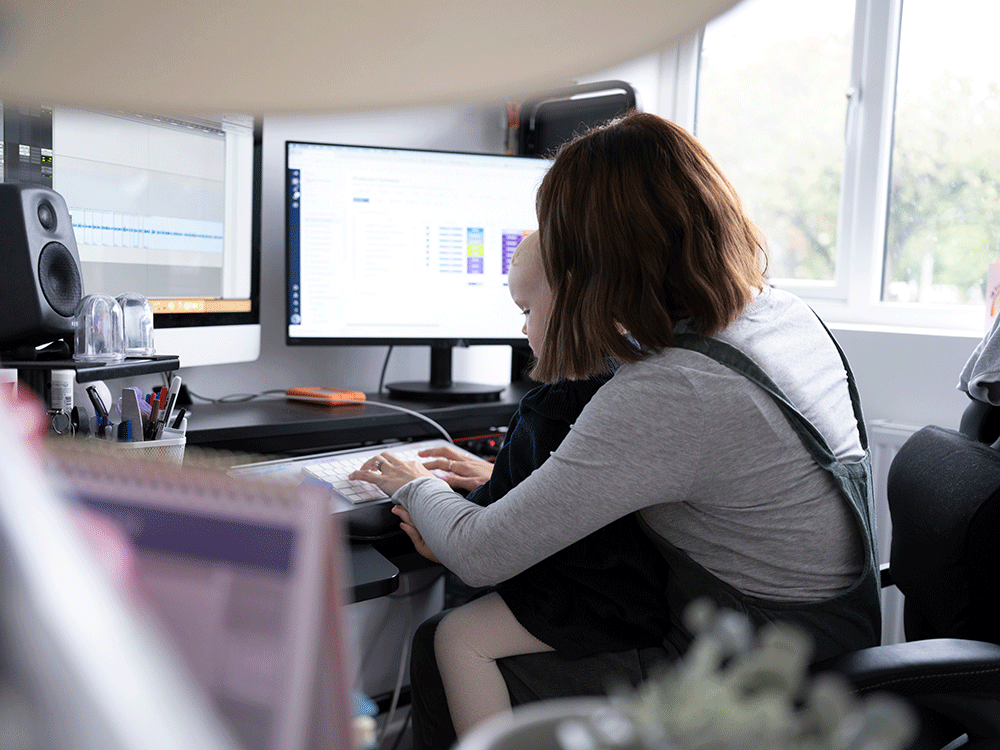
2024

みさお 松本

[会社名]

2024年2月12日



プログラム基礎知識

内容

[プログラム基礎知識 1](#_Toc165674586)

[VS2019communityでプロジェクトを作成 1](#_Toc165674587)

[イベントとイベントハンドラーとは？ 6](#_Toc165674588)

[イベントハンドラーの構文 13](#_Toc165674589)

[VisualBasicの場合 13](#_Toc165674590)

[C#の場合 14](#_Toc165674591)

[「ASP. NET 式」 イベントドリブンモデル の 特殊 性 16](#_Toc165674592)

[ASP.NETページにおけるイベント処理 20](#_Toc165674593)

## プログラム基礎知識

### VS2019communityでプロジェクトを作成

VisualStadio２０１９を起動して、「新しいプロジェクトの作成」ダイアログ（下図）を開き、下図の黄色のドロップダウンで「VisualBasic」を選択してください。次に「ASP.NET　Webアプリケーション」を選択して、右下の「次へ」ボタンをクリックします。

|  |
| --- |
| グラフィカル ユーザー インターフェイス, テキスト, アプリケーション  自動的に生成された説明 |

次に、プロジェクト名を「SkyOceanLecture」と入力して、その他はデフォルトのまま、「作成」ボタンをクリックします。（下図）

|  |
| --- |
| グラフィカル ユーザー インターフェイス, アプリケーション  自動的に生成された説明 |

下図のようなダイアログが表示されるので、そのまま「作成」ボタンをクリックしてください。

|  |
| --- |
| グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール  自動的に生成された説明 |

下図のように、ソリューション「SkyOceanLectue」の配下に「SkyOceanLecture」プロジェクトが出来たことを確認してください。

|  |  |
| --- | --- |
|  | 左側の「ツールボックス」右側の「ソリューションエキスプローラ」が表示されていない場合はメニューの「表示」を操作してください。 |

