dpi

Christian Pfeil

Adobe AIR

RIAs für den Desktop entwickeln

Know-how für HTML/Ajax- und Flash/Flex-Entwickler

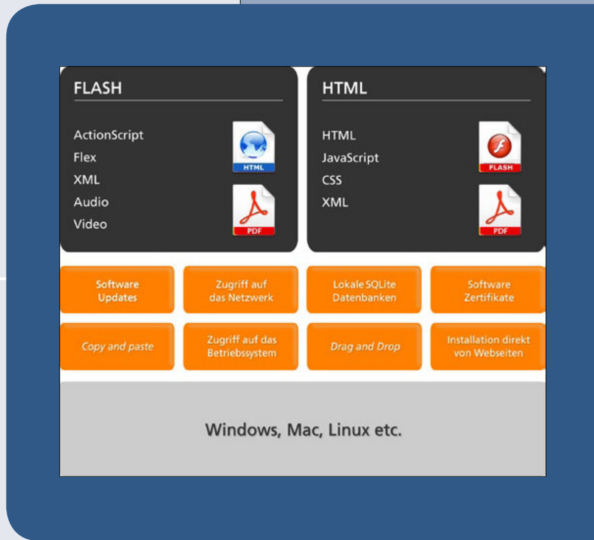
Mit Technologie-Überblick

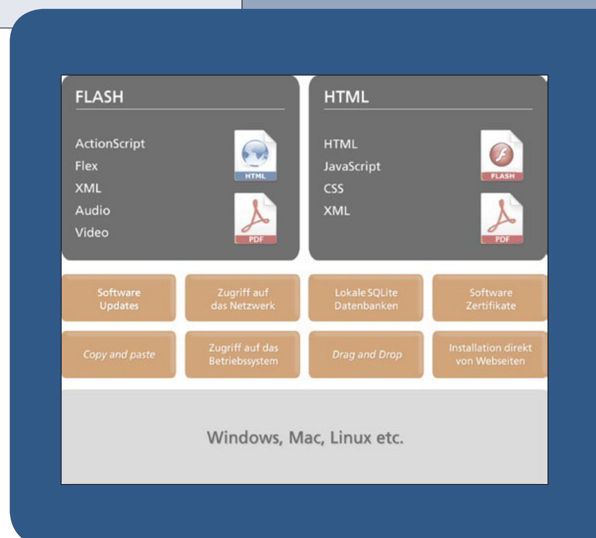
DVD Adobe AIR- und ADOBE Flex-Trials

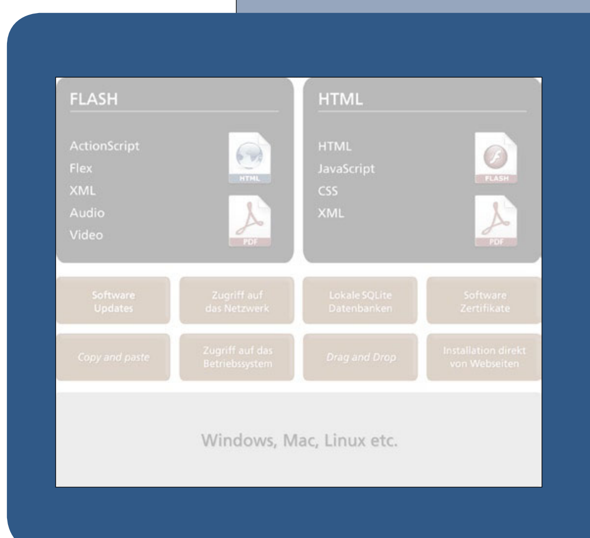
ADDISON-WESLEY

3

DIE ARCHITEKTUR VON AIR







Adobe AIR (Adobe Integrated Runtime) ist eine systemunabhängige Laufzeitumgebung. Mit AIR sind Sie als Entwickler in der Lage, Ihr bestehendes Wissen im Bereich Flash, Flex sowie HTML, Ajax, CSS etc. zu nutzen, um bereits bestehende Webapplikationen auf den Desktop zu bringen oder *RIA* (*Rich Internet Applications*) für den Desktop zu erstellen. Doch was heißt das genau? Lassen Sie uns einen Blick auf die Architektur von Adobe AIR werfen. Ich möchte Ihnen erklären, wie Adobe AIR aufgebaut ist und wo ich die Vorteile der Architektur sehe.

