face = document.all.GKitSpread1.face;
// 関数の実行
face.setText(0, 0, 0, "Hello World!");
```

**GKitSpread1**はテーブルのIDを指定しています。  
IDがGKitSpread1以外の場合は、個別のIDを指定してください。

「イベント」と「フィルタリング関数」はSpreadsheetの本体より呼び出されます。  
onload時に、関数を登録する処理をおこなってください。

```
<script language="javascript">
  window.attachEvent('onload', load);
  function load() {
    // イベントの登録
    document.all.GKitSpread1.face.onTextChange = function(row, col, rec) {

      // 処理
      ...

    }
  }
</script>
```

### 基本関数

基本関数は、お客様のアプリケーション側で呼び出します。

#### getText(row, col, rec)

セルのテキストを取得します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
戻り値	テキストの文字列

#### setText(row, col, rec, text)

セルのテキストを設定します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
text	テキストの文字列
戻り値	成功、失敗を表す真偽値

### **getCheck(row, col, rec)**

チェックボックス、ラジオボタンの状態を取得します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
戻り値	チェックボックスの状態を表す真偽値

### **setCheck(row, col, rec, check)**

チェックボックス、ラジオボタンの状態を設定します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
check	チェックボックスの状態を表す真偽値
戻り値	成功、失敗を表す真偽値

### **getSelectedNumber(row, col, rec)**

ドロップダウンリストで選択されているアイテムの要素番号を取得します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
戻り値	選択されているアイテムの要素番号を表す整数値

### **setSelectedNumber(row, col, rec, index)**

ドロップダウンリストで選択されているアイテムの要素番号を設定します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
index	選択されているアイテムの要素番号を表す整数値
戻り値	成功、失敗を表す真偽値

### **getSelectedText(row, col, rec)**

ドロップダウンリストで選択されているアイテムのテキストを取得します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
戻り値	選択されているアイテムのテキストを表す文字列

### **setSelectedText(row, col, rec, text)**

ドロップダウンリストで選択されているアイテムのテキストを設定します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
index	選択されているアイテムのテキストを表す文字列
戻り値	成功、失敗を表す真偽値

## setBackColor(row, col, rec, color)

背景色を設定します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
color	背景色を表す文字列(例 . red、blue、yellow、#030303)
戻り値	なし

## getCellType(row, col, rec)

セルタイプを取得します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
戻り値	セルタイプを表す文字列(Text、Password、CheckBox、DropDownListなど)

## copy()

選択範囲をコピーします。

戻り値      なし

## paste()

選択範囲をペーストします。

戻り値      なし

## cut()

選択範囲を切り取ります。

戻り値      なし

## del()

選択範囲を削除します。

戻り値      なし

## isOpened()

セルが編集集中かどうかを判定します。

戻り値      編集集中かどうかを表す真偽値(true:編集集中、false:編集集中でない)

## close()

セルを閉じます。

戻り値      正常にセルを閉じることができたかを表す真偽値(true:成功、false:失敗)

## **clear()**

セルの範囲選択を解除します。

戻り値      なし

## **formatCheckAll()**

すべてのセルのフォーマットをチェックします。

戻り値      なし

## **イベント**

イベントは、何らかのアクションが発生したときに、Spreadsheetの本体より呼び出されます。戻り値を指定する必要はありません。

### **onTextChange(row, col, rec)**

テキストが変更されたときのイベントです。  
setText関数でテキストを変更したときはイベントは発生しません。

row          行のインデックスの整数値  
col          列のインデックスの整数値  
rec          レコードのインデックスの整数値  
戻り値      指定しないでください

### **onCheckBoxChange(row, col, rec)**

チェックボックスが変更されたときのイベントです。  
setCheck関数で変更したときはイベントは発生しません。

row          行のインデックスの整数値  
col          列のインデックスの整数値  
rec          レコードのインデックスの整数値  
戻り値      指定しないでください

### **onRadioButtonChange(row, col, rec)**

ラジオボタンが変更されたときのイベントです。  
setCheck関数で変更したときはイベントは発生しません。

row          行のインデックスの整数値  
col          列のインデックスの整数値  
rec          レコードのインデックスの整数値  
戻り値      指定しないでください

### **onDropDownListChange(row, col, rec)**

ドロップダウンリストが変更されたときのイベントです。  
setSelectedNumber、setSelectedText関数で変更したときはイベントは発生しません。

row          行のインデックスの整数値  
col          列のインデックスの整数値  
rec          レコードのインデックスの整数値  
戻り値      指定しないでください

## onButtonClick(row, col, rec)

ボタンがクリックされたときのイベントです。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
戻り値	指定しないでください

## フィルタ関数

フィルタ関数は、何らかの処理をおこなうために、Spreadsheetの本体より呼び出されます。戻り値はSpreadsheetの本体が利用しますので、正しく記述する必要があります。

### doFilter(row, col, rec, text)

テキストのフィルタリング処理です。独自のエラー処理を記述できます。RecordCellクラスのRestrictionプロパティがCustomのとき呼び出されます。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
text	フィルタリングをおこなうテキストを表す文字列
戻り値	結果を格納したObject型の連想配列
	ret["result"]      テキストが正しいかを表す真偽値
	ret["message"]    エラーメッセージを表す文字列
	ret["text"]        テキストを表す文字列
	ret["realText"]    表示テキストを表す文字列
	ret["viewText"]    実テキストを表す文字列
	realTextが存在しない場合は、textで設定した値が有効になります。
	viewTextが存在しない場合は、textで設定した値が有効になります。

### doSortAsc(row, col, a, b)

昇順ソートをおこなう際に、値を比較するために呼び出されます。

row	行のインデックスの整数値
col	列のインデックスの整数値
a	比較対象の1つ目の文字列
b	比較対象の2つ目の文字列
戻り値	比較結果を表す整数値
	a > b のとき      正の整数を返してください
	a = b のとき      0を返してください
	a < b のとき      負の値を返してください

### doSortDesc(row, col, a, b)

降順ソートをおこなう際に、値を比較するために呼び出されます。

row	行のインデックスの整数値
col	列のインデックスの整数値
a	比較対象の1つ目の文字列
b	比較対象の2つ目の文字列
戻り値	比較結果を表す整数値
	a > b のとき      負の整数を返してください
	a = b のとき      0を返してください
	a < b のとき      正の値を返してください



## 10. 行と列のサイズ変更禁止

ユーザーがマウスを使って行と列のサイズを変更できるかを設定できます。  
GKitSpreadクラスのAllowUserResizingプロパティに真偽値を設定します。

trueのとき行と列のサイズ変更を許可します。  
falseのとき行と列のサイズ変更を禁止します。

## 11. 自動編集モードの変更

自動編集モード(AutoEditがtrue)のとき、テキスト型のセルで文字キーを入力したときの動作が変更されました。

### 従来仕様

「a」を入力したとき、既存のテキストの先頭に「a」を挿入して、編集を開始。

### アップデート版仕様

「a」を入力したとき、既存のテキストがクリアされ、「a」を挿入して、編集を開始。

アップデート版では、IMEが全角のときでも、正しく入力できるようになりました。  
自動編集モードをサポートするIMEのバージョンは2002と2003、OSはWindows XPです。  
KeyDown、KeyUpイベントが正常に取得できない場合は、自動編集できません。  
編集集中にEscキーで編集を中断できるようになりました。

## 12. アクションボタン

従来仕様では、ボタンを押したときは、サーバー側に通知されて、イベントが発生しました。  
アップデート版では、ボタンを押したときのイベントをクライアント制御関数で取得できるようになりました。

RecordCellクラスのButtonTypeプロパティにButtonType.Buttonを設定してください。  
イベントの登録は以下のように、記述してください。

