



MAI 2011

RICH-INTERNET-
ENTWICKLUNG MIT
QOOXDOO

Norbert Schröder
A:gon Solutions GmbH

■ Abstract

Das in Deutschland entwickelte JavaScript-Framework *qooxdoo* gehört zweifellos zu den Schwergewichten im Ring, wenn es um Entwicklungssysteme für browsergestützte Web- und Intranet-Anwendungen geht. Der folgende Beitrag stellt Ihnen dieses Open Source Projekt vor, beschreibt seine wichtigsten Merkmale und skizziert, wie die Arbeit mit diesem innovativen Werkzeug aussieht.

■ Was ist qooxdoo?

qooxdoo (QX) ist – ebenso wie z. B. ExtJS, ZK, GWT oder Adobe AIR – ein Framework zur Erstellung von Rich Internet Applications (RIA). Das sind Web-Anwendungen, die in Aussehen und Bedienung weitgehend einem üblichen Desktop-Programm entsprechen und mit einer traditionellen Web-Seite nur noch wenig gemein haben. QX ist ausschließlich in JavaScript (JS) implementiert, benötigt also keine Plugins (wie z.B. AIR) oder Compiler (wie z.B. GWT). Auf Grundlage der objektorientierten Fähigkeiten von JS ist mit QX ein vollständig klassenbasiertes, äußerst leistungsfähiges Entwicklungssystem entstanden, das bei der Gestaltung anspruchsvoller Benutzerschnittstellen oder der Programmierung ausgefeilter Client-Server-Kommunikation kaum Wünsche offen lässt.

Initiator von qooxdoo (gesprochen: "guckst du") ist der Internet Service Provider 1&1, der das Open Source-Projekt im Jahre 2005 ins Leben rief und ein derzeit 10-köpfiges Entwicklerteam beschäftigt, um das Projekt weiter voranzutreiben. Da der QX-Quellcode unter einer LGPL (Lesser General Public License) - bzw. EPL (Eclipse Public License) -Lizenz veröffentlicht wird, fallen für einen QX-Nutzer auch keine Lizenzgebühren an. Inzwischen existieren zahllose kleinere und größere QX-Anwendungen, die allerdings überwiegend in Firmen-Intranets genutzt werden und deshalb nicht öffentlich zugänglich sind; die (in Deutschland) wohl bekannteste mit QX erstellte RIA ist der neue Webmail-Client von GMX (Abb. 1).



Abbildung 1: Webmail-Client von GMX

■ Hauptmerkmale

Da qooxdoo primär auf die Erstellung komplexer grafischer Benutzeroberflächen (GUI) ausgerichtet ist, deren Komponenten über weite Strecken ereignisgesteuert interagieren, lässt sich das Framework nicht mit sog. JS-Toolkits, wie z. B. jQuery oder Prototype, vergleichen, die im wesentlichen zur Anreicherung von HTML-basierten Websites geeignet sind. Stattdessen hat QX eher den Charakter einer eigenen Programmiersprache, die zwar auf einem JS-Fundament ruht und die typischen JS-Charakteristika beibehält, aber eine zumindest in Teilen eigenständige Syntax sowie eine komplexe, in Schichten gegliederte Architektur aufweist (Abbildung 2).

