

Story Paper

Hangman - Umgesetzt mittels .NET-Webservices

Februar 2005, © InterSolutions GmbH



Alle Rechte vorbehalten. Alle erwähnten Marken, Dienstleistungsmarken und Logos sind für die entsprechenden Firmen rechtlich geschützt und unterliegen dem Urheberrecht sowie auch anderen Gesetzen zum Schutz des geistigen Eigentums. Das Fehlen von Kennzeichnungen geschützter Marken bedeutet nicht, dass es sich um einen freien Namen, ein freies Bild oder einen freien Text im Sinne des Markenzeichnungsrechts handelt.

Microsoft Schweiz führte auch 2005 den bekannten Programmierwettbewerb „Codeduel“ durch. Dabei ging es dieses Mal um die Umsetzung des Spielklassikers „Hangman“ mittels .NET Webservices.

InterSolutions entschloss sich kurzfristig, am Wettbewerb teilzunehmen. Die Teilnahme war ein voller Erfolg: Neben der Bestätigung des firmeninternen Know-Hows machte die Umsetzung tierisch viel Spass...

ANFORDERUNGEN

Das Spiel Hangman

Hangman, bekannt auch als "Galgenraten", hat als Spielziel, ein gesuchtes Wort zu erraten. Zug um Zug raten zwei Spieler einzelne Buchstaben, bis das Lösungswort zu erkennen ist (z.B.: H*NGM*N).

Die Aufgabenstellung von Microsoft ist als „Hands-On-Lab“ aufbereitet und dient als praktisch umgesetzte Einarbeitung in die Stärken der .NET-Webservices Technologie.

Microsoft stellt dabei den Hangman-Spielleiterserver zur Verfügung, die Wettbewerbsteilnehmer entwickeln den webservice-basierenden Hangman-Spieler.

Hangman als Webservice

Damit der Spielleiter-Server mit dem Hangman-Webservice kommunizieren kann, muss dieser vier von Microsoft vorgegebene Methoden implementieren. Eine Webservice-Methode ist eine in WSDL (WebService Description Language) beschriebene Methode. Der Spielleiter-Server kann anhand der WSDL-Beschreibung der Methoden den Hangman-Spieler einbinden und aufrufen.

C# - Code Ausschnitt der Anforderung:

```
void ShakeHands(int gameId, string
currentWord);

string TakeTurn(int gameId);

void ReceiveTurnFeedback(int gameId, string
currentWord, string lastAnswer, int
yourLives, int opponentLives);

void ReceiveGameFeedback(int gameId, string
solution, bool youWin);
```

Mit diesen Methoden muss ein Regelwerk des Spiels Hangman abgedeckt werden, welches der Entwickler frei gestalten kann.

CRASHKURS IN SACHEN

WEBSERVICES

Was ist überhaupt ein Webservice?

Webservices arbeiten im Prinzip wie ein Modul eines Webserver-Dienstes (z.B. der NT-Service des IIS). Diese werden wie eine Webseite (z.B. ein CGI) gehostet und aufgerufen.

Ein Webservice ist ebenfalls eine Webapplikation, welche über ein standardisiertes Protokoll (WSDL) kommuniziert.

Ein Beispiel für einen solchen Dienst (Webservice) ist „Microsoft Passport“, welcher Entwicklern die Möglichkeit bietet, die Benutzerauthentifizierung outzusourcen.

Was ist Soap?

Soap ist ein auf XML basierendes Schnittstellen-Protokoll. Über dieses standardisierte Format kommunizieren z.B. Webservices.

Webservices und .NET

Da Webservices eine Definition einer Applikation sind, kann eine solche Applikation in einer beliebigen Sprache oder Umgebung realisiert werden. Jedoch ist das Abbilden der Protokolle WSDL/SOAP für die Schnittstelle eines Webservices eher aufwändig, wenn man dies selbst realisieren möchte. Durch das .NET-Framework ist allerdings das Erstellen eines Webservices in VS.NET über wenige Mausklicks möglich.

Sicherheit?

Durch das Textformat XML ist die Übertragung von Daten relativ offen. Die Sicherheit von Webservices ist durch verschiedene Verschlüsselungsmethoden bereits realisiert. Eine Sicherung ist beispielsweise über die SSL-Verschlüsselung des http-Protokolls möglich.

Stärken eines Webservice?

Die Bedeutung eines Webservice liegt in der Unabhängigkeit beim Verbinden von Umgebungen oder Systemen, welche durch die standardisierte Markup-Language XML (SOAP) gegeben ist. Basierend auf dem Internet Protokoll (IP) und http können Services von beliebigen Standorten aus miteinander verbunden werden.

