Willkommen in der Hilfe zu AT HTML Editor 32.3 !

Dieses kleine Tool ist die 32-bit-Umsetzung der 16-Bit-Version von AT HTML Editor. Die 16-Bit-Version wurde entwickelt, um auch auf leistungsschwächeren bzw. älteren Windows-Systemen mit etwas mehr Komfort wie das Windows-Notepad es bietet, HTML-Seiten zu erstellen. Diese Version hatte und hat immer noch einen hohen Beliebtheitsgrad, was nicht zuletzt auch die immer noch steigenden Downloadraten zeigen. So war die logische Folge, diese nun schon die im 3. Update vorliegende 32-bit-Version nachzuschieben.

Was ist neu?
In dieser vorliegenden 3. Version hat sich enorm vieles getan. So sind fast alle Benutzerwünsche berücksichtigt worden (z.B. Standardsyntax einstellen, zuletzt geöffnetes Projekt und Dateien wieder laden, Anbindung der Möglichkeit, JSP- und ASP-Seiten zu entwickeln und per Vorschau anzeigen zu lassen uvm.) Auch sind weitestgehend die Fehler der vorherigen Version beseitigt worden.
Aber es ist auch eine Menge Neues enthalten. Zu erwähnen sind da vor allem die neue und erweiterte Anbindung von Programmen, die Plugin-Schnittstelle für DLL-Plugins und der Sitzungs-Manager. Auch viele kleine Änderungen, die nicht auf den ersten Blick sichtbar, aber wirklich praktikabel sind, machen das Codierer-Leben leichter - z.B. der Code-Browser, die erweiterte Navigation zwischen den Code-Editoren oder die Möglichkeit der einfachen Anordnung der Code-Editoren. Und es gibt auch etwas für die ganz Mutigen: den Makrorecorder für Code-Makros. Auch wurde er den Bedürfnissen im unternehmensinternen Einsatz weiter angepaßt. Aber nach wie vor sind auch die Schmankerl geblieben, die die Arbeit oft entscheidend erleichtern: u.a. WYSIWYG-Editor, Bildviewer, Farben ermitteln etc. Dies ist nicht zuletzt auch den drastisch steigenden Downloadraten und dem wirklich großen Benutzerfeedback geschuldet.
Weitere Infos dazu unter **"Was ist neu?"**.

Diese Hilfe ist kontextsensitiv. Das heißt, daß in ein Programmteil geklickt und die entsprechende Hilfe mit der Taste F1 aufgerufen werden kann.

Also dann, have a lot of fun !

E-mail: **ansatheus@gmx.net**

Internet: **http://www.ansatheus.de**
Forum | **Support**

AT Contenator
http://home.arcor.de/ansatheus

Aufbau

AT HTML Editor 32 ist eine Single-Document-Anwendung. Das bedeutet, jedes Fenster kann beliebig hin- und hergeschoben, in der Größe verändert, aus- und eingeblendet und geschlossen werden. Die Ausnahme bildet der zentrale Projektexplorer, der das Dateisystem, Projekte, Tools, Einstellungen und alle Code-Editoren zugänglich macht. Durch diesen modularen Aufbau können jederzeit neue Komponenten eingebunden werden, ohne das andere (und funktionierende) davon wesentlich berührt werden.

Die zentrale Einheit ist der Projektexplorer. Der Code-Editor greift auf deren Funktionen zu 100 % zu. Der WYSIWYG-Editor sowie der Struktur-Editor und der Syntax-Editor arbeiten im Gegensatz dazu völlig unabhängig und laufen als separate Programme. Alle zu den einzelnen Editoren gehörenden Funktionen arbeiten ausschließlich nur mit dem jeweiligen Editor zusammen. Ganz beonders betrifft dies den Code-Editor. Er kann beliebig instanziiert werden, d.h. es können beliebig viele Code-Editor-Fenster geöffnet und in ihnen gearbeitet werden. Jedes Fenster bildet mit all seinen Funktionen und Bearbeitungsständen eine in sich geschlossene Einheit (im Gegensatz dazu eine Multi-Document-Anwendung, in der eine zentrale Funktionseinheit für alle Editorfenster zuständig ist - z.B. ältere MS-Word-Versionen). Einer der geschätzten Vorteile ist der, das die Arbeitsfläche so groß wie möglich gestaltet werden kann (gerade beim Codieren ist das ein nicht zu unterschätzender Vorteil). Ein anderer ist die flexible Speicherung der Arbeitszustände (in einem Fenster Code, im nächsten Code mit Tags und Farbpalette, im nächsten den Bildviewer, im nächsten das Internet usw.).

Als wahres Sahnestück hat sich der WYSIWYG-Editor entpuppt. Ich erwische mich immer öfter dabei, mit ihm zu arbeiten. Auch diese Hilfe wurde mit diesem Teil nachbearbeitet. Er ist gedacht, visuell Korrekturen an im Code-Editor entwickelten HTML-Seiten vornehmen zu können.

Die Funktion "Farbe holen" hat sich ebenfalls als ganz praktisch und häufig genutzt erwiesen. Mit ihr können desktopweit Farben ermittelt und in den Code-Editor eingefügt werden (für farbenfrohe Freunde der HTML-Entwicklergemeinschaft).

Das gesamte Programm ist auf schnelle Bedienbarkeit optimiert. Icons und sonstige bunte Bildchen zum Raufklicken suchte man in den ersten beiden Versionen zwar vergebens, aber mittlerweile wurden ab Version 3 übliche Symbolleisten auf vielfachen Wunsch integriert. Dennoch sind alle Funktionen nach wie vor per Tastenkombination und/oder Popupmenü erreichbar. Einige Funktionen sind ausschließlich nur so nutzbar.

Um dem Benutzer die Möglichkeit zu geben, die Funktionalität von AT HTML Editor 32 zu erweitern, wurde in der vorliegenden Version eine neue Anbindung von externen Programmen, ein DLL-Interfaces sowie ein Makrorecorder für Code-Makros und weitere Codelisten integriert.

Obwohl doch etliche Mühe auf die fehlerfreie Entwicklung des AT HTML Editors 32 verwandt wurde, ist auch dieses Programm mit Sicherheit nicht frei von Fehlern. Wenn Ihr Spaß damit habt und Fehler sichtbar werden, dann wäre ein Feedback an mich ganz nett.

