「iPhone プログラミング UIKit 詳解テキスト」訂正情報（補足）

最終更新日：2010/02/16

●訂正情報-1：セルの移動

9.3.6 セルの移動（P.301）で一番下のセルを移動不可にする方法として以下のコードを紹介しています。

```
- (BOOL)tableView:(UITableView*)tableView
canMoveRowAtIndexPath:(NSIndexPath*)indexPath {
  // 最後のセル以外なら YES
  return ( dataSource_.count > indexPath.row + 1 );
}
```

これは、UITableViewDataSource プロトコルの **tableView:canMoveRowAtIndexPath:**メソッドで、任意の row に対する戻り値を **NO** とすることでその row の置き換えを制限するものです。これを利用することで、特定のセルが置き換えできないように指定することができます。**9.3.6** のサンプルコードでは、一番下の「新規追加」というセルを移動不可に設定しています(図.A)。

◆図.A

しかし、これだけだと、「新規追加」のセルを移動することはできなくても、他のセルを「新規追加」のセルの下に移動することができてしまうことが発覚しました。今回、この点を訂正しお詫びさせていただきます。

他のセルを「新規追加」の下に移動できなくするには、UITableViewDelegate の **tableView:targetIndexPathForMoveFromRowAtIndexPath:toProposedIndexPath:**メソッドを使います。

```
- (NSIndexPath*)tableView:(UITableView*)tableView

targetIndexPathForMoveFromRowAtIndexPath:(NSIndexPath*)sourceIndexPath
    toProposedIndexPath:(NSIndexPath*)proposedDestinationIndexPath
{
  if ( dataSource_.count > proposedDestinationIndexPath.row + 1 ) {
    return proposedDestinationIndexPath;
  } else {
    return sourceIndexPath;
  }
}
```

これは、セルの移動の挙動を細かく設定するためのメソッドで、

- 移動前の場所が **sourceIndexPath**
- 移動予定の場所が **proposedDestinationIndexPath**

にそれぞれ渡されてきます。

このメソッドの戻り値に指定した場所が実際の移動先になるため、特に挙動を変更しない場合は proposedDestinationIndexPath をそ のまま返します。逆に言えば、移動を許可したくない場合に sourceIndexPath を返してやれば、移動がされなかったという扱いにすることが可能です。

今回で言えば、移動予定の場所(proposedDestinationIndexPath)が「新規追加」の場所よりも下だった場合に sourceIndexPath を返すようにすることで、「新規追加」の場所を常に一番下に保つことが可能になります。

●訂正情報-2：UITableViewCell にコントロールを追加する方法

9.4.8 セルにコントロールを追加 (P.319) でセルのカスタマイズについて以下のようなコードを紹介しています。

■誤ったコード

```
- (UITableViewCell*)tableView:(UITableView*)tableView
  cellForRowAtIndexPath:(NSIndexPath*)indexPath
{
  static NSString* identifier = @"basis-cell";
  UITableViewCell* cell = [tableView
dequeueReusableCellWithIdentifier:identifier];
  if ( nil == cell ) {
    cell = [[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleDefault
                              reuseIdentifier:identifier];
    [cell autorelease];
  }
  cell.textLabel.text =
    [[dataSource_ objectAtIndex:indexPath.section]
objectAtIndex:indexPath.row];
  switch ( indexPath.section ) {
    case 0: //< セルに UIImageView を追加
      [cell.contentView addSubview:[self imageViewForCell:cell
withFileName:@"Samurai.png"]];
      break;
    case 1: //< セルに UISwitch を追加
      [cell.contentView addSubview:[self switchForCell:cell]];
      break;
    case 2: //< セルに UISlider を追加
      [cell.contentView addSubview:[self sliderForCell:cell]];
      break;
    default:
      break;
  }
  return cell;
```

```
}
```

これは、**図.B** のようにセルに様々なコントロールを追加するためのサンプルコードです。

図.B

これは誤ったコードで、この例のようにセルの再利用が発生しにくいケースでこそ問題は顕在化しませんが、**セルの再利用が発生したときに、その再利用されたセルに次々とコントロールが addSubview されて画面がぐちゃぐちゃ**になってしまいます。

このように、**dequeueReusableCellWithIdentifier:**メソッドで取得したセルに対してコントロールを追加することは基本的にやってはいけません。**dequeueReusableCellWithIdentifier:**メソッドで取得したセルに対しては、値の変更のみ行うのが正しいやりかたです。

修正したサンプルコードを以下に添付させていただきます。

なおここではセルを作成する流れがわかる箇所のみ抜粋しています。全てのコードは、

●ヘッダーファイル

http://iphone-book-sample.googlecode.com/svn/trunk/Chapter9/TableSample/Classes/SampleForCustomizedCell.h

●実装ファイル

http://iphone-book-sample.googlecode.com/svn/trunk/Chapter9/TableSample/Classes/SampleForCustomizedCell.m

に置いてありますので、全体が必要な場合にはお手数ですがこちらをご参照ください。

□修正後のサンプルコード

```
- (UITableViewCell*)tableView:(UITableView*)tableView
  cellForRowAtIndexPath:(NSIndexPath*)indexPath
{
  // カスタムセルの種類によって再利用のための ID を変えること
  static const id identifiers[3] = { @"image-cell", @"switch-cell",
@"slider-cell" };
  NSString* identifier = identifiers[ indexPath.section ];
  UITableViewCell* cell = [tableView
dequeueReusableCellWithIdentifier:identifier];
  if ( nil == cell ) {
    // セルにコントロールを追加する場合は、セルの作成時に行う
    // ここでは各カスタムセルごとに UITableViewCell のサブクラスを定義して使っている
    // 各サブクラスの実装は
http://iphone-book-sample.googlecode.com/svn/trunk/Chapter9/TableSample/Classes/SampleForCustomizedCell.m 参照
    // なお、各 initXXX メソッド内でそれぞれのセルに必要なコントロールを addSubview している
    switch ( indexPath.section ) {
      case 0:
        cell = [[[CellWithImageView alloc]
initWithReuseIdentifier:identifier] autorelease];
        break;
      case 1:
```

```
        cell = [[[CellWithSwitch alloc]
initWithReuseIdentifier:identifier] autorelease];
        break;
      case 2:
      default:
        cell = [[[CellWithSlider alloc]
initWithReuseIdentifier:identifier] autorelease];
        [(CellWithSlider*)cell setDelegate:self];
        break;
    }
  }
  // 再利用のときにも通るコードでは、各コントロールの値の変更のみ行う
  switch ( indexPath.section ) {
    case 0:
      [[cell imageView] setImage:[UIImage imageNamed:@"Samurai.png"]];
      break;
    case 2:
      {
        // UISlider 付のセルなら値を設定
        NSNumber* value = [self.sliderValues objectAtIndex:indexPath.row];
        CellWithSlider* cellWithSlider = (CellWithSlider*)cell;
        cellWithSlider.slider.value = [value floatValue];
        cellWithSlider.row = indexPath.row;
      }
      break;
    default:
      break;
  }
  cell.textLabel.text =
    [[self.dataSource objectAtIndex:indexPath.section]
objectAtIndex:indexPath.row];
  return cell;
}
```