Riccardo Becker en Raymond Binnendijk

Patterns and practices bieden redding

HET VINDEN VAN DE JUISTE SOFTWARE FACTORIES, APPLICATION BLOCKS EN END2END GUIDES

Microsoft biedt een groot aantal patterns en practices. Dit artikel is bedoeld als leeswijzer voor het vinden van de juiste software factories, application blocks en end2end guides. Hoe vind ik snel de juiste patterns and practices?

Een ontwikkelaar houdt zich bezig met het oplossen van een specifiek probleem. Vaak is hiervoor een specifieke (maatwerk-) oplossing nodig. Toch lijken veel problemen in meer of mindere mate op elkaar. Het is dan ook niet toevallig dat veel (deel)oplossingen grotendeels hetzelfde zijn. Microsoft onderkent dit en produceert onder de naam 'Patterns and Practices' al een aantal jaren guidance en (deel)oplossingen. Hierdoor hoeft een ontwikkelaar of architect niet steeds zelf het wiel uit te vinden. Op drie vlakken worden producten aangeboden:

- software factories
- application blocks
- end-to-end gidsen voor specifieke deelgebieden

Software factories

Een relatief nieuwe stroming in de softwareontwikkelwereld, waarvan de komende jaren zeer veel wordt verwacht, zijn software factories. Een software factory is een productlijn die ontwikkeltools zoals Microsoft Visual Studio Team System uitbreidt met guidance. Een software factory biedt de ontwikkelaar of architect ondersteuning bij het creëren van high-quality instanties van specifieke applicatietypes. De factory verrijkt de ontwikkelomgeving met functionaliteit die het gemakkelijker maakt specifieke producttypes te ontwikkelen. Een software factory kan bestaan uit herbruikbare stukken code, documentatie, referentie-implementaties, wizards, codegeneratoren en visual designers. Guidance automation ondersteunt ontwikkelaars/architecten tijdens het gehele ontwikkelproces.
Een toepasselijke analogie voor software factories is die met een restaurant. Een software factory bevat drie concepten:

1. **Software factory-schema** - Een software factory-schema is een beschrijving (document) die op logische wijze een structuur aanbrengt in artefacten die gebruikt worden om een applicatie te ontwikkelen. Denk hierbij aan config-files, sourcecode, testcases, xml, modellen, frameworks, enzovoort (analoog aan een recept voor een gerecht).
2. **Software factory-template** - Daadwerkelijke implementatie van het software factory-schema, waarbij de verschillende artefacten worden gedefinieerd en beschikbaar worden gesteld aan developers. Al deze artefacten samen vormen de software factory-template (analoog aan ingrediënten benodigd voor een recept).
3. **Extensible Development Environment** - Indien VSTS wordt geconfigureerd met de software factory-template is deze in staat producten te maken volgens de genoemde recepten (software factory-schema) met behulp van de ingrediënten (software factory-template). VSTS is de keuken waarin de maaltijden (producten) worden gemaakt.

Microsoft levert een aantal kant-en-klare software factories; zie tabel 1 voor een overzicht. Met de software factory kun je direct de betreffende producttypen ontwikkelen of ze kunnen als basis dienen voor een maatwerk software factory. De Guidance Automation Toolkit ondersteunt hierbij.
Het gebruik van een software factory kan leiden tot een toename van productiviteit en productkwaliteit. Het gebruik heeft echter pas zin wanneer je van plan bent meer soortgelijke producten te ontwikkelen volgens een vergelijkbare architectuur. Bovendien moet het probleemdomein goed begrepen zijn en moeten de in de factory gebruikte toevoegingen (frameworks, templates, patterns, enzovoort) goed zijn getest.

