Active FoxPro Pages 3.0

Benutzerhandbuch und Referenz

ProLib Software GmbH

Active FoxPro Pages 3.0: Benutzerhandbuch und Referenz
von ProLib Software GmbH

First Ausgabe
Veröffentlicht 2002

Active FoxPro Pages (AFP) verbinden die Programmiersprache Visual FoxPro mit dem Internet. Hierbei fungieren die AFP als Skriptsprache für den Webserver. Die AFP können von passionierten VFP-Programmierer als auch von WebDesignern ohne VFP-Kenntnisse verwendet werden.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Beispiele

Willkommen zu Active FoxPro Pages 3

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

Active FoxPro Pages (AFP) bieten ein System zum Erstellen dynamischer, tabellengestützter Webseiten mit Microsoft Visual FoxPro, XML und HTML.

Bevor Sie mit den Active FoxPro Pages anfangen, sollten Sie sich die Zeit nehmen und das folgende, wichtige Kapitel lesen:

- Was ist neu in Active FoxPro Pages 3?

Suchen eine bestimmte Auskunft? Probieren Sie diese Referenzen:

- Objekt-Referenz
- Methoden-Referenz
- Eigenschaften-Referenz

Zielgruppe

Dieses Benutzerhandbuch vermittelt Ihnen einige Grundlagen und Hintergründe, damit Sie die Technik der AFP lernen, verstehen und zur Anwendung für Ihre dynamischen Webanwendungen nutzen können.

Wir gehen einfach mal davon aus, dass Sie bereits einige Erfahrungen im Umgang mit HTML oder XML besitzen. Falls Sie bisher erst ein paar HTML Dokumente erstellt ahben sollten, so sind Sie auf alle Fälle im Vorteil. Obwohl wir zwischendurch immer mal kurz auf HTML, XML und strukturierten Aufbau eingehen, sollten Sie dieses Buch nicht als einzige Informationsquelle für HTML und XML heranziehen. Das Internet bietet Ihnen in vielerlei Hinsicht genügend Nachschlagemöglichkeiten.

In diesem Handbuch werden zwischendurch verschiedene Tools und Anwendungen erwähnt. Dabei handelt es sich ausschliesslich um Anwendungen für Microsoft Windows, da auch die Active FoxPro Pages nur auf diesem Betriebssystem lauffähig sind.

Organisation dieses Buches

Dieser UserGuide besteht aus drei Abschnitten:

Part I: Einführung - erstellen Sie dynamische, tabellengestützte Webseiten mit der AFP
Part II: Referenz - Komplettübersicht zu allen Objekten und deren Methoden
Part III: Appendixes - behandelt alle weiteren Aspekte im Zusammenhang mit der AFP

Beispiele aus dem Buch

Alle gezeigten und erläuterten Beispiele in diesem UserGuide stehen auf der offiziellen Webseite zur Verfügung. Die aktuellste Version und neuesten Informationen zu den Active FoxPro Pages finden Sie stets im Internet unter der Adresse: *http://www.afpages.de*

Schicken Sie uns Ihre Kommentare

Bitte helfen Sie uns für zukünftige Ausgaben dieser Dokumentation, indem Sie uns Rechtschreibfehler, Unstimmigkeiten, Fehler in den Beispielen oder schlecht geschilderte Abschnitte sowie einfache Tippfehler angeben. Die jeweils aktuelle Version steht Ihnen auch online auf *http://www.afpages.de* zur Verfügung. Zum Einsenden von Fehlerreports und Kommentare schreiben Sie uns bitte unter *afp@prolib.de* (mailto:afp@prolib.de).

Vielen herzlichen Dank!

I. Einführung

Cogito, ergo sum.
Ich denke, also bin ich.

René Descartes (Meditationes)

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

Part I bietet eine Einführung in die dynamische, datenbasierende Webseitenentwicklung mittels der AFP.

Zuerst möchten wir Ihnen in Kapitel 1 aufzeigen, welche neuen Features und Verbesserungen die Version 3.0 erfahren hat.

Kapitel 1. Was ist neu?

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

Willkommen zur Version 3.0 der Active FoxPro Pages

Diese Ausgabe ist eine komplett neugeschriebene Version. Und falls ich komplett neu sagte, dann ist es das so gemeint. Fast der komplette Quellcode der AFP wurde überarbeitet und neu strukturiert, um höhere Stabilität bei wachsender Performance für Ihre Webanwendungen zu bieten.

Mit dieser Version der AFP betreten wir gemeinsam ein neues Konzept der dynamischen Webseitenerstellung. Vorhergehende Versionen waren stets datei-orientiert, was bedeutet, dass zu einer bestimmten URL-Anfrage eine physikalische Datei zurückgeschickt wurde. Jedoch diese Version bietet Ihnen zukünftig anwendungs-zentrierte Verarbeitung. Sie erstellen somit für Ihre .AFP Seiten eine gemeinsame Web Anwendung, welche die Rahmenbedingungen als Vorlage gibt. Zusätzlich werden die Metadaten hierfür in XML Dateien ausgelagert, damit eine bessere Kontrolle über das verhalten der AFP-Dokumente ermöglicht wird.

Hier die Hauptaspekte und Vorteile des neuen Konzeptes:

- Bessere Performance
- Höhere Stabilität
- Erweiterte Flexibilität
- Mehr Sicherheit für Ihre AFP-Dokumente

Gerade weil das neue Konzept wesentlich mehr Flexibilität als zuvor bietet, ist die aktuelle Version nahezu abwärtskompatibel zu ihren Vorgängern. Wir werden uns die genauen Zusammenhänge später anschauen.

Lassen Sie uns ein wenig genauer in die Details gehen...

1.1. Featureliste

AFP 3.0 wurde vollständig überarbeitet. Würde ich hier alle Neuerungen aufführen, wäre dieses Announcement so lang, daß sie keiner mehr lesen würde. :-) Daher nur einige der Highlights:

- Geschwindigkeitsoptimierung: AFP 3 ist deutlich schneller als AFP 2.4 und kann auf einem einzelnen Rechner mehrere Millionen Hits am Tag verarbeiten. Das gesamte Design der AFP 3 ist auf maximale Performance ausgelegt.
- Keine COM Schnittstelle mehr. Die ISAPI Erweiterung kommuniziert direkt mit der eigentlichen AFP Engine. Dadurch ist keine Konfiguration mit DCOMCNFG mehr erforderlich, kann die gesamte AFP

jederzeit gestoppt werden, ohne den WebServer stoppen zu müssen und gibt es keine Konvertierungsprobleme mit COM.

- Echtes Multithreading: AFP 3 ist eine echte multithreaded VFP Applikation. Dadurch ist der Speicherplatzverbrauch deutlich niedriger als in AFP 2.4.
- Dateiupload
- Verbesserte Sicherheit: AFP 3 kann auch mit Gastrechten laufen. Dateien müssen nicht mehr im wwwroot Verzeichnis sein, sind dadurch vor Zugriffen geschützt.
- ControlCenter: Das ControlCenter bietet einen Überblick über alle laufenden AFP Threads inklusive der gerade bearbeiteten Seite und der Ausführungszeit. Threads können jederzeit entladen oder geladen werden.
- Sessionverwaltung: Die SessionID ist nun immer eindeutig. Die Deklaration von Variablen ist deutlich einfacher. Die Sessionverwaltung ist komplett überarbeitet und erlaubt auch die Erstellung sessionspezifischer Dateien, etwa Abfrageergebnissen oder generierten Berichten. Läuft die Session ab, werden diese Dateien automatisch gelöscht.
- Ereignisse: In jeder AFP Seite und AFP Applikation stehen zahlreiche Ereignisse zu Verfügung, in denen Code beim Laden der Applikation, vor dem Aufruf einer Seite oder dannach ausgeführt werden kann.
- Errorhandling: Sowohl auf Applikationsebene als auch pro Seite kann ein eigener Errorhandler hinterlegt werden. Damit haben Sie die Möglichkeit auf Fehler zu reagieren. Die Möglichkeit eines RETRY wird unterstützt, so daß Sie etwa bei Problemen beim Sperren eines Satzes dies mehrfach wiederholen können. Auch eigene Fehlerseiten können so dynamisch in Abhängigkeit vom Fehler generiert werden.
- Trennung von AFP Applikation. Jede Applikation erhält ihre eigene Datasession und eigenes Set von Variablen.
- Hintergrundtasks erledigen das Aufräumen nicht benötigter Dateien.
- Mit PlugIns kann die AFP nahezu beliebig erweitert werden. Durch die dokumentierte PlugIn Schnittstelle kann jeder eigene PlugIns entwicklen und anderen AFP Entwicklern zur Verfügung stellen.
- Über eine spezielle Syntax können Sie Code bei der Kompilierung ausführen und so die zu kompilierende AFP Datei noch dynamisch beeinflussen.
- Zur Kompilierungszeit können beliebige Dateien eingebunden werden, die ebenfalls wieder AFP Code oder weitere Includes enthalten dürfen. Damit können gemeinsam genutzte Elemente (Kopfzeilen, Menüs, etc.) einfach verwaltet werden. Selbstverständlich wird eine Seite auch dann neu kompiliert, wenn sich lediglich die Includedatei geändert hat.
- Ein Kompatibilitätslayer ermöglicht die Ausführung von AFP 2.4 Applikationen. Das Kompatibilitätslayer stellt das FOX Objekt zur Verfügung.
- Das neue Objektmodell ist an den Quasi-Standard ASP angelehnt. Das verringert die Einarbeitungszeit für neue Entwickler und ermöglicht es außerdem, relativ einfach ASP Beispiele in die AFP zu übernehmen.
- Debuggen ist einfacher geworden. Im Debugmodus läuft die AFP auf dem Web Server direkt in der VFP 7 Entwicklungsumgebung. Dadurch können Sie sämtliche Debugbefehle, wie SET STEP ON, ASSERT, DEBUGOUT, etc. direkt in den AFP Seiten verwenden. Da die AFP Seiten aus dem WebServer heraus aufgerufen werden, können Sie so die Seiten im Applikationskontext testen. Rufen Sie im Browser eine Seite auf und sie erscheint im Debugger.

1.2. Voraussetzungen

Die Systemvoraussetzungen für die Active FoxPro Pages 3.0 sind im Vregleich zur 2.4 ein wenig angestiegen:

- Windows NT/2000/XP
- Webserver mit ISAPI-Schnittstelle (IIS 4 or higher, Apache 2.0.x or higher, WebWeaver, WebSite, etc.)
- Visual FoxPro 7 wird empfohlen. Debugging oder das Entwickeln eigener Plugins setzen VFP 7 voraus.

1.3. Lizenzierungsmodell

Mit der Version 3.0 erfährt auch das bisherige Lizenzierungsmodell der AFP umfangreiche Änderungen. Vielleicht sind die Neuerungen ein wenig hart im Vergleich zum vorherigen Modell. Aber tatsächlich bietet sich nun, einerseits eine bessere und flexiblere Möglichkeit für Sie als Webentwickler und Benutzer von AFP-Anwendungen und andererseits eine gute Chance für uns die Entwicklung der AFP besser zu forcieren und voranzutreiben, um Ihnen neue Features und Tricks in künftigen Versionen bieten zu können.

Neu ist außerdem der Aspekt, dass die Active FoxPro Pages 3.0 in zwei Versionen angeboten werden - Vollversion und Update - was im Prinzip das gleiche Produkt darstellt und sich tatsächlich nur im Preis unterscheidet. Falls Sie eine Liznez der AFP 2.4 besitzen, so sind Sie updateberechtigt und profitieren vom günstigeren Preis.

Weiterer Änderungspunkt ist das sogenannte Hostmanagement der Active FoxPro Pages. Standardmässig unterstützt die AFP drei Domains. Eine Domain setzt sich hierbei aus der generischen Top Lovel Domain (gTLD) wie etwa ’.de’ oder ’.com’ und der benutzereigenen Second Level Doamin (SLD) wie in ’active-foxpro-pages’ oder ’afpages’ zusammen. Erst das zusammengesetzte Resultat ist für das Hostmanagement der AFP entscheidend.

> **Warnung**
>
> Der Hostname Ihres Webservers könnte sich als Engpaß bzgl. der konkurrierenden Lizenzen herausstellen. Sie sollte dabei die Domains für ’localhost’ oder des Hostnamen Ihres Systems, bspw. ’afpserver’ vermeiden. Denn diese Namen werden als SLD gewertet und reduzieren somit auch die Anzahl der tatsächlichen Webverzeichnisse.

Jegliche Subdomains - wie www.afpages.de oder forum.apfages.de - sind gleichbedeutend mit afpages.de und fallen dadurch uneingeschränkt aus der Zählung. Die AFP ermittelt die zulässigen

Domains aus den durchgeführten Anfragen auf AFP-basierende Webseiten und sendet bei Überschreitung des Limits eine HTTP 500 Fehlermeldung als Antwort an den Browser zurück.

Dazu ein Beispiel. Falls für die Domains eine Anfrage erfolgt ist

- http://localhost/index.afp
- http://afpserver/index.afp
- http://www.prolib.de/

wird immer eine gültige Antwort kommen. Kein Problem bis jetzt.

Jede weitere Doamin jedoch verabschiedet sich mit einem HTTP 500 Fehler. Die folgende Anfrage werden somit nicht funktionieren:

- http://www.afpages.de/index.afp

Wie Sie sehen, handelt es sich hierbei um die vierte Kombination aus gTLD und SLD und wir erhalten die erwartete Fehlermeldung. Selbst wenn afpages.de und prolib.de auf der gleichen Maschine laufen sollten so sind es doch unterschiedliche Domänen.

Die folgende Subdomain arbeitet hingegen sorgenfrei:

- http://shop.prolib.de/

Da es sich hierbei nur um eine Subdomain handelt, ist dies gleichgesetzt mit der bereits registrierten Domain ’prolib.de’.

Um die maximale Anzahl an AFP-basierten Domains / Hostnamen erhöhen zu können, gibt es künftig sogenannte Hostlizenzen. Somit brauchen Sie für jede eigenständige zusätzliche Webseite eine weitere Hostlizenz.

Falls Sie als Webhoster für andere Firmen oder Personen AFP Sites anbieten, müssen Sie jeweils eine Hostlizenz für den Kunden erwerben. Für genauere Details und Preisinformationen setzen Sie sich bitte mit uns in Verbindung (sales@prolib.de).

1.4. Erweiterbarkeit

Waren Sie jeweils in der Situation, dass Ihnen ein ganz spezielles Feature der AFP gefehlt hat? Und das Sie am besten seit gestern bräuchten?

Falls Sie wissen, wovon ich spreche, dann seien Sie erfreut über die Erweiterbarkeit der neuen AFP über Plugins. Mit dieser Schnittstelle ist es möglich, dass Sie selbst entscheiden, welchen Funktionsumfang die AFP haben darf oder muss. Sie können die Benutzung der Plugins kontrollieren und nun auch eigene Erweiterungen selbst programmieren.

Kapitel 2. Los geht’s

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

In diesem Kapitel werden wir uns erste Einblicke in die Funktionsweise der Active FoxPro Pages gönnen und einfache AFP-Dokumente erstellen.

In den dargestellten Konfigurationen verwenden wir C:\inetpub\wwwroot als Basisverzeichnis für den Webserver. Sobald Sie diesen im Browser aufrufen, wird versucht, die Standard-HTML-Seite darzustellen. Die verschiedenen Webserver verwenden dafür unterschiedliche Standardseiten und Suchindices.

Im IIS lautet die Seite `default.htm`, beim Apache `index.htm`. Sie können diese Angaben jedoch frei wählen und jederzeit ändern. Für den Einsatz der AFP bietet es sich an, die Standardseite auf `default.afp` und/oder `index.afp` einzustellen.

Um Ihren Webserver im Browser zu erreichen, geben Sie den Domainnamen des Computers in der Adressleiste ein. Falls Sie keine eigene Domain für den Computer haben sollten, so verwenden Sie entweder den Rechnernamen oder die IP-Adresse als Zieladresse. Sollten Sie auf dem Rechner direkt arbeiten, können Sie auch *http://localhost/* verwenden.