次にプロジェクトルート内に新しいフォルダ「Lecture-1」を作成して、そのフォルダの中に下図のように「マスターページを含むWebフォーム」「Hello.aspx」を作成してください。

|  |
| --- |
| グラフィカル ユーザー インターフェイス, アプリケーション  自動的に生成された説明 |

次に、下図のようにHello.aspxのデザイン画面とソースコードビューに示した様にページを作成してください。

|  |
| --- |
| グラフィカル ユーザー インターフェイス, アプリケーション  自動的に生成された説明 |

次に、「btnSend」のクリックイベントを作成して、下記のコードを追加してください。

|  |
| --- |
| Protected Sub btnSend\_Click(sender As Object, e As EventArgs) Handles btnSend.Click  lblGreet.Text = "こんにちは、" & txtName.Text & "さん！"  End Sub |

### イベントとイベントハンドラーとは？

ASP. NET の ページ では、 さまざま な イベント が 発生 し ます。 イベント とは、 たとえば、「 ボタン が クリック さ れ た」「 テキスト ボックス の 内容 が 変更 さ れ た」「 チェック ボックス が チェック さ れ た」 など、 ページ で 発生 する でき ごと の こと です。 イベントドリブンモデル（ イベント 駆動 モデル） とは、 この 随所 で 発生 し た イベント に 応じ て 実行 す べき コード を 記述 する プログラミング モデル の こと を 言い ます。 たとえば、

先ほど、下図のようなページを作成しました。

|  |  |
| --- | --- |
| グラフィカル ユーザー インターフェイス  中程度の精度で自動的に生成された説明 |  |

下図のように各コントロールのプロパティを設定してください。

|  |
| --- |
| テキスト  自動的に生成された説明 |

ブラウザーで表示すると下図のように表示されます。

|  |
| --- |
|  |

上図のプログラムの目的は

**［送信］ ボタン が クリック さ れ たら（ ＝ イベント）、**

**「こんにちは、 ＜ テキスト ボックス の 内容 ＞ さん！」**

**という 文字列 を ラベル に 表示 し なさい（ ＝ 実行 す べき コード）**

という プログラムを作りたいと思います。

この とき、 ボタン クリック という イベント に対して、 その 処理 を 定義 する コード の こと を、 **イベントハンドラー、** または **イベント プロシージャ** と 言い ます。

下図にイベントの処理の様子を示しました。

**イベントとイベントハンドラー**

|  |
| --- |
| ダイアグラム  中程度の精度で自動的に生成された説明  Protected Sub btnSend\_Click(sender As Object, e As EventArgs) Handles btnSend.Click  'lblGreet.Text = "こんにちは、" & txtName.Text & "さん！"  End Sub  あらた  あらた |

赤線の下がプログラムによる、処理。（サーバー側が処理する）

|  |
| --- |
|  |

イベントハンドラー の 生成 イベント と イベントハンドラー を 紐 付ける には、 いくつ かの 方法 が あり ます。

① デザインビュー で コントロール を ダブル クリック する 最も シンプル な 方法 です。

これ によって、 それぞれ の サーバー コントロール で あらかじめ 決め られ た デフォルト の イベントハンドラー（ 最も よく 利用 さ れる イベントハンドラー） が 自動 生成 さ れ ます。 デフォルト の イベントハンドラー は、 使用 する サーバー コントロール によって 異なり ます（ たとえば、 Button コントロール では Click イベント が デフォルト です）。 なお、 デザインビュー の なに も ない 場所 で ダブル クリック し た 場合 には、 Page\_ Load イベントハンドラー（ ページ ロード 時 に 実行） が 自動 生成 さ れ ます。

② プロパティウィンドウ から 選択 する イベントハンドラー は、 プロパティウィンドウ からも 生成 でき ます。 デザインビュー で コントロール を 選択 し てから プロパティウィンドウ 上部 の（ イベント） ボタン を クリック し ます。 対応 する イベント 一覧 が 表示 さ れる ので、 目的 の イベント 名（Click） を ダブル クリック し て ください（下図）

|  |
| --- |
|  |

イベントハンドラー 名 が 自動的 に セット さ れ、 同時に 自動 生成 さ れ た イベントハンドラー が コード エディター に 表示 さ れ ます。

なお、 イベントハンドラー を すでに 用意 し て いる 場合 には、 選択 ボックス に 候補 と なる イベントハンドラー が 表示 さ れる ので、 そこ から 選択 し ても かまい ませ ん。（イベントハンドラーの内容を変更、更新する時にこの機能を使用すると便利です）