Der Autor

Tätigkeitsfelder

- Software-Entwicklung vorrangig im Bereich Visual Basic und Delphi
- Datenbank-Entwicklung im Bereich Access und MySQL
- Web-Development vorrangig im Bereich PHP/SQL/HTML/JavaScript
- diverse andere Bereiche und Berührungspunkte (MS-Office-Programmiereung, CATIA V5-Programmierung, Perl, Java, Tcl/TK, PureBasic, RapidQ etc.)

Referenzen

- **http://home.arcor.de/ansatheus**
- **http://www.ansatheus.de**
- **http://contenator.ansatheus.de**

Haftungsauschluß

Es wird keine Haftung in irgendeiner Form für irgendwelche Schäden übernommen, die angeblich von der Nutzung dieses Programmes herrühren sollen. Auch eine Gewährleistungspflicht für AT HTML Editor 32.x ist jederzeit ausgeschlossen. Aber dieses Programm darf frei kopiert und weitergegeben werden. Kommerzielle Nutzung und Vermarktung ist untersagt (es sei denn, der Autor wurde gefragt und hat zugestimmt - einfach melden bei mir).

Es steht dem Autor frei, jederzeit Änderungen in irgendeiner Form an AT HTML Editor 32.x ohne vorherige Ankündigung vorzunehmen. Auch kann das Programm um Funktionalitäten ergänzt werden, die nicht in dieser Hilfe dokumentiert sein müssen. Auch wird keine Gewährleistung für die Vollständigkeit dieser Hilfe übernommen, obwohl sie gewissenhaft zusammengestellt wurde. Im übrigen wird hier auf weitere Hilfen verwiesen, die unter **http://www.ansatheus.de** zur Verfügung stehen.

Was ist neu?

AT HTML Editor 32.3 ist komplett runderneuert. Die meisten Änderungen und Neuerungen befinden sich unter der "Motorhaube". Einiges ist dennoch sichtbar. Vieles ist auf das Feedback und den Wünschen der vielen, vielen Benutzer zurückzuführen. Deshalb an dieser Stelle ein dicker **Dank** an alle, die AT HTML Editor 32 benutzen, weiterbenutzen oder noch benutzen werden!

In der nachfolgenden Liste sind die meisten und wichtigsten Änderungen und Neuerungen aufgelistet.

Neue Funktionalitäten

- DLL-Plugin-Schnittstelle (Interface) für eigene DLLs
- Programmliste für beliebige Programme
- Sitzungs-Manager
- Option "Letzte Sitzung beim Programmstart öffnen"
- Code-Browser, auch für benutzerdefinierte Begriffe
- Suchen und Ersetzen-Funktionen
- kontextsensitive Hilfe
- Option "Neuen Editor beim Programmstart laden"
- Option "Andere Editorkomponente einsetzen"
- Liste von Dateiarten, die geöffnet werden können
- festlegen einer Standardsyntax - auch für benutzerdefinierte Syntax
- Option "Programm mit Datei im Code-Editor starten"
- Icons für schnelle Fensterpositionierungen
- Run-Modus für benutzerdefinierte Syntax und Hostadresse (z.B. für ASP, JSP)
- Größenveränderungen von Ausgabekonsole und Code-Browser
- Projektnotizen
- Projekt um Strukturdateien erweitert
- Navigationspfeile in den Code-Browsern
- benutzerdefinierte Code-Listen im Code-Manager
- Schnellbuttons für PHP-Kontrollstrukturen im Code-Manager
- Makrorekorder für Code-Makros im Code-Manager
- Wechselbutton für benutzerdefinierte Syntax im Syntax-Manager
- neue Funktionen im Vollbildmodus

Änderungen

- Herauslösung von Editorprogrammen wie WYSIWYG-Editor, Struktur-Editor als eigenständige Programme
- verschiedene Icons anders platziert
- Register "Programmstarter" in "Diverses" verändert mit zusätzlichen Funktionen
- Schnellbuttons für HTML-Tags überarbeitet und erweitert im Code-Manager
- Überarbeitungen im Design
- Handling der Manager verbessert und erweitert

Behobene Fehler

- Runtime-Error 5, wenn von einem anderen Register als "Code" in die Vorschau gewechselt wurde
- Probleme bei Speicherung des Dateinamens, wenn sich der SWR3-OnlineCounter in die Titelleiste einklinkte
- "Eintrag hinzufügen / ändern" bei benutzerdefinierten Compilern/Interpretern
- Probleme bei der Aktivierung von Code-Editoren unter bestimmten Umständen
- fehlerhafte Breitenanpassung des Editorfensters, wenn Projektexplorer nicht am linken Bildschirmrand, aber noch in linker Bildschirmhälfte war
- Registrierungs-Batchdatei u.a. für Runtime-Error 372 beigelegt
- fehlerhafte Funktion einiger Schnellbuttons für HTML-Tags im Code-Manager
- fehlerhafte Positionierungen, wenn die Windows-Taskleiste an anderer Position ist oder andere Maße hat
- Absturz beim Aufräumen bei Programmende

Nichtänderbare Fehler

- Symbolanzeige des Dateimanagers im Register "Dateien" des Projektexplorers - leider herstellerabhängig

Projektexplorer

Der Projektexplorer ist die zentrale Steuerungseinheit von AT HTML Editor 32, ohne der nichts geht. Er sit in der Größe veränderbar und beliebig auf dem Bildschirm positionierbar. In Abhängigkeit von der Position werden Code-Editoren (sofern sie nicht mittig gestartet werden), rechts oder links neben dem Projektexplorer bzw. rechts oben, wenn der Projektexplorer als Vollbild angeigt wird, angeordnet. Das Ganze funktioniert auch in Abhängigkeit von der Position und Größe der Windows-Taskleiste.

Grundkomponenten des Projektexplorers

Das Menü. Einige Menüpunkte werden auch vom Code-Editor aus bedient (z.B. Neues-Codeeditorfenster oder Sitzungs-Manager). Das Projekt-Menü kann im Register "Projekte" per Rechtsklick auch als Kontextmenü aufgerufen werden.

Alle verfügbaren Register.