Unser erstes AFP-Beispiel speichern wir also in das Verzeichnis C:\inetpub\wwwroot. Zur Erstellung ist irgendein Texteditor wie etwa Notepad vollkommen ausreichend. Wir verwenden einfach Notepad, da es normalerweise auf jedem Windowssystem zur Verfügung steht.

2.1. Hello World

Wie in allen Programmiersprachen üblich, beginnen wir klassisch mit dem ’Hello World’ Beispiel:

Beispiel 2-1. Hello World mit AFP - hello.afp

```
<%
? "Hello World!"
%>
```

Sie sehen innerhalb des Quellcodes ganz regulären HTML-Code. Die Verwendung der AFP wird ähnlich wie bei anderen Skriptsprachen (ASP, PHP und andere) durch ein bestimmtes Trennzeichen - hier das Prozentzeichen (%) - eingeleitet und beendet. Dies dient zur strukturellen und syntaktischen Trennung von HTML-Tags und Visual FoxPro Code.

Die AFP besitzt keine Einschränkung beim Vermischen der unterschiedlichen Codeblöcke mit AFPScript oder HTML . Es ist genauso möglich HTML innerhalb eines Skriptblockes auszugeben. Die späteren Beispielen werden dies verdeutlichen.

> **Tipp:** Jedes der gezeigten Beispiele wird mit der AFP selbst auf CD ausgeliefert und steht Ihnen auch auf der Onlineseite der Active FoxPro Pages *http://www.afpages.de* zur Verfügung.

Okay, weiter mit dem nächsten Beispiel.

2.2. Wie spät ist es?

Dieses Beipiel hat ein wenig mehr Dynamik als das vorherige. Es zeigt lediglich das aktuelle Datum samt Uhrzeit auf Ihrem Webserver an. Geben die folgenden Zeilen ein und speichern Sie die Datei unter dem Namen `datetime.afp` im Basisverzeichnis des Webservers.

Beispiel 2-2. Your First Active-FoxPro-Page - datetime.afp

```
<html>
<head>
<title>Test</title>
</head>
<body>

<%
?datetime()
%>

</body>
</html>
```

Jetzt rufen Sie diese Seite im Browser auf: *http://localhost/datetime.afp*

Sie sollten nun das aktuelle Datum und die aktuelle Uhrzeit angezeigt bekommen.

Falls ein Fehler auftreten sollte, schauen Sie bitte in die Ereignisanzeige für Anwendungen, dort ist das Problem ausführlich dokumentiert. Erläuterungen und ausführliche Informationen zum Debuggen von AFP-Seiten finden Sie im entsprechenden Kapitel des Benutzerhandbuches. Erfahrene AFP-Programmierer nennen diese Ausgabe "Red Screen of Death" (Ähnlichkeiten mit dem Betriebssystem sind rein zufällig und nicht beabsichtigt).

2.3. Datenzugriff

In diesem Beispiel wollen wir alle Kundennamen der Customer.dbf aus der Tasmanischen Handelsgesellschaft, die mit Visual FoxPro als Beispielanwendung geliefert wird, anschauen.

Beispiel 2-3. Datenzugriff - allcust.afp

```
<html>
<head>
<title>All customers of table CUSTOMER</title>
</head>
<body>
<h1 color="#0000FF">
<%?"All customers of table CUSTOMER"%>
</h1>
<hr>
<table border="1" width="95%">
<tr>
<td width="20%" bgcolor="#008000">cust_id</td>
<td width="40%" bgcolor="#008000">company</td>
<td width="40%" bgcolor="#008000">contact</td>
</tr>
<%
if not used("customer")
    use JustPath(FILE.GetLocation()) + "\data\customer in 0
endif

select customer
scan
%>
<tr>
<td><%?customer.cust_id%></td>
<td><%?customer.company%></td>
<td><%?customer.contact%></td>
</tr>
<%endscan%>
</table>
</body>
</html>
```

2.4. Selektiver Datenzugriff

In unserem nächsten Beispiel möchten wir die Zahl der angezeigten Kunden durch Angabe des Anfangsbuchstaben einschränken.

Dazu erweitern wir das vorige Beipiel um eine Eingabemöglichkeit für den Suchbegriff und schränken unsere Abfrage auf diesen Suchbegriff ein:

Beispiel 2-4. Selektiver Datenzugriff - somecust.afp

```
<html>
<head>
<title>Some customers of table CUSTOMER</title>
</head>
<body>
<h1 color="#0000FF">
<%?"Some customers of table CUSTOMER"%>
</h1>
<hr>
<form method="POST" action="somecust.afp">
<p>Please specify the starting letter:
<input type="text" name="searchkey" size="5">
<input type="submit" value="Query" name="B1">
</p>
</form>

<table border="1" width="95%">
<tr>
<td width="20%" bgcolor="#808080">cust_id</td>
<td width="40%" bgcolor="#808080">company</td>
<td width="40%" bgcolor="#808080">contact</td>
</tr>
<%
if not used("customer")
   use JustPath(FILE.GetLocation()) + "\data\customer in 0
endif

select customer
lcsearchkey = upper(Request.Form("searchkey"))

if empty(lcsearchkey)
   lcsearchkey=" "
endif

scan for cust_id = lcsearchkey
%>
<tr>
<td><%?customer.cust_id%></td>
<td><%?customer.company%></td>
<td><%?customer.contact%></td>
</tr>
<%endscan%>
</table>
</body>
</html>
```

Beim ersten Aufruf der Seite werden keine Kundendaten ausgegeben, da kein Suchbegriff angegeben ist. Sobald jedoch ein Suchbegriff eingegeben wird, erfolgt die Abfrage auf die Kundendaten und liefert Ihnen das gewollte Ergebnis zurück.

Damit Visual Foxpro bekannt ist, welche Daten durch die Formulareingabe ermittelt werden sollen, verwenden wir in unserem Beispiel die Systemfunktion fox.GetFormVar("suchkey"). Die Funktion gibt uns den Wert des Eingabefeldes zurück. Der Ausdruck "suchkey" entspricht hier dem Namen, welchen wir innerhalb des <form>-Tags für das Eingabefeld <input ... name="suchkey"> vergeben haben.

Kapitel 3. Erstellung dynamischer Webseiten

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

Dieses Kapitel befasst sich mit der Verbindung von statischen Darstellungen in HTML/XML mit der Logik und Magie der AFP Wir werden zeigen, wie man dynamische, tabellengestützte Webanwendungen erstellt.

3.1. Ereignisablauf

Die AFP 3.0 hat keine Ereignisse im eigentlichen Sinne für die Erstellung der Rückantwort, sondern eine Art "Hook-Mechanismus", welcher es gestattet, in den Ablauf einzuwirken. Dazu erstmal eine alphabetische Liste aller verfügbaren Ereignisse für Ihre Webanwendungen.

- AppCall
- Error
- Init
- InitVariables
- Load
- LoadVariablesBefore
- PageAfter
- PageBefore
- PageCall
- PageCallAfter
- PageCallBefore
- Recompile
- SaveVariablesAfter

Die Verfügbarkeit einiger Ereignisse hängt von der Art des AFP-Dokuments - freie Seite oder Anwendung - ab. Sehr ausführliche Informationen zu den einzelnen Ereignissen finden Sie im entsprechenden Referenzteil.

Für die Entwicklung von Webanwendungen sind die Ereignisse standardmäßig leer. Somit liegt es beim Entwickler, ob er darauf zurückgreifen möchte oder nicht. Werfen Sie mal einen Blick auf die kompilierten .PRG Dateien im %cache% Verzeichnis, um einen ersten Eindruck davon zu bekommen.

Wie kann ich das aber nutzen? - Nehmen wir mal eine Webanwendungen mit einigen AFP-Dokumenten, die bei jeder Abfrage die gleichen Tabellen benötigen. Statt wie vielleicht zuerst versucht, den Code in die einzelnen Dokumente zu platzieren, nehmen wir ein Ereignis zur Hilfe, dass bei jeder Anfrage eh ausgeführt wird. Dies erspart Ihnen viel Zeit und Aufwand beim Entwickeln.

Zum besseren Verständnis, hier ein Beispiel zum Init-Ereignis, welches immer beim allerersten Start der Webapplikation ausgelöst wird. Sie finden den Code im Upload-Beispiel, welches mit der AFP ausgeliefert wird.

Beispiel 3-1. Verwendung von Ereignissen für die Webapplikation - upload.afpa.code

```
Procedure Event_Init
    ************************************************************
    * Open all tables
    ************************************************************
    USE AddBS(JustPath(FILE.GetLocation())) + "upload.dbf"
EndProc
```

Schauen wir uns die Struktur des Codebeispiels einmal genauer an. Zuerst einmal sind Ereignismethoden nur in den `.code` Dateien gestattet - also .afp.code oder .afpa.code. Da wir `upload.afpa.code` verwenden, bedeutet dies wiederum, dass es sich um einen Bestandteil einer Webanwendung (`.afpa`) namens `upload` handelt.

Als nächstes erstellen wir das eigentliche Event mittels der Procedure..EndProc Anweisung von FoxPro. Dabei ist es stets notwendig, dass Sie das Ereignis mit dem EndProc abschliessen. Der hier geschriebene Code wird eins zu eins als ’Baustein’ komplett in den Kompilierungsprozess übernommen. Falls Sie also das EndProc vergessen sollten, beeinflussen Sie das gewünschte Resultat negativ und erzeugen dadurch ungewollte Fehler.

Innerhalb der Ereignismethoden dürfen Sie nur reine VFP-Anweisungen und Methoden verwenden. AFPScript - also mittels < und &62; eingeschlossene Segmente - sind für die Ereignisprogrammierung nicht gestattet. Eine weitere Einschränkung ist die Vermeidung der Anweisung ’RETURN’, die durch den Kompilierungsprozess als ein ’EXIT’ interpretiert wird und dadurch das spätere Parsen eines AFP-Dokuments unterbricht und meistens nur eine leere Seite an den Browser zurückschicken.

Ereignisse können mehrfach in der tatsächlichen Ergebnisseite (siehe PRGs im %cache%) auftreten. Dies wird durch verschiedene Faktoren beeinflusst - der AFP interne Kompilierungsverlauf, ob das Ereignis sowohl in der einzelnen Seite und in der Webanwendung definiert ist, ob es sich überhaupt um ein freies AFP-Dokument oder um ein anwendungsgebundenes handelt und weitere mehr. Zusätzlich sollten Sie bedenken, dass es keinen linearen Zusamenhang mit der Definition des Ereignis in der `.code` und der tatsächlichen Position in der kompilierten Fassung. Das entscheidet allein der AFP-Compiler. Falls Sie sich dafür interessieren, aktivieren Sie die Einstellung Build Protokoll und schauen Sie sich die Protokolle einfach an.

Kapitel 4. Hilfe

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

In diesem Kapitel schauen wir die verschiedenen Aspekte und Zusammenhänge der Installation der AFP an, und werfen einen ersten Blick in den Debugmodus für Ihre Webapplikationen.

4.1. Installation

Die Installation der AFP ist sehr einfach, fast zu einfach. Führen Sie die Datei `AFP3Setup.exe` aus und folgen Sie den Anweisungen.

Okay, kleiner Spass am Rande. Natürlich schauen wir uns die Installationsroutine ein wenig genauer an und ich versuche Ihnen einen kleinen Einblick hinter die Kulissen zu vermitteln. Wie bereits erwähnt, beginnen wir die Installation mit dem Ausführen der Datei `AFP3Setup.exe`. Sie benötigen dafür administrative Rechte auf Ihrem System. Falls Sie sich unsicher sind, ob Sie die haben, keine Bange, das Setup prüft diese ab und 'beschwert' sich schlimmstenfalls mit einem Abbruch.

Zusätzlich werden weitere Systemvoraussetzungen durchgeführt, bevor das eigentliche Setup beginnt. Wenn alle Bedingungen erfüllt sind, präsentieren sich Ihnen die Lizenzbedingungen. Schauen Sie sich an, auf was Sie sich einlassen möchten.

Abbildung 4-1. Durch Akzeptanz der Lizenzbestimmungen installieren Sie die AFP

Wenn Sie die Lizenzbestimmungen annehmen, sehen Sie den 'Hauptdialog' des Setups als Nächstes. Hier haben Sie die Möglichkeit sich über den aktuellen AFP-Status in der LiseMich-Datei zu informieren, bestimmen welchen Installationstyp - Empfohlen, Vollständig oder Benutzerdefiniert - Sie in welches Zielverzeichnis installieren möchten.

Was bedeutet das nun wieder? Tja, 'Empfohlen' steht für einen Standardinstallationstyp, der Ihnen die reine AFP 3.0 ohne weiteren 'Schnickschnack' - weder Abwärtskompatibität noch neue Zusatzmodule - installiert. Die 'Vollständige' Installation hingegen bietet genau das Gegenteil, nämlich alles was das Setup bietet, wird auf Ihre Maschine installiert und auch gleich entsprechende vorkonfiguriert. Für die Anwender, die es genauer wissen möchten, bieten wir die 'Benutzerdefinierte' Installation an. Hier können Sie jeden einzelnen Schritt und dessen Optionen selbst beeinflussen. Klar, dass Sie mehr Dialoge zu beantworten und zu bestätigen haben. Dennoch liegt die Wahl bei Ihnen allein.

Abbildung 4-2. Geben Sie das Zielverzeichnis an

Nach dem Dialogdschungel erfolgt das eigentliche Kopieren, Installieren und Registrieren der AFP-Komponenten in das Betriebssystem. Gegen Ende dieses Schrittes wird auch der AFP3 Dienst installiert sowie gestartet und die Einträge im Startmenü werden erstellt. Zu diesem Zeitpunkt läuft die AFP bereits.

Doch wir sind noch nicht am Ende angelangt. Das Setup versucht zusätzlich die notwendigen Dateierweiterungen korrekt im vorhandenen Webserver einzurichten. Falls Sie benutzerdefiniert installieren, können Sie die Optionen selbst wählen. Zum gegenwärtigen Zeitpunkt können nur die Microsoft Internet Information Services und VisNetic WebSite automatisch konfiguriert werden. Apache leider noch nicht.

Noch mehr Dialoge? Herrje, was eine Tortour, nicht wahr? Okay, aber nun haben wir es geschafft und der Finish-Dialog erscheint. Hier haben Sie noch zusätzliche Möglichkeiten bevor Sie das Setup abschliessen. Es ist sicherlich eine gute Entscheidung mal auf der offiziellen Webseite vorbeizuschauen - vielleicht gibt’s eine neuere Version oder Fehlerbehebungen?

Abbildung 4-3. Endlich geschafft - Benötigen Sie weitere Informationen?

Anmerkung: Falls Sie noch mit Windows NT 4 arbeiten, müssen Sie vor der Benutzung des AFP 3.0 ControlCenter einen Neustart Ihres Systems durchführen. Erst danach steht Ihnen die volle Funktionalität zur Verfügung. Obwohl die Kernbereiche der AFP bereits gestartet sind und laufen und der Webserver korrekt konfiguriert sein dürfte, können Sie noch nicht die volle Power nutzen...

4.2. Interaktion mit dem Webserver

Nun die AFP ist korrekt installiert, doch wie stellen wir die Interaktion mit dem Webserver her? Auch das ist relativ einfach, ehrlich. Wir werden in diesem Abschnitt exemplarisch die Integration in die Microsoft Internet Information Services (IIS) realisieren. Wie bereits erwähnt funktioniert die AFP mit allen ISAPI-kompatiblen Webservern, darunter fallen auch Apache 2.0.x oder höher, VisNetic WebSite und sicherlich noch weitere.

Falls Sie eine empfohlene oder vollständige Installation durchgeführt haben sollten, dürfte die AFP bereits korrekt mit dem ermittelten Webserver interagieren. Als einer der letzten Schritte der benutzerdefinierten Installation sahen Sie einen Dialog, der Ihnen die Entscheidung stellte, ob Sie das automatisch wünschen oder selbst durchführen möchten.