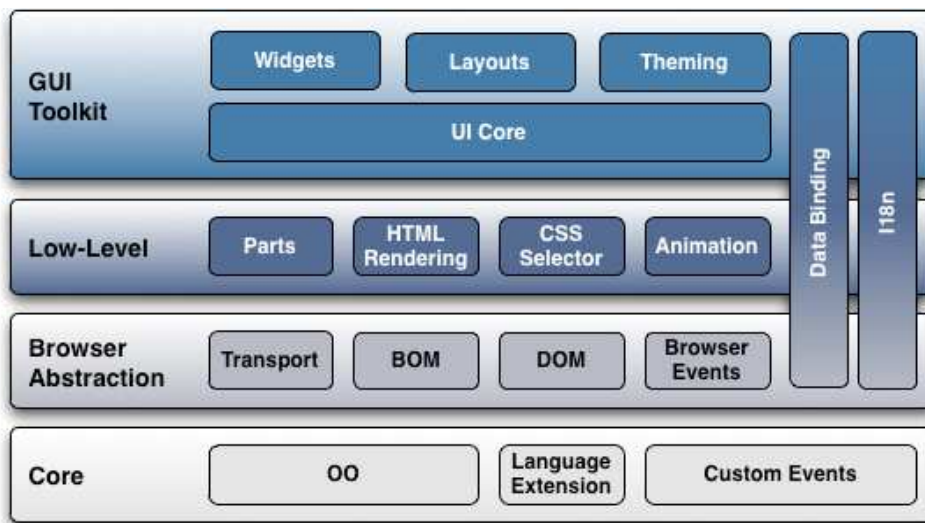


Abbildung 2: Architektur von qooxdoo

Die Core-Schicht liefert die Grundlagen des Frameworks. Hier wird z. B. das `qx.core.Object` definiert, das die Basisklasse fast aller übrigen QX-Objekte darstellt; auch das QX-eigene Event-System hat hier seinen Ursprung. Über diesem Kern ist eine Browser-Abstraktionsschicht angesiedelt, die u.a. das DOM (Document Object Model) abbildet und für die Server-Kommunikation zuständig ist. In der Low-Level-Schicht sind alle Basisfunktionalitäten für die Bildschirmausgabe gekapselt, dazu gehören beispielsweise Objekte für die DOM-Manipulation und verschiedene HTML-Elemente (Input, IFrame usw.). Die GUI-Schicht schließlich besteht wiederum aus einem Kern mit grundlegenden UI-Objekten und entsprechender Infrastruktur (Queues, Event Handling) sowie den darauf aufbauenden Top-Level-Widgets (Button, ComboBox, TabView usw.), dem Layout-System und den Theming-Mechanismen.

Als QX-Entwickler ist man üblicherweise nur mit den oberen beiden Schichten des Modells konfrontiert. Der große Vorteil: man braucht sich nicht mit Details des DOM oder mit CSS und HTML herzuschlagen. Hat man die unvermeidliche Einarbeitungsphase hinter sich gelassen, offenbart sich ein sauber implementiertes Konzept, das sich in der Praxis als äußerst stabil und zuverlässig erweist. Wer sich mit anderen OO-Sprachen wie Java, C++ oder Delphi

auskennt, wird kaum Probleme haben, die QX-Klassenbibliothek zu durch-schauen, mit ihr zu arbeiten und sie bei Bedarf auch zu erweitern. Um ein QX-Programm zu erstellen, sind weder HTML- noch CSS- oder DOM-Kenntnisse notwendig, und auch die in der traditionellen Web-Entwicklung so gefürchteten "Browser-Eigenheiten" bleiben QX-Entwicklern gleichgültig: die Programme laufen in allen modernen Browsern mit identischer Oberfläche und Funktionalität. Sogar der IE6 wird noch unterstützt, der in vielen Unternehmen immer noch zum Standard zählt. Lediglich die Performance variiert naturgemäß, je nach Leistungsfähigkeit der im jeweiligen Browser implementierten JS-Engine.

Bei der Programmentwicklung wird man durch das Framework in vielfältiger Weise unterstützt:

- Beispielsweise lässt sich die notwendige Infrastruktur eines neuen QX-Projektes quasi auf Knopfdruck erzeugen (per Python-Skript);
- das 440-seitige Benutzerhandbuch behandelt die relevanten Aspekte der QX-Programmierung im Allgemeinen gut verständlich und enthält auch ein ausführliches Tutorial;
- mit dem API-Viewer steht eine umfassende Dokumentation der Klassenbibliothek zur Verfügung;
- der Demo-Browser mit seinen hunderten von Beispielprogrammen ist eine uner-schöpfliche Quelle der Erkenntnis und Inspiration;
- und wenn sonst nichts mehr weiterhilft, kann man sich über eine Mailing-Liste jeder-zeit an das QX-Entwicklerforum wenden, wo man in der Regel innerhalb weniger Stunden auf fast jede Frage detaillierte Antworten erhält.

Sobald die ersten Hürden genommen sind, wird man feststellen, dass die RIA-Entwicklung mit qooxdoo einfach Spaß macht – nicht zuletzt deshalb, weil man zumeist innerhalb kurzer Zeit zu vorzeigbaren Ergebnissen kommt.