Damit jederzeit eine Information über geöffnete Dokumente vorhanden ist und auch zu ihnen gewechselt werden kann, werden diese in dieser Liste angezeigt. Per Klick auf ein Dokument genügt, um es wieder in den Vordergrund zu holen. Und um auch einem bißchen Statistik gerecht zu werden, eine Information über die bisherige Gesamtanzahl von geöffneten Dokumenten. Auch bereits geschlossene werden mitgezählt - manchmal erstaunlich, wieviel man so manches mal bearbeitet.
Zudem greift die Navigation in den Code-Editoren darauf zu. Desweiteren werden u.a. auch hierüber Dateien in ein Projekt geladen.

Menü "Datei"
Menü "Projekte"
Menü "Hilfe"
Register "Dateien"
Register "Projekte"
Register "Config"
Register "Run"
Register "Plugins"

Menü "Datei"

Dieses Menü erlaubt Zugriffe auf Funktionalitäten, die mit dem Dateihandling zu tun haben.

Sitzungs-Manager - Zuletzt geöffnete Dateien und Projekte ...	Öffnet den Sitzungs-Manager.
Neues Code-Editorfenster ...	Öffnet einen neuen leeren Code-Editor.
Alle Code-Editorfenster speichern	Speichert alle geöffneten Code-Editoren.
Alle Code-Editorfenster schließen	Schließt alle geöffneten Code-Editoren.
Verzeichniswurzel setzen ...	Setzt ein neues Ausgangsverzeichnis. Dabei wird das Ausgangsverzeichnis die neue Verzeichniswurzel, ab der nur alle Verzeichnisse und Dateien unterhalb sichtbar sind. Die oberen werden ausgeblendet.
Verzeichnisbaum aktualisieren	Da sich die Dateikomponente nicht automatisch aktualisiert sobald eine neue Datei hinzukommt, muß dies hiermit manuell erfolgen. Dabei wird auch die Verzeichniswurzel (leider) auf den Ausgangszustand gesetzt.
Beenden	Beendet AT HTML Editor 32.

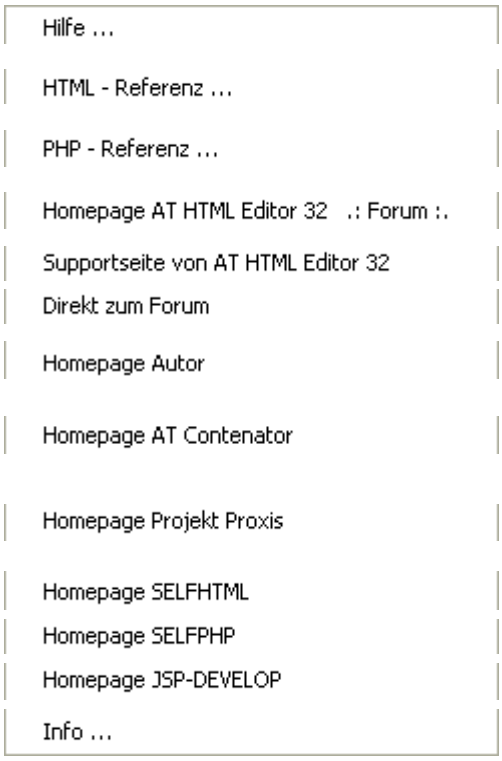
Menü "Projekte"

In diesem Menü passiert das Projekthandling von Dateien. Im Register "Projekte" kann dieses per Rechtsklick in das Projektfenster als Kontextmenü aufgerufen werden.

Neu	Legt ein neues Projekt an.
Öffnen ...	Öffnet ein vorhandenes Projekt.
Speichern	Speichert alle Änderungen an einem Projekt.
Speichern als ...	Speichert ein Projekt unter einem Namen ab.
Gruppe hinzufügen	Zeigt den Gruppen-Editor an, der auf dem Projektnamen erscheint.
In der Liste markierte Datei hinzufügen	Fügt eine in der Dateiliste markierte Datei einer markierten Projektgruppe hinzu. **Vorgehensweise** 1. Datei in der Dateiliste markieren 2. Projektgruppe im Projekt markieren 3. auf diesen Menüpunkt klicken
Dateien hinzufügen ...	Um vorhandene Dateien in ein Projekt zu integrieren, kann dieser Menüpunkt genutzt werden. Dabei ist eine Mehrfachauswahl, also gleichzeitiges Einfügen mehrerer Dateien, möglich.
Laufwerk/Verzeichnis/Netzwerkpfad hinzufügen ...	Das ist die Projektfunktion für "faule Leute" :-). Hiermit läßt sich jeweils immer ein Verzeichnis zum Projekt hinzufügen. Es ist dafür keine Projektgruppe erforderlich. Wird dieses Verzeichnis ausgewählt, erscheint der windowsübliche Öffnen-Dialog, aus der dann die zu bearbeiende Datei ausgewählt und geladen werden kann.
Strukturdatei hinzufügen ...	Fügt eine Strukturdatei zum Projekt hinzu. Diese wird automatisch im Unterverzeichnis "projekte" angelegt und erhält den Namen des Projektes aber mit der Endung .str. Gleichzeitig wird der Struktur-Editor mit ihr geladen. Es lassen sich auch Untereinträge zu einer Strukturdatei hinzufügen.
Markierten Eintrag aus dem Projekt entfernen	Entfernt einen markierten Eintrag aus dem Projekt.
✔ Projektnotizen anzeigen	Blendet die Projektnotizen ein oder aus. Die Einstellung wird dauerhaft gespeichert. Zudem kann die Größe proportional zum Projektfenster verändert werden. Der Inhalt wird automatisch im Unterverzeichnis "projekte" gespeichert. Der Dateiname entspricht dem Namen des Projektes, aber mit der Endung .ntz. Einfach, kleine Projektnotizen sind oft nicht zu unterschätzen. Insbesondere kleine ToDo-Listen lassen sich damit schnell und unkompliziert realisieren.

Menü "Hilfe"

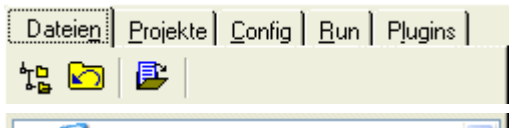
Damit man gut mit AT HTML Editor 32 arbeiten und seine Webseiten codieren kann, gibt es hier neben dieser Hilfe auch einiges an zusätzlicher Hilfe.