Durch die SOAP-Schnittstelle können sehr flexibel Daten über Tags und Attribute übertragen werden - bis hin zu serialisierten Klasseninstanzen.

Schwächen eines Webservice?

Webservices erben sowohl Stärken als auch Schwächen des http-Protokolls. Eine Verbindung basiert somit immer auf dem Prinzip „Nachfrage->Antwort“. Eine beständige Verbindung ist nicht möglich. Wie bei einer Webapplikation muss eine „Unterhaltung“, welche über einen einzelnen Aufruf hinausgeht, über eine Session – d.h. einem Absender-Vermerk, mit dem der Server einen Client wieder identifizieren kann - gesteuert werden.

PLANUNG

Für die Umsetzung des „Hangman-Projekts“ wurde ein Aufwandsrahmen von zwei Wochen definiert. In diesem Rahmen sollten auch weitere Technologien und Produkte wie z.B. UML und Borland Together analysiert werden. So wurden folgende Teilziele formuliert:

- ▶ UML-Tauglichkeit in der Praxis
- ▶ Nutzen der LiveSource-Technologie des Borland Together-AddIns für VS.NET
- ▶ OO-basierende Umsetzung in VS.NET
- ▶ Nutzen der Serialisierung bei Webservices
- ▶ Wahrscheinlichkeitsprüfung anhand DB-Abfragen
- ▶ Remote Debugging

UMSETZUNG

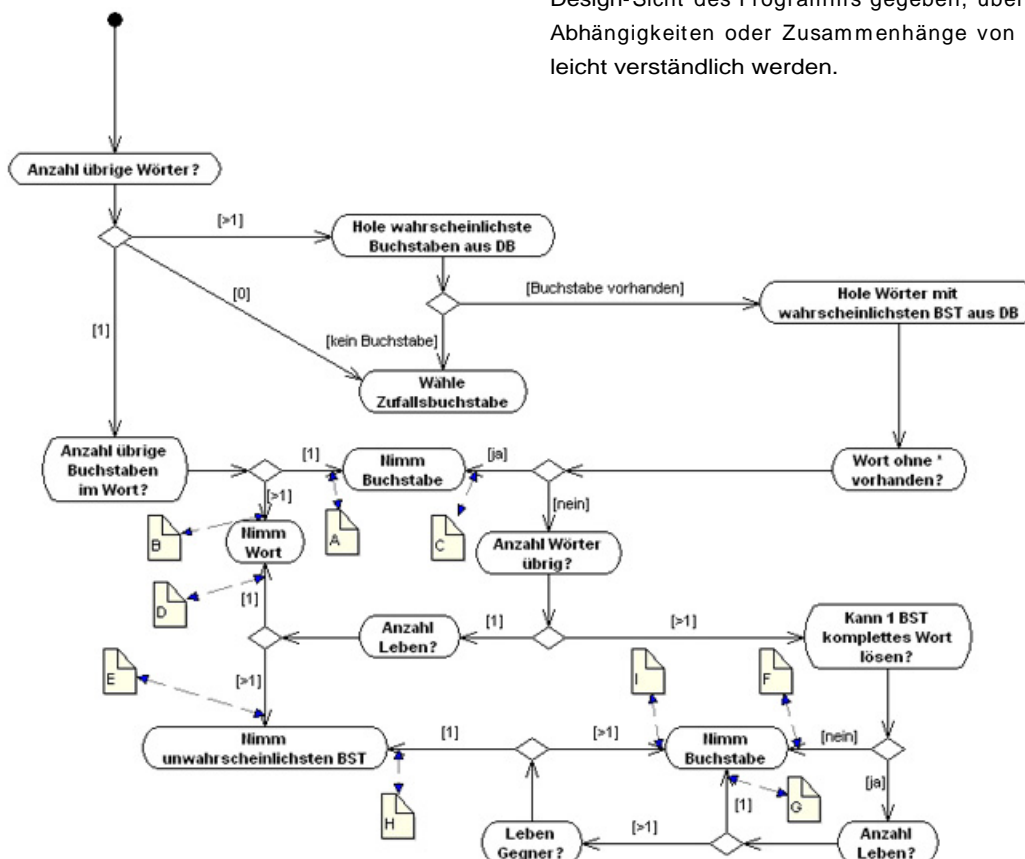
Die Evaluierungsziele Schritt für Schritt:

Um die Spieltaktik zu planen, wurde ein UML-Aktivitätsdiagramm mit ModelMaker erstellt. Ein Abbilden des Diagramms mit dem Together Add-in für VS.NET war aufgrund der noch fehlenden Funktionalität nicht abbildbar.