Oberflächlichkeiten

Für die GUI-Gestaltung steht dem Entwickler eine reichhaltige Auswahl von Komponenten ("widgets") zur Verfügung, also z.B. Eingabefelder, Buttons, diverse Auswahlboxen, Menüs oder Fensterelemente. Leider existiert für qooxdoo bisher keine integrierte Ent-wicklungsumgebung, so dass man gezwungen ist, die Bildelemente "von Hand" anzu-ordnen. Dies wird jedoch – zumindest teilweise – durch eine Reihe von "intelligenten" Layout-Klassen (wie z. B. Grid, Flow oder Dock) wettgemacht, die selbständig dafür sorgen, dass die enthaltenen Widgets stets korrekt positioniert und arrangiert werden.

Wie in einer objektorientierten Sprache üblich, kennt auch jedes QX-GUI-Objekt eine klassenspezifische Anzahl von Standardereignissen, die durch Benutzeraktionen oder datengesteuert ausgelöst und mittels Event-Handlern ausgewertet werden können. Außerdem ist es möglich, eigene Events zu definieren und beliebigen Objekten zuzuordnen. Da QX ferner Drag & Drop sowie ein fast ausschließlich tastaturgesteuertes Bedienkonzept unterstützt, ist eine gute QX-Anwendung kaum von einem herkömmlichen Desktop-Programm zu unterscheiden.

Mit dem Server auf Du und Du

Zur Kommunikation mit einem Server stehen einem QX-Client mit der Request- bzw. der RPC-Klasse prinzipiell zwei Wege offen. Während ein instantiiertes Request-Objekt im Wesentlichen für "klassische" AJAX-Aufrufe nutzbar ist, steht mit der RPC-Klasse eine Schnittstelle für Remote Procedure Calls zur Verfügung (Abbildung 3). In dieser Hinsicht bietet qooxdoo mehr Komfort als z.B. das ansonsten vergleichbare ExtJS. In beiden Fällen spielt die jeweilige Server-Infrastruktur für qooxdoo keine Rolle, solange gewisse formale Regeln bei der Strukturierung der Response eingehalten werden. Bei Verwendung der RPC-Schnittstelle sind diese Regeln etwas strikter als für einfaches AJAX, d. h. auf dem Server muss ein RPC-Backend existieren, das im wesentlichen das JSON-RPC-Protokoll implementiert. Für Java, PHP, Perl und Python sind entsprechende Backends bereits vorhanden; falls der Server eine andere Sprache spricht, muss die Response-Schnittstelle ggf. nachgerüstet werden.

```
// Instantiierung des RPC-Objekts
// Parameter: URL, Service-Name
var rpc = new qx.io.remote.Rpc(
    "http://localhost:8080/qooxdoo/.qxrpc",
    "qooxdoo.test"
);

// Callback-Funktion für asynchronen Aufruf
var handler = function(result, exc) {
    if (exc == null) {
        alert("Ergebnis des Aufrufs: " + result);
    } else {
        alert("Fehler während des Aufrufs: " + exc);
    }
};

// Und ab geht die Post
// Parameter: Callback-Handle, Service-Methode, Methoden-
// parameter
rpc.callAsync(handler, "echo", "Test");
```

Abbildung 3: RPC-Beispiel (mit Java-Backend)

Do you speak I18n?

Auch die Internationalisierung von Programmen ist mit qooxdoo problemlos möglich. Der sogenannte LocaleManager des QX-Kerns stellt verschiedene Methoden zur Lokalisierung und Übersetzung von Strings bereit und unterstützt sämtliche bekannten Sprachen, "at least on

this planet" (qooxdoo-Eigenwerbung). In der Praxis bedeutet dies u.a., dass die Benutzerschnittstelle einer entsprechend vorbereiteten RIA jederzeit von einer zur anderen Sprache umschalten kann – und zwar während der Laufzeit.

