

Architektur Richtlinien für SharePoint 2010 Anwendungen Teil 3 (SPC 2009)

Im 3. Teil der Architektur Richtlinien für SharePoint 2010 aus dem Vortrag von Mike Ammerlaan auf der SharePoint Conference 2009, geht es um Architektur Richtlinien für Logik, Building Blocks und der Benutzerschnittstelle in SharePoint.

Autor: Reiner Ganser

Im 3. Teil der Architektur Richtlinien für SharePoint 2010 aus dem Vortrag von Mike Ammerlaan auf der SharePoint Conference 2009, geht es um Architektur Richtlinien für Logik, Building Blocks und der Benutzerschnittstelle in SharePoint.

Im dritten und letzten Teil dieser kleinen Serie geht es um die Logik, Building Blocks und die Benutzerschnittstelle beim Aufbau von SharePoint 2010 Anwendungen.

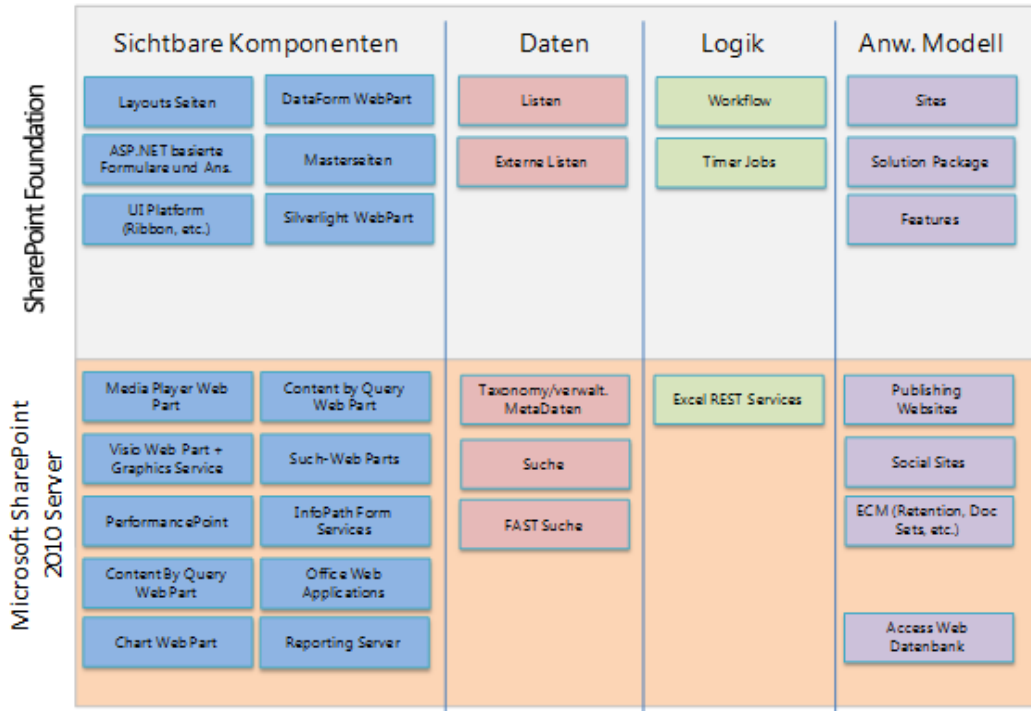
- [Teil 1: Einführung und Website Strukturen](#)
- [Teil 2: List Strukturen](#)
- **Teil 3: Logik, Building Blocks und Benutzerschnittstelle**

Dieser Blog ist angelehnt an den Vortrag "Architecture guidance for building applications in SharePoint 2010" von Mike Ammerlaan auf der SharePoint Conference 2009 in Las Vegas.

Die Ausführungen in diesem Teil beziehen sich fast ausschliesslich auf SharePoint 2010.