Application blocks

Daar waar software factories zich richten op het ontwikkelen van (meer instanties van) een specifiek producttype, helpt een application block de ontwikkelaar bij het oplossen van veel voorkomende ontwikkelvraagstukken in een willekeurige applicatie. Application blocks zijn herbruikbare sourcecode-componenten die proven solutions beschikbaar stellen aan

Product	Beschrijving
Mobile Client Software Factory	Guidance bij het ontwikkelen van Windows Mobile-applicaties die communiceren met backend-systemen op verschillende netwerken.
Smart Client Software Factory	Geautomatiseerde ondersteuning bij het ontwerpen en ontwikkelen van occasionally connected smart-client applications.
Web Client Software Factory	Uitgebreide set van uitbreidingen binnen Visual Studio 2005 voor het ontwikkelen van webapplicaties met de laatste technologieën zoals ASP.NET en Windows Workflow Foundation.
Web Service Software Factory	Ook wel bekend als service factory. Verzameling van tools, patterns, sourcecode en guidance voor het ontwikkelen van webservices gebaseerd op ASMX of Windows Communication Foundation.
Application Block Software Factory	De Application Block Software Factory helpt bij het ontwikkelen van nieuwe application blocks (zie verder in dit artikel).
Guidance Automation Toolkit	Met de Guidance Automation Toolkit is het mogelijk guidance packages voor Visual Studio te maken. Deze kunnen bestaan uit templates, wizards, actions en recepten. Hiermee kun je snel functionaliteit toevoegen aan Visual Studio om specifieke problemen op te lossen.

Tabel 1. Overzicht software factories

Application block	Beschrijving
Caching	Levert mechanismen voor client- en serverside caching.
Smart Client – Composite UI	Ondersteuning bij het bouwen van een complexe windows desktop-UI.
Cryptography	Maakt het gemakkelijk om cryptografiefunctionaliteit in de applicatie te gebruiken.
Data Access	Beperkt de hoeveelheid code die nodig is om data access-lagen te maken, te testen en te onderhouden.
Exception Handling	Maakt het gemakkelijk om consistente exception handling-policies te implementeren.
Logging	Voor het met logging uitrusten van jouw applicatie. Biedt ondersteuning voor filtering en formatting. Standaard mogelijkheid om te loggen naar eventlog, databases en WMI.
Policy Injection	De PIAB biedt de mogelijkheid policies via configuratie en zonder code te wijzigen en toe te kennen aan instanties van klassen. Bovendien is het mogelijk handlers toe te voegen en zelf te bouwen en deze via matching toe te passen op policies. De PIAB kan goed ingezet worden voor logging, exception handling en het meten van de performance van methoden. Het codevoorbeeld geeft aan hoe je PIAB kan gebruiken en wat de toepassing ervan is.
Security	De SAB biedt ontwikkelaars functionaliteit voor standaard autorisatiemechanismen.

Tabel 2. Beschikbare Application blocks

veelvoorkomende ontwikkelvraagstukken. De blocks kunnen worden geïntegreerd in een applicatie. Het is mogelijk de blocks aan te passen en/of uit te breiden. At runtime is het gedrag van de blocks te configureren. In tabel 2 is te zien welke application blocks beschikbaar zijn.
Codevoorbeeld 1 maakt duidelijk wat jij je bij een application block kunt voorstellen. Het voorbeeld heeft betrekking op het Policy Injection Application Block. Om gebruik te kunnen maken van policy-injection moet een klasse interceptable zijn. Een klasse is interceptable als deze een interface implementeert of afgeleid is van MarshalByRefObject (remoting). In dit geval kiezen we voor een interface IPerson die wordt geïmplementeerd door de klasse Person. Op de klasse Person gaan we policy-injection toepassen.

Let op: het Tag-attribuut is al toegevoegd en is terug te vinden in namespace Microsoft.Practices.EnterpriseLibrary.PolicyInjection. We gaan nu via het PIAB de methode SayHi interceptable maken. Ten eerste gaan we via Enterprise Library Configuration twee policies definiëren: TypePolicy(Match=Person) en TagPolicy (Match=LogThisCall). Hiermee kunnen we op basis van een type (Person) en op basis van een tag (LogThisCall) de

```
public interface IPerson
{
   void SayHi();
}

public class Person : IPerson
{
   [Tag("LogThisCall")]
   public void SayHi()
   {
      Console.WriteLine("Hi!");
   }
}
```

Codevoorbeeld 1.

```
IPerson person = PolicyInjection.Create<Person, IPerson>();
person.SayHi();
```

Codevoorbeeld 2.

