Razor構文による学習支援システムの構築

青木恒夫＊

1．は　じ　め　に

中日本自動車短期大学（以下「本学」とする）では，基礎学力の不足する学生への学習支援を明確に推進するため，平成23年度に学習支援センターを設立した。この運用を効果的にサポートする目的で，データベース，学内LANおよびWeb技術を活用した学習支援システムを構築することになった。今回，本システムをWebプログラミングの最新技法であるRazor構文によって作成する機会があったので，システムの概要と共に構築に必要な環境整備ならびにプログラミング技術を解説する。

2．学習支援システムの概要

2．1　構築の目的

近年の入試多様化に伴い，入学する学生の学力レベルや就学に対するモチベーションには大きな差が生じている[1]。この状態を放置すると，本学の目的の一つである自動車整備士教育が十分機能しないばかりか，学生個々の満足度を高く維持することも難しくなる。全ての学生が当初の目的を果たし，満足できる学生生活を送るためには，出来るだけ早い段階で個別に最適化された学習や生活全般にわたる支援が必要である。

学習支援センターの設立準備としてワーキング・グループが組織され，前年度末からさまざまな課題や対策を議論してきた。その結果，これまで教員個々が講義やゼミ，あるいは日常的な接触で断片的に把握していた学生の様態を，例えば「学生カルテ」のような形で集約し，これを全ての教員が共有すれば，今まで気付かず埋もれてしまっていた様々な情報が明確になり，より効果的な個別支援が可能であるという結論を得た。

学生個々の情報を集約し，これを把握する手段として，これまで学生カードが活用されてきた。厚手のカード用紙に学生番号，氏名，読み，住所や携帯電話番号などの連絡先，保護者情報などといった基本情報，ゼミや面談等を通した支援履歴，さらには特記事項を手書きで記入し，担任がこれを保管する。カードに記載できる情報量は限られているので，欠席や成績などの情報は教

＊ 中日本自動車短期大学 MSE学科 准教授

務課などの別チャンネルから入手して指導の基礎資料として活用する。この方法は基本的に指導する担任のみがカードを閲覧，利用することを前提としているので，学生のプライバシー保護には有効であるが，情報ソースの少なさや情報の共有が出来ないなどの欠点を持つ。指導のポイントとなる情報を出来る限り多く把握し，より早く的確に支援するためには，複数教員からの情報が集約され，この情報を教員全体で共有出来るシステムが必要である。今回，この目的を実現するため，学生情報を学内 LAN で閲覧，追加できる電子化された学生カードの構築を行うことになった。

2. 2　機能の概要

学生の個人情報を扱うことから，ユーザ ID とパスワードを使った認証画面からログインする（図 2. 1)。ユーザ ID とパスワードについては，事前に全教員へ配布している。ID にはパーミッション（権限）が付与されていて，一般教員は学習支援履歴の追加，自分自身が記入した学習支援内容の修正および教員自身の基本情報を更新できる。

図 2. 1　ログイン認証画面

ログイン直後の Welcome ページでは，新しく追加された支援履歴の見出し一覧が表示され，最新情報へのアクセスを容易にしている（図 2. 2)。図では個人情報保護の観点から，学生番号と氏名をマスクしているが，表示されている学生番号をクリックすると直接，個人情報ページ（以下「学生カルテ」とする）を表示することが出来る。

図 2. 2　ログイン直後の Welcome 画面

学習支援をサポートする全教員が等しく情報を共有する必要があることから，見逃しが無いよう，最近 7 日以内 に記載された支援履歴について，見出し一覧を表示している。見出しの内容は，記録日時，学生番号，学生名，支援の種別である。なお，ログイン後は，全てのページ上部にログインユーザ名が表示されている。また，ログイン後，何も操作をしないで 1 時間放置すると，自動的にログアウトするようセキュリティに配慮している。更に，ログイン，ログアウトの日時，ログイン中の閲覧履歴などを全てログに記録しているので，あってはならないが，漏洩事故が発生した場合の調査に利用できる。

ログイン後の各ページ上部に表示されているメニュータブをクリックすると，目的のページへ

移動することができる。多くのユーザが最初に利用するのが Table ページで，支援履歴の一覧を表示するインデックス的な働きをする（図 2. 3)。Table タブをクリックすると，既定では，現年度の 1 年生全ての支援履歴概要が一覧表示される。既定で 1 年生としたのは，入学直後の出来るだけ早い時期に支援を開始することが効果的であり，教員に早めの全体把握をしてもらいたいという意図がある。Table ページには，選択されたグループの学生番号，学生名，支援記載日，支援種別，支援の概要（先頭40文字）が一覧表示され，学生番号をクリックすると該当する学生カルテへ移動できる。

図 2. 3　支援一覧を表示する Table ページ

Table ページの上部には検索用のフォームが置かれ，ここに一覧表示を希望する検索対象（学年，学科，クラスなど）を示す学生番号の一部を入力，検索 ボタンを押して表示する仕組みになっている。検索は前方一致で行われ，例えば「2011年度入学の MSE 学科生」を表示したい場合は，「2011M」を検索ワードとして検索を行う。また，学生名の読み順，支援日の逆順，支援コード順に並べ替える検索オプションも用意されている。

通常，一人の学生の支援を開始すると，複数個の支援履歴が発生する。一覧表示で，同じ学生番号の支援履歴が羅列する場合，学生番号欄と学生名欄に「↑」（上矢印）を表示し，複数履歴が存在することを明確にしている。

Table ページで学生カルテを表示したい学生が見つかれば，表示されている学生番号をクリックする。ブラウザで閲覧履歴を残すよう設定をしている場合，以前に学生カルテを表示したことがあれば，学生番号はピンク色で表示され，既表示の有無を明確にした。アクセス履歴のない学生番号は，通常文字と同じネイビー色に設定している。