Nun wollen wir uns dennoch die Details dazu ansehen bzw. überprüfen. Das Setup modifiziert die MetaBase des IIS in der Art, dass die Konfiguration des WWW-Dienstes für die Verwendung der AFP verändert wird nicht die Konfiguration der einzelnen virtuellen Webseiten. Dies wird jedoch nur erfolgreich durchgeführt, wenn keine bestehende Konfiguration vorhanden ist. Das ist insbesondere für Upgrades von Version 2.4 auf 3.0 interessant. Nun zurück zu den Details. Die Konfiguration sollte so aussehen, dass die Dateierweiterung '.AFP' auf die Datei `AFP3.DLL` (inkl. vollem Pfad) eingestellt ist.

Zur Konfiguration einzelner (virtueller) Webverzeichnisse öffnen Sie den Eigenschaftendialog, auf dem neuen Dialog wählen Sie Basisverzeichnis (siehe Abbildung 4-4). Öffnen Sie die Konfiguration, um die bestehenden Scripterweiterungen (Abbildung 4-5) zu bearbeiten. Wahrscheinlich dürften bereits einige Dateierweiterungen eingerichtet sein, fügen Sie die AFP, falls nicht schon vorhanden, zu Ihrer Liste hinzu. Danach 'weiß' der IIS, wie er mit AFP-Seiten umzugehen hat und übergibt die Anfrage an die `AFP3.DLL`.

Abbildung 4-4. Configure IIS - Home Directory dialog

Abbildung 4-5. Configure IIS - Configuration dialog

Das war's. Verlassen Sie die offenen Dialoge über die OK-Buttons und starten Sie das AFP-aktivierte Webverzeichnis neu. Das Neustarten ist eigentlich nicht notwendig, es schadet aber auch nicht und wir würden es durchführen. Sicher ist sicher.

Falls Sie den Apache HTTP Server ab Version 2.0.x verwenden, so wurde vom AFP-Setup eine Datei Namens `httpd.afp.conf` in das 'conf'-Verzeichnis rüberkopiert. Bitte schauen Sie sich die Unterschiede zur Ihrer bestehenden an, dann sollte es ein leichtes sein, dem Apachen die AFP nahe zu legen. Ja, das ist sicherlich die feine Art und wir arbeiten auch an einer Verbesserung, jedoch ist ein Eingriff in die komplexe Konfiguration mit ihren 'tausenden' Direktiven nicht ganz einfach und deshalb, unterlassen wir gegenwärtig die automatische Konfiguration. Wir hoffen, dass dies auch Ihren Vorstellungen entspricht.

4.3. Konfigurationsdatei der AFP

Die Active FoxPro Pages können auf zwei sehr unterschiedliche Arten konfiguriert werden - einfach und eben schwer. Wenn Sie sich jetzt für den einfachen entscheiden, starten Sie dazu das AFP 3.0 ControlCenter und lesen in Abschnitt 5.4 weiter. Ansonsten bleiben Sie hier und kämpfen sich durch die nächsten Seiten.

Dieser Abschnitt behandelt jede verfügbare Option der `afp.config` - der zentralen Konfigurationsdatei der AFP 3.0. Die interne Struktur der Datei ist einfaches XML und relativ selbsterklärend, so dass

Änderungen auch direkt durchgeführt werden können. Dennoch empfehlen wir, dass Sie das ControlCenter verwenden.

Zusätzlich zur angepassten Konfiguration installiert die AFP noch eine sogenannte Standarddatei `afp.default.config`, welche Sie verwenden können, falls Ihre existierende nicht mehr lauffähig sein sollte. Sozusagen als Backup. Das ControlCenter bietet Ihnen die Erstellung einer neuen, leeren Konfiguration auf Knopfdruck.

Beispiel 4-1. Standardkonfiguration - afp.default.config

```
<?xml version="1.0" encoding="Windows-1252"?>
<config>
  <afp>
    <path root="" common="%root%" cache="%common%\cache" log="%root%\log"
          session="%common%\Session" bin="%root%"/>
    <server memory="1256000" valid-extensions="afp,htm,html,asp" threads="4"
            isolated="YES" error=""/>
    <session http-sessionid="YES" http-sessionname="afpcookie" afp-sessionid="YES"
             afp-sessionname="afpcookie" file="%common%\Session.dbf"/>
    <build compile-log="NO"/>
    <debug afpengine="NO" cache-fxp="NO" show-error="NO"/>
    <plugins>
    </plugins>
    <virtual name="/" secure="%common%\secure" securelevel="1">
    </virtual>
  </afp>
</config>
```

Klar erkennbar, lautet die Basis der XML-Datei <config>, welche wiederum das Element <afp> beinhaltet. Erst dann stehen die Einstellungen samt ihrer Werte wie im ControlCenter zur Verfügung. Für die Tags, die Attribute und auch teilweise die eigentlichen Werte ist auf alle Fälle die Groß/Kleinschreibung zu beachten - Bitte denken Sie daran, wenn Sie die Datei selbst editieren.

4.3.1. <path>

Hier werden alle Pfadangaben der AFP hinterlegt, vergleichbar mit der Seite Pfad (Abbildung 5-4). Zur Auswahl stehen %root%, %common%, %cache%, %log% und %session%, die Verzeichnisse können sowohl absolut als auch relativ zueinander angegeben werden.

Die Einstelleung %root% hält die Information, wo die AFP 3.0 Engine läuft. Falls der Wert leer, wird das Verzeichnis, indem sich die `afp.config` befindet, genommen. Standardmässig ist dies der Fall.

Falls Sie die AFP als Teil einer Clusterlösung einsetzen, so verwenden Sie einen gemeinsamen - %common% - Ordner für alle angebundenen Webserver. Dies dient dazu, dass systemweite Dateien

wiedergefunden werden. Da im Normalfall kein Cluster vorliegt, entspricht %common% dem Wert von %root%.

Bei jedem ersten Zugriff auf ein AFP-Dokument erfolgt der Kompilierungsprozess durch die AFP. Das Ergebnis wird unterhalb des Cacheverzeichnisses %cache% gespeichert.

Zusätzlich zum vorhandenen Ereignisprotokoll des Betriebssystems kann die AFP auch in einen definierten Ordner %log% ihre Resulte ausgeben. Dies ist vorallem für die zusätzliche Fehlerlogging als auch für das eventuell aktivierte Buildlog relevant. Eine durchaus sinnvolle Möglichkeit bestünde darin, die Ausgaben in das Log-Verzeichnis des Webservers zu schreiben.

Mittels des Session-Wertes sagen Sie der AFP, wo die temp. Sessionhandles abgelegt werden sollen. Diese werden natürlich nach Ablauf der Session wieder entfernt. Meistens handlet es sich um ein Verzeichnis unterhalb des %common%.

4.3.2. <server>

Das nächste Tag: <server> beschäftigt sich mit den Optionen für die AFP Core Engine. Hier erfolgen die Angabe zur max. Speichernutzung, zulässige Dokumenterweiterungen, die von der APF bearbeitet werden dürfen, die Anzahl und die Art der initialen Threads beim Start der AFP und letztendlich die Pfadangabe, ob und wo ein alternatives Fehlerausgabedokument auffindbar ist. Siehe dazu Abbildung 5-5.

Der Wert bzgl. der Speichernutzung der AFP entspricht der Zuweisung für die Pufferspeichergröße mittels der SYS(3050) Funktion von VFP. Dies dient hauptsächlich der optimalen Anpassung an die vorhandenen Systemressourcen, um das Optimum an Leistung und Performance aus der AFP rauszukitzeln.

Zusätzlich zu den Einstellungen des Webservers bzgl. den von der AFP zu verarbeitenden Dateierweiterungen besitzt die AFP nochmals eine eigene Kontrolle darüber, welche Dateitypen überhaupt verarbeitet werden dürfen.

Der Wert für threads definiert die Anzahl der bei Start der AFP sofort zu aktivierenden Engines. Über das Attribut isolated bestimmen Sie, in welcher Arbeitsumgebung die AFP selbst läuft. Ja, die AFP hat mindestens zwei unterschiedliche Modi wie AFP-Dokumente intern verarbeitet werden - Singlethread und Multithread. Durch Aktivierung des Isolated-Modus setzen Sie die AFP in die Singlethread-Umgebung und verwenden die `AFP3Host.exe` für jede laufende Engine. Dies ist unter anderem der empfohlene Modus, falls Sie das C24-Plugin aktiviert haben sollten. Im Mutlithread-Modus bietet Ihnen die AFP zwar leichte bessere Performance bei geringerer Systembelastung, dafür erhalten Sie jedoch ein paar Einschränkungen seitens VFP. Mit der Angabe eines eigenen Fehlerdokumentes haben Sie die Möglichkeit auf eventuelle HTTP 500 Fehler selbst zu reagieren, um die Standardmeldung zu unterdrücken und den Surfer besser in Kenntnis zu setzen.

4.3.3. <session>

Über die Eigenschaften des <session>-Tags spezifieren Sie genauere Angaben zum Handling von Cookies und Sessions innerhalb der AFP - Abbildung 5-6.

Dabei sind die beiden ’http’ Eigenschaften für die Cookies und die die beiden ’afp’ Werte für die Sessions zuständig. Die Namensgebung ist leider noch historisch bedingt, dürfte sich aber mit einem der nächsten Major-Releases villeicht ändern. Aus Kompatibilitätsgründen zur Vorgängerversion verwenden wir ’afpcookie’ - Sie können die Bezeichnung frei wählen.

> **Anmerkung:** Bei aktiviertem C24 ist die maximale Länge der SessionID auf 10 Zeichen beschränkt und wird mittels SYS(2015) - Eindeutiger Prozedurname - erstellt. Ohne C24 ist die Bezeichnung normalerweise ’sid’ und besitzt eine Länge von ungefähr 44 Zeichen.

Zur Verwaltung der einzelnen Sessions verwendet die AFP eine eigene Tabelle `Session.dbf`. Dort finden sich sämtliche Information über laufende und aktive Websession. Diese Tabelle dient auch als Grundlage zur autmoatischen Löschung abgelaufender Sessions.

4.3.4. <build>

Sie können durch Aktivierung der Option compile-log die AFP dazu veranlassen, dass der Kompilierungsprozess Ihrer AFP-Dokumente ausführlich protokolliert wird. Die Ausgabe erfolgt in das mittels %log% definierte Verzeichnis. Falls Sie also Probleme mit Ihren Webanwendungen haben sollten, aktivieren Sie diese Option. Die Protokolle bieten Ihnen sehr detaillierte Einblicke in die AFP und sollten sowohl aus Sicherheits- und Performancegründen auschliesslich auf Ihrer Entwicklungsmaschine aktiviert werden. Siehe auch Abbildung 5-7.

4.3.5. <debug>

Die Einstellungen für <debug> (Abbildung 5-8) sollten Sie mit Vorsicht geniessen und wenn möglich ausschliesslich auf Ihrem Computer zum Entwickeln verändern.

Mittels ’afpengine’ de-/aktivieren Sie die Interaktion mit dem Standard-Debugger von Visual FoxPro. Bitte ändern Sie dafür auch die Anzahl der Threads beim Start der AFP, um unnötige Instanzen zu vermeiden.

wie bereits mehrfach erwähnt, kann die AFP ihre Fehlermeldungen auch in eine Textdatei rausschreiben. Verwenden Sie hier die Option ’logfile’. Falls Sie nur einen Dateinnamen ohne Pfadangaben angeben, finden Sie die Ausgabe im System32-Verzeichnis unterhalb Ihres Windowsordners. Auch hier sind absolute und relative Pfade zulässig.

"Na toll, ich habe aber kein Visual FoxPro, was jetzt?" - Kein Problem, setzen Sie show-error auf YES und erfreuen Sie sich der Fehlermeldungen im Browser. ABER aktivieren Sie diese Einstellungen nur in einer gesicherten Umgebung - etwa in einem Intranet - da sehr viele Detailinformationen präsentiert werden.

Gepufferte FXP-Dateien können während der Entwicklung extrem nervig sein. Daher aktivieren Sie das lediglich zur Erhöhung der Verarbeitung auf Ihrem Produktivsystem

4.3.6. <plugins> / <plugin>

Die Standardkonfigurationsdatei der AFP beinhaltet keine zusätzlichen Plugins. Diese sind separat zu aktivieren. Dabei empfehlen wir Ihnen dringend wirklich nur die Erweiterungen zu verwenden, die auch tatsächlich in Ihren Webanwendungen genutzt werden. Jedes Plugin beansprucht ein wenig Performance. Okay, wir sprechen hier über Millisekunden, aber auf einer viel besuchten Webseite kann dies schon eine Rolle spielen. Das ControlCenter bietet die Einrichtung von Plugins garantiert die bessere Schnittstelle: Abbildung 5-9.

Jede Erweiterung wird durch ein eigenes <plugin> Tag definiert. Das Attribut location bestimmt dabei den Ordner, in dem sich die ausführbare Datei befindet. Auch hier sind wiederum beide Pfadtypen zulässig.

4.3.7. <virtual>

Nun, könnte es ein wenig komplizierter werden. Das ist mitunter auch eine Begründung, dass Ihnen dieses Tag nicht im ControlCenter zur Verfügung steht. Mittels des <virtual> Tag und den entsprechenden Eigenschaften erhalten Sie die Möglichkeit, sogenannte virtuelle Verzeichnisstrukturen innerhalb der Active FoxPro Pages aufzubauen.

Hm, ein kurzes Beispiel sollte dies anschaulicher machen...

Abbildung 4-6. Anfrage eines AFP-Dokumentes innerhalb der virtuellen Struktur

Stellen Sie sich eine Anfrage auf die URL http://localhost/index.afp vor. Normalerweise befände sich eine physikalische Datei `index.afp` unterhalb Ihres öffentlichen Webordners, bspw. C:\inetpub\wwwroot. Mittels der virtuellen Verzeichnisstruktur kann das gewünschte Dokument jedoch an ganz anderer Stelle, etwa in einem gesicherten Ordner, liegen. Die AFP besitzt zusätzlich zur virtuellen Struktur innerhalb des Webservers eine eigene virtuelle. Spassig, nicht wahr? Über diesen Mechanismus wird die Anfrage zuerst durch den Webserver und danach durch die AFP-eigene Struktur gesucht.

4.4. Debug Modus

Natürlich ist jede Applikation frei von syntaktischen und/oder semantischen Fehlern. Falls dem jedoch nicht so sein sollte, sind die Debug-Kapazitäten eine echte Wohltat beim Entwickeln.

Demnächst mehr...

Hier schnell ein Beispiel wie Sie den Debugging-Modus innerhalb Ihrer AFP-Dokumente aktivieren:

Beispiel 4-2. Activate debug mode - debug.afp

```
<%
ASSERT .F. MESSAGE "Debug me, please..."
? "Hello World!"
%>
```

Dadurch öffnet sich das Debugfenster der VFP-Instanz auf dem Webserver und Sie können in aller Ruhe den fehlerbehafteten Quellcode analysieren.

> **Anmerkung:** Das Debugfenster öffnet sich auf dem Webserver. Also, vermeiden Sie die Einstellung afpengine=YES auf Ihrem öffentlich verfügbaren Produktivsystem. Die Konsequenzen sind sicherlich nicht spassig...

4.5. Error Handling

Jeder Softwareentwicklungsprozess ist in Begleitung von Fehlern oder Fehlfunktionen. Um Ihnen die Arbeit mit den Active FoxPro Pages zu erleichtern, bieten wir Ihnen verschiedene Techniken, um auftretenden Fehlern auf die Schliche zu kommen. Dieser Abschnitt behandelt die unterschiedlichen Systeme und zeigt Ihnen auch Möglichkeiten, wie Sie selbst eingreifen und Veränderungen durchführen können.

Die Einstellungen in der `afp.config` beeinflussen das Vorhandensein und sowie einige Parameter der zur Verfügung stehenden Fehlerausgaben der AFP.