```
<script language="javascript">
    window.attachEvent('onload', load);
    document.all.GKitSpread1.face.onButtonClick = function(row, col, rec)
    {
        alert(row + ", " + col + ", " + rec + "が押されました");
    }
</script>
```

ButtonTypeプロパティにButtonType.Submitを指定すると従来通り、サーバーに通知されます。

## 13. 入力部品のスタイルシート

入力部品(テキストボックス、ボタン等)にスタイルシートを設定することができます。  
RecordCellクラスのInputCssClassプロパティにCSSクラス名を指定してください。

動作検証済みのスタイルシートは以下の通りです。

```
height
back-groundcolor
filter:flipV()
```

## 14.1 レコードテーブル

アップデート版では、1レコードのテーブルを快適に操作できるよう機能拡張がされています。  
1レコードのテーブルとは、GKitSpreadクラスのRecordCountを「1」に設定したテーブルです。

### テーブルモード

選択方式(SelectMode)がCellModeのとき、行ヘッダ部をクリックしたときは、レコード単位で選択されました。  
選択方式を新しく追加されたTableModelにすることにより、行単位での選択が可能になりました。

### RecordCellクラスの拡張

従来の仕様では、Check、Selectの設定はRecordItemクラスでおこなっていました。

```
GKitSpread1.Record.Rows[0].Cells[0].Items[0].Check = true;
```

アップデート版では、RecordCellクラスでもSelectとCheckが設定できます。