Table ページの学生番号をクリックするか，Karte ページの学生番号検索により，目的とする学生カルテ（学習支援記録）が表示できる（図 2. 4)。

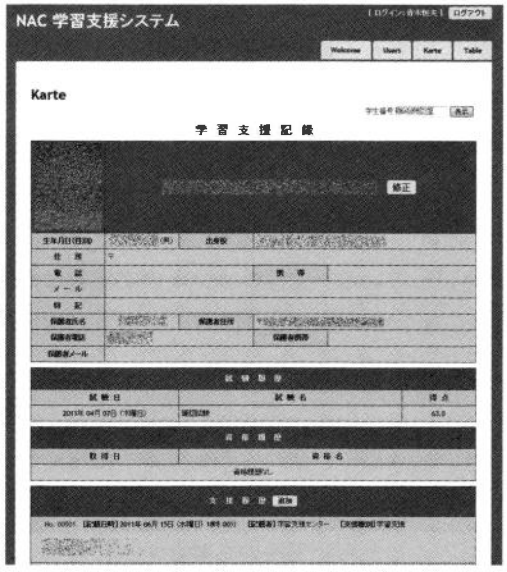

図 2. 4　学生カルテ（Karte ページ）

カルテは診療録（Karte：ドイツ語のカード

の意，英語では Medical Record）のことであるが，馴染みやすく分かりやすいということから，他のページ名に使っている英語表記とは無関係に採用した。

学生カルテは学習支援システムのメインページとなる。表示される情報をカテゴリ別に以下に示す。

上段から，①学生基本情報としての，学生顔写真，学生番号，学生名，学生名読み，生年月日，性別，出身高校，高校の設置区分，全日制区分，高校学科名，卒業年度，連絡先（住所，電話番号，携帯番号，ｅメールアドレス）および特記事項を，②保護者情報として，保護者氏名，連絡先（住所，電話番号，携帯番号，ｅメールアドレス）を表示する。続くカテゴリは③試験履歴である。これは，定期試験などとは別に実施される学力試験，選抜試験，自動車整備士講習での学科，実技試験などの得点が100点満点に換算した実数で表示される。試験履歴の下は④資格履歴である。資格履歴は学内で行われる各種資格講習並びに卒業後に判明する自動車整備士国家資格などの資格取得状況が表示される。なお，試験履歴および資格履歴の何れにも発生した日付が併記されている。

ここまでの情報が基本情報で，学習支援の基礎資料となる。基本情報はシステム管理者の方でデータベースに登録されるが，連絡先などの一部情報は，教員が修正することも出来る。また，執筆時点では実装されていないが，合わせて欠席状況（学内で稼働している欠席システムからの情報を利用）および成績情報（教務システムからの情報を利用）の表示も，2011年度中に完成するよう作業を進めている。

これら基本情報に引き続き，末尾に支援履歴が続く。支援履歴は学習支援システム最大の特徴で，学生に対する支援情報を教員が自由に追記できる。教員は講義，ゼミ，日常の接触，学習支援，学生指導などで得た情報を支援コメントとして自由に記述していく。支援コメントを追加すると，追加した単位ごとに，日時，教員名，支援種別，支援コメントが履歴として学生カルテの末尾に追加表示される。支援の種別は，執筆時点において，1. 学生指導，2. 学習支援，3. 要面談，4. 経過観察，5. 特別支援，6. その他，9. 保留の 7 種類で，支援のカテゴリを大まかに分類している。支援コメントの文字数は最大漢字 4000文字まで許されており，教員は自由に支援内容を記述できる。

図 2. 5　支援コメント追加ページ

全ての教員が支援履歴を追加できることから，担任が知り得ない情報も学生カルテに反映されてくる。また，全ての教員が学生カルテを閲覧出来ることから，支援の必要な学生に対する状況把握もスムーズに行える。支援コメントは追

加した教員のみ修正可能で，一般教員の持つパーミッションでは削除できない。一度追記したコメントが削除できない仕様は，いたずら書きの防止にもなるが，各教員が責任を持って記入して欲しいという意図である。なお，管理者のパーミッションを持つユーザは，自分を含めて他人の履歴を削除することが出来る。

このほかに，ログインしているユーザ自身のログイン・パスワードと連絡先（電話番号，携帯番号）が Users ページで修正可能である。

2. 3　計画している機能追加

前述したが，執筆時点において計画はしているが実装されていない機能が二つある。一つは，学内で稼働している欠席システムとのリンクである。欠席システムは本学独自のサービスで，各教員が講義ごとに学生の欠席状況をデータベースに登録し，夜間にこれが自動集計される。科目ごとの累計欠席時数が既定に達するごとに，担任および通知を希望してオンラインで登録を行った保護者宛にメールで通知するものである。学科科目の場合，欠席が 6 時間（1 コマ 2 時間カウント，合計 3 回講義分）を超えると欠席オーバで失格となるが，欠席が 4 時間となった時点，6 時間となった時点，および 6 時間を超えた三つの時点で警告メールが自動発信され，欠席オーバの早期発見，ドロップアウトの防止に一役買っている。欠席システムで得た欠席状況を学生カルテに表示できると，日々の履修状況が明確になり，就学指導に活用できると思われる。欠席システム自体は，学習支援システムとは別のサーバで稼働している別種のデータベースであるため，外部データベース参照の技術的裏付けが必要となる。

二つ目は，教務システムで稼働している成績情報の閲覧である。従来，成績データは個人情報の最たるものとして，学生個人および保護者に対する成績証明書の発行，担任に対しては履修指導時に成績一覧を開示するのみに限定されたものであった。学習支援システムをより完成させるためには，成績情報の学生カルテへの表示が不可欠と思われるが，個人情報保護の観点から導入を躊躇していた。今回，学内で調整した結果，情報の機密保持体制を確立することを条件に学習支援システムでの成績情報閲覧が許可されたので，先の欠席システムと合わせて，2011年度中に機能を追加するよう作業を進めている。