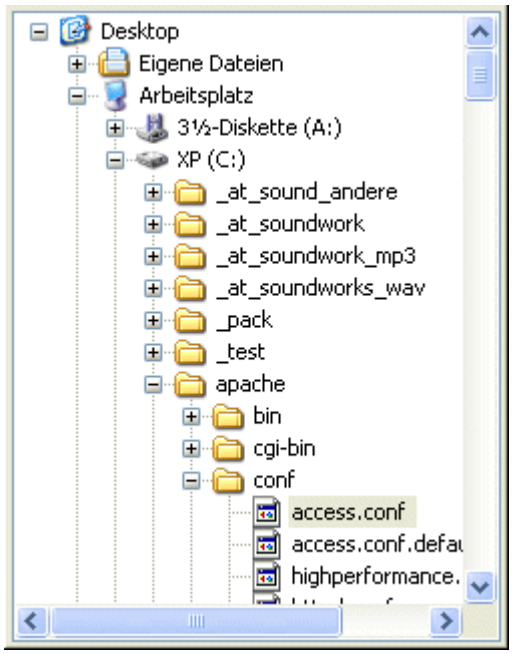
Hilfe ...	Zeigt diese Hilfe zu AT HTML Editor 32 an. Sie ist "etwas" kontextsensitiv - mit F1 läßt sich zu den Hauptelementen die entsprechende Hilfeseite laden.
HTML - Referenz ...	Lädt eine kleine HTML-Referenz, die dem Programm beiliegt, in der die Grundlagen von HTML einfach und an Beispielen vermittelt wird.
PHP - Referenz ...	Lädt die dem Programm beiliegende PHP-Referenz von SELFPHP. So hat man auch hier schnellen Zugriff auf entsprechende Hilfe und Beispiele.
Homepage AT HTML Editor 32 .: Forum :.	Wechselt zur Homepage von AT HTML Editor 32, über die auch auf das Forum zugegriffen werden kann.
Supportseite von AT HTML Editor 32	Wechselt zur Supportseite von AT HTML Editor 32.
Direkt zum Forum	Wechselt direkt zum Forum von AT HTML Editor 32.
Homepage Autor	Wechselt zur Homepage des Autors [also meine :-)], mit Personenprofil, Arbeitsmuster und Referenzen.
Homepage AT Contenator	Wechselt zur Homepage von AT Contenator, einem PHP-basierendem System zur automatischen Generierung von Websites, Dokumentationen, Tutorials uvm..
Homepage Projekt Proxis	Wechselt zur Homepage von Projekt Proxis, einem kleinen Portal, in der einige Leute Freeware zum Download und Referenzen zur Ansicht bereitgestellt haben.
Homepage SELFHTML	Wechselt zur Homepage von SELFHTML.
Homepage SELFPHP	Wechselt zur Homepage von SELFPHP.
Homepage JSP-DEVELOP	Wechselt zur Homepage von JSP-DEVELOP
Info ...	Lädt die Programminformation.

Register "Dateien"

Hier erfolgt der Zugriff auf das gesamte Dateisystem. Dabei werden Laufwerke, Verzeichnisse und Dateien gleichermaßen in einem Dateibaum und nicht getrennt angezeigt. Das bietet gewisse Vorteile. Die Verzeichniswurzel kann aber auch individuell eingestellt werden, quasi als Snapshot. Dann werden alle Dateien und Verzeichnisse unterhalb dieser Verzeichniswurzel angezeigt und alles, was darüber ist, ausgeblendet.



Iconleiste, die oft genutzte Funktionen schneller zugänglich macht.

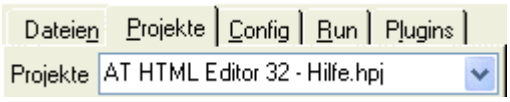


Ein typischer Dateibaum mit Laufwerke, Verzeichnisse und Dateien.

Register "Projekte"

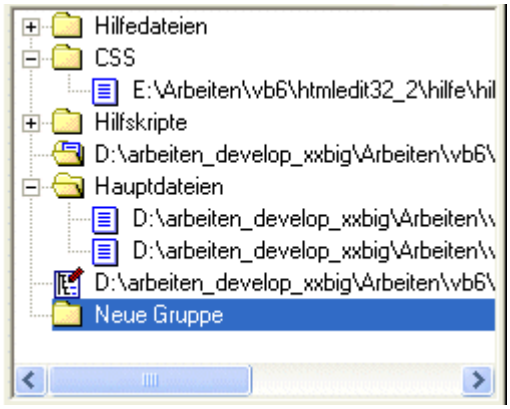
Hier erfolgt der Zugriff und die Verwaltung auf Projekte und Projektdateien.

Mögliche Projektelemente
- Projektgruppen
- Dateien in den Projektgruppen
- Laufwerke, Verzeichnisse und Netzwerkpfade
- Strukturdateien

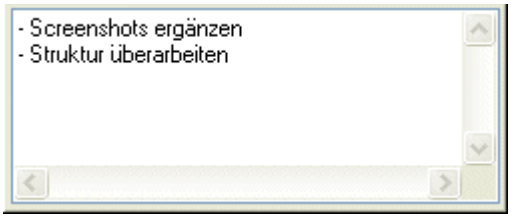


Lädt ein Projekt direkt aus der Auswahlliste. Dies funktioniert aber nur, wenn im Register "Config" unter "Projektordner" ein ebensolcher eingestellt ist. Dadurch wird ein schnelles wechseln zwischen verschiedenen Projekten ermöglicht.

Über diesen kleinen Gruppen-Editor werden Projektgruppen angezeigt. Sichtbar wird er direkt auf Pfad und Name des Projektes.



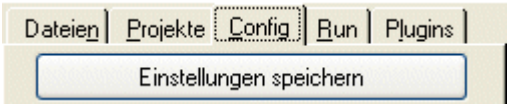
Hier ist eine beispielhafte Projektstruktur sichtbar. Es sind alle Projektelementarten enthalten. Die Strukturdatei wird automatisch im Unterverzeichnis "projekte" mit dem Projektnamen und der Endung .str angelegt, wenn sie hinzugefügt wird.



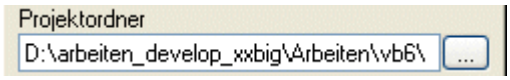
Hier können ganz schnell einfache Notizen zum Projekt festgehalten werden. Die Notizen werden ebenfalls im Unterverzeichnis "projekte" mit dem Projektnamen und der Endung .ntz angelegt. Die Größe kann proportional zum Projektfenster geändert werden.
Die Projektnotizen können dauerhaft ein- und ausgeblendet werden.