繰り返し実行してクライアント側とサーバー側の関連を想像しながら確認をして、自分がコードを記入した内容が実行される様子をしっかりと把握してください。このことがプログラミングの出発点になります。

参考：下記はC#のコードです。

|  |
| --- |
| protected void btnSend\_Click(object sender, EventArgs e)  {  lblGreet.Text = $"こんにちは、{txtName.Text}さん！";  //lblGreet.Text = Server.HtmlEncode($"こんにちは、{txtName.Text}さん！");  } |

### イベントハンドラーの構文

### VisualBasicの場合

Visual Studio を 利用 し て いれ ば、 イベントハンドラー の 最低限 の 骨組み は 自動 生成 さ れ ます。 その ため、 特に 最初 の うち は 構文 そのもの を 意識 する 機会 は ない かも しれ ませ ん。 しかし、 イベントハンドラー の 構文（ シグニチャ） を 最低限 理解 し て おく こと は、 今後、 プログラミング を 行っ て いく 中 で 欠か せ ない ステップ です。 先程イベントを作成 し た Hello.aspxのHTMLコードビューを再び 確認 し て み ましょ う。

|  |
| --- |
| < div > 名前：< asp: TextBox ID =" txtName" runat =" server"></ asp: TextBox >  < asp: Button ID =" btnSend" runat =" server" Text =" 送信" /> < br />  < asp: Label ID =" lblGreet" runat =" server"></ asp: Label > </ div > |

プログラムコード

|  |
| --- |
| Protected Sub btnSend\_Click(sender As Object, e As EventArgs) Handles btnSend.Click  lblGreet.Text = "こんにちは、" & txtName.Text & "さん！"  End Sub |

コード（. aspx. vb） に「 Handles btnSend. Click」 の よう な 記述 が ある のが 確認 でき ます。 Visual Basic では、 これ によって「 Button コントロール btnSend の Click イベント」 と btnSend\_ Click イベントハンドラー（ メソッド） とを 紐 付け て いる わけ です。

### C#の場合

ソースコード

|  |
| --- |
| < div > 名前：< asp: TextBox ID =" txtName" runat =" server"></ asp: TextBox >  < asp: Button ID =" btnSend" runat =" server" Text =" 送信" OnClick =" btnSend\_ Click" /> ─ ─ ─ ❶  < br /> < asp: Label ID =" lblGreet" runat =" server"></ asp: Label > </ div > |

プログラムコード

|  |
| --- |
| グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, Word, メール  自動的に生成された説明 |

#### ❶ イベント と イベントハンドラー の 紐 付け を 理解 する

サーバー コントロール と イベントハンドラー とを 関連付け て いる のは、 OnXxxxx プロパティ の 部分 です。 この 場合、 Button コントロール btnSend が クリック さ れ た タイミング で（ Click イベント が 発生 し た タイミング で）、 btnSend\_ Click イベントハンドラー が 呼び出さ れる こと を 表し ます。

＜ コントロール の ID 値 ＞\_＜ イベント 名 ＞

と する のが 一般的 で、 Visual Studio でも この 規則 で 命名 さ れ ます が、 名前 そのもの に 関連 付け の 意味 は あり ませ ん（ Hoge メソッド を Click イベント に 紐 づけ ても かまい ませ ん）。 あくまで コード を 読み やすく する 意味 での 命名 と 理解 し て ください。

Note： その 性質 上、 イベントハンドラー を 削除 する 際 には 要注意 です。 というのも、 分離 コード（. aspx. cs ファイル） の 側 で メソッド（ イベントハンドラー） を 削除 する だけでは 不十分 です。. aspx ファイル に OnXxxxx 属性 の 記述 が 残っ て しまう から です。 イベントハンドラー を 削除 する には、 プロパティウィンドウ で 関連 付け を 削除 し てから、 イベントハンドラー を 削除 する よう に し て ください。