3．Razor 構文について

3. 1　Razor 構文の採用

学習支援システムを構築するにあたり，実行環境や開発言語について種々検討をした。導入コストが比較的安価でメンテナンスが容易であることも必要なので，サーバ OS としては Linux 系か Windows 系，DBMS（Data Base Management System：データベース管理システム）としては PostgreSQL や MySQL，Microsoft SQL Server，Oracle などが選択肢に入る。開発言語は，データベースとのやり取りを伴うサーバサイドのプログラミングが主となるため，従来から多用されている PHP（PHP Hypertext Preprocessor），Windows ベースではあるが Microsoft Visual

Web Developer による ASP.NET Web アプリケーション開発環境下での C# や Visual Basic などが候補に上がる。

このように開発環境の選択肢が多岐に渡るため，開発の初期段階で少々躓いたが，調査のためにネットを検索していると，2010年 7 月に Microsoft から発表された WebMatrix という新しい開発環境が注目された。よく似た名称の「ASP.NET Web Matrix」とは別物で、正式版であるバージョン1.0は2011年 1 月11日にリリースされており，Microsoft ダウンロードセンターから入手できる。

WebMatrix は，Microsoft 社が無償で提供する Web アプリケーション開発環境で，環境内に開発用の Web サーバである IIS Developer Express やデータベース MS SQL Server の簡易版となる SQL Server Compact が同梱されている。また，データベースやネットワーク連携を伴う Web プログラムの開発においては，プログラミングから動作テスト，デバッグ，完成コードのサーバへのアップロードまでを総合的にサポートしており，効率よく高機能なプログラムが作成可能である。この環境内では，ASP.NET のサーバベースコードを動的な Web ページに埋め込むため，Razor 構文と呼ばれるシンプルな書式が用いられる。今回，統合開発環境の WebMatrix については，実運用上いくつかの制約を伴うため利用を断念した。しかし，WebMatrix で採用されている Razor 構文と C# 言語との組み合わせが，非常に高い開発生産性を備えることが分かったので，未経験の開発環境ではあったが学習支援システムの構築に採用することを決定した。

3. 2　Razor 構文の概要

前述した WebMatrix で採用され，学習支援システムの開発に利用した Razor 構文についての概要を説明する。

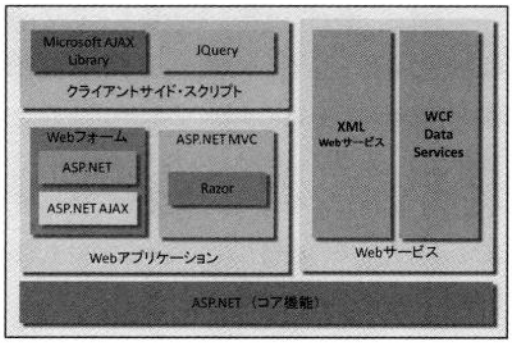

図 3. 1　ASP.NET 4 構成　2）より引用

Microsoft 社が推奨する Web アプリケーションの開発環境においては，Microsoft Visual Web Developer（Microsoft Visual Studio にも含まれる）を開発ツールとする ASP.NET（ASP: Active Server Pages の .NET Framework 対応版）で提供されるフレームワークの Web フォームや ASP.NET MVC（Model-View-Controller）が主流となっている。このうち ASP.NET MVC は Visual Studio のバージョンに合わせて次々とバージョンアップされて機能が充実している。2011年 1 月14日には ASP.NET MVC 3 がリリースされ（執筆時点では，図 3. 1の MVC 4 が最新)、これに合わせて登場したのが新しいビュー・エンジン Razor 構文（Razor Syntax）である。

図3. 2に示すように，Windows や XML Web サービス，Web アプリケーションなどの全ての Microsoft アプリケーションの開発，実行環境の基盤となるのが Microsoft .NET Framework である。ASP.NET（Active Server Pages for .NET）は，.NET Framework を利用するための Web

アプリケーション・フレームワークの開発モデルであり，Razor 構文は ASP.NET の機能を引き出す新しい View Engine の記法（Syntax）という位置付けになる[4]。従来の記法に比べ，記述コードの少なさや関連ファイルの構造化等により，シンプルでサイズの小さいプログラムの生成が可能である。'Razor' には「カミソリ」の意味があるが，Razor 構文は余分な記述を極力「そぎ落とした」構文であることを示している。また，Visual Studio などの環境が無くても単独で Web アプリケーションの開発が可能であることも特筆される。しかしながら，Visual Studio の各エディションや Visual Web Developer の環境においては，IntellisSense の補完機能や強力なデバッガーが使用可能になるため，より効率の良い開発が可能である。なお，今回開発した学習支援システムの構築においては，基本的には OS 付属の IIS 7.5およびテキストエディター（秀丸エディター）のみで開発していることを付記しておく。

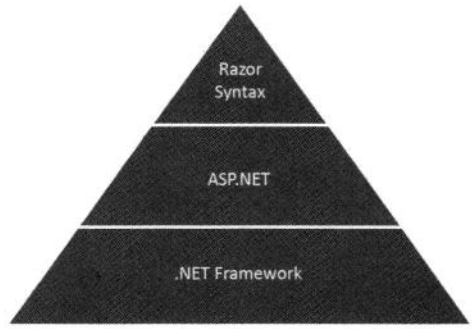

図 3. 2　Razor 構文の位置付け　3）より引用

3. 3　Razor 構文の特徴

ここで Razor 構文の主な特徴を紹介する。

3. 3. 1　対応する言語と動作

Razor 構文では，通常の HTML 文（クライアントコンテンツ）にサーバコードを埋め込んでプログラミングする。サーバはブラウザにページを送出する前にサーバコード実行し，結果を HTML にエンコードしてからブラウザに返す。サーバコードは，コードナゲットと呼ばれる「@」（アットマーク）と共に C# または Visual Basic で記述する。C# で記述されたファイル名の拡張子には「.cshtml」が，Visual Basic では「.vbhtml」が用いられる。