```
GKitSpread1.Record.Rows[0].Cells[0].Check = true;
```

1レコードのテーブルを作成した場合は、RecordCellクラスのText、Select、Checkに値を設定してください。

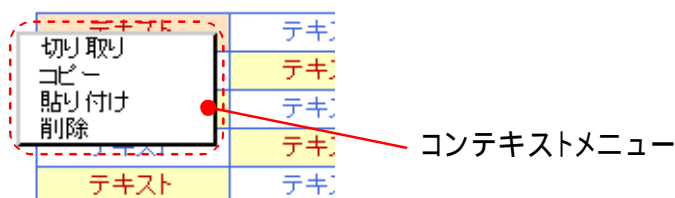
### 永続化関数

1レコードのテーブルを作成した場合は、ポストアックしたときのデータの永続化関数として  
新しく追加されたCopyRequestToRecordCell()を利用してください。

CopyRequestToRecordCellは、ポストアックした一番最初のレコードのデータを  
RecordCellクラスのText、Check、Selectにコピーします。

## 15. コンテキストメニュー

セルを右クリックしたときにコンテキストメニューが表示されるようになりました。



コンテキストメニューでは、「切り取り」、「コピー」、「貼り付け」、「削除」を実行できます。

## 16. Copy、Paste、Cut、Delete、Editの制限

RecordCellクラスの以下のプロパティを設定することにより、操作の許可、禁止を設定できます。  
true: 許可、false: 禁止

AllowCopy	コピー
AllowPaste	貼り付け
AllowCut	切り取り
AllowDelete	削除
AllowEdit	編集

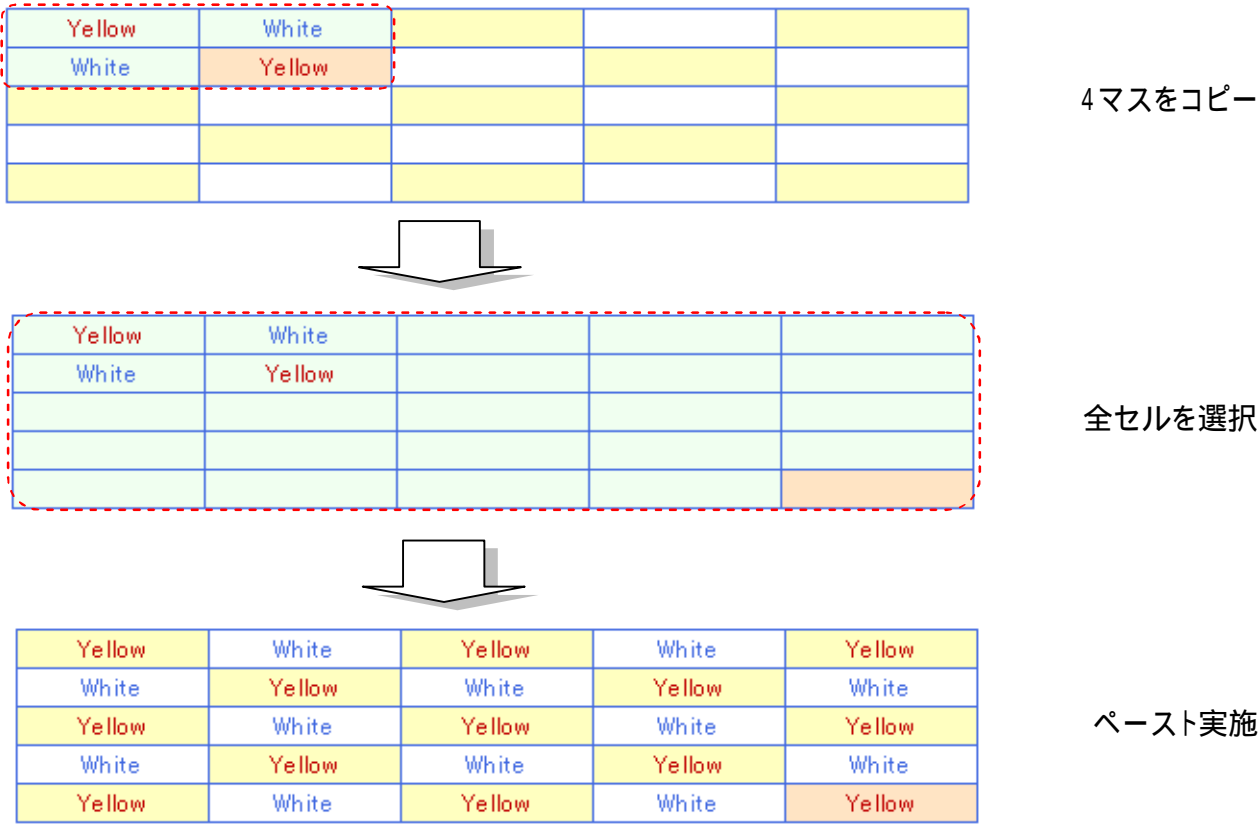
編集とは「テキスト編集」のことです。セルタイプがテキスト型のときのみ意味を持ちます。

# 17. 状態によるコピーアンドペースト

GKitSpreadクラスのCopyPasteModeにStatusを設定することにより、状態によるコピーアンドペーストが有効になり、チェックボックスとドロップダウンリストの状態をコピーアンドペーストできます。

# 18. ペーストのリピート

GKitSpreadクラスのRepeatPasteプロパティがtrueのとき、ペーストのリピート機能が有効になります。ペーストのリピート機能とは、選択範囲に対してペーストをおこなったときは、選択されたすべてのマスに対応するテキストで埋める機能です。



# 19. 高度な書式指定

アップデート版では書式機能が拡張されています。

## 日付型

日付型(Restriction=Date)のフォーマットを和暦に対応しました。

記述子	説明
yy	西暦年(00 ~ 99)
yyyy	西暦年(1900 ~ 2100)
e	和暦年(1 ~ 99)
ee	和暦年(01 ~ 99)
g	年号(M, T, S, H)
gg	年号(明、大、昭、平)
ggg	年号(明治、大正、昭和、平成)
m	月(1 ~ 12)
mm	月(01 ~ 12)
d	日(1 ~ 31)
dd	日(01 ~ 31)

設定例は以下のようになります。

フォーマット	入力値(年/月/日)	表示内容
yy/mm/dd	2004/9/12	04/09/12
yyyy/m/d	2004/9/12	2004/9/12
yyyy年m月d日	2004/9/12	2004年9月12日
d-m-yyyy	2004/9/12	12-9-2004
yymmdd	2004/9/12	040912
yyyymmdd	2004/9/12	20040912
gee/mm/dd	2004/9/12	H16/09/12
gee.mm.dd	H16/9/12	H16.09.12
gge-m-d	H16/9/12	平16-9-12
ggge年m月d日	H16/9/12	平成16年9月12日
(ggg)ee年mm月dd日	2004/9/12	(平成)16年09月12日
ggeemdd	H16/9/12	平160912

## 金額型

金額型のフォーマット形式が変更されました。

記述子	説明
#,##0	整数部をカンマ区切り
0	整数部をカンマ区切りなし

設定例は以下のようになります。

フォーマット	入力値(年/月/日)	表示内容
#,##0円	1234.56789	1,235円
0	1234.56789	1235
#,##0.00	1234.56789	1,234.57
#,##0.0000	1234	1,234.0000
\$0.00	1234	\$1234.00

## 実テキストと表示テキスト

実テキストと表示テキストの表示形式を分離することが可能になりました。  
 実テキストとは、編集時のテキストと、サーバに送信されるテキストです。  
 表示テキストとは、編集状態でないとき、セルに表示されるテキストです。

実テキストのフォーマットは、RecordCellクラスのRealFormatプロパティに記述します。  
 表示テキストのフォーマットは、RecordCellクラスのViewFormatプロパティに記述します。

### ・テキストの編集時

H16/04/04	テキスト
テキスト	テキスト

実テキストのフォーマット  
gee/mm/dd

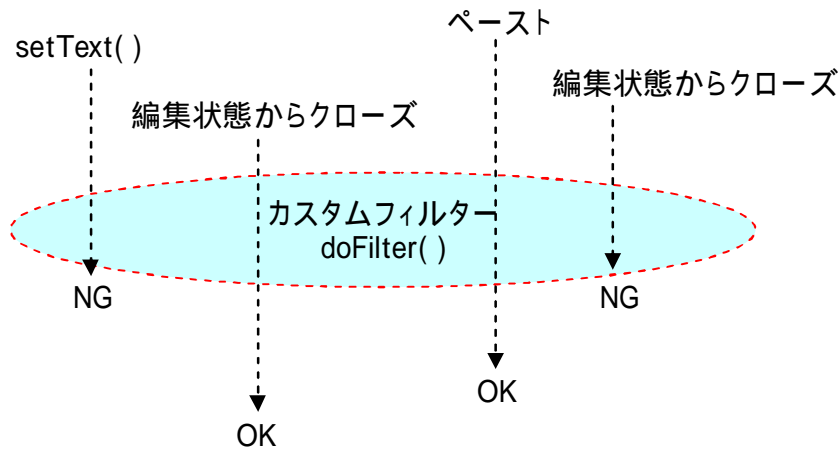
### ・表示状態

平16年4月4日	テキスト
テキスト	テキスト

表示テキストのフォーマット  
gge年m月d日

## 20. カスタムフィルタリング

カスタムフィルタリングとは独自のエラー処理をおこなう機能です。  
カスタムフィルタリングを用いることにより、デフォルトの日付型や金額型と同等のエラー処理ロジックも記述できるため、非常に強力な書式処理が可能です。



カスタムフィルタリングを有効にするには、RecordCellクラスのRestrictionプロパティにCustomを設定します。