In der QX-Anwendung des Lyrik-Portals *di-lemmata* wird dies anschaulich demonstriert: Hier kann der Benutzer zwischen Deutsch, Englisch und Russisch als GUI-Sprache wählen (Abbildung 4 und <http://www.di-lemmata.de/lib.>)

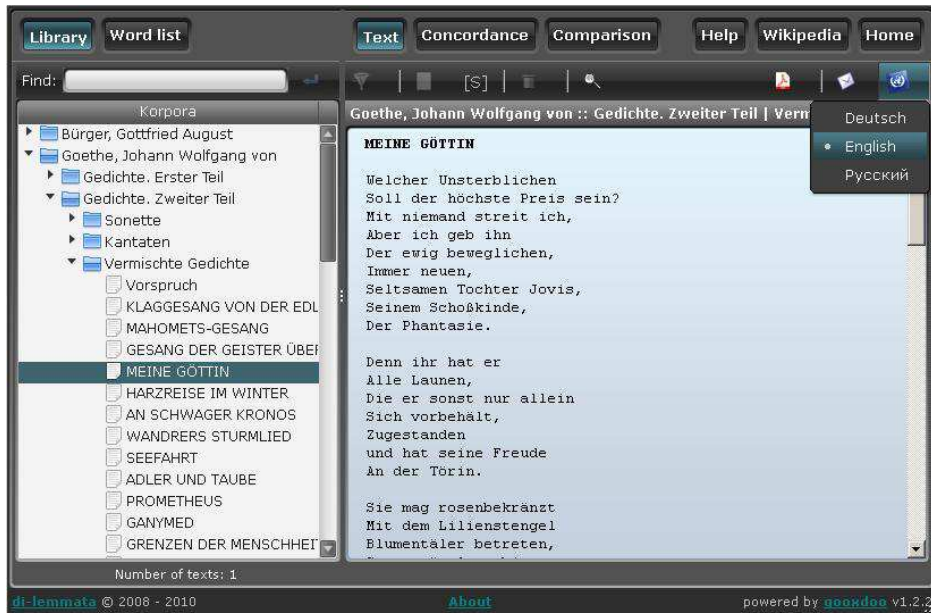


Abbildung 4: Eine dreisprachige qooxdoo-Anwendung

Des Kaisers wechselnde Kleider

Weiterer Pluspunkt des QX-Frameworks: Sämtliche Elemente, aus denen sich eine GUI zusammensetzt, können visuell den eigenen Vorstellungen oder Erfordernissen (Stichwort "Corporate Design") angepasst werden (sog. "Theming"). Neben den drei standardmäßigen QX-Themes gibt es inzwischen auch einige weitere, von enthusiastischen Anwendern beige-steuerte Oberflächen, die vielleicht nicht jedermanns Geschmack sind, aber immerhin demonstrieren, was prinzipiell möglich ist. Einziger Wermutstropfen: Das Theme einer QX-RIA kann nicht zur Laufzeit ausgewechselt werden, dazu ist ein Programmneustart erforderlich.

■ Ein Rezept für qooxdoo

Zutaten

Für die Entwicklung eines QX-Programms sind neben dem qooxdoo-SDK noch ein gängiger Web-Browser, ein UTF-8-fähiger Text-Editor sowie ein Python-Interpreter notwendig. Letzte-

res ist die Voraussetzung für einige Batch-Skripte, die z.B. zur Optimierung des JS-Codes, zur Generierung von API-Dokumentationen oder im Rahmen der Internationalisierung von Programmen wertvolle Dienste leisten. Auf Unix-Systemen ist Python in der Regel bereits installiert, unter Windows lässt es sich leicht nachrüsten. Das QX-SDK selbst enthält den Quellcode des kompletten JS-Frameworks, inklusive aller Ressourcen, Utilities und Demo-Programme, und ist in der entpackten Version 1.4 ca. 72 MB groß. Jedes QX-Programm kann direkt aus dem Dateisystem heraus gestartet werden. Deshalb sind auf dem Entwicklungsrechner keine zusätzlichen Komponenten erforderlich, sofern eine QX-Anwendung selber nicht mit einem HTTP-Server kommuniziert. Andernfalls muss ein lokaler HTTP-Server mitsamt der dazugehörigen Backend-Software installiert sein.

Jetzt wird gekocht!