3.1 Die Laufzeitumgebung von AIR

Die Architektur von Adobe AIR ermöglicht es Entwicklern, Desktop-Applikationen zu erstellen, wie man sie aus dem Java- oder .NET-Bereich kennt. Der Benutzer kann dann mit AIR-Applikationen wie mit einer normalen Desktop-Anwendung interagieren. Der Vorteil der AIR-Architektur liegt jedoch klar auf der Hand: Bestehende HTML-, Flash- oder Flex-Projekte können problemlos in AIR integriert und auf den Desktop *deployed* werden. Da die AIR-Architektur im Grunde genommen sehr übersichtlich und einfach aufgebaut ist, können HTML- und Flash-Entwickler ihr bestehendes Wissen nutzen, um mit ein wenig Mehraufwand AIR-Applikationen zu erstellen. Insbesondere die *API* ist momentan noch sehr überschaubar.

Laufzeitumgebung und Architektur von AIR sind nicht von einem bestimmten Betriebssystem abhängig. Sie entwickeln daher nicht speziell für ein Betriebssystem, das auf einem System wie Windows, Mac oder Linux läuft.

Im Normalfall würde dies bedeuten, dass Sie je nach Betriebssystem umfangreiche Anpassungen an Ihrer Applikation vornehmen müssten. Dies ist in den meisten IT-Projekten ein echter Kostenfaktor und Budget-Killer. Die Architektur von AIR ist so aufgebaut, dass Entwickler die Laufzeitumgebung über eine API direkt ansprechen können. Die Laufzeitumgebung sorgt dann dafür, dass Ihre Applikation auch wirklich auf allen Betriebssystemen ohne Fehler lauffähig ist und Sie sich nicht mit einer zeitraubenden Optimierung für verschiedene Systeme befassen müssen. Die lästige Fehlersuche auf unterschiedlichen Betriebssystemen entfällt komplett. Sie können sicher sein, wenn Sie Adobe AIR auf einem Windows-Rechner entwickeln, dass sich Ihre Applikation auf einem Mac- oder Linux-Rechner identisch verhält und (fast) alle Features ebenfalls verfügbar sind. Toll, nicht wahr?

Ich möchte Ihnen ein kleines Beispiel geben, nach welchem Prinzip die Laufzeitumgebung von AIR arbeitet. Stellen Sie sich vor, Sie müssten zehn Menschen in zehn verschiedenen Sprachen zum gleichen Zeitpunkt ein Kapitel aus einem Buch vorlesen. Nahezu unmöglich, oder? Stellen Sie sich nun vor, es gäbe eine Art Maschine, die Ihr gesprochenes Wort zeitgleich und ohne Verzögerung in alle zehn Sprachen übersetzt, so dass Sie jeder verstehen kann. Diese Maschine ist im Prinzip die AIR-Laufzeitumgebung. Sie ist in der Lage, die Kommandos im Quellcode so umzusetzen, dass jedes Betriebssystem sie versteht und anwenden kann.

Die Laufzeitumgebung dient als *Vermittler* zwischen der Applikation und dem jeweiligen Betriebssystem. Sie sorgt dafür, dass bestimmte Funktionen zum Start oder beim Beenden der Applikation ausgeführt werden können. Außerdem stellt diese ein *API* (*Application Programming Interface*) zur Verfügung, das auf bestimmte Funktionen des Betriebssystems zurückgreifen kann.

Ein Beispiel hierfür wären die Methoden, die einen Zugriff auf das Dateisystem des Betriebssystems ermöglichen. Unter Umständen müssten Sie beim Schreiben einer

Datei auf die lokale Festplatte je nach Betriebssystem verschiedene Optimierungsschritte vornehmen.