Building Blocks und Logik

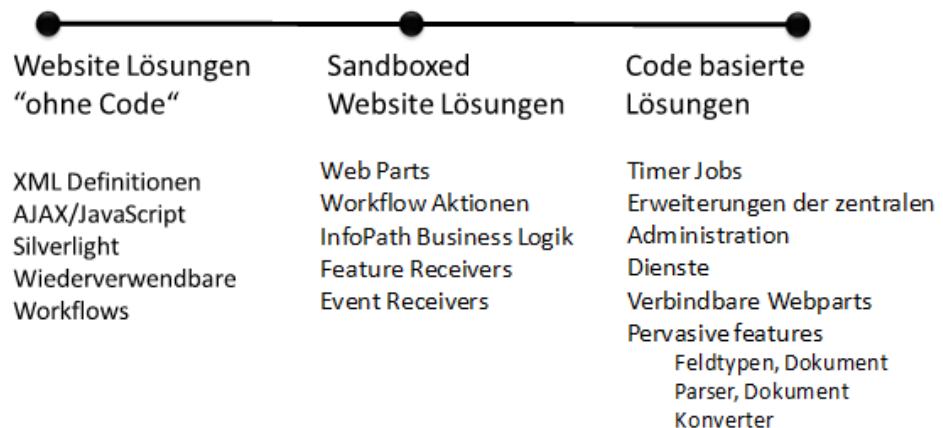
SharePoint Server 2010 als auch SharePoint Foundation bieten bereits etliche Bausteine (Building Blocks) wie Komponenten, Daten, Logik und Anwendungsmodelle, auf denen man eigene Anwendungen bauen kann.



Oftmals reichen diese Möglichkeiten aber nicht aus. Es gibt aber inzwischen sehr viele Microsoft Partner, die Lösungen auf der Basis von SharePoint anbieten. Zudem finden sich etliche Erweiterungen auch auf www.codeplex.com. Der erste Schritt vor der Erstellung von eigener Software sollte also immer die Evaluation der bestehenden Funktionalität, sowie der Recherche und Test von Komponenten von Partners bzw. Codeplex.

Building Block Strategien

Sind tatsächlich keine bestehenden Lösungen vorhanden bestehen mehr oder weniger die folgenden 3 Optionen:



Muster: Website Lösungen "ohne Code"

Bietet die meiste Flexibilität beim Deployment. Erfordert aber auch die Fokussierung auf die Basis und die Einschränkung auf die verfügbaren Funktionen.

- Mit Hilfe von Silverlight oder dem Client Objektmodell kann die Lösung erweitert und damit besser benutzbar gemacht werden
- Benutzung von ScriptLink/angepassten Masterseiten: ScriptLink ist eine neue Technologie in SharePoint 2010, bei dem eine Custom Action auf ein JavaScript verweisen kann. Durch diese Technik muss die Masterpage nicht immer wieder erhalten, um generelle Änderungen einzubringen.
- Xslt ListView Webpart, Inhalts Editor Webpart: Es gibt etliche Webparts, die XSLT basiert sind. Es muss daher nicht immer ein neues Webpart programmiert werden, sondern es genügt oft, ein XSLT Webpart mit den richtigen Daten zu füttern und ein angepasstes Stylesheet zu verwenden. Der Inhaltseditor Webpart ist ebenfalls sehr flexibel einsetzbar. Er erlaubt die Verwendung von beliebigem HTML und somit auch JavaScript und anderen auf dem Client gerendertem Code.

Muster: Sandboxed Lösungen

Erlaubt die Benutzung von eingeschränktem Code. Bietet aber dennoch eine gute Flexibilität beim Deployment von Lösungen.

- Alle Techniken von Lösungen "ohne Code" können verwendet werden
- Das Erstellen von RIAs (AJAX, Silverlight) für die Benutzerschnittstelle sollte berücksichtigt werden
- Nicht möglich sind folgende Elemente
 - Timer Jobs: Somit können auch keine lang laufenden Operationen einfach abgebildet werden.
 - Webservice Aufrufe: Dies ist momentan noch eine Limitation in Beta 2. Es ist abzuwarten, ob diese Limitation in der finalen Version aufgehoben wurde.
 - Elevation/Impersonation

Weitere Dinge, die man bei Sandbox Solutions wissen sollte:

- Die Sandbox kann erweitert werden
 - Dies ist eine privilegierte Operation und erfordert Full Trust Code. D.h. Dass dieser Code über den Administrator installiert werden muss. Über diesen Weg können dann spezielle Funktionen abgebildet werden. Z.B. wär auf diesem Weg die Unterstützung von Elevation/Impersonation oder Webservice Aufrufe möglich.
- Performance Einbussen sind wahrscheinlich: Da Aufruf des Objektmodells hat eine gewissen Routing Overhead, der die Aufrufe auf Gültigkeit prüft. Im Moment ist es jedoch zu früh, um hier genauere Performance Aussagen zu machen. Ein Full-Trust Webpart wird jedoch immer performanter arbeiten als ein Sand Boxed Webpart.