#### ❷ イベントハンドラー の 引数 を 理解する

続い て、 イベントハンドラー に対して 渡さ れる 引数 に 注目 し て み ましょ う。 まず、 第 1 引数 の object（ sender） は イベント の 発生 元 と なっ た オブジェクト を 指し ます。 Hello. aspx. cs の 例 で 言う なら ば、 Button コントロール です。 これ は すべて の イベントハンドラー において 共通 の 引数 です。 一方、 第 2 引数 は イベントハンドラー の 種類 によって 異なり ます。 発生 し た イベント に関する 情報 を 格納 する EventArgs クラス、 もしくは EventArgs クラス の 機能 を 引き継ぐ 派生 クラス（ EventArgs 派生 クラス） を 指定 でき ます。 たとえば、 後述 する GridView コントロール（ グリッド 表） では、 表 内 の［ 削除］ ボタン を クリック し た タイミング で RowDeleting イベント が 発生 し ます が、 その とき に 呼び出さ れる イベントハンドラー の シグニチャ は、 次 の とおり です。 protected void grid\_ RowDeleting( object sender, GridViewDeleteEventArgs e) { // 任意 の コード } RowDeleting イベントハンドラー では、 第 2 引数 として GridViewDeleteEventArgs オブジェクト e を 受け取り ます。 イベントハンドラー では、 たとえば その RowIndex プロパティ に アクセス する こと で、 削除 行 の インデックス 番号 を 取得 でき ます。 lblResult. Text = $"{ e. RowIndex + 1} 行 目 が 削除 さ れ まし た。"; GridView コントロール そのもの について は、 改めて 解説 し ます。 ここ では EventArgs 派生 クラス を 利用 する こと で、 イベントハンドラー の 中 で イベント に 関係 する さまざま な 情報 を 知る こと が できる という こと を おさえ て おい て ください。 イベント と イベントハンドラー で 受け取る こと が できる EventArgs 派生 クラス の 関係 は、 あらかじめ 決まっ て い ます。 主 な 対応 関係 について は、 それぞれ 関連 する サンプル で 触れ て いる ので、 そちら を 参照 し て ください。

C#の場合＿ここで終了。

### 「ASP. NET 式」 イベントドリブンモデル の 特殊 性

イベントドリブンモデル による 開発 の 基本 が 理解 でき た ところ で、 もう少し 詳細 を 見 て いく こと に し ましょ う。 イベントドリブンモデル は、 もともと Visual Basic などに 代表 さ れる Windows アプリケーション 開発 で 採用 さ れ て い た プログラミング モデル です。 Windows アプリケーションと 限り なく 同じ 感覚 で Web アプリケーション を 開発 できる のは、 ASP. NET の 最大 の 特長 です。 もっとも、 ASP. NET の イベントドリブンモデル は あくまで 擬似 的 な もの で あり、 Windows アプリケーション の それ とは あくまで 異なる 点 に 注意 し て ください。 具体的 には、 イベント を 処理 する 場所 が 違い ます。 Windows アプリケーション では、 クライアントマシン の 上 で イベント が 発生 し たら、 それ は そのまま クライアント 側 で 処理 さ れ ます。 つまり、 イベント が 発生 する 場所 と イベント を 処理 する 場所 は、 同一 です。 しかし、 ASP. NET では、 クライアント で 発生 し た イベント は、 いったん サーバー 側 に 送信（ ポスト） さ れ た うえ で 処理 さ れ ます。 たとえば、 ブラウザー 上 で ボタン が クリック さ れ た とき、「 ボタン が クリック さ れ た」 という 情報 は、 いったん サーバー 上 に ある 現在 の Web ページ（ 自分自身） に対して 送信 さ れ ます。 ASP. NET では、 これ を イベントハンドラー で 処理 し た 後、 その 結果 を 反映 し た ページ を 応答 する わけ です（下図）

#### ポストバック処理

|  |
| --- |
| 初回以外のイベント発生時は、全てがポストバック　＝　True　と言える。  ユーザーの操作  初回  ポストバック　＝　True  ポストバック　＝　False  あらた |