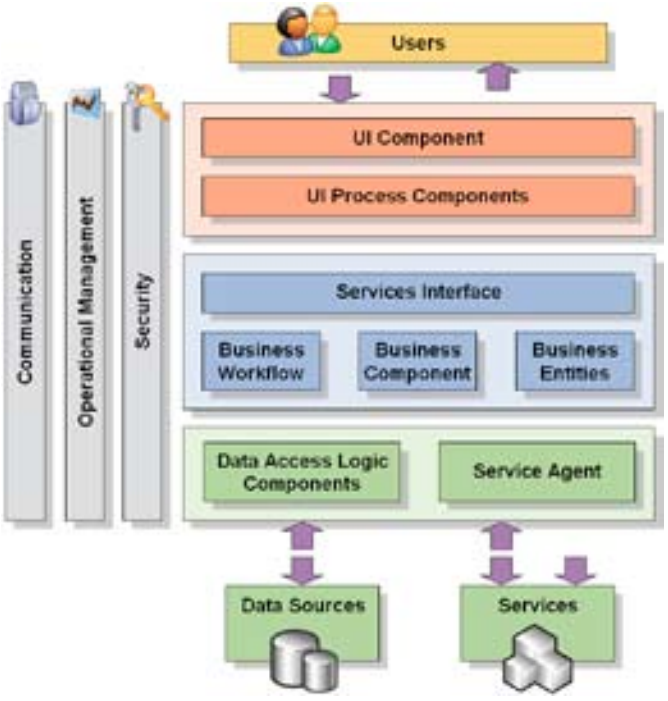
methode onderscheppen. Vervolgens kunnen we handlers toevoegen voor beide policies. In codevoorbeeld 2 voegen we een logging handler toe (zorg er wel voor dat je ook het Logging Application Block hebt toegevoegd aan je app.config). Geef in de BeforeMessage en AfterMessage aan wat er gelogd moet worden.
Het PIAB zorgt er nu voor dat de handlers worden aanroepen als een object van het type is IPerson aangemaakt (TypePolicy, Match=Person) en als de SayHi()-methode wordt aangeroepen (TagPolicy, Match=LogThisCall). Het gebruik van de caching handler, die is toegevoegd aan de TypePolicy, is heel handig om bijvoorbeeld zware berekeningen te cachen of voor het versnellen van het ophalen van stamtabellen uit een database. Het eenvoudigweg toevoegen van deze handler is voldoende om de aanroep naar de methode cacheable te maken en zo de performance te verbeteren. Uiteraard is het ook mogelijk je eigen handler te schrijven en deze via Enterprise Library Configuration toe te voegen aan de PIAB.

Enterprise Library

De patterns and practices Enterprise Library is een verzameling van een aantal application blocks voor het oplossen van de meest voorkomende problemen. Enterprise library 3.1 bestaat uit de blocks Caching, Cryptography, Data Access, Exception Handling, Logging, User Interface Process, Policy Injection, Security en Validation. Door gebruik te maken van application blocks kun je jezelf veel werk besparen bij het oplossen van een aantal veel voorkomende vraagstukken. Er is veel informatie beschikbaar over de manier waarop de blocks zijn te gebruiken. Natuurlijk moet je per project bekijken of het zin heeft een bepaalde application block (of de enterprise library) te gebruiken.

Patterns and Practices Guides

Naast de software factories en application blocks staat er op de site van Microsoft Patterns and Practices een groot aantal geschreven guides over verschillende probleemdomeinen en software-engineering practices (voor een overzicht van alle gui-



Afbeelding 1. De kern van de guide

Detail-guides	Primair gebied
Designing Data Tier Components and Passing Data Through Tiers	All layers
Design and Implementation Guidelines for Web Clients	Presentation layer
Smart Client Architecture and Design Guide	Presentation layer
.NET Data Access Architecture Guide	Business layer / Data-access layer.
Data Patterns	Data-access layer
Caching Architecture Guide for .NET Framework Applications	Communication
Designing Application-Managed Authorization	Security
Improving Web Application Security: Threats and Countermeasures	Security
Web Service Security Guidance	Security
Authentication in ASP.NET: .NET Security Guidance	Security
Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication	Security
Exception Management Architecture Guide	Operational Management

