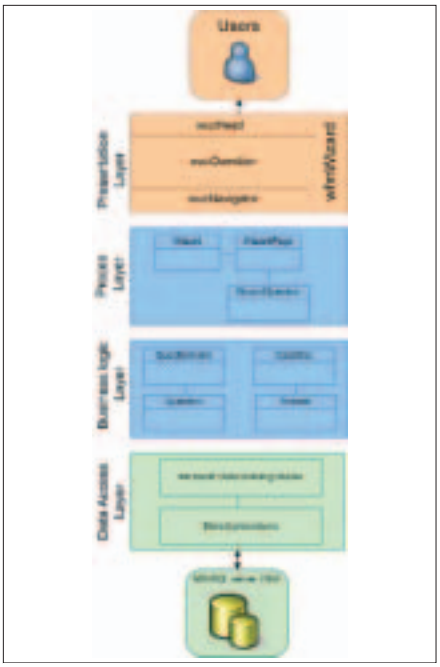
Rob van der Meijden en René Schrieken
zijn consultants Public Sector bij LogicaCMG

Vragen in een ASP.NET wizard

HET DYNAMISCHE OSIRIS-SYSTEEM VAN HET RIVM

Bij het Rijksinstituut voor Volksgezondheid en Milieu worden jaarlijks tienduizenden meldingen geregistreerd van infectieziekten. Bij aanmelding worden, afhankelijk van de infectieziekte, algemene en specifieke vragen gesteld. Een belangrijke eis aan het systeem, genaamd Osiris, is dat het geschikt moet zijn wijzigingen in de vragenlijsten zonder tussenkomst van een ontwikkelaar door te kunnen voeren.

Dit betekent dat de procesinrichting, vragenlijsten, infectieziekten en rechtenstructuren een dynamisch karakter hebben. Op basis daarvan is ervoor gekozen het gehele systeem te baseren op een metamodel. Dit metamodel wordt onder andere gebruikt voor de wizard waarmee de vragenlijsten worden ingevuld. De belangrijkste gebruikte technieken om deze wizard te implementeren in een ASP.NET-applicatie worden in dit artikel besproken. Voor metamodelgestuurde applicaties is een uitgekiende architectuur van wezenlijk belang (zie afbeelding 1). In de data-access-laag hebben wij gebruik gemaakt van de Data-Building-Blocks die door Microsoft via hun website aan de .NET-ontwikkelaars beschikbaar worden gesteld.



Afbeelding 1. Architectuur is van wezenlijk belang

Businesslogica-laag

In onze businesslogica-laag leggen we ons metamodel vast. Daarin is de dynamiek van de applicatie in classes vastgelegd. Als voorbeeld gebruiken we hier een metamodel met twee classes waarmee we een questionnaire kunnen opbouwen. Allereerst de class-questionnaire. Deze class heeft als belangrijkste property een collectie met instanties van de question-class, de vragen waaruit de vragenlijst is opgebouwd. De class-question heeft property's als Text, Type, ControlHeight en ControlWidth voor het sturen van de userinterface. Om onze productiedata, de ingevulde vragenlijsten, te kunnen vastleggen hebben we twee classes: casedoc en answer. Een instantie van de casedoc-class heeft een relatie met de questionnaire-class, die gebruikt is bij de totstandkoming van deze case en een collectie van answers. De answer-class kent de questionnaire waar hij bijhoort en onthoudt het ingevoerde antwoord.

Proceslaag

Nu we de basis van ons model hebben, dienen we een aantal classes te definiëren waarmee we de metamodel-classes en businessdata-classes tot elkaar brengen. De wizard-class gaat ons metamodel gebruiken om de vragen en de mogelijk te geven antwoorden op één of meer WizardPages te tonen. Het WizardPage-object gaat bijhouden welke WizardQuestion-objecten getoond moeten worden. Een WizardQuestion heeft een relatie met de Question die hij toont en het answer dat er bijhoort (zie afbeelding 2).