Die Laufzeitumgebung von AIR nimmt Ihnen diese Aufgabe ab und bündelt alle notwendigen Schritte in nur einer Methode und stellt diese dem Entwickler über die Adobe API bereit.

Unterschiede zwischen Betriebssystemen

Unter Umständen kann es natürlich sein, dass auf dem einen Betriebssystem bestimmte Features vorhanden sind und auf einem anderen nicht. Ein einfaches Beispiel wäre die Methode, um das Mac-*Dock* (die *Menüleiste* des Macintosh-Betriebssystems) hüpfen zu lassen. Mit einem einfachen Befehl fängt das zu Ihrer AIR-Applikation gehörende Icon im *Dock* zu hüpfen an, um auf sich aufmerksam zu machen. Auf dem PC gibt es kein *Dock*, sondern eine *Taskleiste*. Demnach bliebe der Aufruf der Methode auf dem PC ohne Funktion, würde jedoch trotzdem keinen Fehler oder Absturz des Programms verursachen.

Ich bin mir sicher, dass Sie sich bei Ihrer Arbeit als HTML- oder Flashentwickler mit Sicherheit schon einmal mit der lästigen Frage beschäftigen mussten, welche Flash-Version für ein Projekt angemessen ist. Die Laufzeitumgebung von AIR enthält einen integrierten Flash Player. Die lästige Frage, ob auf dem Zielsystem Flash installiert ist und in welcher Version, entfällt somit vollkommen. Sofern Sie Ihre AIR-Applikation auf Basis der Technologie Flash entwickeln oder in Ihrer HTML-basierten AIR-Applikation Flash-Elemente einsetzen, können Sie immer davon ausgehen, dass Ihre Inhalte korrekt dargestellt werden und dass der Nutzer kein Plug-in nachträglich installieren muss, um Ihre Applikation nutzen zu können.

Unterstützung für unterschiedliche Sprachen

Die Installation von Adobe AIR 1.1 und alle laufzeitspezifischen Dialoge sind für die folgenden Sprachen lokalisiert:

- Traditionelles Chinesisch
- Vereinfachtes Chinesisch
- Englisch
- Französisch
- Deutsch
- Italienisch
- Japanisch
- Koreanisch
- Brasilianisches Portugiesisch

3.2 Unterstützung für HTML, JavaScript und Ajax

Die Entwicklung von Webkit Engine wird seit 2003 von Apple koordiniert und steht unter LGPL und BSD-Lizenz.

Adobe AIR bietet zudem die Möglichkeit, HTML-Seiten laden und anzeigen zu lassen. Ebenfalls ist eine Unterstützung für Ajax- und JavaScript-Code innerhalb der HTML-Seiten vorhanden. Die Technologie, die für die Anzeige und Ausführung von HTML-, JavaScript- und Ajax-Code verantwortlich ist, basiert auf dem OpenSource-Projekt Webkit Engine. Es handelt sich hierbei nicht etwa um einen neuen Browser, denn es gibt schließlich schon genug Browser, auf die Webseiten optimiert werden müssen, sondern um eine HTML-Render-Bibliothek, die bereits seit langem in Apples Safari und anderen Browsern zum Einsatz kommt. Adobe AIR setzt damit auf eine etablierte Technologie namens Webkit Engine zur Darstellung von HTML-Seiten.

Hauptgrund für die Wahl von Webkit Engine ist mit Sicherheit die Kompaktheit der Rendering-Bibliothek. Adobe möchte die Zielgröße der Laufzeitumgebung möglichst gering halten.

In AIR können Sie sowohl HTML als auch Flash als Kerntechnologie einsetzen. Doch dies ist noch nicht alles! Die wohl durchdachte Architektur von Adobe AIR ermöglicht es Ihnen, Inhalte beliebig miteinander kommunizieren zu lassen. Dies bedeutet, dass Sie mit ActionScript-Code in der Lage sind, auf das *DOM* (*Document Object Model*) einer geladenen HTML-Seite zuzugreifen. Natürlich funktioniert das Prinzip auch umgekehrt. Mit HTML-Code können Sie bestimmte Ergebnisse auf ActionScript-Ebene auslösen. Sie schreiben ActionScript-Code direkt in HTML und haben somit Zugriff auf die AIR *API*.