```
GKitSpread1.Record.Rows[0].Cells[0].Restriction = Restriction.Custom;
```

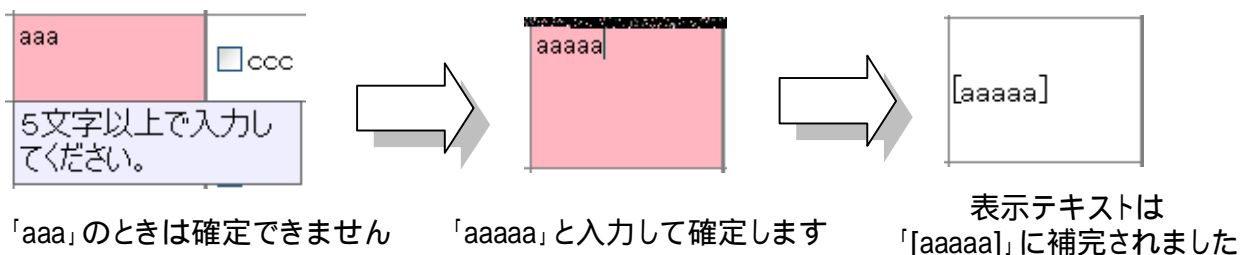
次に、クライアントスクリプト関数のdoFilterの登録をおこないます。  
以下は、5文字以上のテキストでない場合は、エラーと見なします例です。

```
<script language="javascript">
    window.attachEvent('onload', load);

    function load() {
        document.all.GKitSpread1.face.doFilter = function(row, col, rec, text)
        {
            var ret = new Object();

            ret["result"] = (text.length >= 5);
            ret["message"] = "5文字以上で入力してください。";
            ret["realText"] = text;
            ret["viewText"] = "[" + text + "]";