Register "Config"

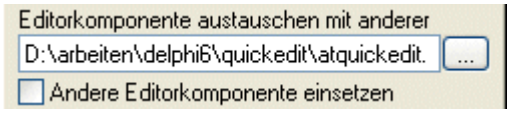
Hier werden verschiedene allgemeine Programmeinstellungen gemacht, die das Verhalten von AT HTML Editor 32 beeinflussen.



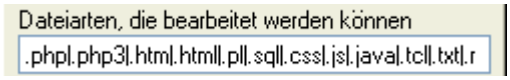
Hiermit werden alle Einstellungen in den Registern "Config" und "Run" gespeichert.



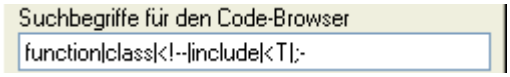
Ist hier ein Projektordner angegeben, kann im Register 'Projekt' zwischen verschiedenen Projekten wechseln.



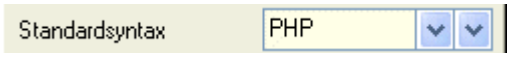
Ist hier eine alternative (selbstprogrammierte) Editorkomponente angegeben, wird jede Datei aus dem Dateimanager und Projektmanager mit dieser gestartet, sofern das Häkchen gesetzt ist. Das funktioniert natürlich auch mit anderen Editoren, sofern diese mit einer Datei gestartet werden können. Alle so geladenen Dateien werden nicht in die Dateiliste übernommen.



Alle Dateiarten, die angezeigt werden sollen, werden hier eingestellt. Das Format muß unbedingt beachtet werden:
.Dateiendung|.Dateiendung|usw. -> Bsp.: **.php|.html|.pl**
Hinter der letzten Dateiendung ist kein "|" erforderlich. Ohne "." davor funktioniert es genauso. Es werden Dateiendungen bis maximal 6 Zeichen unterstützt.



Hier werden die einzelnen Suchbegriffe für den Code-Browser ebenfalls mittels "|" getrennt. Durch diese Flexibilität können beliebig viele Suchbegriffe bis maximal 12 Zeichen definiert werden, die dann im Code-Browser angezeigt werden. Zu diesen kann dann ganz bequem im Code-Editor per Klick gesprungen werden.



Hiermit kann jeder Code-Editor beim Laden mit der hier eingestellten gewünschten Syntax versehen werden. Die erste Liste zeigt die in AT HTML Editor 32 integrierten Syntaxmodi an und die zweite Liste alle benutzerdefinierten Syntaxmodi.
Ist keine Standardsyntax eingestellt, ist HTML die Standardsyntax.

Ist hier eine Startdatei definiert, kann im Register "Vorschau" des Code-Editors die gesamte, momentan bearbeitete Website gestartet werden.

Damit der Sitzungs-Manager nicht so nach und nach immer größeren Ballast mit sich schleppt, kann hier die Anzahl der maximalen Einträge begrenzt werden.

Gibt die Spaltenanzahl für die Dateiliste an. Je mehr Spalten, desto mehr Dateien sind in der Dateiliste gleichzeitig sichtbar.

Ist das Häckchen gesetzt, wird die komplette letzte Sitzung inklusive des geöffneten Projektes sowie aller bis dahin **geöffneten** Dateien beim nächsten Start von AT HTML Editor 32 geöffnet. Sind zusätzlich Dateien im Dateispeicher des Sitzungs-Managers abgelegt und danach wieder geschlossen worden, werden auch diese ebenfalls geöffnet.

Ist das Häkchen gesetzt, wird jeder Code-Editor unabhängig von der Position des Projektexplorers in der Bildschirmmitte geladen. Ansonsten ordnet sich jeder Code-Editor in Abhängigkeit von der Position des Projektexplorer entweder rechts oder links (wenn sich der Projektexplorer z.B. am rechten Bildschrimrand befindet). Das Ganze funktioniert auch in Abhängigkeit von Position und Größe der Windows-Taskleiste.

Ist das Häkchen gesetzt, wird beim Start von AT HTML Editor 32 ein neuer Code-Editor geladen. Arbeitet man an einem Projekt, daß sich jedesmal wieder laden soll, ist es nützlich, das Häkchen nicht zu setzen.

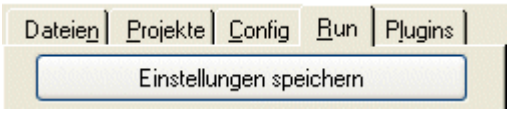
Ganz praktisch, wenn man nicht jedesmal auf 'Speichern' klicken möchte. Insbesondere bei PHP-Dateien hat sich das als ganz nützlich erwiesen, da diese ja erst interpretiert werden, wenn sie gespeichert sind.

Register "Run"

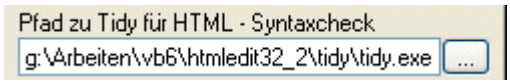
Hier werden alle Compiler-/Interpreter-Anbindungen an AT HTML Editor 32 gemacht. Die jeweiligen Überschriften über den Feldern zeigen den Zweck eines jeden Feldes an. Der 'Run'-Modus der Code-Editoren greift dann gezielt darauf zu, wenn des jeweilige Syntaxhighlighting dafür eingeschaltet ist. Das gilt auch für alle benutzerdefinierte Syntax-Modi.

Compiler/Interpreter haben in der Regel sogenannte Schalter, die man bei der Compilierung/Interpretierung setzen kann. Diese können ebenfalls wie in der DOS-Kommandozeile eingegeben werden. So kann bei PHP die Ausgabe eines Infotextes mit dem Schalter -q unterdrückt werden. Also heißt der Eintrag im Feld 'Pfad zum PHP - Interpreter' c:\php\php.exe -q. Gleiches gilt für Tidy, PureBasic, Rapid-Q und den benutzerdefinierten Compilern/Interpretern. Man sollte in der jeweiligen Hilfe nachschauen, wie die einzelnen Schalter heißen und wie sie eingesetzt werden.