Die eigentliche Programmentwicklung erfolgt, wie bei einer Interpreter-Sprache üblich, zu- meist in zwei Schritten:

1. Bearbeitung des Quellcodes
2. Testlauf (im Browser).

Ein QX-Programm muss jedoch hin und wieder "generiert" werden, so dass gelegentlich noch ein dritter Schritt hinzukommt. Dieses Generieren kann als ein kombiniertes Kompilieren und Linken aufgefasst werden: Das QX-SDK besteht aus hunderten einzelner Dateien (pro Klasse eine Datei), die einerseits durch eine verzweigte Klassenhierarchie miteinander in Beziehung stehen, von denen andererseits jedoch nur ein bestimmter Teil für die Programmausführung benötigt wird. Damit sich der Entwickler nun nicht selber darum kümmern muss, die erforderlichen Dateien zusammenzustellen und in sein Programm einzubinden, existiert mit dem in Python geschriebenen "Generator" ein Kommandozeilen-Tool, das ihm diese Arbeit abnimmt. Der Generator erzeugt in der sog. "Source"-Version des Programms einen JS-Loader, der sämtliche benötigten QX-Klassen zusammenführt und seinerseits durch Aufruf einer HTML-Datei geladen wird.

Des Weiteren ist der Generator auch für das ordnungsgemäße Einbinden grafischer Ressourcen oder für die Erzeugung der Textdateien für die Internationalisierung zuständig. Typischerweise muss die Generierung immer dann gestartet werden, wenn dem Programm eine neue Klasse oder Grafikdatei hinzugefügt wurde. Der vom Generator erstellte Loader braucht vom Entwickler übrigens niemals angetastet zu werden, man kann sich also voll und ganz auf die Bearbeitung des eigentlichen Quellcodes konzentrieren.

Zwischendurch abschmecken...

Zur Fehlersuche in einem QX-Programm kann der Entwickler auf verschiedene Hilfsmittel zurückgreifen. Zum einen stellt QX eine eigene Konsole bereit, die mit Hilfe QX-eigener Sprachkonstrukte für die Ausgabe von Variableninhalten oder zur Verfolgung des Programmablaufs jederzeit eingeblendet werden kann; zum anderen existiert mit dem in QX geschriebenen In-

spektor ein mächtiges Werkzeug, das es erlaubt, die Eigenschaftswerte aller GUI-Objekte und QX-Klassen einer laufenden Anwendung zu inspizieren und zu manipulieren. Darüber hinaus können QX-Programme natürlich auch mit einem der zahlreichen Debugging-Tools untersucht werden, die für jeden gängigen Browser frei verfügbar sind (besonders beliebt bei QX-Entwicklern ist die Firefox-Erweiterung "Fire-bug").