            return ret;
        }
    }
</script>
```



## 21. 入力エラー時の動作

アップデート版では、入力エラー時の動作を詳細に設定できるようになりました。

### 背景色の変更

入力エラー時に背景色を変更することができます。  
背景色は、GKitSpreadクラスのErrorColorプロパティに設定します。  
背景色を変更するかは、GKitSpreadクラスのChangeColorOnErrorプロパティに設定します。

2004/2/30	

存在しない日付を入力したので、エラー。  
背景色がパープルに変化。

平16年2月28日	

正しい日付なのでOK。  
背景色は正常の状態。

### エラー時の確定禁止

エラー時の確定を禁止できます。(デフォルトは確定禁止です。)  
GKitSpreadクラスのWriteProtectOnErrorプロパティに設定します。  
WriteProtectOnErrorが有効のとき、テキストの不正入力時は、以下の動作に従います。

- ・編集状態から確定の禁止
- ・ペーストの禁止
- ・setText()の禁止

### 初期エラーチェック

ブラウザに表が表示されたときや更新されたときは、自動的にエラーチェックはおこないません。  
自動的にエラーチェックをおこなうためには、GKitSpreadクラスのInitFormatCheckプロパティを有効(true)にしてください。  
InitFormatCheckプロパティが有効のときは、ページの読み込み時に、自動的にフォーマットチェックをおこない、補完やエラー時の背景色を設定します。

なお、カスタムフィルタリングを使用している場合は、InitFormatCheckを無効にして、doFilter()を登録した後で、手動でformatCheckAll関数を呼んでください。  
InitFormatCheckプロパティが有効のときは、自動でformatCheckAll関数が呼ばれる仕組みです。

## 22. 編集開始、編集終了時の動作

編集開始時の動作、編集終了時の動作を設定することができます。  
OpenCellModeとCloseCellModeはGKitSpreadクラスで設定できます。

### OpenCellMode

CursorLeft	セルを開いたとき、カーソルの初期位置を一番左側にします
CursorRight	セルを開いたとき、カーソルの初期位置を一番右側にします
SelectAll	セルを開いたとき、テキストを選択状態にします

### CloseCellMode

KeepCursor	セルを閉じたとき、カーソル位置を保持します。
MoveNext	セルを閉じたとき、閉じたときのキーによってセル移動します。

## 23. リストアイテムの設定方法の拡張

従来の仕様では、リストの要素は、RecordCellクラスのListItemsプロパティに設定していました。アップデート版では、RecordCellクラスとListItemクラスのTextプロパティにも設定できるようになりました。Textプロパティに設定したリストアイテムを有効にするには、ListItemsプロパティが空の必要があります。ListItemsプロパティが設定されているときは、優先的にListItemsで設定した内容が有効になります。Textプロパティに設定するリストアイテムはカンマ区切りで指定してください。

```
GKitSpread1.Record.Rows[0].Cells[0].Text = "リンゴ,ミカン,イチゴ";
```

## 24. カスタムソート

デフォルトのソートは、文字列で比較しているのので、数値では比較されません。カスタムソートは、独自のソート処理を記述することにより、より柔軟なソート処理を支援します。

クライアント制御関数に、ソートのための比較関数を登録すれば独自のソートを実現できます。以下の例は、数値でソートを実装しています。

```
<script language="javascript">
    window.attachEvent('onload', load);

    function load() {
        // 昇順ソートの処理
        document.all.GKitSpread1.face.doSortAsc = function(row, col, a, b){
            {
                var ai = parseInt(a);
                var bi = parseInt(b);

                if( isNaN(ai) || isNaN(bi) ){
                    return 0;
                }
                return (ai - bi);
            }

            // 降順ソートの処理
            document.all.GKitSpread1.face.doSortDesc = function(row, col, a, b){
                {
                    var ai = parseInt(a);
                    var bi = parseInt(b);

                    if( isNaN(ai) || isNaN(bi) ){
                        return 0;
                    }
                    return (bi - ai);
                }
            }
        }
    }
</script>
```

## 25 . Home、Endキーの割り当て

GKitSpreadクラスのHomeAndEndKeysプロパティを設定することにより、HomeキーとEndキーを割り当てることができます。

TopBottom     HomeキーとEndキーを押すと、テーブルの上端と下端に移動します。  
LeftRight     HomeキーとEndキーを押すと、テーブルの左端と右端に移動します。

## 26 . テーブルのショートカット移動

アップデート版ではCtrlキーと方向キーの組み合わせでテーブルの端へ移動できるようになりました。

## 27 . 枠線表示の拡張

アップデート版では枠線をより詳細に表示できるようになりました。

苗字/名前	鈴木	大輝
生年月日/血液型	昭50年6月15日	〇型
住所	東京都新宿区	
趣味	プログラミング	
積立額/使用額	1,000,000円	500,000円
会員	<input checked="" type="checkbox"/> データを送信	

セル単位で枠線を設定しています

GKitSpreadクラスにCellStyleが追加され、枠線のスタイルを設定できます。  
また、HeaderRow、HeaderCell、RecordRow、RecordCellに追加された以下のプロパティを用いて、行ごと、セルごとに枠線を設定できます。

BorderColor	枠の色です。
BorderStyle	枠のスタイルです。
BorderWidth	枠の幅です。
BorderTopColor	上枠の色です。
BorderTopStyle	上枠のスタイルです。
BorderTopWidth	上枠の幅です。
BorderBottomColor	下枠の色です。
BorderBottomStyle	下枠のスタイルです。
BorderBottomWidth	下枠の幅です。
BorderLeftColor	左枠の色です。
BorderLeftStyle	左枠のスタイルです。
BorderLeftWidth	左枠の幅です。
BorderRightColor	右枠の色です。
BorderRightStyle	右枠のスタイルです。
BorderRightWidth	右枠の幅です。

### [制約事項]

GKit.NETの枠の表示スタイル(border-collapse)は、collapseが設定されているため、隣接するセルのボーダーを重ねて表示する仕組みです。  
HTMLの描画規則に従いますので、ご了承ください。



## 28. レコード単位のプロパティ設定

行とセルの背景色やフォント色等を、レコード単位で設定できるようになりました。

### セルのプロパティをレコード単位で設定する場合

セルのプロパティをレコード単位で設定する場合は、RecordItemクラスのプロパティを利用します。RecordItemクラスでは以下のプロパティを利用できます。

BackColor	背景色
	RecordCellクラスのBackColorItemsプロパティで背景色を設定した場合、BackColorItemsの設定が優先されます。
ForeColor	文字色
NavigateUrl	リンク先のURL

上記のプロパティはスプレッドデザイナーからも設定できます。

例えば、レコードごとにBackColorを設定する場合は、