Presentatielaag

De presentatielaag wordt geïmplementeerd door standaard ASP.NET-classes: De Page-class met de naam wfmWizard dient als ankerpunt voor al onze objecten. We willen de gebruiker graag in staat stellen om door de pagina's van de wizard heen te bladeren. Hiervoor implementeren we een WebUserControl (wucNavigator), die toont hoeveel pagina's er in de wizard zitten en ervoor zorgt dat we naar de gewenste pagina kunnen navigeren. Bovenaan onze pagina willen we de gebruiker informeren over de vragenlijsttitel. Dus daar maken we ook een WebUserControl (wucHead) voor. De laatste en belangrijkste WebUserControl (wucQuestion) is verantwoordelijk voor het daadwerkelijk tonen van de gegevens die

door het WizardQuestion-object voor ons samengevoegd zijn.

Wizard-presentatie

De wfmWizard.aspx is niet veel meer dan een bijna lege template, waar alleen de wucNavigator (voor het navigeren) en de wucKop (voor onze informatie over de pagina) een vaste plaats hebben gekregen. Wat verder opvalt is de PlaceHolder Control. Deze control zal straks gebruikt worden om de user control(s) voor de vragen aan de pagina toe te voegen.

UserControl - Head

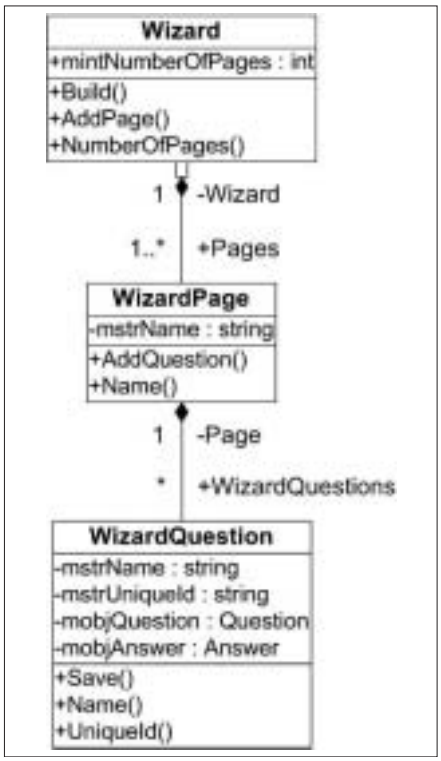
De wucHead usercontrol heeft een ascx-file waar één Label-control in zit. In de code-behind-file hebben we één property geïmplementeerd, waarmee we de Titel die we willen tonen kunnen vullen. Het lijkt overbodig om deze UserControl te hebben met deze eenvoudige functionaliteit. Maar het is niet zo moeilijk voor te stellen dat deze control in de toekomst meer functionaliteit zal bevatten; denk bijvoorbeeld aan versie-info, laatste wijzigingsdatum, statusinformatie enzovoort.

UserControl Navigator

De UserControl wucNavigator bestaat uit een tweetal vaste controls, namelijk de Vorige en Volgende Button. Ook hier maken we gebruik van een PlaceHolder. Deze staat op de plek waar de buttons voor het aantal webpagina's moeten komen. Op de wucNavigator zitten drie property's: Vorige, Huidige en Aantal. De wucNavigator bewaart de inhoud van deze property's tussen pagina-reposts in de viewstate van de pagina. De wfmWizard-pagina zal de property Aantal op de juiste waarde zetten. In het Load-event voeren we een for-lus uit waar we Button-controls met het juiste paginanummer toevoegen op de PlaceHolder met behulp van de Add-methode. Bovendien vullen we de OnClick met een delegate naar de PageClick event-handler. De Huidige en Vorige property doen wat ze beloven: ze geven aan wat de pagina is die de gebruiker nu ziet en wat de vorige pagina was die de gebruiker zag.

```
public void SetQuestion(WizardQuestion pobjWZQuestion) {
  this.lblQuestion.Text = pobjWZQuestion.QuestionObject.Name;
  WebControl lctrQuestionControl; // here is our overall WebControl
  switch(pobjWZQuestion.QuestionObject.QType)
  {
    case QuestionType.Text:
    {
      TextBox lctrText = new TextBox();
      lctrText.Text = pobjWZQuestion.AnswerObject.Value;
      lctrQuestionControl = lctrText;  // store this local control
                                       // in the overall object
      break;
    }
    case QuestionType.Radio:
    {
       // do the same but now for RadioButtonList
    }
    //  more cases for other WebControls
  }
  // set our heigth on the table
  this.tblQ.Height = pobjWZQuestion.QuestionObject.Height;
  // setting this ID is essiantial, if you don't use it ASP.NET will
  // generate something like _ctl[number] for you.
  lctrQuestionControl.ID= pobjWZQuestion.QAID;
  // now add the overall webcontrol to the usercontrol
  this.plcQuestion.Controls.Add(lctrQuestionControl);
}
```