3.3 Integration und Darstellung von PDF-Inhalten

Neben der Darstellung von HTML-Inhalten lassen sich ebenfalls PDF-Dokumente laden und darstellen. Adobe AIR hat keine eigenständige *Rendering Engine* für PDF-Dokumente, sondern greift auf eine installierte Instanz des Acrobat Reader zurück, um PDF-Dokumente darzustellen. Die Darstellung von PDF-Dokumenten ist identisch mit der Darstellung, wie Sie sie vom Browser kennen. Eine AIR-Applikation kann nicht nur aus einem einzelnen PDF-Dokument bestehen, sondern benötigt entweder HTML oder Flash als Container, in das es integriert werden kann.

3.4 Brücken bauen mit Script Bridging

Der Begriff *Script Bridging* beschreibt die Eigenschaft von Adobe AIR, über die Grenzen einer Technologie hinweg Aktionen auszuführen. HTML, Flash und PDF-Inhalte

können mithilfe von *ECMAScript* miteinander kommunizieren und Informationen austauschen.

Was im Adobe-Jargon als *Bridge* (engl. Brücke) zwischen zwei Technologien bezeichnet wird, gibt es in ähnlicher Form unter dem Namen *Flex-Ajax Bridge* schon etwas länger.

Natürlich muss klar sein, wenn hier die Rede von Flash, HTML und PDF ist, dass ich diese Technologien immer im Kontext mit Adobe AIR sehe und niemals nur eine isolierte Technologie betrachte. Die hier beschriebenen Features könnte man niemals (oder nur sehr mühsam) ohne die AIR-Laufzeitumgebung erreichen. Doch wie funktioniert *Script bridging* in der Praxis genau? Ich gebe Ihnen ein einfaches Beispiel, damit Sie den Begriff *Script Bridging* besser verstehen.

In einer AIR-Applikation wird ein Flash-Film mithilfe eines HTML-Kontrollelements dargestellt. Dieser Flash-Film beinhaltet eine HTML-Seite mit einem einfachen Formular. Für eine Applikation besteht das Szenario, dass genau dieses Formular mithilfe von AIR ausgefüllt werden soll. Mit AIR und dem internen Scripting haben Sie die Möglichkeit, direkt aus Flash das *DOM* (*Document Object Model*) der HTML-Seite anzusprechen. Mithilfe von Flash können Sie im HTML-Kontrollelement das komplette *DOM* der HTML-Seite abrufen. Sie könnten nicht nur das HTML-Formular aus Flash ohne Umweg ausfüllen und überwachen, sondern auch auf alle anderen Elemente zugreifen, die sich innerhalb der HTML-Seite befinden. Nach dem Laden der HTML-Seite sind Sie in der Lage, mit einer einfachen Methode das Formular aus Flash auszufüllen und abzusenden, obwohl es sich im HTML-Code befindet.

Dank Script Bridging sind HTML und ActionScript in der Lage, direkt miteinander zu kommunizieren.

Dieses Feature, das *Script Bridging* genannt wird, eröffnet Entwicklern ganz neue Perspektiven in der Entwicklung von Anwendungen. Ich möchte zwar nicht sagen, dass die Grenzen zwischen Technologien verschwinden, aber das Feature erlaubt es zumindest, eine *Brücke* zu anderen Technologien zu bauen, um zu kommunizieren. Somit sind viele neue Szenarien in der Entwicklung von Applikationen möglich.