...und dann servieren

Ist die Entwicklung einer QX-RIA abgeschlossen, muss das Programm für das Deployment auf einem Web- oder Intranet-Server vorbereitet werden. Dazu wird wieder der Generator benötigt, der für produktionsreife Anwendungen neben dem oben beschriebenen "Kompilieren und Linken" auch diverse Optimierungen vornimmt, den JS-Code komprimiert (entfernen von Leerzeichen und Zeilenumbrüchen, verkürzen von Variablen- und Funktionsnamen) und die benötigten Projektressourcen zusammenstellt. Im Ergebnis erhält man eine Ordnerstruktur, die alle für den Regelbetrieb erforderlichen Dateien enthält. Dazu gehören im Standardfall die komprimierte JS-Datei, eine index.html (Abbildung 5) zum Laden des JS-Programms sowie ein Ressourcen-Ordner mit allen in der GUI verwendeten Grafikdateien. Die Funktionsfähigkeit auch dieser Produktionsversion der QX-Anwendung kann bei Bedarf auf dem Entwicklungsrechner abschließend getestet werden, bevor sie endgültig auf den Server übertragen wird.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://
www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" /><br/>
  <title>QX-Beispiel</title>
  <script type="text/javascript" src="script/
beispiel.js"></script>
</head>
<body></body>
```

Abbildung 5: HTML-Loader eines qooxdoo-Programms

■ Hinein ins Vergnügen!

Wer sich mit der Erstellung von HTML-basierten Web-Seiten einigermaßen auskennt, mit JavaScript aber noch nichts oder wenig zu tun hatte, wird bei näherer Beschäftigung mit einem JS-Framework wie qooxdoo möglicherweise sehr erstaunt sein, welche Perspektiven sich daraus eröffnen. Ohne sich Gedanken über die Zielplattform machen oder über eigenwilliges Browser-Verhalten ärgern zu müssen, lassen sich Web- und Intranet-Anwendungen entwickeln, die wirklich interaktiv, datenbankgestützt und visuell sowie funktional in allen Browsern identisch lauffähig sind.

Es ist fraglos nicht von der Hand zu weisen, dass die Einarbeitung in qooxdoo einer durchaus steilen Lernkurve folgt. Wenn man noch keine JS-Erfahrung hat, mag außerdem hinzukommen, dass das eine oder andere Sprachkonstrukt zunächst etwas seltsam anmutet. Andererseits kann man auf ein ausführliches Benutzerhandbuch des Gesamtsystems zurückgreifen, die QX-Klassenhierarchie ist ausgezeichnet dokumentiert, es gibt zahllose Beispielprogramme für fast jeden denkbaren Anwendungszweck und, last but not least, mit der QX-Mailingliste steht jedem Interessierten ein Informationsforum zur Verfügung, das – laut Aussage vieler "Umsteiger" – in punkto Reaktionszeit und Qualität seinesgleichen sucht.

Der größte "Nachteil" des QX-Frameworks besteht vermutlich darin, dass er bisher noch eine relativ geringe Publizität genießt. Für IT-Dienstleister könnte sich dieses Manko aber auch als Vorteil erweisen – wenn es nämlich gelingt, genügend Know-how aufzubauen, um sich im zukunftssträchtigen Markt der RIA-Anwendungen als Dienstleister für QX-Services aller Art zu etablieren. Innerhalb Deutschlands gibt es auf diesem Gebiet bisher keine erkennbare Konkurrenz.

■ Referenzen

Homepage:	http://qooxdoo.org
Demo-Showcase:	http://demo.qooxdoo.org/current/showcase
QX-Klassenbibliothek (API):	http://demo.qooxdoo.org/current/apiviewer
Benutzerhandbuch:	http://manual.qooxdoo.org/1.4.x
Entwicklerforum:	http://forum.qooxdoo.org
Ausgewählte QX-Anwendungen:	http://qooxdoo.org/community/real_life_examples
qooxdoo-Anwendung di-lemmata	http://www.di-lemmata.de/lib
Bericht über Qooxdoo in „Ajaxschmiede“	http://www.ajaxschmiede.de/ajax-frameworks/qooxdoo-ein-web-entwicklungsframework-aus-deutschland/
ExtJS – eine Alternative zu qooxdoo	http://www.sencha.com/products/extjs/

■ A:gon Solutions GmbH

Die A:gon Solutions GmbH, 2004 gegründet, ist ein unabhängiges IT-Dienstleistungsunternehmen mit Firmensitz in Eschborn bei Frankfurt und weiteren Standorten in Hamburg und Berlin. Das branchenübergreifende Dienstleistungsportfolio von A:gon umfasst das „A:gon-proven IT-Consulting“, eine bewährte, herstellernerneutrale IT-Beratung; sowie die „A:gon-tailored IT-Solutions“, zu denen maßgeschneiderte, individuelle Softwareentwicklung, passgenaue Softwareintegration und effiziente Business Intelligence Lösungen gehören. A:gon stellt sich mit professionellem Projektmanagement, proaktivem Anforderungsmanagement und änderbaren Softwarearchitekturen auf die individuellen Bedürfnisse seiner Kunden ein. In ausgewählten Branchen wie Banken, Versicherungen, Aviation und Health Care bietet A:gon gemeinsam mit seinen Partnern Lösungen, die auf fundiertem Geschäftsprozess-Know-how beruhen und speziell auf die jeweilige Branche zugeschnitten sind: die „A:gon-tailored Business Solutions“. Die plattformübergreifende technologische Kompetenz bei A:gon reicht von klassischen Mainframe-Architekturen bis hin zu modernen Java/JEE Web- und Portal-Architekturen. Zu den Referenzkunden von A:gon gehören unter anderen die AOK Berlin-Brandenburg, die Commerzbank, die Deutsche Bank, die Deutsche Bank Bauspar, die Deutsche Börse, die Finanz Informatik und die Deutsche Lufthansa.

Copyright:

A:gon Solutions GmbH

Frankfurter Strasse 71-75
D-65760 Eschborn
Telefon : +49 6196 80269 0
Telefax : +49 6196 80269 11
<http://www.agon-solutions.de>

Handelsregister Frankfurt HRB 58185
St.-Nr. 4022826171
Geschäftsführer: Udo Peters