Tabel 3. Patterns and Practices Guides

Overige guides	Gebied
Application Interoperability: Microsoft .NET and J2EE	Applicatie-integratie. Integratie van een .NET-applicatie met een J2EE-applicatie.
Deploying .NET Framework-based Applications	Applicatie-deployment
Describing the Enterprise Architectural Space	Architectuur
Enterprise Solution Patterns Using Microsoft .NET	Architectuur
Improving .NET Application Performance and Scalability	Performance en scalability
Integration Patterns	Applicatie-integratie
Testing .NET Application Blocks - Version 1.0	Testing
Upgrading Visual Basic 6.0 Applications to Visual Basic .NET and Visual Basic 2005	Migratie VB6 -> Visual Basic .NET
Guidance Automation Toolkit	Software factories

Tabel 4. Overige Patterns and Practices Guides

des zie: http://msdn2.microsoft.com/en-us/library/aa137892.aspx). Een guide beschrijft een aan te bevelen aanpak om een specifiek vraagstuk op te lossen.

Application Architecture for .NET Guide

Zeer interessant, maar ondertussen helaas nogal gedateerd is de 'Application Architecture for .NET: Designing Application and Services'-guide. Een belangrijke uitdaging bij het ontwikkelen van een applicatie is telkens weer het benoemen van de juiste subsystemen en het plaatsen van de juiste code in het juiste subsysteem. De guide helpt hierbij. In de guide staat een overkoepelend recept voor het ontwikkelen van een 'standaard' n-tier .NET-applicatie. Dit recept kan dienen als basis voor het ontwikkelen van veel voorkomende applicaties. De kern van de guide is te zien in afbeelding 1.
De architectuur van veel applicaties is geheel of gedeeltelijk op het diagram in afbeelding 1 te mappen. De guide behandelt de verschillende subsystemen en hun onderlinge relatie. Een groot aantal veelvoorkomende design patterns en best-practices passeert de revue. Op sommige plekken in de guide staan verwijzingen naar detail-guides over het betreffende subsysteem.

Orde in de chaos

Er zijn 22 guides beschikbaar; zie tabel 3 en 4. We zien de 'Application Architecture for .NET'-guide als een zeer interessante bij het ontwerpen en bouwen van veelvoorkomende applicaties. Vanuit deze guide wordt verwezen naar verschillende detail-guides. Bovendien bestaan er guides die andere probleemdomeinen behandelen. Deze guides zijn niet direct aan het diagram in afbeelding 1 op te hangen.
De door Microsoft aangeboden guides bevatten een enorme hoeveelheid informatie over een groot aantal probleemdomeinen. Het is altijd aan te bevelen een guide door te lezen, wanneer je worstelt met een probleem dat er geheel of gedeeltelijk in beschreven is.

Patterns & practices vinden

We hopen en verwachten dat dit artikel helpt bij het vinden van patterns and practices die kunnen ondersteunen bij applicatieontwikkeling. Er zijn ondertussen een aanzienlijke hoeveelheid patterns and practices beschikbaar. Door gebruik te maken van proven patterns and practices (in de vorm van software factories, application blocks, guides of een combinatie ervan) kan je de ontwikkeltijd drastisch verkorten en de kwaliteit en consistentie van het eindproduct verhogen. Dit artikel is bedoeld als leeswijzer die gebruikt kan worden om snel de juiste patterns and practices te vinden. Kijk op www.microsoft.com/patterns voor meer informatie.

Riccardo Becker en Raymond Binnendijk zijn beiden werkzaam als softwarearchitect bij LogicaCMG, www.logicacmg.com/nl. Voor vragen of opmerkingen zijn ze te bereiken via riccardo.becker@logicacmg.com en raymond.binnendijk@logicacmg.com.

Referenties

Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools, ISBN13: 9780471202844, auteurs: Jack Greenfield and Keith Short with Steve Cook and Stuart Kent.
Algemeen P&P: www.microsoft.com/patterns: www.davidhayden.com/blog/