Natürlich habe ich in diesem Beispiel nur einen Weg beschrieben, nämlich wie Sie mit einer HTML-Seite aus Flash kommunizieren können. Glauben Sie mir, das Ganze funktioniert natürlich auch umgekehrt, nämlich aus HTML auf die Flash *API* zuzugreifen. Mit AIR sind Sie als HTML-Entwickler in der Lage, eine HTML-Seite zu erstellen, die die Flash *API* nutzen kann. Sie können also innerhalb Ihrer HTML-Seite ActionScript-Code in seiner reinsten Form schreiben, um so die Funktionalität Ihrer HTML-Seite zu erweitern. Der Vorteil für HTML-Entwickler liegt klar auf der Hand: Wenn Sie ein Feature in einer Applikation mit JavaScript umsetzen möchten, das nicht unterstützt wird oder dessen Entwicklung zu viel Zeit in Anspruch nehmen würde, dann werfen Sie einen Blick in die AIR *API*, ob es dort eine Funktionalität gibt, die die Umsetzung des Features erlaubt und nutzen Sie diese in HTML direkt.

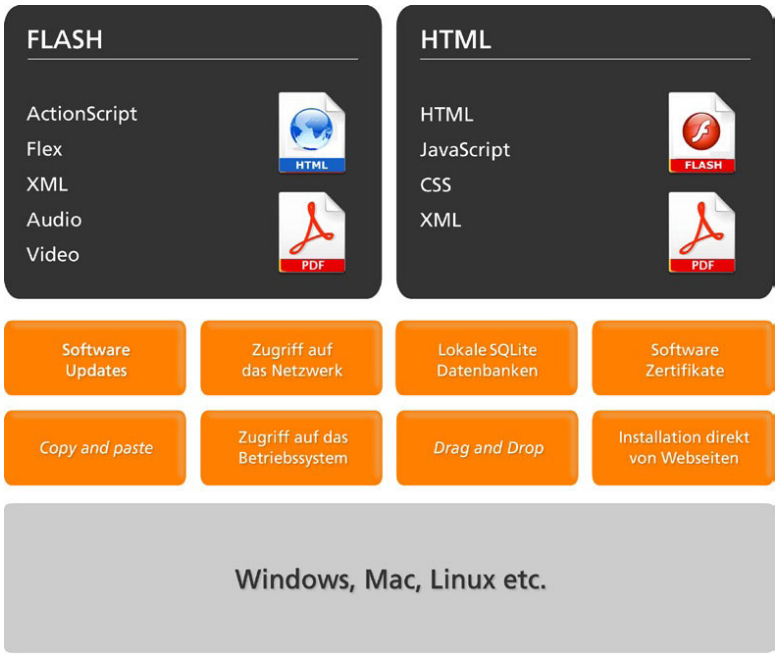
In HTML-Seiten können Sie direkt ActionScript-Code schreiben, um auf die AIR API zuzugreifen. Flash wiederum kann das Document Object Model einer HTML-Seite in ActionScript nutzen.

Wenn Ihnen das nun alles schon etwas zu technisch war oder Sie mit meinen Beispielen nichts anfangen konnten, dann möchte ich Sie bitten, einmal bei den Codebeispie-

len nachzusehen. Dort finden Sie mehr Informationen zu dem Thema und ein paar praktische Beispiele, wie Sie *Script Bridging* in der Praxis nutzen können. Ich finde, diese Art der Kommunikation mit anderen Technologien ist eines der stärksten Features von Adobe AIR.

Sie wissen nun, welche Rolle die Laufzeitumgebung übernimmt, dass HTML und PDF problemlos in AIR integriert werden können und eine Kommunikation über alle Inhalte hinweg möglich ist. Lassen Sie mich die Architektur mit einem Schaubild zusammenfassen. Ich habe mich bemüht, das Schaubild möglichst einfach und übersichtlich zu halten. In Abbildung 3.1 sehen Sie einen Überblick über die Architektur von AIR.

ABBILDUNG 3.1
Die Architektur von AIR



Die AIR API und Features

Plattformunabhängig

Das Schaubild zeigt im oberen Teil welche Technologien in AIR möglich sind und wie diese verschachtelt werden können. Sie sehen außerdem, welche Inhalte (XML, Audio, Video etc.) für die einzelnen Kerntechnologien in AIR unterstützt werden. Auf der mittleren Eben finden Sie eine Auflistung der Möglichkeiten der AIR *API* und Features. Ich möchte später noch auf die einzelnen Features eingehen. Der untere Teil des Schaubilds zeigt die Plattformen für Adobe AIR.