上の図のようにユーザーの操作によって 送信 し、 処理 する こと を、ポスト バック（ Postback） と 言い ます。 また、 イベント の 発生 に 伴っ て、 処理 が サーバー／ クライアント 間 を 行き来 する こと を サーバーラウンドトリップ と 言い ます。 この よう な「 ASP. NET 式」 イベントドリブンモデル を 理解 し て おく こと は、 ASP. NET プログラミング で 思わ ぬ 不具合 に 遭遇 し ない ため にも、 とても 重要 な こと です。 以下 では、「 ASP. NET 式」 イベントドリブンモデル 固有 の 注意 点 を いくつ か まとめ て おき ます。

Note： ここ で 注目 し て いただき たい のは、 ポスト バック も サーバーラウンドトリップ も、 ASP. NET 独自 の 仕組み では ない という こと です。 これら は、 いずれ も HTTP による 一般的 な Web アプリケーション の 仕組み に すぎ ませ ん。 ASP. NET の 魅力 とは、 独自 の アーキテクチャ では ない、 ごく 汎用 的 な プロトコル（ 手続き） の 範疇 で、 新た な 開発 モデル を 実現 し て いる 点 に あり ます。

**スケートボードの達人の言葉　＝　あきらめなければ、絶対出来る。**

**出来たときに、昔の自分を超えた。**

#### ページイベントは常に発生する

たとえば、先ほど作成したHello.aspx.vbを、下記のように書き換えてみましょう。Page\_Loadイベントハンドラーは、ASP.NETページが呼び出されたときにコールされるイベントハンドラーで、ここではTextBoxコントロールtxtNameに初期値として「尾崎」をセットしています

（イベントハンドラーそのものはデフォルトで用意されています）。

|  |
| --- |
| Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load    　　　　　‘page\_Loadが呼び出されるたびにこれを実行する  txtName.Text = "尾崎"  End Sub |

ページを呼び出したタイミングで、今度はテキストボックスに「尾崎」と表示されます。ここま

では正しい動作です。

しかし、テキストボックスに「松本」と入力して［送信］ボタンをクリックするとどうでしょう。テキストボックスに入力した文字列がラペルに反映されません。

名前を入力してボタンをクリック入力した名前がラベルに表示されない

これはどういうことでしょうか。

本来の意図ではボタンクリックによってbtnSend Clickイペントハンドラーが呼び出され、「こんにちは、山田さん」という文字列がラベル上に表示されるはずです。しかし、画面上には「こんにちは、尾崎さん」と表示されてしまうのです。テキストボックスの表示もいつの問にか「尾崎」に戻ってしまっています。　一見、プログラムに誤りはないように見えます。しかし、これこそが「ASP.NET式」イベントドリブンモデルの落とし穴なのです。

### ASP.NETページにおけるイベント処理

#### ページに関連するイベントの処理

ASP.NETでは下図に示す順序で**「イベントの発生」**を決めています。

|  |
| --- |
| ダイアグラム  中程度の精度で自動的に生成された説明 |

上記のイベントの実践についてはオンラインセミナーに於いて講師が解りやすく実践していきます。

#### Page.IsPostBackを理解する

##### メニューからページを呼び出す

マスターページのメニューからHello.aspxページを呼び出すようにメニュー項目を追加してください。網掛け部分

|  |
| --- |
| </button>  <a class="navbar-brand" runat="server" href="~/">アプリケーション名</a>  </div>  <div class="navbar-collapse collapse">  <ul class="nav navbar-nav">  <li><a runat="server" href="~/">ホーム</a></li>  <li><a runat="server" href="~/Lecture\_1/Hello.aspx">Helloページ</a></li>  <li><a runat="server" href="~/About">詳細</a></li>  <li><a runat="server" href="~/Contact">問い合わせ</a></li>  </ul>  </div>  </div> |

|  |
| --- |
| 参考：　<a hrefリンクの場合  “href=”まで入力したら、VisualStadioのスマート機能によりリンク先のTree構造を表示してくれるので、そこから目的のファイルを選択してください。下図    「URLの選択」をクリック。下図が表示されます。    Hello.aspxが入っているフォルダーを開いてHello.aspxを選択して、「OK」ボタンをクリックして終了します。 |