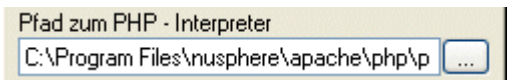
Die Run-Modi unterscheiden automatisch nach folgenden Syntaxmodi:
HTM, HTML, PHP, PureBasic (.pb), Rapid-Q (.rq) und benutzerdefinierten Compilern/Interpretern. Das heißt: Wird der Run-Modus ausgelöst, schaut AT HTML Editor 32 sich die Dateiendung an. Ist es eine HTM- oder HTML-Datei, wird Tidy ausgeführt. Ist es eine PHP-Datei wird in Abhängigkeit der Run-Option im Register nur die Codesyntax überprüft oder die Seite in der Vorschau ausgeführt. Ist es eine PB-Datei, wird sie mit PureBasic compiliert und ausgeführt. Ist es eine RQ-Datei, passiert dasselbe mit dem Rapid-Q-Compiler. Ist die Datei keine von den erwähnten, wird nachgeschaut, ob ein benutzerdefinierter Compiler/Interpreter eingestellt ist. Ist das der Fall, wird dieser mit der Datei ausgeführt. Das funktioniert dann mit jeder benutzerdefinierten Syntax.



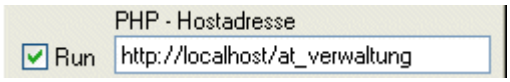
Hiermit werden alle Einstellungen in den Registern "Config" und "Run" gespeichert.



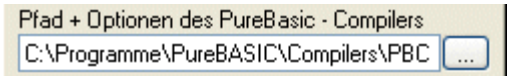
Bindet den HTML-Syntaxchecker "Tidy" an. AT HTML Editor 32 liefert eine etwas ältere, aber dafür deutsche Version mit.



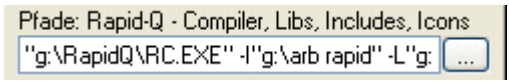
Der Pfad zum PHP-Interpreter wird hierüber eingestellt. Dies ist dann nützlich, wenn lokal ein Syntaxcheck durchgeführt werden soll. Die Ausgabe erfolgt dann im Debugfenster des Code-Editors.



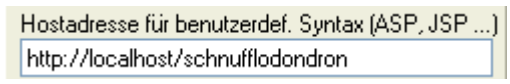
Ist das Häkchen bei "Run" gesetzt und die Hostadresse eingetragen, wird die bearbeitete PHP-Datei im Register "Vorschau" des Code-Editors wie im Browser aufgerufen. So kann sofort das Ergebnis auch "online" überprüft werden. Auch die Fehlermeldungen werden dann in der Vorschau angezeigt.



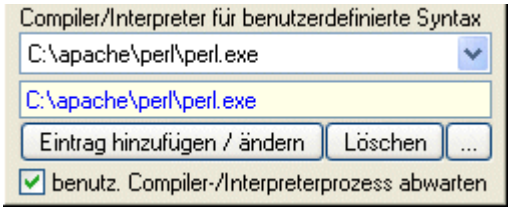
Will man PureBasic-Programme mit AT HTML Editor 32 kompilieren, kann hier der Pfad zum PureBasic-Compiler sowie seine Optionen eingestellt werden. Die Compilerausgaben erfolgen dann ebenfalls im Debugfenster des Code-Editors.



Gleiches wie bei PureBasic gilt auch hier für Rapid-Q. Der Unterschied ist jedoch in der Angabe Compileroptionen und Pfade. Diese müssen teilweise in Anführungszeichen gesetzt werden.
Rapid-Q wird nicht mehr weiterentwickelt. Es ist aber nach wie vor ein beliebtes Basic-system mit beachtlichem Funktionsumfang. Interessant ist sicher die Möglichkeit, mittels Rapid-Q-Programmen auf MySQL-Datenbanken zugreifen zu können.



Wenn jemand z.B. ASP (Active Server Pages)- oder JSP (Java Server Pages)-Seiten mit AT HTML Editor entwickeln und in der Vorschau ausführen möchte, kann er hier die entsprechende Hostadresse einstellen. Im Code-Editor steht dann dafür ein entsprechendes Run-Icon zur Verfügung.



Alle Felder in diesem Bereich dienen zur Anbindung und Bearbeitung von beliebigen Compilern/Interpretern. Will z.B. jemand JSP-Seiten entwickelt, kann hierüber der Java-Compiler angebunden werden. Per Run-Icon im Code-Editor kann dann der Compiler aufgerufen und die Java-Klasse erzeugt werden. Die Fehlerausgabe erfolgt dann im Debugfenster des Code-Editors. Per Button "..." kann ein neuer Compiler/Interpreter gesucht und in das gelbe Bearbeitungsfeld geladen werden. In diesem Feld können eventuell nutzbare Schalter hinzugefügt werden (siehe oben). Mit dem Button "Eintrag hinzufügen / ändern" wird der Eintrag im gelben Feld der Liste hinzugefügt bzw. wenn vorhanden, geändert. Mit dem Button "Löschen" wird ein in der Liste markierter Eintrag gelöscht.

Wenn die Anbindung des Compilers/Interpreters nicht richtig funktioniert oder AT HTML Editor 32 scheinbar abgestürzt zu sein scheint, hilft die Option "benutz. Compiler-/Interpreterprozess abwarten" weiter. Es kann unter Umständen sein, daß ein Compiler-/Interpreterprozeß erst beendet werden muß, bevor Windows die weitere Abarbeitung anderer Programme zuläßt. Ist hier ein Häkchen gesetzt, wartet AT HTML Editor 32 diesen Prozeß ab, bevor er weiter arbeitet.

Register "Plugins"

Die in diesem Register enthaltenen Funktionalitäten erweitern die Handhabbarkeit und Funktionalität von AT HTML Editor 32 in beliebigem Maße. Hier erfolgt eine Unterteilung in Programme und Plugins, um dem Benutzer Flexibilität zu verleihen, was die Anbindung von Programmen und Plugins betrifft.

Unterteilung in die Register "Programme" (ausführbare EXE-Dateien) und "Plugins" (DLLs - Delphi, C++, aber keine ActiveX).

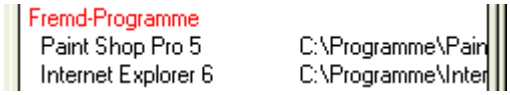
Register "Programme"
Register "Plugins"

Register "Programme"

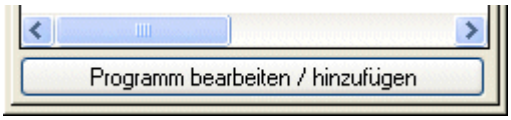
Hier können beliebige externe Programme angebunden werden. Auch wir hier nochmals zwischen Editor- und Fremd-Programmen unterschieden.