3.5 Die Adobe AIR Features im Überblick

Nachdem ich Ihnen nun einen kleinen Überblick darüber gegeben habe, wie andere Technologien in AIR integriert werden können und sogar in der Lage sind, miteinander zu kommunizieren, möchte ich nun ein wenig auf die Features von AIR eingehen.

Die Funktion einer Applikation, die hier als *Feature* beschrieben wird, basiert auf der *API* von Adobe AIR. Wenn Sie ein konkretes Feature umsetzen wollen, wie z.B. *Drag and Drop*, dann sollten Sie immer zuerst einen Blick in die dokumentierte *API* von Adobe AIR werfen, um herauszufinden, ob das Feature sich überhaupt mit der *API* umsetzen lässt und wie es sich umsetzen lässt.

Ich möchte Ihnen eine kurze Einführung zu den AIR Features geben. Bisher haben Sie nur oberflächlich von AIR Features erfahren, dies soll sich nun ändern.

Sie erhalten mit dieser Übersicht nur einen Überblick über die aus meiner Sicht wichtigsten Features von AIR. Wenn Sie im Internet nach mehr Informationen zu Adobe AIR suchen, dann werden Sie schnell feststellen, dass diese Liste mit Features fast überall auftaucht, wo etwas zu AIR geschrieben steht. Dies hat nichts damit zu tun, dass die Autoren voneinander kopieren. Vielmehr zeigt sich, dass die hier vorgestellten Features die API und die Laufzeitumgebung von AIR am besten beschreiben und daher oft kommuniziert werden.

Die wichtigsten Features von Adobe AIR sind:

- Zugriff auf das Dateisystem
- *Drag and Drop*
- *Copy and Paste*
- Lokale SQLite-Datenbanken nutzen
- HTML- und PDF-Inhalte nutzen
- Unterstützung für *DRM* (*Digital Rights Management*)
- Interaktion mit dem Betriebssystem
- Zugriff auf das Netzwerk

Ab dem achten Kapitel werden Sie alle Features in Form von praktischen Beispielen wiederfinden. Doch vorab ein wenig Theorie. Ich möchte Ihnen die wichtigsten Features von AIR kurz erklären.

Zugriff auf das Dateisystem

Adobe AIR ermöglicht es Ihnen, auf das Dateisystem zuzugreifen. Die wichtigsten Features sind: Dateien und Verzeichnisse lesen, schreiben, verschieben und kopieren. Sie können den Inhalt eines bestimmten Verzeichnisses abrufen oder Dateien und Ordner in den Mülleimer verschieben. Mit AIR sind Sie außerdem in der Lage, zahlreiche Informationen über eine bestimmte Datei abzurufen.

Diese Funktionalität ermöglicht es Ihnen, relevante Dateien für Ihre Applikation auf der Festplatte abzulegen und zu nutzen. Was für Flash- und HTML-Inhalte im Browser innerhalb der *Sandbox* unmöglich ist, macht AIR auf dem Desktop möglich.

Mehr Informationen über die Arbeit mit Dateien und Verzeichnissen erhalten Sie in Kapitel *12 Arbeiten mit Dateien und Verzeichnissen.*

Drag and Drop

Drag and Drop (engl. *Ziehen und Fallenlassen*) beschreibt die Möglichkeit einer Interaktion mit grafischen Benutzeroberflächen durch das Bewegen grafischer Elemente mithilfe des Mauszeigers.

Mithilfe des *Drag and Drop*-Feature in AIR können Sie entweder bestimmte Datentypen aus Ihrer Applikation heraus auf den Desktop (das sogenannte *drag-out*) oder vom Desktop in Ihre Applikation (das sogenannte *drag-in*) ziehen. Unterstützte Datentypen für eine solche Aktion sind: Bitmap, Dateien, HTML-formatierter Text, Text, URLs, Objekte und Verweise auf Objekte.