Durch die Planung in UML ergibt sich eine Unabhängigkeit von der Entwicklungssprache. So ist kein spezielles Entwickler-KnowHow nötig, um bei der Planung mitwirken zu können. Die Übersicht macht es für jeden leicht verständlich und bietet dem Entwickler einen sauberen Ablauf beim Umsetzen der Logik in Code.

Nutzen der LiveSource-Technologie des Borland Together-Add-Ins für VS.NET:

Borland bietet mit dem Produkt „Together“ eine Lösung zum Planen von Entwicklungsprojekten mit Hilfe von UML an. Eine nahtlose Integration durch ein MS VS.NET-Add-In soll das Entwickeln mit der Planungssoftware verbinden. Ein Klassendiagramm kann so auf der Basis von bestehendem Code erstellt werden. Im erstellten Diagramm können Klassen erweitert, per drag&drop platziert und ergänzt werden. Über die LiveSource-Technologie werden Änderungen im Code oder Änderungen im Klassendiagramm automatisch entsprechend ergänzt. Durch das Klassendiagramm ist eine Design-Sicht des Programms gegeben, über welche Abhängigkeiten oder Zusammenhänge von Klassen leicht verständlich werden.

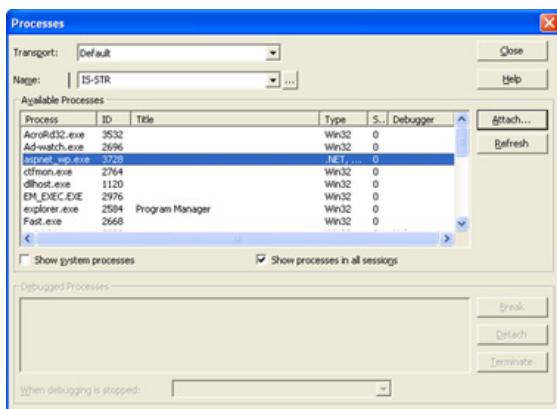


OO-basierende Umsetzung in VS.NET: Das Windows .NET Framework verwendet ein komponentenbasierendes Konzept, das Integrationsarbeiten (z.B. WindowsAPI-Zugriffe bei Eventlogs oder Performancecountern) überflüssig macht und Entwicklern somit ermöglicht, sich auf die Programmierung der Businesslogik zu konzentrieren. Somit muss „das Rad nicht immer neu erfunden werden“, was sich in der umfangreichen Komponenten-Sammlung widerspiegelt.

Nutzen der Serialisierung bei Webservices: Mit der Soap-Schnittstelle (basierend auf XML) können neben Images und Binärdaten auch Abbildungen von Klasseninstanzen übergeben werden. Dies geschieht über das „Serialisieren“ (*System.Runtime.Serialization.Formatters.Soap.SoapFormatter*) von Klasseninstanzen. Im Projekt Hangman wurde das Serialisieren verwendet, um die Session-Informationen in die Datenbank auszulagern. So erhielt der Webservice den grossen Vorteil von SOAP ("Simple Object Access Protocol") - die Plattformunabhängigkeit - indem er „cookie-less“ arbeitet.

Wahrscheinlichkeitsprüfung anhand DB-Abfragen: Um das Erraten des gesuchten Begriffs zu optimieren, wurde eine Wortliste in einem MS SQL Server abgelegt. Bei jedem Ratedurchgang wurden anhand von Stored Procedures die noch möglichen Begriffe gefiltert und die Wahrscheinlichkeit für den optimalen, nächsten wählbaren Buchstaben vorausberechnet.

Remote Debugging: Der publizierte Webservice kann von remote durch die „intermediate language“ vom Debugger abgefangen werden. Dadurch ist das Einklinken (Fernzugriff) der Entwicklungsumgebung an ein Testsystem (Integrationsumgebung) möglich. Hierzu muss lediglich der Framework-Prozess des entfernten Servers in VS.NET eingebunden werden.



FAZIT

Die Kombination zwischen Produkteevaluation und Spielwettbewerb hatte einen ganz besonderen Reiz. Der grosse Ansporn lag u.a. auch daran, eine gute Spiellogik zu entwerfen, da dieser im Wettkampf gegen andere Teilnehmer antrat.

Obwohl es nicht ganz aufs Podest gereicht hat, so hat sich die Zeit für die Wettbewerbsteilnahme auf alle Fälle gelohnt.

Technologien:

- MS IIS 6.0
- MS SQL Server 2000
- MS VS.NET 2003
- Borland Together .net Add-In
- Object Management Group - ModelMaker



Weitere Informationen unter
www.intersolutions.ch

Intersolutions GmbH
Hauptstrasse 23
CH - 4142 Münchenstein

Mail: info@intersolutions.ch
Phone: +41 61 416 86 00
Fax: +41 61 416 86 01