Muster: Code basierte Lösungen

Ermöglichen die Anpassung aller Aspekte von SharePoint.

- Die Treiber auf Kernel-Ebene von SharePoint: Sehr Leistungsfähig aber auch gefährlich bei falscher Anwendung
- Full-Trust WSP Lösungen werden für das Deployment verwendet
- Etliches an Funktionalität ist nur Full-Trust Code verfügbar:
 - Erweiterbare Feldtypen
 - Selbst erstellte Server Controls

- Administration & Timer Jobs

Wann sollten Code basierte Lösungen verwendet werden

- Web Content Management: Grosse Internet Website
 - Custom Server Controls
 - Verwendung von Benutzerdefinierten Feldtypen
 - Es wird dedizierte Hardware/Farm bereitgestellt, die auf die Erfordernisse dimensioniert wurde
- Grosse Anwendungs-Website
 - Es werden produktive und performante Workflows benötigt
 - Hohe Volumen erfordern native Performance

Präsentation

Performance der Benutzerschnittstelle

Ziel ist eine performante Benutzerschnittstelle zu bauen.

- Drei Aspekte, die die Benutzererfahrung bestimmen:
 - PLT1 (Page Load Time): Zeit für das Laden mit leerem Browser Cache -> Download aller CSS Dateien, Grafiken, HTML Dateien und andere Ressourcen, die zur Seite gehören. Bei einer schnellen Netzwerkverbindung ist dies zumeist kein Problem. Bei langsamer Netzwerkverbindung führt dies aber bzgl. der Benutzerakzeptanz eine sehr wichtige Rolle.
 - PLT2: Zeit für das Laden nach dem ersten Aufruf: Dies sind die dynamischen Inhalte, die sich bei jedem Seitenaufruf ändern.
 - "PLT3": Gesamt-Erfahrung
- Initiales Laden der Seite sollte kleiner als 600 KB sein
- Anzahl von initialen Aufrufen sollte kleiner als 30 sein (z.B. CSS-Dateien, Bilder, HTML Fragmente usw.). Gerade bei Seiten auf denen sich vielen Webparts befinden, ist dies nur schwer zu kontrollieren.
- Idealerweise 1 oder 2 SQL Abfragen/WCF Aufrufe pro Seite, möglichst < 5 WCF Aufrufe + Abfragen + queries: Vor allem sollte man dies bei der Homepage einer Website berücksichtigen und diese nicht mit Listen Webparts vollpflastern, da dies die Performance der Startseite erheblich bremsen kann. Besser ist es hier, die Webparts auf mehrere Seiten zu verteilen. Dies steigert zudem die Übersicht. Beispiel ist hier die persönliche Website, in welcher die Informationen in Reitern unterteilt sind, um nicht alles auf einer Seite zu haben und entsprechend viele Abfragen machen zu müssen. Bei weniger frequentierten Seiten muss man hierauf nicht unbedingt den Fokus legen.
- Client Anwendungen sollten Caching verwenden

Muster: Webpart-basierte Benutzerschnittstelle

Wenn Ihre Anwendung niedrige oder moderate Komplexität hat, sollte ein Webpart-basierter Ansatz gewählt werden. Dieser ist sehr flexibel, da die einzelnen Webparts jederzeit ausgewechselt werden können.

- Sowohl Admin, als auch Endbenutzer Komponenten können als Webparts erstellt werden