Unterstützt ein Programm den Parameteraufruf mit einer übergebenen Datei, so daß diese beim Laden des Programmes angezeigt wird, kann hier ein Häkchen gesetzt werden. So kann z.B. die aktuelle Datei im WYSIWYG-Editor unkompliziert aufgerufen und bearbeitet werden. Gleiches gilt z.B. auch für den Aufruf eines Web-Browsers zur externen Vorschau. Auch kann hierüber ein einfacher FTP-Uploder mit dieser Datei geladen werden.

Alle Editor-Programme befinden sich im Unterverzeichnis "plugins" des Installationsverzeichnisses von AT HTML Editor 32. Der Name wird automatisch anhand des Namens der EXE-Datei ausgelesen. So z.B. heißt die EXE-Datei für den WYSIWYG-Editor WYSIWYG-Editor.exe und für den Struktur-Editor Struktur-Editor.exe. Werden diese Namen verändert, so erscheint der veränderte Name in dieser Liste. Werden in das Plugin-Verzeichnis weitere Programme kopiert, dann erscheinen diese ebenfalls in dser Liste. Dabei sollte aber darauf geachtet werden, daß diese möglichst keine weiteren umfangreichen Zusatzdateien benötigen.
Alle Editor-Programme sind zur Unterscheidung blau gekennzeichnet. Der Pfad wird daneben zu Informationszwecken ebenfalls angezeigt. Gestartet wird ein Programm per Doppelklick darauf.

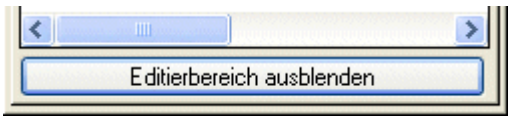


Hierüber können zusätzlich beliebig viele externe Fremd-Programme angebunden werden. Diese Auflistung ist in scharzer Schrift gehalten. Name und Pfad werden hier durch den entsprechenden Editor eingestellt. Gestartet wird ein Programm per Doppelklick darauf.
Es können auch Dateien hinzugefügt werden, die automatisch das damit verknüpfte Programm starten!



Zeigt den Editor zur Anbindung externer Programme an.

Der Eintrag unter "Programmname" erscheint in der Liste in dieser Spalte sowie auch der "Programmpfad" neben dem "Programmnamen" angezeigt wird. Alle weiteren Buttons erklären ihre Funktion mit ihrer Aufschrift bzw. der Kurzhilfe, wenn die Maus kurz darübergehalten wird.



Wird der Editor zur externen Programmanbindung angeziegt, ändert sich die Aufschrift des Buttons, mit dem er eingeblendet wird als Hinweis, daß er damit auch wieder ausgeblendet wird.

Register "Plugins"

Hier sind selbstgeschriebene Erweiterungen in Form von DLLs zugänglich. Damit können dem AT HTML Editor 32 weitere Funktionen "verpaßt" werden.

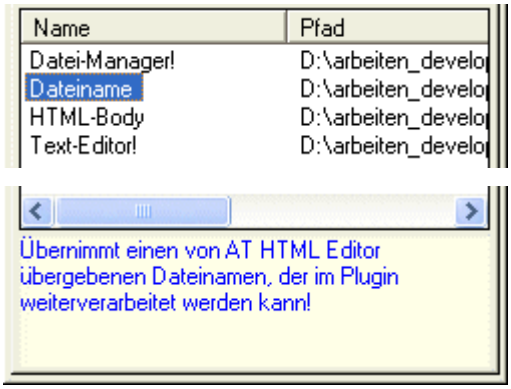
Gibt eine Information über den Rückgabetyp aus.

Es gibt 4 Rückgabetypen
markierter Text zum Plugin
Hiermit wird der im Code-Editor markierte Text an das Plugin zur weiteren Bearbeitung übergeben ohne daß das Plugin diesen Text wieder zurück gibt. Das ist z.B. dann sinnvoll, wenn der Text anderweitig weiterverarbeitet werden soll.

markierter Text zum/vom Plugin
Hiermit wird der im Code-Editor markierte Text an das Plugin zur weiteren Bearbeitung übergeben. Nach Bearbeitung durch das Plugin erfolgt eine Rückgabe des bearbeiteten Textes. Dieser ersetzt dann den im Code-Editor markierten Text. Ist kein Text markiert, wird der Text aus dem Plugin an der Cursorposition zusätzlich im Code-Editor eingefügt. Das ist dann nützlich, wenn z.B. per Plugin eine Tabelle oder Frameset generiert werden soll.

Dateiname zum Plugin
Hier wird lediglich der Dateiname des aktuellen Code-Editors an das Plugin übergeben. Das ist z.B. dann nützlich, wenn man eine bearbeitete Datei mit einem kleinen FTP-Uploader automatisch schnell mal in das FTP-Verzeichnis "schieben" möchte. Auch wäre denkbar, hiermit einen kleinen externen Debugger oder Syntaxchecker zu realisieren.

Dateiname vom Plugin
Hiermit ergibt sich die Möglichkeit, z.B. aus einem externen Dateimanager oder FTP-Downloader schnell die übergebene Datei in AT HTML Editor 32 zur Bearbeitung zu öffnen.



Alle DLLs, die sich im Unterverzeichnis "plugins" des Installationsverzeichnisses von AT HTML Editor 32 befinden, werden hier aufgelistet. Per Einfachklick werden zusätzliche Informationen zum Plugin ausgegeben und per Doppelklick wird ein Plugin gestartet.
Möchte man eigene Plugins nutzen, dann müssen diese in das Plugin-Verzeichnis kopiert werden. Beim nächsten Start von AT HTML Editor 32 stehen diese dann zur Verfügung.

Informationen über die Funktionalität bzw. über den Autor werden hier ausgegeben.

Eigene Plugins schreiben
Um eigene Plugins schreiben zu können, sind die Informationen unter dem Punkt "Plugin-Schnittstelle" hilfreich. Weitere Informationen finden sich auf der **Homepage von AT HTML Editor 32**.
Desweiteren stehen Beispiele und ein Plugin-Tester zum Download bereit.