4.5.1. Windows Ereignisprotokoll

Gehen wir einfach die einzelnen Möglichkeiten der AFP durch. Unabhängig von den vorgenommen Einstellungen schriebt die AFP ihre Meldungen in das Ereignisprotokoll für Anwendungen unter Windows. Zum Betrachten starten Sie die Ereignisanzeige und selektieren das Protokoll für Anwendungen. Sie haben somit eine sehr gute und flexible Möglichkeit mit systemeigenen

Anwendungen der AFP auf die Finger zu schauen. Das Ereignisprotokoll gestattet Ihnen sogar das Überwachen von Fremdmaschinen.

Abbildung 4-7. Standardfehlerausgabe - Windows Ereignisprotokoll

Falls Sie eine Clusterlösung betreiben sollten, können Sie alle AFP-basierten Nachrichten der einzelnen Maschinen an zentraler Stelle empfangen und auswerten. Äußerst praktisch, oder?

4.5.2. Einfache Textdatei

Aufgrund von Sicherheits- oder Firewallrestriktionen könnte die Verwendung des Ereignisprotokolls eingeschränkt sein. Aber keine Verzweiflung aufkommen lassen, die AFP bietet hierfür einen zweiten Weg, um an Ereignis- und Fehlermeldungen zu kommen. Dazu sollten Sie sich das logfile-Attribut der Konfiguration genauer anschauen. Hier definieren Sie eine einfache Textdatei (siehe Abbildung 4-8) für die Ausgabe. Das ist vorallem dann sehr nützlich, wenn Sie entweder eingeschränkte Zugriffsmöglichkeiten haben oder das Protokoll per eMail an sich versenden möchten oder um auf einfache Weise den Status per Mobilfunk abzufragen. Die Möglichkeiten sind vielfältig. Schauen Sie sich dazu Abschnitt 4.3.5 und Abschnitt 5.4.2.5 für weitere Details an. Diese Protokollierung ist standard nicht aktiviert.

Abbildung 4-8. Einfache Textdatei

In beiden Abbildungen - Abbildung 4-7 und Abbildung 4-8 - können Sie exakt die gleiche Information in unterschiedlichen Ausgabeformaten erkennen. Wählen Sie Ihre bevorzugte.

4.5.3. Angepasste HTML Fehlermeldung

Suchen Sie einen geschickteren Weg als die HTTP 500 Meldung, um Ihre Besucher über eventuell auftretende Webserverfehler zu informieren? Okay, dann erstellen Sie eine eigene HTML-Datei dafür und konfigurieren Sie <server error=""> der AFP-Konfiguration entsprechend. Falls Sie keinen Wert angeben, sucht die AFP zuerst nach einer Datei `HTTPError.html`.

Aber das wäre nicht die AFP, wenn nicht mehr möglich wäre. Raten Sie mal. Ja, auch innerhalb dieser HTML-Datei haben Sie die Möglichkeit, dynamische Elemente/Informationen zu integrieren. Der einzige Unterschied besteht jedoch darin, dass Sie zum einen reinen VFP Code verwenden müssen und zum anderen die Delimiter jetzt nicht mehr <% or %> sind, sondern << und >> - die Standarddelimiter -

von Visual FoxPro. Mit dieser Erkenntnis können Sie zusätzliche, just ermittelte Informationen an den Surfer zurücksenden.

Beispiel 4-3. Angepasste HTTP 500 Fehlermeldung

```
<html>
<head>
<title>Sorry, work-in-progress!</title>
</head>
<body>
<h2>Dear visitor,</h2>
due to some re-configuration on our billing system, it’s not possible to send any orders
at the moment. We are working on that and the system should be running perfectly in about
<<22 - HOUR(DATETIME())>> hours this day.
<p>
Please be patient, sincerely.
</body>
</html>
```

Sinnigerweise eignet dich das error Attribut der AFP somit, um Ihre Besucher darüber zu informieren, dass gegenwärtig etwas mit dem Webserver oder der Dienstleistung nicht stimmt. Weiteres dazu finden Sie in Abschnitt 4.3.2 und Abschnitt 5.4.2.2.

4.5.4. Detailliertes HTML Feedback

Die bereits vorgestellten Möglichkeiten für Fehlerreporte sind sicherlich nützlich für den normalen Gebrauch. Doch wie sieht es eigentlich während der Entwicklung aus? - Auch für diesen Fall bietet Ihnen die AFP ein nettes Feature - "Red Screen of Death". Klingt interessant, oder? Die entsprechende Einstellungsmöglichkeit show-errors finden Sie bei den Debugoptionen. Bitte lesen Sie in Abschnitt 4.3.5 und Abschnitt 5.4.2.5 weiter, falls dies neu für Sie klingt.

Zur Verdeutlichung betrachten Sie sich untenstehende Abbildung 4-9. Es ist absolut unnötig die detaillierte Fehlerausgabe auf einem Produktivsystem der Öffentlichkeit anzubieten. Sie bewirken damit sogar eher das Gegenteil, in dem Sie zuviele Informationen über Ihr System und dessen Konfiguration - einschliesslich Pfadangaben und Systemvariablen - preisgeben.

Abbildung 4-9. Detailliertes HTML Feedback - Red Screen of Death

Kapitel 5. AFP Applikationen

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

Die Active FoxPro Pages sind sehr modularisiert aufgebaut und bieten umfangreiche Einstellungsmöglichkeiten, um den Entwicklungsprozess für Webanwendungen zu verbessern und um die Antwortzeiten auf dem öffentlich verfügbaren Webserver auf ein Mimnum zu optimieren. Damit dieses Ziel erreciht werden konnte, besteht die AFP aus diversen einzelnen Anwendungen. Dieses Kapitel gibt Ihnen einen Überblick darüber.

Allgemein gesagt, besteht die AFP aus vier Komponenten, die miteinander agieren und kommunizieren.

- ISAPI-Schnittstelle
- AFP Engine
- AFP Server
- Konfiguration

5.1. ISAPI-Schnittstelle zum Webserver

Die AFP ist eine Erweiterung für ISAPI-kompatible Webserver wie IIS oder Apache. Die dafür notwendige Systembibliothek ist die Datei `AFP3.DLL`, welche die Kommunikation zwischen dem Webserver und der AFP Engine kontrolliert und regelt.

Der tatsächliche Ablauf ist einfach: jede eingehende Anfrage für ein AFP-Dokument wird vom Webserver - in Abhängigkeit seiner Konfiguration für die Dateierweiterung - an die ISAPI-Schittstelle `AFP3.DLL` übergeben, welche diese wiederum an die AFP Engine weitergibt und im Gegenzug das entstehende Result an den Webserver zurückliefert.

Für die ISAPI erweiterung gibt es eine optionale Konfigurationsdatei `AFP3.INI` mit einem einzigen Parameter 'BusyTimeout'. Dieser spezifiziert die maximale Wartezeit in Sekunden, die die `AFP3.DLL` bis zur Antwort der AFP Engine verweilen lässt, bevor eine Time-Out-nachricht an den Webserver zurückgeliefert wird. Falls die Einstllung fehlt, beträgt die Standardzeit 30 Sekunden.

5.2. Varianten der AFP Engine

Was bedeutet das eigentlich? - Die AFP Engine ist sozusagen das 'Arbeitstier' in der Gesamtheit der einzelnen Komponenten der Active FoxPro Pages. Die Engine kümmert sich um die eingehenden

Informationen aus der ISAPI Erweiterung, interagiert mit den Tabellen und erzeugt die zurückzusendende Antwort an den Webserver. Unter Varianten gruppieren sich die unterschiedlichen Laufzeit-Umgebungen für die AFP Engine. Wie bereits in Abschnitt 4.3.2 erwähnt, kann die AFP in unterschiedlichen Modi betrieben werden - Multithreaded, Singlethreaded aber Mehrfachbenutzung und Debugmodus. Die richtige Umgebung ergibt sich aus den Konfigurationseinstellungen für isolated und afpengine. Genauere Details erschliessen aus Abschnitt 4.3.

Normalerweise wird die AFP in der multithreaded Umgebung - AFP3Engine.DLL (MTDLL) - ausgeführt und bietet dabei die beste Performance bei geringster Systembelastung. Leider fordert der MTDLL-Modus von VFP ein paar Einschränkungen bzgl. der Verwendung einiger VFP-Anweisungen. Siehe dazu Anhang A für unzulässige sowie problematische Funktionen.

Falls dennoch die Notwendigkeit besteht einen oder mehrere dieser Befehle nutzen zu müssen, so betreiben Sie die AFP mit anderen Optionen, wie etwa aktivem isolated. Dies verursacht, dass die AFP Engine nun als singlethreaded Task - AFP3Host.exe (EXE) - gestartet wird, dafür aber mehrfach. Die Konsequenz daraus führt zu höheren Systemanforderungen bzgl. Arbeitsspeicher und etwas geringfügiger Leistung. Bedenken Sie dabei, dass wir uns hierbei im Millisekundenbereich bewegen.

Schlußendlich bietet die AFP noch den Debugmodus speziell während der Entwicklung Ihrer Webanwendungen an. Dies erfordert jedoch, dass Sie eine vollständige Microsoft Visual FoxPro Entwicklungsumgebung (ab Version 7) installiert haben. Um den Debugmodus zu verwenden, aktivieren Sie die Einstellung afpengine in der Datei `afp.config` (siehe Abschnitt 4.3.5) oder setzen einfach ein Häkchen bei ’Debug Engine’ auf der Seite Debug im AFP 3.0 ControlCenter (siehe Abschnitt 5.4.2.5).

Die AFP Engine selbst kann nicht als einzelne Applikation verwendet werden, sie setzt immer einen AFP Server voraus.

5.3. Varianten des AFP Server

Somit beschäftigen wir uns auch gleich mit der Zentraleinheit - dem AFP Server. Der Server kontrolliert und überwacht jegliche Aktivitäten zwischen den beteiligten AFP-Komponenten und lädt unter anderem die Konfiguration. Genau wie Sie eben die Auswahl aus verschiedenen AFP Engines hatten, bietet die AFP 3.0 auch mehrere Server an, um die AFP in Aktion zu setzen. Das Standardverhalten bedient sich des AFP3 Dienstes (`AFP3SRV.EXE`), welcher automatisch nach einem Betriebssystemstart ohne Benutzer anläuft. Dies ist übrigens auch die empfohlene Variante für das Betreiben der Active FoxPro Pages auf Ihrem Webserver.

Während der Entwicklung von Webapplikationen kann das Starten und Stoppen eines Dienstes ziemlich zeitraubend sein. Daher ist hierfür die Einzelanwendung `AFP3.EXE` besser geeignet, da sie dem Entwickler ein direktes Eingreifen in die AFP Engine bietet.

Abbildung 5-1. AFP 3.0 Desktop Application with Quick Launch

> **Tipp:** Erstellen Sie sich eine Verknüpfung zur AFP3.exe in der Schnellstartleiste. Das spart Ihnen Zeit und unnötige Mausklicks.

Als letztes sei noch die Ansteuerung des AFP Servers über COM erwähnt. Mittels der ausgelieferten Typbibliothek können Sie die AFP problemlos als OLE Automationsserver betreiben und beispielsweise in eigene Anwendungen integrieren. Hierzu ein kurzes Beispiel, wie das ablaufen könnte.

Beispiel 5-1. Verwendung der AFP 3.0 COM-Schnittstelle - VFP-Beispiel

```
ON SHUTDOWN CLEAR EVENTS

oAFP = CREATEOBJECT("AFP3.Server")
oAFP.StartUp("C:\Program Files\AFP3\afp.config")

READ EVENTS

oAFP.ShutDown()
```

Das ist eigentlich alles, um die AFP als COM Server zu betreiben.

5.4. AFP 3.0 ControlCenter

Das AFP 3.0 ControlCenter stellt die ’Kommandozentrale’ der AFP dar. Im ControlCenter bekommen Sie Auskunft über die aktuellen Stati der einzelnen AFP Threads, fügen neue Instanzen hinzu oder entfernen welche und modifizieren auf die einfache Weise die Konfiguration der AFP.

Abbildung 5-2. The AFP 3.0 ControlCenter

5.4.1. Status

Der Statusdialog gibt Ihnen Informationen über die laufenden Instanzen. Dies umfasst auch das zuletzt ausgeführte AFP-Dokument samt Ausführungszeit. Zusätzlich können Sie weitere AFP Instanzen auf

Knopfdruck erstellen - sowohl im Normal- als auch Debugmodus. Falls ein Thread hängen sollte, können Sie diesen mittels des ControlCenters beenden ohne gleich die komplette AFP neustarten zu müssen.

Abbildung 5-3. Statusübersicht laufender AFP Instanzen

5.4.2. Konfiguration

Schauen wir uns doch einmal ein wenig die AFP Konfiguration an. Die Datei `afp.config` ist ein kleines XML-Dokument, das bei jedem Start der AFP, also des einen Threads, eingelesen wird. Wenn Sie also Änderungen vornehmen, denken Sie dran, die Threads neuzustarten. Wir denken, dass diese Vorgehenweise besser als ein direktes Anwenden der Änderungen ist. Somit können Sie zeitgesteuerte Aktivitäten mit unterschiedlichen Konfigurationen durchführen.

5.4.2.1. Pfad

Der konfigurationsdialog im AFP 3.0 ControlCenter bietet mehrere Tabs bzw. Seiten, welche die entsprechenden XML-Tags innerhalb der Konfigurationsdatei darstellen. Wir besprechen jede einzelne Seite.

Abbildung 5-4. Pfadangaben

Auf der Seite Pfad hinterlegen Sie die Verzeichnisse in der die AFP agieren darf. Das Basisverzeichnis zeigt meist auf das Applikationsverzeichnis, welches bei der Installation angeben wurde. Kein Eintrag ist gleichbedeutetnd mit dem Ordner, in dem sich entweder die AFP3.EXE oder die AFP3Srv.EXE befindet.

Die Direktive für die Common Files gibt Auskunft darüber, wo sich bei Einsatz der AFP in einem Cluster, die gemeinsamen Metadaten befinden, meist entspricht dies einer Netzwerkfreigabe auf einem anderen System. Da in den meisten Fällen kein Cluster vorliegt, entspricht der Wert dem Basisverzeichnis.

bei jedem erstmailgen Aufruf eines AFP-Dokuments wird dieses durch die Visual FoxPro Laufzeitbibliothek kompiliert und im Cache abgelegt. Dies garantiert für alle folgenden Aufrufe wesentlich schneller Reaktionszeiten. Das Cacheverzeichnis befindet sich normalerweise unterhalb des Common.

Falls Sie während der Entwicklung genaue Informationen zum kompilierungsprozess wünschen, so werden diese Protokolle im Logpfad abgelegt.

Die Verwaltungstabelle für die einzelnen Session liegt zwar im Commonverzeichnis, die jeweilige, temporäre Einzelsession wird jedoch mittels der Pfadangabe für Session festgelegt. Auch dieser Wert liegt standesgemäß im %common%.

5.4.2.2. Server

Wie bereits im vorhergehenden Abschnitt erwähnt, gibt es für die AFP unterschiedliche Servervarianten. Die Einstellungen auf dem Tab Server legen dabei die initialen Startparameter, wie allozierende Speicherbelegung (in Bytes) für jede AFP Engine sowie die Gesamtanzahl fest. Abhängig von der eingesetzten Hardware lässt sich hier noch zusätzlich Leistung herauskitzeln.

Abbildung 5-5. Einstellungen zur Komponente AFP Server

Die Angabe von zulässigen Dateierweiterungen ist absolut notwendig für das Verarbeiten von AFP-Dokumenten. Falls die entsprechende Erweiterung nicht hinterlegt ist, verweigert die AFP die Interpretation der Anfrage. Es handelt sich hierbei um einen zusätzlichen Kontrollmechanismus zum Webserver. Wenn Sie mit der AFP beispielsweise WAP-Seiten dynamisch erzeugen wollen, so ist die Endung .WML entsprechend hinzuzufügen. Ansonsten bekommen Sie einen schlichten HTTP 500 Fehler.

> **Anmerkung:** Die zulässigen Erweiterungen betreffen nur die für die AFP erlaubten Dateiendungen, welche von der AFP Engine verarbeitet werden dürfen. Dies entspricht NICHT zwangsläufig den Skripterweiterungen des Webservers. Die AFP führt demnach nur eine zusätzliche Prüfung durch.