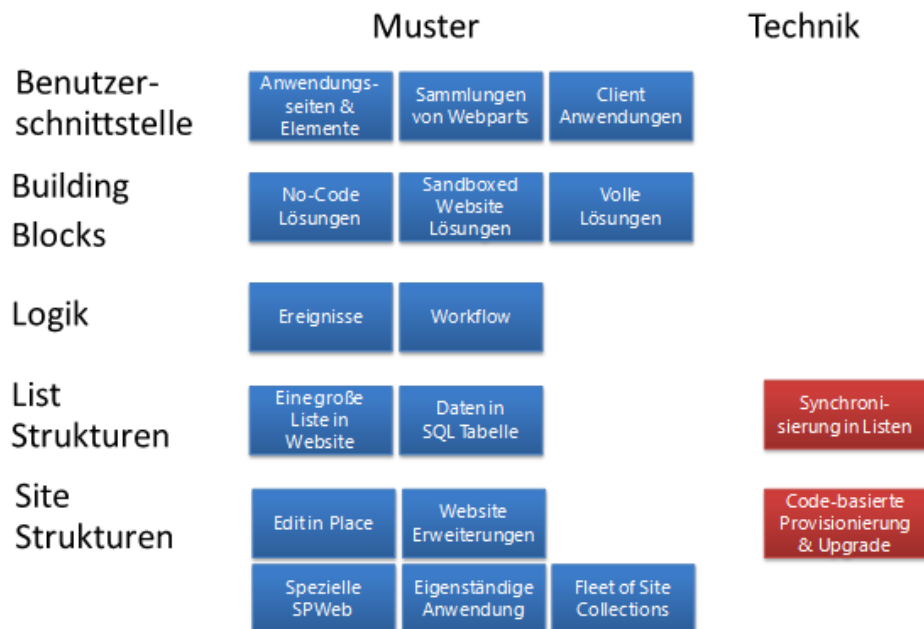
- Um die Webparts zu benützen können Webpart Seiten eingesetzt werden. Dies ist vor allem bei Sandboxed Solutions notwendig, da dort keine Layouts Seiten möglich sind
- Damit die Webparts miteinander interagieren können, lassen sich Webpart Verbindungen nutzen (z.B. Master-Detail Ansichten oder ein Webpart dient als Datenquelle oder Filter und die anderen Webparts zeigen die Daten an)
 - Webpartverbindungen sind nur Serverseitig möglich

Muster: Benutzung von Clients als Benutzerschnittstelle

Für sehr interaktive , reichhaltige Benutzerschnittstellen sollten Teile oder die ganze Seite in AJAX oder Silverlight entwickelt sein.

- Wahrscheinlich werden diese Techniken immer mehr zunehmen (bei der heutigen Rechenpower langweilt sich der Client sowieso ;-)
- Ribbon + Client Objektmodell + REST macht die Erstellung von interaktiven Benutzeroberflächen (viel) einfacher
- Silverlight
 - Für die meisten ASP.NET Entwickler wird es einfacher sein, mit Silverlight Oberflächen zu bauen als mit Skript
 - Unterstützung des lokalen System Cache eröffnen neue Möglichkeiten für die Erstellung von reichhaltigen Anwendungen

Zum Abschluss nochmal das im ersten Teil gezeigte Grafik, welche Möglichkeiten für die Umsetzung von SharePoint Anwendungen zur Verfügung stehen:



Zusammenfassung

Zum Schluss sind nachfolgend noch einmal einige der in dieser Blogserie behandelten Themen zum schnellen Überblick zusammengefasst:

- Website Struktur
 - Für einfache Anwendungen sollte eine einzelne Website genutzt werden. SharePoint ist für diese Anwendungen optimiert und viele Funktionen sind dadurch ohne Zusatzaufwand einfach nutzbar.
 - Wenn es sich um große Massen handelt oder wegen einfacherer Administration, sollte überlegt werden, die Anwendung auf mehrere Websitesammlungen zu verteilen. Man erkaufte sich dadurch aber mehr Aufwand durch die Websitesammlungs Barrieren (Dinge die zwar Websitesammlungsübergreifend genutzt werden sollen, die aber nur innerhalb einer Websitesammlung verfügbar sind). Einige der Barrieren existieren allerdings im Gegensatz zu MOSS 2007/WSS 3.0 in SharePoint 2010 nicht mehr in dieser Form (z.B. Übergreifende Inhaltstypen)
- Listen Struktur
 - Die Funktionalität in Listen wurden deutlich erweitert, so dass diese immer mehr in Richtung einer Datenbank Tabelle gehen. Nichts desto Trotz, sind viele Datenmodelle nur sehr schwer in SharePoint abzubilden und erfordern die Verwendung einer Datenbank. In SharePoint 2010 ist es jedoch auch einfacher geworden, Datenbanken einzubinden. Bevor man also versucht, krampfhaft die Daten in SharePoint Listen abzubilden, sollte man eher darüber nachdenken, ob es nicht besser wäre, die Datenstrukturen in einer Datenbank zu modellieren und diese dann in SharePoint einzubinden
- Benutzerschnittstelle
 - Das wichtigste ist eine gute Performance!!
 - Richtwerte: <5 Abfragen, <600KB Inhalt