```
GKitSpread1.Record.Rows[0].Cells[2].Items[0].BackColor = Color.Blue;  
GKitSpread1.Record.Rows[0].Cells[2].Items[1].BackColor = Color.Red;  
GKitSpread1.Record.Rows[0].Cells[2].Items[2].BackColor = Color.Green;
```

のようにコードを記述します。

### 行のプロパティをレコード単位で設定する場合

行のプロパティをレコード単位で設定する場合は、RowRecordItemクラスのプロパティを利用します。RowRecordItemクラスでは以下のプロパティを利用できます。

BackColor	背景色
	RecordRowクラスのBackColorItemsプロパティで背景色を設定した場合、BackColorItemsの設定が優先されます。
ForeColor	文字色

上記のプロパティはスプレッドデザイナーでは**設定できません**。

例えば、レコードごとにBackColorを設定する場合は、

```
GKitSpread1.Record.Rows[0].Items[0].BackColor = Color.Blue;  
GKitSpread1.Record.Rows[0].Items[1].BackColor = Color.Red;  
GKitSpread1.Record.Rows[0].Items[2].BackColor = Color.Green;
```

のようにコードを記述します。

## 29. ボタンタイプの拡張

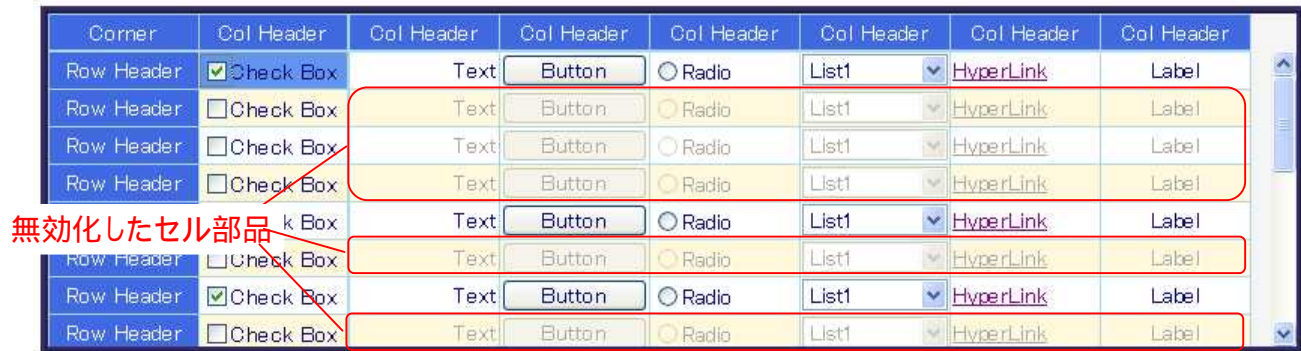
ButtonType列挙体にBoth項目が追加されました。Bothを使用すると、クライアント側でonButtonClickイベントを処理した後、サーバーに通知され、サーバー側でButtonClickイベントが発生します。SubmitとButtonの両方の機能を兼ね備えています。

クライアントイベントのonButtonClickの戻り値がfalseのときは、サーバーへの通知をキャンセルします。戻り値がその他の場合は、サーバー通知をおこないます。



## 30. セル部品の無効化

Enabledプロパティを設定することにより、セル部品を有効化/無効化できるようになりました。デフォルトではEnabledはtrueに設定されており、セル部品は有効な状態です。Enabledをfalseに設定すると、セル部品が無効化され灰色表示になり、操作ができなくなります。



Enabledを設定できるクラスは以下の4つです。

RecordRow	行ごとに設定します。(GKitSpread1.Record.Rows[ ].Enabled)
RowRecordItem	行の各レコードに対して設定します。(GKitSpread1.Record.Rows[ ].Items[ ].Enabled)
RecordCell	セルごとに設定します。(GKitSpread1.Record.Rows[ ].Cells[ ].Enabled)
RecordItem	セルの各レコードに対して設定します。 (GKitSpread1.Record.Rows[ ].Cells[ ].Items[ ].Enabled)

Enabledを設定するコード例を示します。