ユーザがフォームを通してサーバへ送った要求やシステム情報などの動的データを IIS Web サーバが受け取ると，ASP.NET のパーサー（構文解析器）がコード文を解釈，必要なサーバ処理を行ったあとに，その結果を HTML でエンコードして，元となる HTML 文に挿入する形でブラウザに送出する。例えば，データベースへ接続して特定のデータの検索し結果を表示するといった要求では，ASP.NET はコード文に記されたユーザ要求の検索条件をデータベースに渡して検索を実行，検索後にデータベースから返される結果データを Web ブラウザで表示可能な HTML 文にエンコードしてブラウザへ返す。プログラム中に記されている HTML タグ部分はそのままに，埋め込まれたコード文の実行結果のみ HTML に置き換えるので，CSS（Cascading Style Sheets：スタイルシート）や JavaScript などのクライアントスクリプトも使える。ブラウザはエンコードされた HTML 文を普通に表示するだけであるが，ASP.NET の全ての機能を利用可能な動的プログラムが作成でき，C# などのオブジェクト指向プログラミングの技法もそのまま利用す

ることができる。学習支援システムの構築では，筆者がC言語およびC++言語の利用経験があることから，比較的文法の近いC#を採用した。

3. 3. 2　コードナゲット「@」

Razor構文最大の特徴にコードナゲット「@」（アットマーク）がある。コードナゲットは，HTML文に埋め込まれるコード文や変数をHTML部と区分するために用いる。一般にコードナゲットはコード文などの前後に2カ所配置するが，Razor構文では先頭に開始タグ「@」を1個配置するだけでよい。

```
<html>
<body>
時間は <?php echo date（"Fj, Y, g:i a"）  ?> です。
</body>
</html>
```

図3. 3　PHPでの記述例（.php）

```
<html>
<body>
時間は <%= DateTime.Now %> です。
</body>
</html>
```

図3. 4　Webフォームでの記述例（.aspx）

```
<html>
<body>
時間は @DateTime.Now です。
</body>
</html>
```

図3. 5　Razorでの記述例（.cshtml）

図3. 3～3. 5は何れもブラウザに現在日時を表示させるプログラムである。図3. 3のPHPでは，date（）という指定フォーマットで日付文字列を取り出す関数にコードナゲットを適用している。図3. 4および図3. 5は，いずれもDateTimeオブジェクトから現在時刻を取得するDateTime.Nowプロパティにコードナゲットを適用している。Webフォームの場合，プロパティの前後にコードナゲットを配置しているが，Razor構文ではプロパティの先頭に「@」を配置するだけでプロパティと認識される。単純な例ではあるが，Razor構文によって記述すると，これだけでもシンプルに書き上げられる。さらに，PHPやWebフォームによる記述では，前後に配するコードナゲットが異なり，しかも複数の文字から出来ている。プログラミング時のタイプ数は必然的に多く，プログラムも見づらくなる。Razor構文では，コード文や変数の先頭に「@」を一つ配するだけで，その開始を宣言することができ，プログラムは見やすくすっきりと書ける。なによりタイプ数が大幅に削減されることは，生産性アップに繋がる。なお，複数のコード文を記述する場合は，「@{コード文1；コード文2；……；コード文n;}」のように，「{}」（ブレース）で括ってコードブロックとすることができる。コードブロック内では，適宜改行を入れて複数のコード文を記述できる。

3. 3. 3　コード文の末尾記号「;」

コードブロック内のコード文は「;」（セミコロン）で文を完結する。図3. 6の例は，コードブロック内で変数「ThisYear」と「ThisMonth」を宣言し，それぞれに現在の年と月を格納して

```
@{
   var ThisYear = DateTime.Now.Year;
   var ThisMonth = DateTime.Now.Month;
}
<p> 今は @ThisYear 年 @ThisMonth  月です。</p>
```

図3. 6　コードブロック内のコード文

いる。またブロック外では，格納された変数の値を HTML の「<p>」要素を使って表示している。このときの「@ThisYear」「@ThisMonth」はインライン式とよばれ，セミコロンは必要ない

3. 3. 4　一貫性のある外観の作成

前述したコードナゲット「@」以外にも省力化の工夫が色々見受けられる。例えば，ヘッダやフッタ，ログインユーザ名など，どのページにも共通で表示するコンテンツを，一つのファイル（コンテンツブロック）にまとめることができる。サーバはページを表示するときに，「@RenderPage{ ファイル名 }」のメソッドを見つけると，表示しているメインページ内に別ファイルで記述されたコンテンツブロックを挿入し，最終結果をマージして出力する。

コンテンツブロックの機能を更に進めて，Web ページの構造を定義するレイアウトページという機能もある。レイアウトページはメインページの先頭で読み込まれ，メインページの HTML 文やコンテンツブロックを挿入する位置を指示する。メインページはレイアウトページ内の「@RenderBody ()」メソッドで 1 回だけ呼び出し可能だが，コンテンツブロックは何度でも呼び出せる。レイアウトページ内ではスタイルシートの指定も可能で，サイト全体を一貫したデザインで作成することが出来る。

コンテンツブロックやレイアウトページを利用するメリットは，コード数を大幅に減らせ，構造の把握が容易でプログラムも見やすくなることである。さらに，共通部分の変更が単一ファイルで行えるため，メンテナンス性も非常に良くなる。