Falls Sie die AFP mit aktiviertem C24 Plugin betreiben, sollten Sie die Threads isolated betreiben. Das begründet sich unter anderem darin, dass die Vorgängerversion 2.4 für den MTDLL-Modus unzulässige Befehle verwendet hat. Als problematisches Beispiel sei nur der CD-Befehl zum Wechseln von Verzeichnissen genannt.

Informieren sie den Surfer mit Ihrer eigenen Fehlerhinweismeldung (HTML-Datei). Interessanterweise darf es sich hierbei um ein skriptsprachen angereichertes Dokument handeln. Nutzen Sie diese Feature, denn eine simple Hinweismeldung dürfte allemal besser ein als nichtaussagender HTTP 500 oder im schlimmsten Fall ein Red Screen of Death sein. Schauen Sie dafür auch mal in Abschnitt 4.5 nach.

5.4.2.3. Session

Schauen wir uns nun die Einstellungen für die Interaktion mit dem Besucher an. Dafür gibt’s effektiv zwei Varianten, wie dies erfolgen kann - Cookies oder Sessions. Was ist dabei der Unterschied? Nun Cookies sind kleine Textdateien, die auf dem Computer des Besuchers zwischengespeichert werden können, während Sessioninformationen serverseitig verwaltet werden. Mittels beider Techniken bekommt die Möglichkeit den Surfer auf der eigenen Website zu ’verfolgen’ oder um zusätzliche

Informationen eines einzelnen Benutzers mitzuführen. Aufgrund der Tatsache, dass Cookies beim Besucher eventuell aus Sicherheitsgründen deaktiviert sein könnten, bieten sich Sessions als gute Alternative an.

Sessions sind eigentlich nichts anderes als die ’AFP Cookies’ der Vorgängerversion. Der Begriff mag damals vielleicht ein wenig unglücklich gewälht worden sein. Über das Sessionmanagement brauchen Sie keine benutzerbezogenen Informationen auf dem Fremdsystem abzulegen. Zusätzlich bleiben die Informationen (bedingt) erhalten, falls der Surfer seinen PC oder Browser zwischenzeitlich wechseln sollte. Dazu werden die Daten einer Session zentral auf dem Server im %session% Verzeichnis abgelegt und mittels der Session.DBF überwacht. Für die Entwicklung Ihrer Webanwendungen bauen Sie dazu die SessionID immer in Ihre verlinkten URLs, auch in die FORM ACTION ein. Dadurch haben Sie stets den aktuellen Standort und Status eines Surfers innerhalb Ihrer Webseiten.

Die Konfigurationsseite offeriert beide Möglichkeiten. Hier wird definiert, ob überhaupt Cookies und/oder Sessions verwendet werden und wie diese lauten sollen. Der jeweilige Name ist auch hier weitestgehend frei wählbar, sollte aber keine deutschen Sonderzeichen enthalten, um eventuellen Zeichensatzproblemen direkt aus dem Weg zu gehen.

Die Angabe der Sessiondatei bezieht sich auf die Verwaltungstabelle aller aktiven Sessions, welche meistens unterhalb des %common% Verzeichnis abgelegt wird. Timeout spezifiziert die maximale Lebensdauer der einzelnen Sessions bevor diese automatisch aus der Verwaltung entfernt werden. Details finden sich in Abbildung 5-6.

Abbildung 5-6. Sessionmanagement - die einfache Konfiguration

Abbildung 5-7. Kompilierungsprotokoll erwünscht?

5.4.2.4. Protokoll

Aufgrund der Tatasache, dass jedes AFP-Dokument beim ersten Aufruf durch die AFP Engine kompiliert - auch bei Änderungen - können dazu die entsprechenden Protokolle zum Erstellungsprozess unterhalb des %log% Verzeichnisses abgelegt werden. Dies dürfte vorallem beim Scuhen von Fehlern interessant sein. Siehe dazu auch Abbildung 5-7).

5.4.2.5. Debug

Schauen wir uns mal die Optionen für das Ausgeben und Aufspüren von eventuellen Fehlfunktionen mit

der AFP 3.0 an. Hierbei handelt es sich um eine vollständig neue, faszinierende Möglichkeit innerhalb der Entwicklung von dynamischen Webanwendungen. Noch nie war es einfacher. Was ist dafür notwendig? - Nun, zuerst einmal eine Debug Engine der AFP selbst und eine (optionale) installierte Version von Visual FoxPro 7 oder höher. Wie, ich benötige eine Visual FoxPro Entwicklungsumgebung? - Nicht direkt, aber es erleichtert die Sache ungemein und der Debugmodus lässt in vollem Maße nutzen. Der Abschnitt Debugmodus behandelt die Materie etwas umfangreicher.

Also on the Debug page you define if the generated compiled AFP documents - FXP files - should be cached. For performance reason it's better to enable caching on your live system. In consequence the documents reside in the memory of your Web server and respond quicker. On a development machine preference lies on disabled caching cause then you are able to cleanp up the %cache% directory without stopping the AFP engine.

Abbildung 5-8. Controlling the Debug options of the AFP Engine

Show error message in browser - this (de-)activates the Red Screen of Death if any parsing or systematic error occurs inside your AFP documents. Take care of this option! For security reasons activate this only on your development enviroment but not your live system. The generated error message contains a lot information about your system and might open wide doors to crackers to knock down your Web server. Details are discussed in the section about Error Handling.

5.4.2.6. Plugin

An absolutely new highlight of the AFP version 3.0 is the ability to increase and decrease its functionality and power dynamicly. With Plugin management you are able to set up your own AFP enviroment which best fits your needs. And as the state-of-the-art you are even able to write your own plugins to enhance the AFP. To manage the plugins you specify them on the Plugin page, re-start the AFP engine and enjoy them.

Abbildung 5-9. Specifying additional Plugins

Currently the AFP ships with three and a 'half' plugins - C24 Fox and C24 Cookie, Voodoo Web Controls and DirectCall. It's recommended to specify the relative path to the plugins with the Path variables. Default, the plugins reside below the %root% path but absolute paths are possible, too.

Detailed information about plugins and their development is described in the Developer's Guide to Active FoxPro Pages 3.0.

5.4.2.7. License

Although the AFP is fully functional in demo mode, it’s always a nice move to purchase a license. Entering your license is quite easy. Specify the Name and Serial Number, send the resulting Hardware Key to your distributor to obtain an Activation Key.

Abbildung 5-10. Entering the License informations and Activation key

A licensed version of the AFP lacks the appearence of the demo table at the top of every processed AFP document and the nag screen every 15 minutes.

Kapitel 6. Plugins

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

In this chapter we're talking about the plugin interface of the Active FoxPro Pages. Plugins give you a flexible way to enhance the power of the AFP to your belongings. As you already might have seen, the standard edition of the AFP comes with some optional plugins like the compatibility plugins for the previous AFP version 2.4.

6.1. C24 - AFP 2.4 Compatibility

The C24 plugin offers you a backward compatibility to run existing Web sites build with AFP 2.4 with version 3.0. The C24 is a layer to represent the older FOX object and its functionality translated to the new objects and PEMs. The C24 is not full-compliant to the original version, i.e. the FOX.browse() is missing.

6.1.1. Comparison between AFP 2.4 and 3.0

Here is an alphabetical overview of the properties and methods the C24 plugin takes care of and some suggestions which properties and methods should be used in version 3.0 instead. For your explanation these tables do NOT represent the functionality of the C24 plugin but a conversion help and reference for you to get a starting point for the new AFP 3.0 syntax without the need of C24.

Tabelle 6-1. AFP 2.4 - system variables

Version 2.4	possible substitution - Version 3.0
FOX.aConvert[]	Server.HTMLEncode()
FOX.aFormVar[]	REQUEST.Form(@aForm)
FOX.aParam[]	REQUEST.QueryString(@aQueryString)
FOX.cAFPHook	Handled by the event procedures in your *.afp[a].code files
FOX.cAFPInit	Handled by the events EVENT_Init and EVENT_InitVariables in your *.afp[a].code files
FOX.cAFPPath	PATH.cRoot
FOX.Call	RESPONSE.Call()
FOX.cAPPId	APP.cID
FOX.cBrowseSeparator	
FOX.cContenttype	RESPONSE.ContentType

Version 2.4	possible substitution - Version 3.0
FOX.cCookie	SESSION.SessionID()
FOX.cDefines	Place your #DEFINEs in a procedure called DOCUMENT_INCLUDEALL in your *.afpa.code file
FOX.cEmailEHost	
FOX.cEmailEReceiver	
FOX.cEmailESender	
FOX.cEmailHost	
FOX.cEmailReceiver	
FOX.cEmailSender	
FOX.cForm	REQUEST.Form()
FOX.cForm1	FILE.cLocation
FOX.cHttpCookie	RESPONSE.Header()
FOX.cHttpCookieString	REQUEST.ServerVariables("HTTP_COOKIE")
FOX.cHttpHeader	Primarly this is handled by the Web server. To set your own HTTP header extensions refer to RESPONSE.AddHeader()
FOX.cIni	* obsolete * Due to the new communication structures between AFP 3.0 and the Web server, there is no need for the .INI files used by version 2.4 anymore
FOX.cIniFile	* obsolete * Due to the new communication structures between AFP 3.0 and the Web server, there is no need for the .INI files used by version 2.4 anymore
FOX.cLanguage	REQUEST.ServerVariables("HTTP_ACCEPT_LANGAGE")
FOX.cLogFile	* obsolete * Use the capabilities of the Event Viewer of Microsoft Windows NT and higher, the separate error-log file and the native VFP Debugger for this property.
FOX.cParameter	REQUEST.QueryString()
FOX.cPath	FILE.cLocation
FOX.cPathInfo	FILE.cVirtualLocation

Version 2.4	possible substitution - Version 3.0
FOX.cSQLProt	Due to structural enhancements not possible. You can implement something simi- lar using SYS(3054) from VFP
FOX.cTimeStamp	* obsolete * Use the log files of the Web server instead. If needed, implementation is very easy using FILETOSTR() with a customized string in combination with some server variables and time stamp.
FOX.cVersion	SERVER.Version()
FOX.lAFP2ASP	
FOX.lAutoCookie	afp.config: http-sessionid="YES" | "NO"
FOX.lAutoLocalize	
FOX.lCloseAll	* obsolete *
FOX.lConvertAlways	* obsolete * Use SERVER.HTMLEncode() instead
FOX.lCookieHandling	afp.config: http-sessionid="YES" | "NO"
FOX.lcstring	* obsolete * Might use SERVER.Tag instead
FOX.lDebug	afp.config: afpengine="YES"|"NO"
FOX.lError	Define your own error handler - EVENT_Error - in your *.afp[a].code files
FOX.lExecuteFXP	* obsolete *
FOX.lFoxCookieHandling	afp.config: afp-sessionid="YES" | "NO"
FOX.lFXPOnly	
FOX.lKeepAlive	* obsolete *
FOX.lLocalize	
FOX.lMadeHtml	* obsolete *
FOX.lNewCookie	SESSION.IsNew()
FOX.lRecompile	* obsolete *
FOX.lScriptMap	This is handled by the virtual tree structure defined by the <virtual> tags in `afp.config`
FOX.lSQLProt	see FOX.cSQLProt
FOX.lTab2Return	* obsolete * CHRTRAN(cString, CHR(9), CHR(13))
FOX.lWorkAsFXP	* obsolete *

Version 2.4	possible substitution - Version 3.0
FOX.nAktBrowseLine	see FOX.Browse()
FOX.nCookieTimeOut	SESSION.Timeout()
FOX.nError	ERROR()
FOX.nMemory	afp.config: memory="nMemory"
FOX.nSekunden	* obsolete * Might be implemented with some time handling mechanism in EVENT_PageCallBefore

Tabelle 6-2. AFP 2.4 - system methods

Version 2.4	possible substitution - Version 3.0
FOX.Browse()	
FOX.ConvToStr()	Transform()
FOX.CookieDate()	RESPONSE.CookieDate()
FOX.GetFormVar()	REQUEST.Form()
FOX.GetHttpCookie()	REQUEST.Cookies()
FOX.GetIniVar()	REQUEST.ServerVariables()
FOX.GetPar()	REQUEST.QueryString()
FOX.Include()	FileToStr()
FOX.Localize()	
FOX.Out()	SERVER.HTMLEncode()
FOX.Read()	Partly through REQUEST.ServerVariables() Rest is obsolete and easy to implement yourself
FOX.SetHttpCookie()	RESPONSE.AddHeader()
FOX.Write()	Obsolete and easy to implement yourself

6.2. DirectCall - Code-based Programming

The DirectCall uses another technique to parse and response any AFP documents. Actually you are calling your object methods directly to get the results.

6.3. Crypto - Secure your data

The crypto plugin is based on the CryptoAPI of Microsoft Windows 2000 and higher. To use the plugin

add a new <plugin>-Tag to your `afp.config` and re-start the AFP engines.

```
<plugin location="%root%\plugin\crypto.plugin.exe" engine="AFP"/>
```

Now, you have a crypto object available to your AFP documents. Currently there are three methods to work with:

- Encrypt()
- Decrypt()
- Checksum()

And here’s a basic example on how to use the crypto plugin and its syntax.

Beispiel 6-1. Using the crypto plugin - crypto.afp

```
<%
lcString = "This sentence is top secret."
? "Original string: ’" + lcString + "’"
? "<p>Let’s look at the results:"
? "<br>CRYPTO.Encrypt(): " + CRYPTO.Encrypt(lcString)
? "<br>CRYPTO.Decrypt(): " + CRYPTO.Decrypt(CRYPTO.Encrypt(lcString))
%>
```

6.4. Create your own Plugins

"Hey, I like those plugins but how to..." - you might think now. Plugins add a lot of flexibility and power to the AFP and you decide which plugins you need or want. So, in this section let’s see how to write individual plugins.

What is a plugin?.

A plugin for the AFP is just a normal Microsoft Visual FoxPro 7 compiled application - either .APP, .EXE or .FXP. To integrate your VFP application as plugin to the AFP you have to add a new <plugin> tag in the afp.config which refers to your program. After successfully reloading the AFP engine your .AFP documents can use your plugin. That’s all, almost.

An AFP plugin has to fullfil several requirements before it loads to the AFP engine anyway and here they are.

6.4.1. Requirements

Before you can use your own plugins, they have to match a specific interface given by the plugin management of the AFP. In this section we explore this interface and you will see that is very easy to write your own add-ons.

If the AFP starts a plugin one parameter by reference is passed to add this object to its plugin management.

An AFP plugin must have the following properties and methods:

6.4.1.1. cID

A short identifier string to represent or list the loaded plugin on the information page of AFP.

Take a look at the plugins section in `afpinfo.afp`.

6.4.1.2. Load-Method

Obviously the initializing method to specify any defaults or pre-requisities the plugins needs - like the global variable wanted in any AFP document.

6.4.1.3. Unload-Method

While an AFP engine quits, the plugin management calls the Unload method of every loaded plugin to release itself gracefully. So, this method is to clean up the plugin environment and to release any objects to avoid dangling references.

6.4.2. The plugin loader

To use an VFP application as AFP plugin you have to load it through the plugin management. Here is sample code on how this should be done (taken from the voodoo plugin).

Beispiel 6-2. The plugin loader

```
*============================================================================
* The Loader creates an instance of the CPlugIn object and passes it
* back to the caller.
*============================================================================
LParameter roPlugIn
```

```
roPlugIn = CreateObject("CPlugIn_Voodoo")


*============================================================================
* CPlugIn_Voodoo
*
* Implements the REQUEST and WebPage classes as needed by Voodoo.
*============================================================================
Define Class CPlugIn_Voodoo as CPlugin

    cID = "VOODOO"

*============================================================================
* Make the classes available to AFP. All Subclasses must be in the main
* procedure file.
* Otherwise, VFP is not able to find them.
*============================================================================
Procedure Load
    Local lcVoodoo, lcPath
    If not Upper(JustPath(This.cFullFileName)) $ Upper(Set("Path"))
        lcPath = Set("Path")
        If not Empty(m.lcPath)
            lcPath = m.lcPath + ","
        EndIf
        lcPath = m.lcPath + JustPath(This.cFullFileName)
        Set Path To &lcPath
    EndIf
    Set Procedure To ("Voodoo") Additive
    Set Procedure To (This.cFullFileName) Additive
EndProc

*============================================================================
* Get rid of the procedure files
*============================================================================
Procedure Unload
    Release Procedure ("Voodoo")
    Release Procedure (This.cFullFileName)
EndProc

EndDefine
```

First the plugin loader should be the main program of your project that handles the parameter passed from the AFP management. This parameter is passed by reference and returns actually the object.