Ein weiteres tolles Feature ist, dass HTML-Seiten, die in AIR geladen wurden, auch eine *Drag and Drop*-Funktionalität haben. So können beispielsweise Text, Bilder und URLs aus und in das HTML-Objekt *gezogen* werden. Für alle Arten von *Drag and Drop*-Ereignissen (Events) bietet die Adobe AIR API ein umfangreiches *Event-Handling* an.

Dank *Drag and Drop* sind Sie zum Beispiel in der Lage, ein Bilderalbum zu erstellen, und die Bilder hierfür mit dem Mauszeiger von einem beliebigen anderen Ordner in Ihre Applikation zu *ziehen.* AIR kann dieses Bild dann direkt verwenden. Dies ist sowohl auf Basis von HTML als auch mit Flash als Kerntechnologie in AIR möglich.

Mehr Informationen zu *Drag and Drop* sowie zahlreiche praktische Beispiele finden Sie in Kapitel *13 Drag and Drop.*

Copy and Paste

Copy and Paste (engl. *Kopieren und Einfügen*) nutzt den systemeigenen Zwischenspeicher, um Dateien von einem Ort zum anderen zu kopieren. Adobe AIR unterstützt *Copy and Paste* für folgende Dateitypen: Bitmaps, Dateien, Text, URLs, Objekte und Verweise auf Objekte. Mit Adobe AIR sind Sie nicht nur in der Lage, die Zwischenablage des Systems für Ihre *Copy and Paste*-Funktionalitäten zu nutzen, sondern Sie

können auch das Datenformat der aktuellen Datei in der Zwischenablage bestimmen. Genau wie beim *Drag and Drop* gibt es auch für HTML eine Funktion, um *Copy and Paste* nutzen zu können.

Typischerweise funktioniert eine *Copy and Paste*-Anweisung über die Tastenkombinationen Strg+C (Kopieren) und Strg+V (Einfügen) und wird von Adobe AIR unterstützt.

Wie Sie *Copy and Paste* für Ihre eigenen AIR-Applikationen nutzen können, erfahren Sie in Kapitel *14 Copy and Paste*.

Lokale SQLite-Datenbanken nutzen

Wirklich spannend wird es beim Thema *Lokale Datenbanken*. AIR unterstützt lokale *SQL*-Datenbanken und verwendet hierfür das OpenSource-*SQLite*-System. Per *SQL*-Befehlen können Sie mit lokalen *SQLite*-Datenbanken sehr einfach kommunizieren. AIR ist in der Lage, eine lokale *SQLite*-Datenbank zu erzeugen, zu lesen und auch wieder zu löschen. Hierfür verwendet AIR *SQL*-Befehle.

Der Vorteil von *SQLite* ist, dass es keinerlei Konfiguration oder gar eine Serverumgebung benötigt. Eine neue Datenbank kann mit AIR im Bruchteil einer Sekunde angelegt und verwendet werden. Sie bestimmen lediglich den Dateinamen und den Zielpfad, wo die Datenbank auf Ihrer Festplatte gespeichert werden soll. Jede Datenbank wird in einer einzelnen Datei gespeichert.

Die Datenbank lässt sich sehr einfach auf ein anderes System umziehen. So können Sie die Datenbank ganz einfach von einem Rechner auf ein anderes System kopieren und nutzen. Zudem gibt es eine ganze Reihe von Tools, die in der Lage sind, *SQLite* zu verwalten. Eines meiner persönlichen Lieblingstools zur Verwaltung von lokalen *SQLite*-Datenbanken ist das Firefox-Plug-in *SQLite Manager*.

Mehr Informationen zu lokalen SQLite-Datenbanken erhalten Sie in Kapitel *15 Lokale SQLite-Datenbanken verwenden*.

HTML- und PDF-Inhalte nutzen

Dass Sie HTML- und PDF-Inhalte in AIR nutzen können, habe ich auf den letzten Seiten sicherlich schon ausführlich erklärt. Mehr Informationen zur Nutzung von HTML- und PDF-Inhalten erhalten Sie in Kapitel *17 Rich media content*.