3. 3. 5　ASP.NET Razor ヘルパー

Razor 構文では，データベースから得たデータを表形式で表示したり（WebGrid Helper），データをグラフ化する（Chart Helper）など，パッケージ化された高度な機能を簡単に Web ページに取り込む「ヘルパー」と呼ばれる仕組みが用意されている。例えば，あらかじめ Twitter ヘルパーをインストールしておくと，「@Twitter.Search（“@username”）;」という簡単な 1 行を記すだけで，「username」変数で定義されたユーザを Twitter から検索し，その Twitter フィードを表示することが出来る。また，これらのライブラリは NuGet（ぬげっと：http://nuget.org/，Visual Studio 2010の拡張機能）と呼ばれる機能で管理でき，そのホームページからライブラリを直接ダウンロードすることも出来る。

3. 3. 5　8つのプログラミング Tips

文献 3）には，これまで述べてきた特徴を含む Razor 構文におけるプログラミングエッセンスが「8 つのプログラミング Tips」としてまとめられている。以下に 8 つのプログラミング Tips（一部名称の変更）を引用しておく。なお，具体的な記述については，文献 3）を参照されたい。

① @ 文字を使ってページにコードを追加する

② 中カッコでコードブロックを囲む

③ ブロックの中で，各コード文をセミコロンで終わらせる

④ 値を保存させるために変数を使う

⑤ ダブルクォーテーション記号でリテラル文字列値を囲む

⑥ コードは大小文字を区別する

⑦ フォーム値を獲得する Request オブジェクト

⑧ 条件処理 if 文

そのほか，文献 3）には具体的なプログラム例が豊富であり，Razor 構文の入門書として推薦する。

3. 3. 6 デバッグ

Razor 構文により記述したプログラムに文法的なバグが存在する場合，ローカルの開発環境であれば実行時にエラーが表示され，エラー特定の一助となる。ASP.NET が文法チェックを行ってくれるので，プログラミングはエラーが出なくなるまでデバッグを続ければよいことになる。

エラーには，プログラムの処理を続行できない「例外」も含まれる。例外が出来る限り発生しないように，例外を想定したプログラミングが必要であるが，開発当初に予知できなかった例外も，何度かのテストによりサーバから返されるエラーで発見出来る場合がある。ただ，実環境においては，ユーザのブラウザにエラーが表示されることは望ましくないので，これを回避する「try / catch」文が用意されており，エラー発生時に適切なページを表示するようプログラムすることも可能である。

ASP.NET にはデバッグ用のヘルパーが用意されている。ServerInfo ヘルパーを使用すると，① サーバの設定，② ASP.NET サーバ変数，③ HTTP ランタイム情報，④環境変数の一覧をブラウザに出力できる。また，ObjectInfo ヘルパーは，プログラム中でやり取りされるオブジェクトの型と値をブラウザに表示することが出来る。これらヘルパーによるデバッグは，追跡が難しいデータ値を可視化することができ，デバッグに有用である。

その他，ブラウザにアドオンとして追加できる Internet Explorer Developer Tools や Firebug（Mozilla Firefox 用）は，ブラウザに渡される HTML コードの検証に役立つ。

4．開発環境の整備

4. 1 環境の構築手順

前述したように，Razor 構文と C# の組み合わせを開発言語としたことから，OS および DBMS は Windows をベースとすることになる。以下，具体的な環境構築について説明する。

4. 1. 1 サーバ用 OS

Razor 構文を採用する WebMatrix のシステム必要条件は，Windows 7, Windows Vista SP1 以上，Windows XP SP3 以上，Windows Server 2003 SP2 以上，Windows Server 2008，Windows Server 2008 R2 とある。WebMatrix 利用の有無に関わらず，実行環境は Razor 構文を解釈する Web サーバである IIS のバージョンに依存するので，WebMatrix のシステム要件に従う必要が出てくる。今回は，手元にあった Windows 7 Professional 64bit Edition がインストールされている

PC 2台（職場と自宅）を開発環境に充てた。また，執筆時点では予算措置の関係で正規サーバを導入することが出来ず，公開用のサーバにおいても同じく Windows 7 Professional 64bit Edition をクリーン・インストールした一般的な PC（CPU：Core 2 Duo E7500，メモリ：PC6400 DDR 2 800MHz 2 GB×2，HDD：1 TB，バックアップ用 HDD：1 TB USB 外付け HDD）を利用する。

サーバ用 OS については，後日，本学で稼働中の Windows Server 2008（IIS バージョンは7.0）サーバに対して試験的な環境構築を行ったが、プログラムの中でファイル名拡張子が省略できないなどの予期しない挙動が発生し、そのまま移行することが出来なかった。筆者が設定を十分精査していない可能性は否定できないが，サーバ用 OS としては，Windows 7 と同じバージョンの IIS 7.5（WebMatrix も同じ IIS 7.5）を搭載する Windows Server 2008 R2 を採用することが望ましい。なお，Windows Server 2008 R 2 は64bit マシンのみ対応しているので，サーバマシンを選定する場合は注意を要する。

4. 1. 2 IIS 7.5のインストールと設定

Microsoft の Web サーバである IIS（Microsoft Internet Information Services）は、Windows OS に同梱されており、サービスを有効にすることで使用が可能となる。ここでは、Razor 構文に対応した IIS 7.5を搭載する Windows 7 Professional（Windows Server 2008 R2 もほぼ同様）を例にインストール方法を示す。なお，別途インストールに必要なプログラムは，インターネットより無料で入手可能であるが，参考に付した URL は将来変更となる可能性があるので注意されたい。