Any plugin should inherit from the CPlugin class of the AFP. It’s a commendation for your plugins but not mandatory. And why not using the offered plugin base? You have the choice.

Now, the source code fullfils the plugin requirements - cID property and load/unload methods. It should be clear what's going on. The Voodoo plugin checks if the needed procedure file `voodoo.fxp` for the Voodoo Web Controls is in path of the plugin at the begin and releases the additive procedure at the end.

II. Reference

Reality is for people who lack imagination.

anonymous

Kapitel 7. Objects

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

Some nice words about objects...

App-Object

Name

`App-Object` — represents a separate application in the AFP enviroment.

Synopsis

App

Notes

The APP object is only available when a page is part of an AFP application. Applications have a private datasession. Files open in onw application are not visible to other applications.

Properties

cFile
cID

See also

File-Object

Name

`File-Object` — represents the curremt executed .AFP document.

Synopsis

File

Notes

The FILE object represents the .AFP document that is executed. In AFP, a file object manages the physical location and the location within the virtual directory tree of a web site.

Properties

cLocation
cVirtualLocation

Methods

AddExtension()
Clone()
GetCacheName()
GetLocation()
Relative()
Reset()
ResolveRelative()

See also

Path-Object

Name

`Path-Object` — contains all informations about path configuration.

Synopsis

Path

Notes

The PATH object manages the various paths used in AFP.

Properties

cCache
cCommon
cLog
cRoot
cSession

Methods

MakePath()

See also

Request-Object

Name

`Request-Object` — holds parameters about the request sent by th web server and browser.

Synopsis

Request

Notes

The request object contains all the incoming data from a web request. These informations are sent from the users browser, the web server handling the request and the requested url in the user's browser.

Methods

Cookies()
Form()
MultiPart()
QueryString()
Reset()
ServerVariables()

See also

Response-Object

Name

`Response-Object` — handles all content sent back to the browser.

Synopsis

Response

Notes

The Response object collects all information to send them back to the users browsers. The object has two different kinds of data - HTTP header and content.

Properties

Content-Type | Cookie | Expires |

Methods

AddHeader()
BinaryWrite()
Body()
Clear()
Cookies()
Document()
Header()
HTTPCookie()
Redirect()
Reset()
Write()

See also

Server-Object

Name

`Server-Object` —

Synopsis

Server

Notes

Methods

DoCmd()
Execute()
HTMLDecode()
HTMLEncode()
MapPath()
Transfer()
URLDecode()
URLEncode()

See also

Session-Object

Name

`Session-Object` — manages the session id.

Synopsis

Session

Notes

However, it does not manage variables or persistent contents! In AFP 3.0 a session is created, when you call the SESSION.SessionID() function. If you don’t call it, no session is created. Sessions are managed in a table SESSION.DBF which sits in the common directory.

Each time you renew a session or a request comes in that contains session information, a time stamp is updated. Session information can be carried in two forms: Either as a HTTP cookie or as a QueryString parameter. The name of both are configurable in the afp.config file. The default is "sid" for session ID, but in the sample config file it has been changed to "afpcookie" for backward compatibility reasons.

The C24_Cookie plugIn replaces this session object. CreateSessionID() returns the old SYS(2015) setting. Also, the ?!_XXXXXXXXX Syntax is regonized as a session ID.

The session object works closely with the ControlServer (AFPCS.EXE). This server is running in the background and removing old session data. This ensures that a request is not slowed down due to garbage collections.

Methods

Abandon()
CreateSessionID()
GetSessionCookie()
GetSessionData()
GetSessionFileName()
IsNew()
NewSession()
ReNew()
SessionID()
SetSessionData()
Timeout()
URL()

See also

Kapitel 8. Properties

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

cCache-Property

Name

`cCache-Property` —

Synopsis

Path.cCache [`= cDirectory`]

Attribute values

Notes

The cache directory contains all the compiled AFP documents. You can safely delete all files in there. The cache can be a local directory. The default is %common%/cache.

Example

See also

cCommon-Property

Name

`cCommon-Property` —

Synopsis

Path.cCommon `[ = cDirectory]`

Attribute values

Notes

Common directory. This directory must be accessible from all AFP instances. Default is %root%.

Example

See also

cData-Property

Name

`cData-Property` —

Synopsis

Request.cData

Attribute values

Notes

Example

See also

cFile-Property

Name

`cFile-Property` — name of the application file in the local cache.

Synopsis

App.cFile

Attribute values

Notes

Example

See also

cID-Property

Name

`cID-Property` — ID of the application.

Synopsis

App.cID

Attribute values

Notes

This ID is used when variables are stored. Two applications with the same ID change the same set of variables, but not the same set of files.

Example

See also

cLocation-Property

Name

`cLocation-Property` — Physical file as specified by the web server in the PATH_TRANSLATED server variable.

Synopsis

File.cLocation

Attribute values

Notes

This value is optional.

Example

See also

cLog-Property

Name

`cLog-Property` —

Synopsis

Path.cLog `[ = cDirectory]`

Attribute values

Notes

Log directory. All log files produced by AFP are saved here. Default is %root%\log

Example

See also

ContentType-Property

Name

`ContentType-Property` —

Synopsis

Response.ContentType `[= cContentType]`

Attribute values

Notes

Example

Change the HTTP header to return a .PDF document:

```
Response.ContentType = "application/pdf"
```

See also

AddHeader-Method

Cookie-Property

Name

`Cookie-Property` —

Synopsis

Response.Cookie

Attribute values

Notes

Example

See also

AddCookie-Method | GetCookieHeader-Method

cRoot-Property

Name

`cRoot-Property` —

Synopsis

Path.cRoot `[= cDirectory]`

Attribute values

Notes

Root directory. This directory can be local in a cluster.

Example

See also

cSession-Property

Name

`cSession-Property` —

Synopsis

Path.cSession

Attribute values

Notes

Session directory. All session information are stored here. Therefore this directory must be accessible by all AFP instances. IMPORTANT: All files in this directory are deleted unless they belong to a valid session. The default is %common%/session.

Example

See also

cVirtualLocation-Property

Name

`cVirtualLocation-Property` — Location of the file in the WebServers hierarchy.

Synopsis

File.cVirtualLocation

Attribute values

Notes

If the location is http://server/foo/foobar.afp, this property contains "/foo/foobar.afp". This value is mandatory and used to determine the cache file name.

Example

See also

Expires-Property

Name

`Expires-Property` — Expiration date of the responsed page.

Synopsis

Response.Expires `[ = Date]`

Attribute values

Date

Notes

Example

The expiration of this page is tomorrow.

```
<%
Response.Expires = DATE() + 1
%>
</para>
</refsect1>
<refsect1><title>See also</title>
<para>
</para>
</refsect1>
</refentry>

<refentry id="response.status">
<refnamediv>
<refname>Status-Property</refname>
<refpurpose>sets the HTTP response status.</refpurpose>
</refnamediv>
<refsynopsisdiv>
<cmdsynopsis>
<command>Response.Status</command>
<arg choice="opt">
  <option>= cHTTPStatusCode</option>
</arg>
</cmdsynopsis>
```

```
</refsynopsisdiv>
<refsect1><title>Attribute values</title>
<para>
</para>
</refsect1>
<refsect1><title>Notes</title>
<para>
</para>
</refsect1>
<refsect1><title>Example</title>
<para>
Invoke a redirection to another URL:
<screen><![CDATA[
Response.Status = "301"
Response.AddHeader("URI", "http://www.active-foxpro-pages.com")
```

See also

AddHeader-Method

Kapitel 9. Methods

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

Abandon-Method

Name

`Abandon-Method` — terminates the current session.

Synopsis

Session.Abandon( )

Return values

Parameters

Notes

The session cannot be recovered. You can call this method to implement a logoff feature to ensure that sensitive data is not available to the next user on a computer.

Example

The following document could be used to log out the current user. Once the session has been abandoned, the browser is redirected to the login page letting the next user log onto the system. When you design applications that might be used from public computers, you should always provide an explicit logout option. Otherwise the next visitor might gain access to private data of a previous user.

`LogOut.AFP`

```
<%
SESSION.Abandon()
RESPONSE.Redirect("Login.afp")
%>
```

See also

AddExtension-Method

Name

`AddExtension-Method` — returns a new file object that has been extended by the string in cExtension.

Synopsis

Response.AddExtension(cExtension)

Return values

Object

Parameters

cExtension

File extension to add to the current file.

Notes

Tjoa...

Example

For example, file.AddExtension(".config") returns a file object that represents the configuration file.

```
<%
oFile = FILE.AddExtension(".config")
%>
```

See also

AddCookie-Method

Name

`AddCookie-Method` — adds a user-defined cookie information to the response.

Synopsis

Response.AddCookie(cName, cValue, cPath, uDate)

Return values

Logical

Parameters

cName

Name of the cookie to send back to the client

cValue

Corresponding value for the specified cookie

cPath

Corresponding path for the specified cookie

uDate

Expiration date of the cookie. Allowed value type are string and date.

Notes

The AddCookie-Method gives the opportunity to send your own cookies with their properties back to the clients browser.

Example

This example creates a cookie.

```
<%
Response.AddCookie("AFP", "Active FoxPro Pages", "/", DATE() + 1)
%>
```

See also

Cookie-Property | CookieDate-Method | GetCookieHeader-Method | Header-Method | Write-Method

AddHeader-Method

Name

`AddHeader-Method` — adds user-defined information to the HTTP header.

Synopsis

Response.AddHeader(cKey, cValue)

Return values

Logical

Parameters

cKey

Keyword to add to the HTTP Header.

cValue

Corresponding value for the specified keyword

Notes

The AddHeader-Method gives the opportunity to add own keywords and values to the HTTP Header. This only possible before any response is sent back to the browser.

Usage of colons for the specified cKey parameter is not permitted.

Example

This example adds an additional key-value pair to the HTTP header:

```
<%
*// write additional HTTP-Header.
*// X-Powered-By: AFP/<version>
Response.AddHeader("X-Powered-By", "AFP/" + SERVER.Version())

*// response normal HTML
? "<html><head></head><body>"
? "This page Page creates an additional entry to the HTTP header.<br>"
? "Show result:"
? "<p>" + Server.HTMLEncode(Response.Header())
? "</body></html>"
%>
```

See also

ContentType-Property | Header-Method | Write-Method

BinaryWrite-Method

Name

`BinaryWrite-Method` — returns any binary file

Synopsis

Response.(cData)

Return values

Character

Parameters

cData

Any data content to send back to the client's browser.

Notes

The method adds the binary content to the response buffer and does not handle any content type information. So if you want to respond, ie. a PDF file you have to set the correct content type additive.

Example

Send back a PDF file

```
<%
lcPDFFile = AddBS(JustPath(FILE.GetLocation())) + "UserGuide.pdf"
Response.ContentType = "application/pdf"
Response.BinaryWrite(lcPDFFile)
%>
```

See also

ContentType-Property

Body-Method

Name

`Body-Method` — returns the HTML content from the response buffer

Synopsis

Response.Body( )

Return values

Character

Parameters

Notes

Example

```
<html><body>
Let’s look at the resulting HTML generated by the AFP:
<p>
<%? Response.Body()%>
</body></html>
```

See also

Call-Method

Name

`Call-Method` — calls a new AFP document to be executed.

Synopsis

Response.Call(cLocation)

Return values

Logical

Parameters

cLocation

Notes

With this method you are able to interrupt and empty your response buffer and continue with a new AFP document to send back to the browser. The current AFP page is no longer executed. This includes any information send either to the HTTP header or cookies.

Formerly known as FOX.Call()

Example

For example in an online calendar you are able to send different responses due to the current year. This is quite useful, cause mostly the office is closed around new year.

```
<%
IF YEAR() = 2003
    Response.Call("happynewyear.afp")
ENDIF

*// Normal code execution, if NOT 2003.
%>
```

See also

Redirect-Method | Transfer-Method

Clear-Method

Name

`Clear-Method` — empties the response buffer.

Synopsis

Response.Clear( )

Return values

Logical

Parameters

Notes

Sets the response buffer back to an empty string. HTTP header and cookies are not affected.

Example

```
<html><body>
<h2>Congratulations, You won!</h2>
<%
IF lcName = "wOOdy"
    Response.Clear()
    %>
    <html><body>
    <h2>Sorry, you’re a team member...</h2>
    <%
ENDIF
```

```
%>
<p>Best wishes, your <i>AFP Team</i>
</body></html>
```

See also

Reset-Method

Clone-Method

Name

`Clone-Method` — returns an exact copy of the current file object.

Synopsis

Response.Document( )

Return values

Object

Parameters

Notes

Example

```
<%
loFile = FILE.Clone()
%>
```

See also

AddExtension-Method

CookieDate-Method

Name

`CookieDate-Method` — converts specified date to a cookie conform string.

Synopsis

Response.CookieDate(dDate)

Return values

Character

Parameters

Notes

Formerly FOX.CookieDate()

Example

Convert tomorrow’s date to a cookie-like string.

```
<%
lcCookieDate = Response.CookieDate(DATE() + 1)
%>
```

See also

Cookie-Property | AddCookie-Method | Cookies-Method | GetCookieHeader-Method

Cookies-Method

Name

`Cookies-Method` — returns a HTTP Cookie sent by the client.

Synopsis

Request.Cookies( )

Return values

Character

Parameters

Notes

Formerly FOX.GetHTTPCookie()

Example

```
<%
lcCookie = Request.Cookies()
%>
```

See also

CreateSessionID-Method

Name

`CreateSessionID-Method` — returns a new session ID, but does not replace the current one.

Synopsis

Session.CreateSessionID( )

Return values

Character

Parameters

Notes

You can use this function to create a unqique ID for other purposes, as well. With the C24_Plugin this method is identical to SYS(2015).

Example

See also

DoCmd-Method

Name

`DoCmd-Method` — executes VFP code and returns the error code

Synopsis

Server.DoCmd( )

Description

Document-Method

Name

`Document-Method` —

Synopsis

Response.Document( )

Return values

Parameters

Notes

Example

See also

Execute-Method

Name

`Execute-Method` — executes AFP code on the fly. The result is added to the response buffer.

Synopsis

Server.Execute(cAFPCode)

Notes

Joh mei...

See also

Da und dort...

Form-Method

Name

`Form-Method` — returns the content of a HTML form variable.

Synopsis

Request.Form([cName | nIndex | @aArray | 0])

Return values

Character, Numeric or Array

Parameters

cName

Specified name attribute of the correspondending input field.

nIndex

Specified index the correspondending input field.

aArray

Specified array to get all form input fields as key/value pair.

Notes

This method maybe a little bit confusing first, but is very powerful. You can achieve your form content by several ways in three different manners. The easiest call Request.Form() gives you all key/pair values as a string. Here's a list about specified parameter and excepted return values:

```
cForm   = REQUEST.Form()
cVar    = REQUEST.Form(cName)
cVar    = REQUEST.Form(nIndex)
nCount  = REQUEST.Form(@aArray)
aForm   = REQUEST.Form(0)
```

Formerly known as FOX.GetFormVar()

Example

A small exmaple for a recursive HTML formular to represent the inserted value. You can simple enchance this example by integrating IF EMPTY() statements for the form variable.

```
<%
lcSearch = Request.Form("search")
%>
<html><head></head><body>
<form action="<%? FILE.cVirtualLocation%>" method="post">
Insert your search string:<br>
<input type="text" size="10" name="search" value="<%? lcSearch%>">
<input type="submit" value=" Go ">
</form>
</body></html>
```

See also

QueryString-Method

GetCacheName-Method

Name

`GetCacheName-Method` — returns the name of the file in the %cache% without any extension.