DRM (Digital Rights Management)

AIR unterstützt geschützte FLV-Videos und ermöglicht so auch im Offline-Bereich eine umfassende Kontrolle für ausgelieferte audiovisuelle Medien.

Der Flash Media Rights Management Server (kurz: FMRMS) ermöglicht es Medienunternehmen, Bewegtbildinhalte gegen unerwünschte Nutzung zu schützen Der Produzent kann diverse Parameter regeln, die Zugriffsrechte und Nutzungsdauer beeinflussen. So kann z.B. ein Benutzer, der bei einem Videodienst ein Premium-Abo erwirbt, einen FLV Stream ohne Werbung abrufen. Für Benutzer ohne Premium-Abo würde Werbung zwischen den Programmteilen angezeigt werden. Der Adobe Flash Media Rights Management Server lässt sich in bestehende Projekte integrieren und unterstützt Ausgabeplattformen wie den Adobe Media Player und Anwendungen auf Basis von Adobe AIR.

Obwohl die *DRM*-Technologie umstritten ist und viele Anbieter bereits darüber nachdenken, diese wieder abzuschaffen, ermöglicht die Kontrolle von audiovisuellen Medien im On- und Offline-Bereich ganz neue Vertriebsszenarien.

Interaktion mit dem Betriebssystem

Hinter der Überschrift *Interaktion mit dem Betriebssystem* verbergen sich eine ganze Reihe von Features, die der Entwickler in AIR nutzen kann. So können Sie mit AIR beispielsweise feststellen, ob der Benutzer gerade aktiv ist oder nicht. Sie können bestimmte Dateien anhand ihrer Endung auf dem Betriebssystem mit Ihrem Programm assoziieren. Wenn ein Benutzer eine bestimmte Datei anklickt und diese ist mit Ihrem Programm assoziiert, öffnet sich dann Ihr Programm und die Datei kann verwendet werden. Sie können bestimmte Aktionen festlegen, die beim Start bzw. beim Beenden Ihres Programms ausgeführt werden. Sie können eine AIR-Applikation, die auf Ihrem Rechner installiert ist, direkt von einer Webseite starten.

Die Liste der Features, die in die Kategorie *Interaktion mit dem Betriebssystem* fallen, lässt sich noch beliebig fortsetzen.

Praktische Beispiele zu diesem Thema finden Sie in Kapitel *16 Interaktion mit dem Betriebssystem.*

Zugriff auf das Netzwerk

Mit AIR können Sie den Status des Netzwerks feststellen. HTTP- und Socket-Verbindungen lassen sich überwachen. Dies ist zum Beispiel für alle Applikationen wichtig, die Daten über das Netzwerk beziehen.

Mehr Informationen erhalten Sie in Kapitel *18 Arbeiten mit dem Netzwerk.*

Kombinationsmöglichkeiten von Technologien in AIR

Ich hoffe, die Kombinationsmöglichkeiten von Technologien innerhalb der Flash-Architektur sind klar geworden. Lassen Sie mich kurz diese Kombinationsmöglichkeiten noch einmal zusammenfassen.

AIR-Applikationen bestehen aus einer Kombination der nachfolgenden Technologien:

- Nur HTML und JavaScript
- HTML/JavaScript mit Flash-Inhalten
- Nur Flash-(Flex-)Inhalte
- Flash mit HTML-Inhalten
- In alle Kombinationen kann PDF integriert werden.

In diesem Kapitel haben Sie etwas über die Architektur von Adobe AIR erfahren. Sie haben gelernt, aus welchen Komponenten die Architektur besteht, wie PDF- und HTML-Inhalte in AIR eingebunden werden können, was *Script Bridging* bedeutet und ich habe Ihnen einen kurzen Überblick über die Features von AIR gegeben. Zwar wissen Sie jetzt noch nicht wirklich, welche Möglichkeiten hinter der API von Adobe stecken, Sie werden dies anhand der praktischen Beispiele in diesem Buch jedoch schnell erfahren.