```
GKitSpread1.Record.Rows[0].Cells[3].Items[0].Enabled = false;
GKitSpread1.Record.Rows[0].Cells[3].Items[1].Enabled = false;
GKitSpread1.Record.Rows[0].Cells[3].Items[2].Enabled = false;
GKitSpread1.Record.Rows[0].Cells[3].Items[3].Enabled = false;
GKitSpread1.Record.Rows[0].Cells[3].Items[4].Enabled = false;
```

### クライアント制御関数

JavaScriptを用いて、クライアントサイドでセルの部品を有効化/無効化できます。

#### getDisabled(row, col, rec)

セル部品の有効/無効を取得します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
戻り値	セル部品が無効ならtrue、有効ならfalseを返します。

#### setDisabled(row, col, rec, disabled)

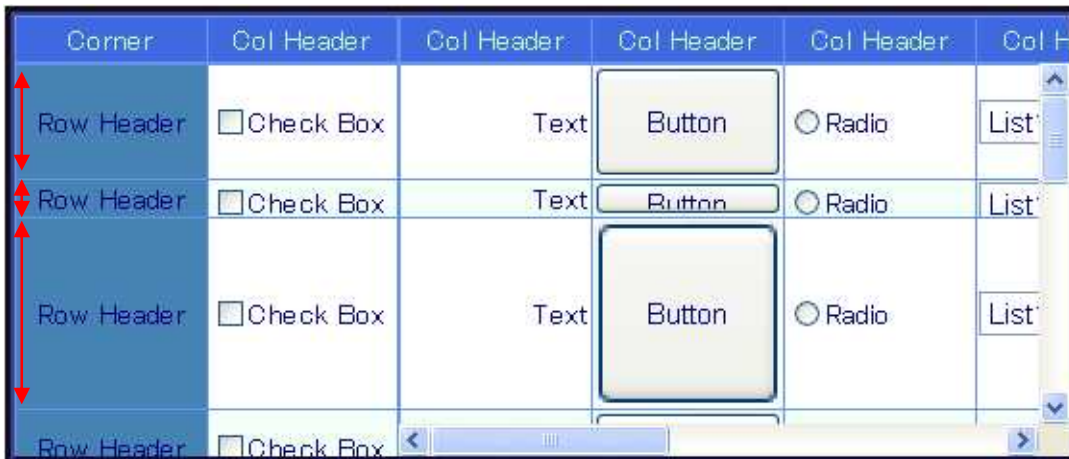
セル部品の有効/無効を設定します。

row	行のインデックスの整数値
col	列のインデックスの整数値
rec	レコードのインデックスの整数値
disabled	セル部品を無効にするならtrue、有効にするならfalseを返します。
戻り値	成功、失敗を表す真偽値

### 3 1 . 特定行の高さ設定

レコード単位で特定行の高さを設定できるようになりました。  
特定行の高さを設定するにはRowRecordItemクラスのHeightプロパティを利用します。  
HeightプロパティはRecordRowクラスとRowRecordItemクラスで設定できますが、  
RowRecordItemクラスで設定した値が優先的に反映されます。  
コード例を以下に示します。

```
GKitSpread1.Record.Rows[0].Items[0].Height = new Unit(60);  
GKitSpread1.Record.Rows[0].Items[1].Height = new Unit(20);  
GKitSpread1.Record.Rows[0].Items[2].Height = new Unit(100);
```



### 3 2 . クライアント制御関数の拡張

クライアント制御関数が拡張されました。

#### forceClose()

セルの編集時、セルがフォーマットエラーのときでも、強制的にクローズします。

戻り値     なし

#### cancel()

セルの編集時、編集をキャンセルします。  
Escキーを押下した場合と同じ動作です。

戻り値     なし

### 3 3 . Visual Studio 2005対応

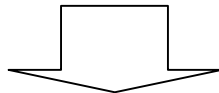
Visual Studio 2005に対応しました。

#### [Visual Studio 2005で利用するときの注意]

GKit.NET Spreadsheet for WebFormは標準準拠モードには対応していません。  
後方互換モードで解釈させてください。  
Visual Studio 2005はデフォルトで標準準拠モードを利用するようになっているので、  
DOCTYPEスイッチを変更してからご利用ください。  
(DOCTYPEスイッチはASPXファイルに記述されています)

標準準拠モード

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" ~>
```



後方互換モード

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
```

### 3 4 . 行の高さのパーセント指定

行の高さをパーセントで指定できるようになりました。(レコード行のみ)  
パーセント指定すると、テーブルの読み込み時、自動的に行の高さを調節ことができます。  
コード例を以下に示します。

```
GKitSpread1.Record.Rows[0].Items[0].Height = new Unit("20%");  
GKitSpread1.Record.Rows[0].Items[1].Height = new Unit("30%");  
GKitSpread1.Record.Rows[0].Items[2].Height = new Unit("50%");
```

#### [制約事項]

スプレッドデザイナーを利用して、行の高さをパーセント指定したとき、Visual Studioのフォーム上の表示は正しくない場合があります。(固定部とスクロール部がずれて表示されます)

↑↓	Hello World!			
↑↓	Hello World!			
↑↓	Hello World!			

### 3 5 . 行の高さの自動調整

レコード行の高さを未設定(Empty)にすることにより、行の高さを自動調整できるようになりました。  
セル内のテキストや要素の高さにより、ページロード時に行の高さを自動調整します。  
コード例を以下に示します。