Synopsis

File.GetCacheName( )

Return values

Character

Parameters

Notes

This name is unique for any file in a web site.

Example

```
<%
lcCacheFile = FILE.GetCacheName
? SERVER.HTMLEncode(lcCacheFile)
%>
```

See also

GetCookieHeader-Method

Name

`GetCookieHeader-Method` — returns the complete cookie information.

Synopsis

Response.GetCookieHeader( )

Return values

Character

Parameters

Notes

This methods returns a complete set of all cookie information found in the response buffer.

Example

```
<%
? Response.GetCookieHeader()
%>
```

See also

Cookie-Property | AddCookie-Method

GetLocation-Method

Name

`GetLocation-Method` — returns the actual location for the file, if one exists.

Synopsis

File.GetLocation( )

Return values

Character

Parameters

Notes

If the file exists in cLocation, this location is returned. Otherwise, cVirtualLocation is determined in the alternate directory tree (virtual director tree) as defined in the afp.config file. If the file doesn't exist there, an empty string is returned. This name is unique for any file in a web site.

Example

This method is useful to refer to relative data files below your public web directory.

```
<%
lcData = "D.\inetpub\data\customer.dbf"
IF NOT FILE(lcData)
    lcData = ADDBS(JUSTPATH(FILE.GetLocation())) + "data\customer.dbf"
ENDIF
IF NOT USED("customer")
    USE (lcData) IN 0 SHARED
ENDIF

SCAN
    ? cust_id + "<br>"
ENDSCAN
%>
```

See also

cLocation-Property
cVirtualLocation-Property

GetSessionCookie-Method

Name

`GetSessionCookie-Method` — returns a cookie string with the current session ID.

Synopsis

Session.GetSessionCookie( )

Return values

Parameters

Notes

If HTTP cookies are enabled, returns a cookie string with the current session ID. This method is automatically called when the response is send back to the client.

Example

See also

GetSessionData-Method

Name

`GetSessionData-Method` — returns session data that has previously been saved by SetSessionData().

Synopsis

Session.GetSessionData( )

Return values

Character

Parameters

Notes

If no data is available, an empty string is returned.

Example

See also

GetSessionFileName-Method

Name

`GetSessionFileName-Method` — returns a file name that you can use to store session specific data.

Synopsis

Session.GetSessionFileName( )

Return values

Parameters

Notes

cExtension must currently start with a period (.). This method returns a file name that you can use to store session specific data. These files are stored in the session directory and automatically deleted when the session times out. The ControlServer checks every 3 minutes for abandoned files and deletes them. This applies to directories, too. For directories, only the directory name must be returned by this method. Files in a directory can have any name.

This method is especially usefull to keep temporary results sets, shopping carts, and the like. However, keep in mind that these files might exist for the duration of a session (30 minutes by default). If you have many hits, you might run out of harddisk space.

Example

```
lcDir = GetSessionFileName(".appdata")
Md (lcDir)
Create Database (AddBs(lcDir)+"foo")
```

See also

Header-Method

Name

`Header-Method` — returns all additional HTTP header information.

Synopsis

Response.Header( )

Return values

Character

Notes

The method returns all information set to the HTTP header via AddHeader-Method

Example

```
<%
*// write additional HTTP-Header.
Response.AddHeader("X-Powered-By", "AFP/3.0")

*// show resulting header information
? Response.Header() && X-Powered-By: AFP/3.0
%>
```

See also

AddHeader-Method

HTMLDecode-Method

Name

`HTMLDecode-Method` — converts HTML escape codes back to normal characters.

Synopsis

Server.HTMLDecode(cValue)

Return values

Character

Parameters

cValue

Encoded string to decode with HTML conversion.

Notes

Tjoa...

Example

```
lcString = SERVER.HTMLDecode(lcFormVar)
```

See also

HTMLEncode-Method

HTMLEncode-Method

Name

`HTMLEncode-Method` — applies HTML escape codes.

Synopsis

Server.HTMLEncode(cValue)

Return values

Character

Parameters

cValue

Encoded string to decode with HTML conversion.

Notes

CHR(10) is converted to "
", all special characters are converted to their "&xxx;" counterparts. Control characters are removed, eg. CHR(7) or CHR(13). FOX.Out() has been updated to use SERVER.HTMLEncode(). The configuration table cannot be configured.

Example

```
lcQuoteString = SERVER.HTMLEncode(lcString)
```

See also

HTMLDecode-Method

HTTPCookie-Method

Name

`HTTPCookie-Method` —

Synopsis

Response.( )

Return values

Parameters

Notes

Example

See also

IsNew-Method

Name

`IsNew-Method` —

Synopsis

Session.IsNew( )

Return values

Parameters

Notes

returns .T., wenn SessionID() has created a new session, if no session has yet been created or a session timed out.

Example

See also

MakePath-Method

Name

`MakePath-Method` — returns a path.

Synopsis

Path.MakePath(cDirectory, cFile)

Return values

Character

Parameters

cDirectory

...

cFile

...

Notes

If needed, backslashes are added or souble backslashes removed. In both parts you can use any of the system variables: %root%, %common%, %cache%, %log% and %session%.

Example

See also

MapPath-Method

Name

`MapPath-Method` —

Synopsis

Server.MapPath( )

Return values

Parameters

Notes

Example

See also

MultiPart-Method

Name

`MultiPart-Method` — returns values from multipart-enctyped HTML forms

Synopsis

Reqeust.MultiPart(cName, cElement)

Return values

Character

Parameters

cName

...

cElement

Specifies the element property to return from form element cName. Currently two elements are available:

filename - transfered filename from clients browser

content-type - Content-Type header of specified filename

Notes

The MultiPart-method is useful with any multipart-enctyped HTML forms, ie. to upload files to the server.

If a specified parameter is not found the methods return an empty string.

Example

Here is an excerpt from the upload sample - safefile.afp

```
<%
Local lcFileName, lcDescription, lcFile

lcFileName = JustFName(Request.MultiPart("txtFile","filename"))

If Empty(m.lcFileName)
     Return
EndIf

lcDescription = Request.Form("edtDescription")
lcFile = Request.Form("txtFile")

Insert into Upload (cFile,cDesc,cData) Values ( ;
        m.lcFileName, ;
        m.lcDescription, ;
        m.lcFile ;
)
%>
<HTML><p>Your file has been saved. <A HREF="upload.afp">Back to list</A></HTML>
```

See also

Form-Method

NewSession-Method

Name

`NewSession-Method` —

Synopsis

Session.NewSession(uSessionID)

Return values

Parameters

uSessionID

Notes

Example

See also

QueryString-Method

Name

`QueryString-Method` — returns a variable in the query string.

Synopsis

Request.QueryString([cKey | nIndex | @aArray | 0])

Return values

Character, Numeric or Array

Parameters

cKey

Specified key attribute of the correspondending field.

nIndex

Specified index the correspondending field.

aArray

Specified array to get all query infos as key/value pair.

Notes

With this method you can handle the query string of an URL reqeusted by the browser. This is very useful if you are dealing with enhanced hyperlink definitions to use the same AFP document always the same time for (slightly) different responses but same static elements.

Formerly FOX.GetPar()

Example

This example is an extract from the official data-driven AFP website (site.afp):

```
<%
lcCategory = ALLTRIM(LOWER(Request.QueryString("cat")))
lcCategory = IIF(INLIST(lcCategory, "home", "docu", "demo"), lcCategory, "home")

SELECT * ;
    FROM category ;
    WHERE cCat == (lcCategory) ;
    INTO CURSOR _Category

IF _TALLY > 0
    SCAN
        SCATTER MEMO MEMVAR
        ? m.de + "<p>"
        GATHER MEMO MEMVAR
    ENDSCAN
ENDIF
%>
```

See also

Form-Method | ServerVariables-Method

Redirect-Method

Name

`Redirect-Method` — redirects the client's browser to another location.

Synopsis

Response.Redirect(cLocation)

Return values

Parameters

cLocation

Notes

The Redirect method generates a HTTP 302 message with the new location and sends this back to the client's browser. In consequence the browser requests the new URL automatically.

Example

Due to some re-structuring your Web site has other paths and/or filenames.

```
<%
? Response.Redirect("newpage.afp")
```

See also

Relative-Method

Name

`Relative-Method` — returns a new file object that represents a file in a position relative to the current one.

Synopsis

File.Relative(cRelativeLocation)

Return values

Object

Parameters

cRelativeLocation

Location to another file relative to the current one.

Notes

The relative location is searched in both trees - the physical and the virtual one.

Example

If you need a standard footer at the end of your page this might be done with this method.

```
<%
loFooter = FILE.Relative("includes/footer.inc.htm")
lcFooter = FileToStr(loFooter.GeLocation())
? lcFooter
%>
```

See also

Clone()

ReNew-Method

Name

`ReNew-Method` — updates the time stamp for the current session.

Synopsis

Session.Renew( )

Return values

Parameters

Notes

Example

See also

Reset-Method

Name

`Reset-Method` — loads the file object with its location properties.

Synopsis

File.Reset([cLocation] [, cVirtualLocation] [, cHost])

Return values

Parameters

cLocation

Physical location of the File object.

cVirtualLocation

Virtual location of the File object.

cHost

Hostname of the Web site to handle the object.

Notes

This method sets the properties for the physical and virtual paths. It’s actually used internal by the file methods Clone() and Relative().

Example

See also

Clone-Method | Relative-Method

Reset-Method

Name

`Reset-Method` — reloads the internal collections of the request object

Synopsis

Request.Reset( )

Return values

Parameters

Notes

The Reset-Method reloads the internal collections for the request methods Form(), QueryString(), Cookies() and ServerVariables() and the request properties cData and cForm.

Example

See also

Cookies-Method | Form-Method | QueryString-Method | ServerVariables-Method

Reset-Method

Name

`Reset-Method` — resets the response to its defaults.

Synopsis

Response.Reset( )

Return values

Parameters

Notes

Resetting the Response object includes the HTTP header with status code, cookie information, the content type and the content itself.

Example

```
<%
Response.Reset()
%>
<html><body>New content to respond...</body></html>
%>
```

See also

Clear-Method

ResolveRelative-Method

Name

`ResolveRelative-Method` — returns an absolute path that is relative to the specified base.

Synopsis

File.ResolveRelative(cBase, cRelative)

Return values

Character

Parameters

cBase

Specifies the base to start from to resolve the relative path to the destination. This can be a file or a path.

cRelative

Specifies the relative path to the destination.

Notes

For instance, FILE.ResolveRelative("/public/shop/foo.afp","../images") returns "/public/images".

Example

```
<%
lcAbsPath = FILE.ResolveRelative(FILE.GetLocation(), "../images")
%>
<img src="<%? lcAbsPath%>/afplogo.png" border="0" alt="AFP Logo">
```

See also

ServerVariables-Method

Name

`ServerVariables-Method` — returns any of the HTTP server variables.

Synopsis

Request.ServerVariables([cKey | nIndex | @aArray | 0])

Return values

Character, Numeric or Array

Parameters

cKey

Specified key attribute of a known server variable.

nIndex

Specified index inside the array of server variables.

aArray

Specified array to get all server variables as key/value pair.

Notes

This methods gives you all informations send by the web server to the AFP documents. With ServerVariables() are able to inter- and re-act with user specific informations sent by their browser everytime. Also it’s possible to get informations about the web server software itself, i.e. to handle web server specific extensions or features.

cData = REQUEST.ServerVariables() cVar = REQUEST.ServerVariables(cKey) cVar = REQUEST.ServerVariables(nIndex) nCount = REQUEST.ServerVariables(@aArray) aArray = REQUEST.ServerVariables(0)

Example

Present your website in the default language the user accepts:

```
<%
gcLang = "en"    && default language

lcLang = LOWER(LEFT(Request.Servervariables("HTTP_ACCEPT_LANGUAGE"), 2))
lcLang = IIF(INLIST(lcLang, "de", "en", "fr"), lcLang, gcLang)

DO CASE
  CASE lcLang = "de"
    ? "Guten Tag"
  CASE lcLang = "en"
    ? "Hello"
  CASE lcLang = "fr"
    ? "Bonjour"
  OTHERWISE
    ? "Undefinied" && or something else...
ENDCASE
%>
```

This example determines the information about the web server software:

```
<%
? Request.ServerVariables("SERVER_SOFTWARE") && Apache 2.0.43(Win32)
%>
```

See also

CData-Property

SessionID-Method

Name

`SessionID-Method` — returns the current session ID.

Synopsis

Session.SessionID()( )

Return values

Parameters

Notes

If needed a new session is created. The default session ID is a 44 character string that bases on GUIDs and therefore is unique. With the C24_Cookie PlugIn installed, a SYS(2015) string is returned.

Example

See also

SetSessionData-Method

Name

`SetSessionData-Method` — makes the string in cData available for future instances of this session.

Synopsis

Session.SetSessionData( )

Return values

Parameters

Notes

Technically, a ".session" file is generated in the session directory. This method is used to save variables. The result of SAVE MEMORY ALL LIKE G* is passed as the cData. You can use this function to store additional variables, or other temporary data like XML strings. cID should be a unqiue string. It shares the name space with the application IDs.

Example

See also

Timeout-Method

Name

`Timeout-Method` — returns or changes the remaining time until the session is discarded.

Synopsis

Session.Timeout([nMinutes])

Return values

Numeric

Parameters

Notes

With the Timeout-Method you are able to get the remaining time until the current session is discarded and by specifying a numeric value to set this timeout in minutes.

Currently the default value for timeout is 30 minutes.

Example

Give feedback to the user when its session will be terminated by the server.

```
This session expires in <%? SESSION.Timeout%> minutes.
```

For an online editing interface it might be useful to increase the timeout value after the user logged in successful.

```
<%
lnMinutes = 60
Session.Timeout( lnMinutes )
%>
Welcome, you are authorized.
<p>Please remember that you will be logged off automatically after
<%? lnMinutes%> minutes inactivity.
<p>Happy working!
```

See also

Transfer-Method

Name

`Transfer-Method` — transfers execution to another document after the current document terminated.

Synopsis

Server.Transfer(cFile)

Return values

Parameters

cFile

Notes

The result of the called page is added to the response buffer, effectively merging both pages together. You can chain as many pages together as you want. The called page is not executed immendiately, but

only after finishing the current one. Inside an .afp document, you can use RETURN to cancel execution. When multiple SERVER.Transfer() calls are made in a page, the last one wins.

Example

```
<%
? "Hello, this is the actually called page.<br>"
? "We are now moving to a second AFP document to response...
Server.Transfer("rest.afp")
%>
```

See also

URL-Method

Name

`URL-Method` — returns a URL string that contains the SessionID.

Synopsis

Session.URL( )

Return values

Parameters

Notes

The format depends on the session object. The default object generates:

You should replace all occurrences of ?FOX.cCookie inside a URL to this new syntax, because it allows the transition to the new unique session ID.

Example

```
Session.URL("info.afp","name=foo")

default session object: "info.afp?sid=9999999999999999&name=foo"
C24_Cookie plugin: "info.afp?!_xxxxxxxxxxxname=foo"
```

See also

URLDecode-Method

Name

`URLDecode-Method` — decodes a string that has been URL encoded.

Synopsis

Server.URLDecode(cValue)

Return values

Parameters

cValue

Encoded string to decode with URL conversion.

Notes

This means that certain characters have been replaced by a percent sign followed by their hex ASCII code. For eample, "AFP 3.0" is encoded "AFP%203.0". Passing the last string as a parameter to this method results in the first string. This method is used to decode form data or query parameters.

Example

See also

URLEncode-Method

URLEncode-Method

Name

`URLEncode-Method` — encodes a string to be used in an URL.

Synopsis

Server.URLEncode(cValue)

Return values

Parameters

cValue

String to encode with URL conversion.