1. IIS のインストール

   ［コントロールパネル］ → ［プログラムと機能］ → ［Windows の機能の有効化または無効化］

   表示されるダイログ ボックス内の［インターネット インフォメーション サービス］の先頭にあるチェックボックスを以下のように選択する。

   ［World Wide Web サービス］ → ［アプリケーション開発機能］ を全てチェック（他は変更しない！）

   ［OK］ ボタンで必要なモジュールがインストールされる。

2. IIS の稼働テスト（ローカル）

   ローカルから http:// サーバ名 / または http://localhost でサンプルページ（iisstart.htm）の表示を確認する。

3. ネットワーク上での表示設定と確認

   Windows 7 の初期設定では，ネットワーク上の他のマシンから表示ができないため，ファイアウォールの設定を行う。

   ［コントロール パネル］ → ［Windows ファイアウォール］ → ［詳細設定］

   ［セキュリティが強化された Windows ファイアウォール］ を開く。

［受信の規則］ 中央の ［受信の規則］ で，［World Wide Web サービス（HTTP トラフィック)］ を選択し，右側の［規則の有効化］ をクリックする。

他のマシンから http:// サーバ名 / で表示されることを確認する。ただし，hosts または DNS にマシン名が登録されていることが前提である。

4. 401エラーの対応

上記までの設定において，適当な「index.html」を ¥wwwroot に置いて表示すると，「**401 - 権限がありません：資格情報が無効であるため、アクセスが拒否されました。**」と正しく表示できない。（ローカル、リモートとも）

これは，C:¥inetpub¥wwwroot のセキュリティ設定のためで，同フォルダのユーザ「Users」に「読み取りと実行」「フォルダー内容の一覧表示」「読み取り」のパーミッションを与える。

5. リモートからのコンテンツアップ対応

C:¥inetpub¥wwwroot の共有設定において，メンテナンスのためアクセスするユーザに，フルコントロールなどのパーミッションを与える。

4. 1. 3　Razor 構文への対応

IIS 標準インストールの状態では Razor 構文に対応していない。以下の追加インストールを実施する。

1. Microsoft .NET Framework 4

   dotNetFx40_Full_setup.exe を実行する。

   (http://www.microsoft.com/downloads/ja-jp/details.aspx?FamilyID= 9 cfb 2 d51- 5 ff 4 -4491-b 0 e 5 -b386f32c0992)

2. AspNetWebPages.msi，AspNetWebPages_LP_JPN.msi を順に実行する。

   (http://www.microsoft.com/downloads/ja-jp/details.aspx?FamilyID=300314da-dedd-4540-a236-a 0 de 0 a 5 a534d&displayLang=ja)

3. IIS マネージャを起動し，以下の設定を確認する。

   ［IIS ハンドラマッピング］ 以下のマネージャハンドルが自動的に追加される。

   ・要求パス：*.cshtm，種類：System.Web.HttpForbiddenHandler，名前：cshtm-Integrated-4.0

   ・要求パス：*.cshtm，種類：
   C:¥Windows¥Microsoft.NET¥Framework¥v4.0.30319¥aspnet_isapi.dll（追加），
   名前：cshtm-ISAPI-4.0_32bit

   ・要求パス：*.cshtml，種類：System.Web.HttpForbiddenHandler，名前：cshtml-Integrated-4.0

   ・要求パス：*.cshtml，種類：
   C:¥Windows¥Microsoft.NET¥Framework¥v4.0.30319¥aspnet_isapi.dll（追加），
   名前：cshtml-ISAPI-4.0_32bit

［アプリケーションプール］ 以下のアプリケーションプールが自動的に追加される。

ASP.NET v4.0

ASP.NET v4.0 Classic

［アプリケーションプールの規定値］ アプリケーションプール表示の右側にリンクがある。

.NET Framework バージョン：v4.0 ← ※2.0から変更

この設定が変更されていないと、「**404.17 - Not Found 要求されたコンテンツはスクリプトであり、静的ファイルハンドラーで処理されない可能性があります**」のエラーが発生する。

［.NET Frame のバージョン変更］ サイト をクリックすると右側にリンクがある。

v4.0.30319

［既定のドキュメント］

既定のドキュメントに「index.cshtm」「index.cshtml」「default.cshtm」「default.cshtml」を追加する。

既定では、「Default.htm」「Default.asp」「index.htm」「index.html」「iisstart.htm」「default.aspx」が設定されている。

4. IIS の再起動

以上で IIS の設定が完了する。なお，Windows 7 Professional と Windows Server 2008 R2 に搭載されている IIS の違いについて，通常の動作については違いがない。ただ Windows 7 の場合，アクセス数が増加するとレスポンスが低下するので，日に5000ページ以上の本格的な運用には Windows Server 2008 R2 を導入することが望ましい。

3. 2. 4 DBMS（データベース）

DBMS の選択について，WebMatrix に付属する SQL Server Compact や別途フリーで提供されている SQL Server 2008 R2 Express を利用することも検討したが，製品版の SQL Server 2008 R2 Standard に比べて，データベースサイズの制限（Compact： 4 GB，R2 Express：10GB，R2 Standard：無制限）や，管理プログラムにおいて直接 CSV ファイルが読み込めないなどのメンテナンス上の制限があることから，無料版の採用を断念した。以下に，有料版である SQL Server 2008 R2 Standard について，インストールの留意点とバックアップ方法について述べる。

1. インストール

SQL Server 2008 R2 Standard のインストールは，インストール用 DVD に含まれる SETUP.EXE を実行し，表示されるメニューに従えば問題なく行える。なお，64bit Windows 7 の場合，共有機能を保存する既定フォルダは「C:¥Program Files (x86) ¥Microsoft SQL Server¥」で，SQL Server 自体のフォルダは「C:¥Program Files¥Microsoft SQL Server¥」にインストールされる。