```
GKitSpread1.Record.Rows[0].Height = Unit.Empty;
```

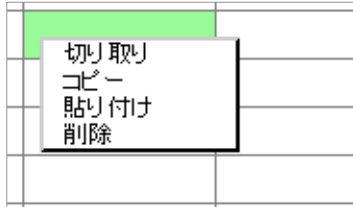
行の高さをパーセント指定したときと同様に、Visual Studioのフォーム上の表示制限があります。

### 36. コンテキストメニューの非表示

コンテキストメニューを非表示にできるようになりました。  
非表示に設定すると、レコード部で右クリックしてもコンテキストメニューは表示されません。  
GKitSpreadクラスのEnableContextMenuプロパティに真偽値を設定します。

trueのとき、コンテキストメニューは表示されます。  
falseのとき、コンテキストメニューは表示されません。

デフォルト値はtrueです。



### 37. ソートによる背景色の入れ替え

ソート時に背景色を入れ替えれるようになりました。  
GKitSpreadクラスのBackColorSwapOnSortプロパティに真偽値を設定します。

trueのとき、ソート時に背景色を入れ替えます。  
falseのとき、ソート時に背景色を入れ替えません。

デフォルト値はfalseです。  
BackColorSwapOnSortプロパティは、プロパティウィンドウには表示されません。  
ソースコードから設定してください。

### 38. ラジオボタンのグループ化

複数のラジオボタンをグループ化できるようになりました。  
グループ化したラジオボタンでは、ひとつだけ選択状態を取ることができます。  
従来は、RecordCellに含まれる全てのレコードアイテムでグループ化していました。  
アップデート版では、グループ化するセルを選ぶことができます。  
RecordCellクラスとRecordItemクラスのRadioButtonGroupNameプロパティにString型のグループ名を設定してください。  
グループ名に利用できる文字は、半角英数文字とアンダーバーです。  
RadioButtonGroupNameプロパティはRecordCellクラスとRecordItemクラスの両方で設定できますが、RecordItemクラスで設定した値が優先的に反映されます。  
コード例を以下に示します。

```
GKitSpread1.Record.Rows[0].Cells[1].RadioButtonGroupName = "Group1";  
GKitSpread1.Record.Rows[0].Cells[2].RadioButtonGroupName = "Group1";  
GKitSpread1.Record.Rows[0].Cells[3].RadioButtonGroupName = "Group1";  
GKitSpread1.Record.Rows[0].Cells[4].RadioButtonGroupName = "Group1";
```

ラジオボタンをグループ化

## 39 . Visual Studio 2008対応

Visual Studio 2008に対応しました。

### [Visual Studio 2008で利用するときの注意]

GKit.NET Spreadsheet for WebFormは標準準拠モードには対応していません。  
後方互換モードで解釈させてください。  
Visual Studio 2008はデフォルトで標準準拠モードを利用するようになっているので、  
DOCTYPEスイッチを変更してからご利用ください。  
(DOCTYPEスイッチはASPXファイルに記述されています)

標準準拠モード

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" ~>
```



後方互換モード

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
```

## 40 . ポストバック時の動作を修正

ハイパーリンク型、ボタン型でのポストバック時の動作を修正しました。

ハイパーリンク型

今回、"CopyRequestToListItemCollection(this.Request)"メソッドを明示的に呼び出すことによって、  
ポストバックを行っても"NavigateUrl"プロパティの設定値は保持されるように対応しました。

ボタン型

今回、"CopyRequestToListItemCollection(this.Request)"メソッドを明示的に呼び出すことによって、  
ポストバックを行っても"Enabled"プロパティの設定値は保持されるように対応しました。

### ["Enabled"プロパティの引継ぎの注意点]

他のセルタイプ (Label、Text、CheckBox、RadioButton、DropDownList、HyperLink)については、  
"Enabled"の状態を引き継ぎません。コード上でButton以外の部品の"Enabled"プロパティを設定した場合、  
画面の描画前に再度、値を設定する必要があります。

## 4 1 . Visual Studio 2010対応

Visual Studio 2010に対応しました。

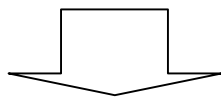
### [Visual Studio 2010で利用するときの注意]

#### ・標準準拠モードについて

GKit.NET Spreadsheet for WebFormは標準準拠モードには対応していません。  
後方互換モードで解釈させてください。  
Visual Studio 2008はデフォルトで標準準拠モードを利用するようになっているので、  
DOCTYPEスイッチを変更してからご利用ください。  
(DOCTYPEスイッチはASPXファイルに記述されています)

標準準拠モード

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" ~>
```



後方互換モード

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
```

#### ・<asp:ContentPlaceHolder />タグについて

GKit.NET Spreadsheet for WebFormは<asp:ContentPlaceHolder />タグには対応しておりません。  
上記タグを利用して呼び出した先の画面に Spreadsheetを配置した場合、製品機能が正常に動作しません  
(本制限事項に関しては、Visual Studio 2010での開発に限りません。)

Visual Studio 2010は、「ASP.NET Webアプリケーション」からプロジェクトを新規作成した場合、  
Site.Masterページ上に<asp:ContentPlaceHolder />タグが配置されます。Default.aspxを呼び出します。  
上記タグにDefault.aspxが紐付けられておりますので、Default.aspxにSpreadsheetを配置した場合、  
正常に動作しません。  
(Site.MasterページにSpreadsheetを配置することは可能です。)

<asp:ContentPlaceHolder />タグから呼び出される画面にSpreadsheetを配置しないか、  
「ASP.NET 空のWebアプリケーション」からプロジェクトを新規作成してください。