Notes

Example

See also

URLDecode-Method

Version-Method

Name

`Version-Method` — returns the version information of the Active FoxPro Pages.

Synopsis

Server.Version()

Return values

Character

Parameters

Notes

Example

See also

Write-Method

Name

`Write-Method` — returns content to the browser.

Synopsis

Response.Write(uValue)

Return values

Parameters

uValue

Value to response to the browser as result of the requested URL.

Notes

The Write-Method is responsible for any content send back to the browser. In analogy to other web products the method can also be used like this: ’? uValue’ and/or ’= uValue’.

All data-types are allowed as parameter. The method converts them internal to strings.

Example

Same behaviour with the synonyms of the write method:

```
<%
```

```
Response.Write("Hello World")
? 40 + 2
= DATETIME()
%>
```

See also

Clear-Method | Reset-Method | HTMLEncode-Method

III. Appendixes

--

Anhang A. Multi-thread Restrictions of VFP

The multi-thread environment of AFP 3.0 offers a lightweight run-time library for in-process servers, many user-input commands and functions are removed. All Object syntax is still available, though events from visual classes such as forms are disabled. The following language categories of Microsoft Visual FoxPro are removed from `AFP3Engine.dll`:

- Menu, Popup and Bar commands and functions
- MESSAGEBOX() and WAIT WINDOW
- READ, @?Get/Says
- User-defined Window commands and functions

A.1. Unsupported commands in AFP multi-threaded mode

The following table is a list of unsupported commands, which will generate one of these errors running AFP in multi-threaded mode.

- Commands: Feature is not available
- Functions: Function is not implemented
- System variables: Variable is not found

Tabelle A-1. Unsupported commands in AFP multi-threaded mode

@?BOX	@?CLASS	@?CLEAR
@?EDIT	@?FILL	@?GET
@?MENU	@?PROMPT	@?SAY
@?SCROLL	@?TO	_ALIGNMENT
_ASSIST	_BEAUTIFY	_BOX
_CALCMEM	_CALCVALUE	_CONVERTER
_COVERAGE	_CUROBJ	_DBLCLICK
_DIARYDATE	_FOXDOC	_GALLERY
_GENMENU	_GENPD	_GENSCRN
_GETEXPR	_INDENT	_LMARGIN
_PADVANCE	_PBPAGE	_PCOLNO
_PCOPIES	_PDRIVER	_PDSETUP
_PECODE	_PEJECT	_PEPAGE

_PLENGTH	_PLINENO	_PLOFFSET
_PPITCH	_PQUALITY	_PSCODE
_PSPACING	_PWAIT	_RMARGIN
_RUNACTIVEDOC	_SCCTEXT	_SPELLCHK
_STARTUP	_TABS	_THROTTLE
_TRANSPORT	_WRAP	ACCEPT
ACTIVATE MENU	ACTIVATE POPUP	ACTIVATE SCREEN
ACTIVATE WINDOW	AGETCLASS()	AMOUSEOBJ()
ANSITOOEM()	ASELOBJ()	ASSERT
ASSIST	BAR()	BARCOUNT()
BARPROMPT()	BROWSE	CALL
CHANGE	CLEAR DEBUG	CLEAR GETS
CLEAR MACROS	CLEAR MENUS	CLEAR POPUPS
CLEAR PROMPT	CLEAR READ	CLOSE DEBUGGER
CLOSE FORMAT	CLOSE MEMO	CNTBAR()
CNTPAD()	COL()	CREATE
CREATE CLASS	CREATE CLASSLIB	CREATE COLOR SET
CREATE FORM	CREATE LABEL	CREATE MENU
CREATE PROJECT	CREATE QUERY	CREATE REPORT
CREATE SCREEN	DEACTIVATE MENU	DEACTIVATE POPUP
DEACTIVATE WINDOW	DEBUG	DEBUGOUT
DEFINE BAR	DEFINE BOX	DEFINE MENU
DEFINE PAD	DEFINE POPUP	DEFINE WINDOW
EDIT		FKLABEL()
FKMAX()	GETBAR()	GETCOLOR()
GETCP()	GETDIR()	GETEXPR()
GETFILE()	GETFONT()	GETPAD()
GETPICT()	GETPRINTER()	HELP
HIDE MENU	HIDE POPUP	HIDE WINDOW
IMESTATUS()	INPUT	KEYBOARD
LOAD	LOCFILE()	MCOL()
MDOWN()	MENU	MENU TO
MENU()	MESSAGEBOX()	MODIFY Commands
MOUSE	MOVE POPUP	MOVE WINDOW
MRKBAR()	MRKPAD()	MROW()
MWINDOW()	OBJNUM()	OBJVAR()
OEMTOANSI()	ON BAR()	ON ESCAPE
ON EXIT Commands	ON KEY	ON KEY LABEL
ON PAD	ON PAGE	ON READERROR

ON SELECTION BAR	ON SELECTION MENU	ON SELECTION PAD
ON SELECTION POPUP	PAD()	PLAY MACRO
POP KEY	POP MENU	POP POPUP
POPUP()	PRMBAR()	PRMPAD()
PROMPT()	PUSH KEY	PUSH MENU
PUSH POPUP	PUTFILE()	RDLEVEL()
READ	READ MENU	READKEY()
REGIONAL	RELEASE BAR	RELEASE MENUS
RELEASE PAD	RELEASE POPUPS	RELEASE WINDOWS
RESTORE MACROS	RESTORE SCREEN	RESTORE WINDOW
ROW()	SAVE MACROS	SAVE SCREEN
SAVE WINDOWS	SCROLL	SHOW GET(S)
SHOW MENU	SHOW OBJECT	SHOW POPUP
SHOW WINDOW	SIZE POPUP	SIZE WINDOW
SKPBAR()	SKPPAD()	SUSPEND
VARREAD()	WAIT	WBORDER()
WCHILD()	WCOLS()	WEXIST()
WFONT()	WLAST()	WLCOL()
WLROW()	WMAXIMUM()	WONTOP()
WOUTPUT()	WPARENT()	WREAD()
WROWS()	WTITLE()	WVISIBLE()
XMINIMUM()	ZOOM WINDOW	

A.2. Disabled commands in multi-threaded mode

The following table is a list of unsupported commands, which will generate one of these errors running AFP in multi-threaded mode. The following table is a list of unsupported commands, which will not generate an error when executed at maulti-threaded mode of AFP 3.0. These functions are still disabled for use in this mode; however, they do not cause an error. When one of these commands or functions is encountered in code, Active FoxPro Pages (VFP) ignores that line of code and continues executing. This includes certain SET commands and SYS functions.

Tabelle A-2. Disabled commands in multi-threaded mode

DOEVENTS		
SET ASSERTS	SET BELL	SET BORDER
SET BROWSEME	SET BRSTATUS	SET CONSOLE

SET COLOR	SET CLEAR	SET CLOCK
SET COVERAGE	SET CONFIRM	SET CURSOR
SET CPDIALOG	SET DEBUGOUT	SET DEBUG
SET DEVELOPMENT	SET DELIMITERS	SET DISPLAY
SET DOHISTORY	SET ESCAPE	SET ECHO
SET EVENTLIST	SET EVENTTRACKING	SET FORMAT
SET FUNCTION	SET HELP	SET INTENSITY
SET MARK OF	SET MACDESKTOP	SET MACKEY
SET MARGIN	SET MESSAGE	SET NOTIFY
SET ODOMETER	SET PALETTE	SET PDSETUP
SET READBORDER	SET REFRESH	SET RESOURCE
SET SAFETY	SET SKIP OF	SET STICKY
SET STATUS	SET SYSMENU	SET TALK
SET TRBETWEEN	SET TYPEAHEAD	SET VIEW
SET WINDOW	SYS(1037)	SYS(18)
SYS(103)	SYS(2002)	SYS(1270)
SYS(2017)	SYS(4204)	SYS(2016)

Anhang B. Lizenzbestimmungen

B.1. AFP Lizenzvertrag

Dies ist ein rechtsgültiger Vertrag zwischen Ihnen, dem Endanwender, und der ProLib Software GmbH mit Sitz in 83358 Seebruck. Bitte lesen Sie diese Bestimmungen sorgfältig durch, bevor Sie dieses Produkt verwenden. Durch Verwendung der AFP erklären Sie sich an die Bestimmungen dieses Vertrages gebunden. Wenn Sie mit den Bestimmungen dieses Vertrages nicht einverstanden sind, geben Sie bitte das Produkt zusammen mit dem Begleitmaterial (Handbücher, Behältnisse usw.) unverzüglich gegen volle Rückerstattung des Produktpreises an die Stelle zurück, von der Sie es bezogen haben.

B.2. Einräumung einer Lizenz

ProLib gewährt Ihnen das Recht, eine Kopie des beiliegenden ProLib Software Programmes (die "SOFTWARE") auf einem einzelnen Computer zu benutzen. Die SOFTWARE wird auf dem Computer "benutzt", wenn sie in den temporären Speicher (d.h. RAM) oder in einen permanenten Speicher (z.B. Festplatte, CD ROM oder eine andere Speichervorrichtung) dieses Computers installiert wird. Jedoch stellt eine Installation auf einem Netzserver für den alleinigen Zweck der Verteilung zu einem oder mehreren anderen Computern keine "Benutzung" dar, für die eine separate Lizenz notwendig ist.

B.3. Erweiterte Lizenzeinräumung

a. Jeder Betreiber einer juristisch eigenständigen Website benötigt eine Lizenz.

b. Wenn ein Provider für seine eigenen Kunden wiederum Dienste auf Basis der AFP anbietet, benötigt dies keine weitere Lizenz Beachten Sie dabei: Sie erklären sich damit einverstanden, Prolib bezüglich aller Ansprüche oder Rechtsstreitigkeiten, einschließlich der Anwaltskosten, die aufgrund des Gebrauchs oder der Verbreitung Ihres Softwareproduktes entstehen können, freizustellen, schadlos zu halten und gegen solche Ansprüche zu verteidigen. Ein von Ihnen verwendetes Produkt, das die AFP verwendet, darf nicht in Leistungskonkurrenz zur AFP selber stehen. Es ist Ihnen nicht erlaubt, AFP oder auch nur Teile davon zu verwenden, um vergleichbare Leistungsmerkmale oder Funktionalität in ein Produkt aufzunehmen, das mit AFP konkurrenziert.

B.4. Urheberrecht

Die SOFTWARE ist Eigentum von Prolib oder deren Lieferanten, und sie ist durch Urheberrechtsgesetze, internationale Verträge und andere nationale Rechtsvorschriften gegen Kopieren geschützt. Wenn die SOFTWARE nicht mit einem technischen Schutz gegen Kopieren ausgestattet ist, dürfen Sie entweder (a) eine einzige Kopie der SOFTWARE ausschließlich für Sicherungszwecke oder

Archivierungszwecke machen, oder die SOFTWARE auf eine einzige Festplatte übertragen, sofern Sie das Original ausschließlich für Sicherungs- oder Archivierungszwecke aufbewahren. Sie dürfen weder die Handbücher des Produktes noch anderes schriftliches Begleitmaterial zur SOFTWARE kopieren.

B.5. Weitere Beschränkung

Sie dürfen die Software selbst weder vermieten noch verleihen, aber Sie dürfen die Rechte an diesem ProLib-Lizenzvertrag auf Dauer an einen anderen übertragen, vorausgesetzt, daß Sie diese ProLib-Lizenznachweiskarte zusammen mit allen Kopien der SOFTWARE und dem gesamten schriftlichen Begleitmaterial übertragen und der Empfänger sich mit den Bestimmungen dieses Vertrages einverstanden erklärt. Zurückentwickeln (Reverse engineering), Dekompilieren und Entassemblieren der SOFTWARE sind nicht gestattet. Eine Übertragung muß die letzte aktualisierte Version (Update) und alle früheren Versionen umfassen.

B.6. Beschränkte Garantie

ProLib garantiert (a) für einen Zeitraum von 90 Tagen ab Empfangsdatum, daß die SOFTWARE im Wesentlichen gemäß dem begleitenden Produkthandbuch arbeitet. Diese Garantie wird von ProLib als Herstellerin des Produktes übernommen; etwaige gesetzliche Gewährleistungs- oder Haftungsansprüche gegen den Händler, von dem Sie Ihr Exemplar der SOFTWARE bezogen haben, werden hierdurch weder ersetzt noch beschränkt.

B.7. Ansprüche des Kunden

Die gesamte Haftung von ProLib und Ihr alleiniger Anspruch besteht nach Wahl von ProLib entweder (a) in der Rückerstattung des bezahlten Preises oder (b) dem Ersatz der SOFTWARE,

Glossary

[Dieses Kapitel ist Bestandteil einer Vorabausgabe und kann sich zukünftig noch ändern. Leere Abschnitte dienen als Platzhalter und werden gerade geschrieben.]

A

Active FoxPro Pages

Script engine created by ProLib Software GmbH based on Microsoft Visual FoxPro to enable HTML pages to be dynamic and interactive by embedding scripts. Since the scripts in AFP pages (suffix .afp) are processed by the server, any browser can work with AFP regardless of its support for the scripting language used therein.
Siehe auch: Visual FoxPro.

C

Cursor

D

Database

A database can be as simple as a shopping list or as complex as a collection of thousands of sounds, graphics, and related text files. Database software is designed to help users organize such information. While early "flat" databases were limited to simple, searchable rows and columns, modern relational databases allow users to access and reorganize data in a variety of ways. Even more advanced databases let users store and retrieve all kinds of nonstandard data, from sound clips to video.

E

Extensible Markup Language

XML is the Extensible Markup Language, a system for defining specialized markup languages that are used to transmit formatted data. XML is conceptually related to HTML, but XML is not itself a markup language. Rather it’s a metalanguage, a language used to create other specialized languages.

F

File Transfer Protocol

This Internet protocol is used to copy files between computers--usually a client and an archive site. It’s old-fashioned, it’s a bit on the slow side, it doesn’t support compression, and it uses cryptic Unix command parameters. But the good news is that you can download shareware or freeware apps that shield you from the complexities of Unix, and you can connect to FTP sites using a Web browser.

H

Hypertext Markup Language

As its name suggests, HTML is a collection of formatting commands that create hypertext documents--Web pages, to be exact. When you point your Web browser to a URL, the browser interprets the HTML commands embedded in the page and uses them to format the page’s text and graphic elements. HTML commands cover many types of text formatting (bold and italic text, lists, headline fonts in various sizes, and so on), and also have the ability to include graphics and other nontext elements.

Hypertext Transfer Protocol

The protocol used to transmit and receive all data over the World Wide Web. When you type a URL into your browser, you’re actually sending an HTTP request to a Web server for a page of information (that’s why URLs all begin with "http://"). HTTP1.1, the latest version, is currently undergoing revisions to make it work more efficiently with TCP/IP

I

Internet Information Services

Internet Information Server, Web-server, part of the Windows NT Server, supports ISAPI.

IP

Internet Protocol, protocol in the TCP/IP internet layer for communication between nets and their hosts. An IP-address uniquely identifies each network and each of its hosts on the internet. Addresses consist of four bytes that can be represented by four integers (0 to 255) seperated by dots, e.g. "157.189.162.75". Dependent on the address' class and a subnet mask, a specific number of bits identify the net or subnet and the rest a PC's address within this net/subnet.

ISAPI

Internet Information Server API, API supported by the MS's Internet Information Server (IIS).

R

Request

Response

Rushmore

T

Thread

Microsoft Visual FoxPro is a 4G programming language with integrated database functionality.

V

Visual FoxPro

Microsoft Visual FoxPro is a 4G programming language with integrated database functionality.

W

World Wide Web

Also known as the WWW, the W3, or most often simply as the Web, it originally developed by CERN labs in Geneva, Switzerland. Continuing development of the Web is overseen by the World Wide Web Consortium. The Web can be described (dryly) as a client/server hypertext system for retrieving information across the Internet. On the Web, everything is represented as hypertext (in HTML format) and is linked to other documents by their URLs. The Web encompasses its native http protocol, as well as ftp, Gopher, and Telnet. The best way to learn about it, however, is to try it for yourself.

Stichwortverzeichnis