マスターページのコードビュー画面を閉じて、「Default.aspx」のコードビュー画面を表示しておいてください。

ASP.NETのアプリを実行している場合、「Page＿Load」イベントは頻繁に発生します。例えばメニューから「○○.aspxページを初めて呼び出した」場合（前者）と○○.aspxページ上の「ボタンをクリック」した場合（後者）、双方の場合に発生します。プログラマーは（前者）、（後者）どちらが原因で「Page＿Load」イベントが発生したか？を知る必要があります。その違いを判別するのに、「Page.IsPostBack」というプロパティが用意されています。

その使い方は？

1. 「ページが最初に呼び出された（例えば、開発中であれば○○,aspxをブラウザーで表示した場合）」は「Page.IsPostBack」は「False」を返します。
2. 「ユーザーが○○.aspxページ上のボタンをクリックした」（サーバーコントロールによるPostBackを発生させた）場合は「Page.IsPostBack」は「True」を返します。

したがって、[VisualBasicのプログラム構文（If. . .Then . . Else）](https://learn.microsoft.com/ja-jp/office/vba/language/concepts/getting-started/using-ifthenelse-statements)

1. If Page.IsPostBack Then. . . （Page.IsPostBackがTrueなら）
2. 又は、If Not Page.IsPostBack Then. . . (Page.IsPostBackがFalseなら)

という式が成り立ちます。

実際に実践してみましょう。下記のコードをPage＿Loadイベントハンドラーに追加してください。

|  |
| --- |
| Protected Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load  ‘セミナーではLabelを配置して、IsPostBackの値を表示する  If Not Page.IsPostBack Then  txtName.Text = "尾崎"  End If  ‘セミナーでは　Else If　も追加したコードを実践します。  End Sub |

尚、上記の②の文章の（サーバーコントロールによるPostBackを発生させた）場合については、変更系（TextBox,DropDownList,CheckBox等のコントロール）のサーバーコントロールに用意されている「AutoPostBack」プロパティによって発生させられたポストバックの事です。

この変更系（TextBox,DropDownList,CheckBox等のコントロール）のサーバーコントロールに用意されている「AutoPostBack」プロパティについては後で実践していきます。

ここまでの重要なキーワードは

「ページが最初に呼び出された（例えば、開発中であれば○○,aspxをブラウザーで表示した場合）」は「Page.IsPostBack」は「False」を返します。

それ以外は「Page.IsPostBack」は「True」を返します

でした。

#### 開発中に素早く実行画面を表示する

次に、プロジェクトを表示してください。赤い四角の部分をクリックする。

|  |
| --- |
|  |
|  |

下図の「Default.aspx」ページがデフォルトで表示されます。

|  |
| --- |
| グラフィカル ユーザー インターフェイス, テキスト, メール  自動的に生成された説明 |

表示された「Default.aspx」の上部メニューの「Helloページ」（赤い四角）の項目をクリックしてください。

表示された画面を見て、自分がプログラムした内容と画面との整合を想像しながら作業を進めてください。

|  |
| --- |
| グラフィカル ユーザー インターフェイス, テキスト, アプリケーション  自動的に生成された説明 |

txtNameの“尾崎”を削除して、“山本”と入力して「ｂｔｎSend」をクリックしてください。下図のような結果になります。

|  |
| --- |
| グラフィカル ユーザー インターフェイス, テキスト, アプリケーション  自動的に生成された説明 |

閉じてください。

**今のうちにイベントの発生とそれを利用するという事を理解しましょう。**

btnSendをクリックー＞ポストバックが発生―＞サーバー側は、決められた順序でページイベントを処理する、下図がイベント発生順序です。

#### イベント発生順序

|  |  |
| --- | --- |
|  | 現時点のHello.aspxでは、＊印の　イベントにプログラムしてあるので、Hello.aspxでのアプリの実行は  「Page＿Load」イベントと「ｂｔｎSend」  イベントのプログラムコードが実行されるのです。VisualStudioがイベントを発生する順序は左図の順序です。 |

現時点でのHello.aspxにおけるイベント名は、「Page＿Load」、「ｂｔｎSend」イベントがプログラムしてあります。

イベントの発生を利用する、しない、はプログラムするか、しないか、によって決まります。（イベントを掴むか、掴まないか　とも言う）

ここまでで、IsPostBackを理解したら、再び「Todo」アプリの「Lecture＿４」の６６ページに戻ります。