プログラムのインストールが完了したら，スタートメニューより Microsoft SQL Server Management Studio（管理用プログラム）を起動し，管理者としてログインする。Microsoft SQL

Server Management Studioは非常に良くできており，通常のデータベースメンテナンスの殆どが，ビジュアルな管理画面で行える。

ここでデータベースの作成，テーブルの設定などを行うが，特定のデータベースに対しWebアプリケーションからアクセスする場合は，作成したデータベースにマッピングされた専用のアカウント（例えば，ユーザweb）を作成する必要がある。Microsoft SQL Server Management Studioの［セキュリティ］→［ログイン］→［新しいログイン］で作成出来る。

2．データベースのバックアップ

実運用を始めるとデータベースのバックアップが必要となる。作成したデータベースの諸設定を含むデータのバックアップは，前述したMicrosoft SQL Server Management Studioにおいて，［タスク］→［バックアップ］から行える。既定では「C:¥Program Files¥Microsoft SQL Server¥MSSQL10_50.MSSQLSERVER¥MSSQL¥Backup」に「データベース名.bak」が作成される。また，バックアップとは別にデータファイル（データベース定義，データ本体，ログ）は，既定で「C:¥Program Files¥Microsoft SQL Server¥MSSQL10_50.MSSQLSERVER¥MSSQL¥DATA」に「データベース名.mdf」と「データベース名_log.ldf」という二つのファイルで保存される。

これらバッアップファイルやデータファイルを多重に複製して保存することは，万が一の事故に備えて安全にシステムを運用する場合に必須となる。SQL Serverをインストールしてからは，SQL Server自体がサービスとして実行されるため，各ファイルはシステムによりロックされている。したがって，バッチ・ファイル等でバックアップを複製する場合はロックを解除（サービスを停止）してからxcopyコマンド等でコピーすることになる。サービスの停止，開始は以下のコマンドにより行える。

サービスの停止：「net stop MSSQLServer /y」

サービスの開始：「net start MSSQLServer /y」

何れもAdministrator権限で実行し，サービス停止中に「**xcopy 元ファイル 複製ファイル**」などで複製を作成する。なお，複製を曜日ごとに分けて作成すると，より安全なシステムが構築できる。重ねて，専用のバックアップアプリケーションやWindows付属のバックアップによってWindowsシステムおよびボリューム単位のバックアップを適用しておけば万全である。いずれにせよ，公開するデータベースについては，事故に対応したシステムを含めたバックアップ体制を何重にも設定することに留意されたい。

5．各機能の実装

動的なWebアプリケーションの作成に必要な機能の実装について，自学自習システムで利用した手法をまとめておく。

5. 1　データベースとの連携

データベースとの連携についても，Razor 構文は簡易な手段を提供している。Web プログラムとデータベースの接点は「ConnectionStrings」と呼ばれる接続文字列を使用する。ConnectionStrings は，サイトのメインフォルダに置かれた「web.config」内で図 5. 1のように記述する。内容は，「name」で定義される呼び出し名(MYConnectionString)と「ConnectionString」で定義される一連のデータベース情報で構成される。この定義中にある「providerName="System.Data.SqlClient"」は Microsoft SQL Server についての記述で，Oracle では「System.Data.OleDb」(Oracle 製の OLE DB ドライバを使用する場合）となる。

```
<connectionStrings>
<add
name="MyConnectionString"
        ConnectionString=
        "Server=db サーバ名;
        uid=db ユーザ ID;
        pwd= パスワード;
        database= データベース名;"
        providerName="System.Data.SqlClient"
/>
</connectionStrings>
```

図 5. 1　データベース定義（web.config）

```
@{
var db = Database.Open（"MyConnectionString"）;
var Query1
 = "SELECT * FROM データベース名
 WHERE テーブル名 . フィールド名 = @0";
}
```

図 5. 2　データベースの操作（定義）

呼び出す側の Web ページでは，コードブロック内で図 5. 2のように定義する。変数「db」はデータベースオープンに関連づけられ，データベースを開くときに利用される。また，変数「Query 1」はデータベースから任意の検索条件に合致するレコードを取り出す SQL 言語を定義している。検索条件に記されている「@ 0」は「パラメータブレースホルダ」と呼ばれるもので，SQL 文へ値を渡す際はパラメータを使う。これは，データベースに対して SQL インジェクションによる悪意あるコマンドが送られることを防止し，セキュリティ対策を目的としている。

```
<table border="1">
<tr><th>data1</th><th>data2</th><th>data3</tr>
@foreach （var row in db.Query（Query1, 検索条件））{
<tr>
<td>@row. フィールド名 </td>
<td>@row. フィールド名 </td>
<td>@row. フィールド名 </td>
</tr>
</table>
```

図 5. 3　データベースの操作（呼出と表示）

定義されたデータベースを呼び出してデータを表示する場合は，図 5. 3のように行う。「@foreach」ループは SQL 文で問い合わせたデータを個別に取り出し，結果を「<td>」要素でブラウザに送出している。

5. 2　ユーザ認証

ユーザを限定したサービスでは，ユーザの認証が必要となる。文献 3）では，メールによるユーザ登録の例（WebSecurity ヘルパー）が紹介されているが，学習支援システムでは厳格なユーザ

認証を必要とすることから，あらかじめ管理者がデータベースに登録したユーザのログイン認証を実装した。

データベースには，図 5. 4の定義がしてあり，このうち教員 ID と教員 PW をログイン時に使用，ログイン後の動作に権限コードを利用した。ログイン ID とパスワードは，学習支援システムで最初に表示される index.cshtml のフォームで入力させ(図 2. 1)，この値をデータベースに保存されている値と照合している。ID，パスワード共に一致した場合に，ログイン用のセッション値にログインを記録する。