Codevoorbeeld 1.



Afbeelding 2. Relatie WizardQuestion

UserControl Question

De usercontrol wucQuestion wordt door de wfmWizard-pagina-runtime aan de pagina toegevoegd. De control heeft een method SetVraag waar we de WizardVraag aan doorgeven. SetQuestion neemt vervolgens het aanmaken van de TextBox, RadioButtonList en andere 'input' controls voor zijn rekening. In WizardQuestion zit een enumerator voor het type Question. Via een switch-statement wordt bepaald welke WebControl voor deze vraag wordt toegevoegd. Aangezien de WebControl die moet worden toegevoegd iedere keer anders kan zijn, gebruiken we een object van het type WebControl om onze toe te voegen control in op te slaan (zie codevoorbeeld 1). Nadat de control is gecreëerd, worden de generieke zaken voor de vraag afgehandeld zoals het vullen van de vraag- en helptekst. Voor iedere WizardQuestion is een unieke identificatie gegenereerd. Het is belangrijk om de identificatie-property van de WebControl te zetten. Hiermee forceren we dat de data van de Control tijdens het posten van de pagina naar de server weer identificeerbaar is.

Wizard Pagina – GET

Nu we onze bouwstenen hebben, kunnen we beginnen met de eindassemblage. De wucNavigator en de wucHead staan op hun vaste plek. We bekijken eerst wat gebeurt

als de pagina voor de eerste keer wordt opgevraagd. In het Load-event roepen we de Build-method van het nog lege Wizard-object aan met een instantie van Questionnaire en een instantie van CaseDoc. Hierna kunnen we de wucNavigator.Count vertellen hoeveel pagina's er zijn. Aangezien het Load-event van de Page eerst vuurt, en daarna pas de Load-events van de UserControls, is dit precies op tijd. In de wfmWizard hebben we een private method CreatePage(). Deze method wordt aangeroepen in het PreRender-event. De CreatePage bestaat in feite uit een for-each lus voor alle WizardQuestion-objecten van de huidige pagina. De huidige pagina (WizardPage) wordt verkregen door aan de Wizard te vragen uit zijn Pages-collectie een pagina te halen. De property Current van de wucNavigator bepaalt welke WizardPage door de Wizard wordt teruggegeven. Vervolgens wordt in de lus voor iedere WizardQuestion de method LoadControl aangeroepen, waarmee we een nieuwe instantie van wucQuestion aanmaken. LoadControl laadt de ASCX-pagina en maakt een UserControl-object aan. Als we dit object naar wucQuestion casten, kunnen we de eerder beschreven method SetVragen aanroepen. Op basis van het WizardQuestion-object worden de specifieke controls voor de echte vraag toegevoegd, de tekst van de vraag en de hoogte gezet. Aan het eind van de PreRender zijn alle controls gemaakt en zal het ASP.NET Framework alle controls renderen en naar de browser sturen.