教員 ID	TeID	char（15）
教員 PW	TePass	char（15）
教員名	TeName	varchar（4）
権限コード	TePerm	char（1）
電話番号	TePhone	varchar（17）
携帯番号	TeMobile	varchar（17）
メール	TeEmail	varchar（30）

図 5. 4　教員テーブルのデータベース定義

5. 3　セッションの管理

ログイン後にユーザ名などのログイン情報を保持したまま複数のページを渡り歩く場合や，ブラウザが一定時間を超えて操作されない場合に自動ログアウトするなどの処理には，状態を保持するセッション管理が必要となる。セッション管理には，①クライアント側のクッキー（HTTP Cookies）を使う方法，②データベースに保存する方法，③ IIS の状態サーバを利用する方法などがある。クッキーについてはユーザ PC の環境に依存しているので，ブラウザでクッキーを有効にしていないと利用できない。インターネットで公開する大規模なサイトでは，大量のセッションを保存するデータベースが必要だが，今回のように中規模なシステムでは Windows の状態サーバが向いている。

状態サーバを利用するには，あらかじめ Windows Aspnet_state.exe 状態サービスを有効にしておく。状態サーバは，アプリケーションを実行するワーカプロセスの外にセッション状態を保存する。このため，アプリケーションのワーカプロセスがリサイクルされても、セッション状態を保持することが出来る。状態サーバを有効にするには，コントロールパネル → 管理ツール → サービス → ASP.NET 状態サーバの起動方法を［手動］から［自動］に切り替えておく。その後，IIS マネージャを起動し，［ASP.NET］ → ［セッション状態］を開き，状態サーバにチェック，接続文字列：tcpip=localhost:42424，タイムアウト(秒)：3600秒などと設定する。また，データベース定義と同じく「web.config」内に状態サーバの定義を記述する（図 5. 5)。

```
<system.web>
  <sessionState
    mode="StateServer"
      stateConnectionString="tcpip=localhost:42424"
      stateNetworkTimeout="3600"
      timeout="60">
  </sessionState>
</system.web>
```

図 5. 5　状態サーバの定義（web.config）

続いて，図 5. 6のようにページ中で参照するセッション変数の定義を行う。この定義はレイアウトシートなどに記し，ページがレンダリングされる最初に実行させる。各ページ

では，セッション変数に値を代入・参照が出来るので，表示するページが変更されても，ログイン状態を把握でき，ログインユーザ名を画面に表示することが可能となる。

```
string Login = （string）（Session["Login"]）　　;// ログイン状態
string UserID = （string）（Session["Uld"]）　　;// ユーザ ID
string UserName = （string）（Session["UName"]);// ユーザ名
string UserPerm = （string）（Session["UPerm"]）;// パーミッション
```

図 5．6　セッション変数の定義

5．4　アクセスログの作成

実装の最後に，アクセスログについて説明する。学習支援システムのように個人情報を扱うシステムでは，ユーザが訪れたページや訪問時刻を細かく記録してログに保存することが望ましい。学習支援システムでは，日時，訪問ページ，ユーザ ID をテキスト形式で，訪問ページごとに記録することにした。

```
@{
string path = "~/App_Data/log_" +@DateTime.Now.ToString
("yyyyMMdd") + ".txt";
// ファイル名定義
fdata = @DateTime.Now.ToString（"yyyy/MM/dd HH:mm:ss"）+
" login "+ @row.TeID.TrimEnd（） + "\r\n";
// 書き込みデータ
var LogFile = Server.MapPath（@path）; // 物理 path
File.AppendAllText（@LogFile, @fdata）; // 書き込み
}
```

図 5．7　ログファイルの書き込み

ログ記録は「File.AppendAllText（ファイル path, 書き込み文字列）」メソッドが利用でき，ログを記録したい場所で図 5．7のようなコードを挿入すればよい。「string path =」は日付入りのログファイル名を生成している。続く「fdata =」は書き込みデータで，書式付き日時とページ名「login」，ログインユーザの ID を文字列として繋いでいる。「var LogFile =」でサーバの物理パスを生成してから，最後に「File.AppendAllText（）」によりログにデータを書き込んでいる。「File.AppendAllText（）」は，書き込みファイルが存在しないと新たにファイルを生成し新規書き込みを，既に存在する場合は同ファイルに追記書き込みを行う。学習支援システムでは，1日単位でファイルを生成するようになっており，日ごとのアクセス統計やセキュリティ調査に利用している。

6．ま　と　め

本稿は学生支援システムを概説したあと，Razor 構文によるシステム構築の技術解説を行った。Razor 構文は単純化された構文を使用しつつ ASP.NET の全パワーを発揮できる潜在能力を持っている。現在のところ Windows 系の限られた稼働環境ではあるが，Linux でも ASP.NET が利用できる研究開発が進められており，将来的には Linux の Apache 上でも Razor 構文が利用可能になる可能性は高い。どちらかと言えば旧来技術の延長線上にある PHP による古い Web アプリケーション開発から，未来発想による新しい Web アプリケーション開発へ大きく一歩踏み出したように思える。今しばらく学習支援システムの完成までは Razor 構文と付き合うことになるが，

今後も引き続きシェイプアップされたスマート技術を満喫したいと考えている。

参 考 文 献

1）清水啓司，学習支援センター設置までの経過と今後の取り組み，神野学園夏季研修会講演資料，(2011)
2）山田祥寛，Web アプリケーション・フレームワークの新たな選択肢，@IT，(2011)
3）Microsoft，Razor 構文と ASP.NET Web ページ，Microsoft ASP.NET デベロッパーセンター，(2011)
4）Scott Guthrie, Introducing "Razor" - a new view engine for ASP.NET, ScottGu's Blog, (2011)