Wizard Pagina – POST

We nemen aan dat de gebruiker de gestelde vragen op pagina 1 beantwoordt en uiteindelijk op een button in de Navigator klikt om naar de volgende pagina te gaan. Als je een gewone ASPX-pagina gemaakt zou hebben, waar alle controls gewoon opstaan, dan had je nu een eenvoudige taak. In ons geval zijn de controls die we tijdens de PreRender hebben aangemaakt op het moment van het laden van de pagina nog niet aanwezig. ASP.NET kan hierdoor de door de gebruiker ingevoerde gegevens niet aan de controls koppelen. Er is rekening gehouden met dit probleem door bij het genereren van de Control in de SetVragen method een unieke identificatie aan de control mee te geven.

```
private void SavePage(){
// get the previous page
WizardPage lobjPage = (WizardPage) mobjWizard.Pages[wucNavigator.Previous-1];
// iterate through our wizard questions
foreach(WizardQuestion lobjWizQues in lobjPage.WizardQuestions){
   // get the answerobject for this question
   Answer lobjAnswer =
      this.CaseObject.GetAnswer(lobjWizQues.QuestionObject);
   // Use our key to get the answer from Request.Form
   string lstrValue =
      this.Request.Form[String.Format("{0}:{0}",lobjWizQues.QAID)];
   if (lstrValue==null){
      // the user didn't provide an answer
      lobjAnswer.Value = ""
  } else {
      // yes! they answered, store the value;
      lobjAnswer.Value = lstrValue;
  }
  // make sure our answer is stored with the CaseDoc
  this.CaseObject.SetAnswer(lobjWizQues.QuestionObject,lobjAnswer);
}
```

Codevoorbeeld 2.

Op het moment dat het PreRender-event start, roepen we de method SavePage aan. Deze method doorloopt met een for-each-lus de collectie van WizardQuestion-objecten voor de vorige pagina (wucNavigator.Previous). In de Request.Form-collectie zitten alle strings, die door de webbrowser in de POST aan de server zijn teruggestuurd. De index in deze collectie is benaderbaar door middel van de identificatie van de Control zoals deze op het form stond. Deze kunnen we nu gebruiken om in de Request.Form-collectie de string op te halen, waarmee het antwoord van de gebruiker nu weer aan het juiste Answer-object kan worden gekoppeld (zie codevoorbeeld 2). Vervolgens kunnen we weer de CreatePage uitvoeren om de controls voor de 'nieuwe' Current-pagina toe te voegen.

Belangrijk om te weten is dat op het moment dat je zelf tijdens runtime-controls gaat toevoegen aan Page-objecten, zaken als (view)state management en event-handling niet werken tenzij maatregelen zijn genomen. Je moet er dan namelijk voor zorgen dat de controls op het juiste moment (en in dezelfde volgorde) weer aanwezig zijn in de Control-hiërarchie. Voor onze oplossing was het makkelijker de gegevens uit de Request.Form-collectie op te halen. Het eerst opnieuw creëren van alle UserControls van de 'vorige' pagina kan hierdoor achterwege blijven.

500 dagelijkse eindgebruikers

De hier beschreven oplossing laat zien dat het goed mogelijk is om met gebruikmaking van de kracht van UserControls en de Controls.Add-method een complexe userinterface real-time te creëren. Het voordeel is dat optimaal gebruik wordt gemaakt van zowel Visual Studio .NET, omdat de UserControls via de IDE zijn te manipuleren, als het .NET Framework. De hier beschreven techniek voor het opbouwen van een wizard is als fundament toegepast in de ASP.NET meldingenapplicatie van het Osiris-systeem. Het implementeren van een andere userinterface (WinForms, PDA) vraagt geen ingreep in de structuur van het meta- en/of businessmodel. Het Osiris-systeem gaat in het eerste kwartaal van 2003 live en zal beschikbaar worden gesteld aan ongeveer 500 eindgebruikers die dagelijks meldingen zullen registreren.

Nuttige internetadressen

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/distapp.asp