Mögliche Probleme
Die Tatsache, daß ich hier ein Visual-Basic-Programm mit einer Plugin-Schnittstelle zum dynamischen Anbinden von DLLs versehen habe (was nebenbei gesagt etwas "Unmögliches" zu sein schien), birgt einige Probleme in sich. Insbesondere, was die Übergabe von Text aus einem Plugin heraus in einen Code-Editor betrifft. VB hat nicht ein so ausgefeiltes Speicher-Management wie z.B. Delphi oder C++. Werden Zeichenmengen von mehr als 1200 Zeichen (1200 Byte) vom Plugin zurückgegeben, kann das zum Absturz führen. Aber es ist auch möglich, daß das erst bei Mengen von mehr als 10000 Zeichen (10000 Byte) passiert. Das scheint von System zu System unterschiedlich zu sein, deshalb habe ich mich entschlossen, die Zeichenmenge nicht zu begrenzen.
Desweiteren kann es zu Schwierigkeiten bei der Übergabe von Dateinamen aus einem Plugin kommen, die dann von AT HTML Editor 32 geöffnet werden soll. Kurze Dateinamen führen öfter zu diesen Schwierigkeiten als lange Dateinamen. Aber auch hier war das von System zu System unterschiedlich.

Wohlgemerkt, diese Probleme sind nicht in erster Linie fehlerhafte Programmierung, sondern auf das Verhalten von VB-Programmen zu Delphi- oder anderen Programmen zurückzuführen, weil ich hier ein VB-Programm mit Delphi- und anderen Programmen quasi "verheiratet" habe.

Sitzungs-Manager

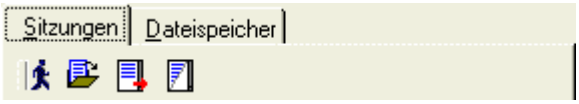
Der Sitzungs-Manager erlaubt den Zugriff auf Dateien und Projekte früherer Sitzungen von AT HTML Editor 32. Sobald AT HTML Editor 32 beendet wird, werden alle offenen Dateien und/oder das offene Projekt gespeichert und beim Neuladen wiederhergestellt, sofern die Option "Letzte Sitzung beim Programmstart öffnen" im Register "Config" des Projektexplorers aktiviert ist.
Sind zusätzliche Dateien im Register "Dateispeicher" vorhanden, werden auch diese gespeichert und wiederhergestellt. Dabei müssen diese nicht beim Beenden von AT HTML Editor 32 geöffnet sein.

Die zu speichernden Einträge im Register "Sitzungen" können im Register "Config" des Projektexplorers unter "Maximale Einträge im Sitzungs-Manager" begrenzt werden.

Register "Sitzungen"
Register "Dateispeicher"

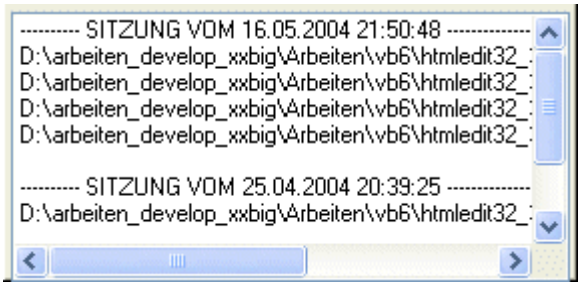
Register "Sitzungen"

Alle hier aufgeführten Einträge werden automatisch beim Beenden des Programmes gespeichert. Zusätzlich werden alle Dateien im Dateispeicher gespeichert, auch wenn sie nicht mehr geöffnet waren. Ist die Option "Letzte Sitzung beim Programmstart öffnen" im Register "Config" des Projektexplorers angehakt, wird das zuletzt geöffnete Projekt sowie deren geöffnete Dateien beim nächsten Start von AT HTML Editor 32 wieder geöffnet.



Hier stehen verschiedene Funktionen zur Verfügung, die das Beenden, das Öffnen und Entfernen von markierten Einträgen sowie das komplette Löschen der Sitzungen erlauben.

Alle Projekte und Dateien werden hier einer Sitzung zugeordnet, die auch Informationen über Datum und Zeit enthalten. Dabei erscheint die letzte Sitzung immer oben. Ältere Sitzungen erscheinen dementsprechend in der Reihenfolge immer weiter unten. Wird auf einen Eintrag doppelt geklickt, dann öffnet sich ie Datei oder das Projekt. Das funktioniert auch mit einer alternativ im Projektexplorer im Register "Config" eingestellten Editorkomponente.



Möchte man einen oder mehrere Einträge öffnen oder entfernen, müssen diese auf folgende Art und Weise markiert werden:

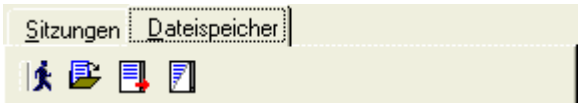
Zusammenhängende Einträge
1) bei gedrückter linker Maustaste über die Einträge ziehen
2) ersten Eintrag markieren und bei gedrückter UMSCHALT-Taste mit der aus auf letzten Eintrag klicken

Nichtzusammenhängende Einträge
Einträge nacheiander mit gedrückter STRG-Taste mit der Maus markieren.

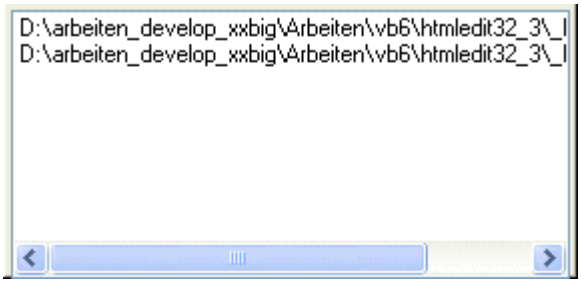
Datum- und Zeitformat der Sitzungsdaten
TT.MM.JJJJ HH:MM:SS

Register "Dateispeicher"

Der Dateispeicher ermöglicht das Ablegen von Dateien in einer aktiven Sitzung, ohne das diese beim Beenden von AT HTML Editor 32, also einer Sitzung, geöffnet sein müssen. Das hat zur Folge, daß auch diese Dateien beim nächsten Start automatisch geladen werden.



Hier stehen verschiedene Funktionen zur Verfügung, die das Öffnen und Entfernen von markierten Einträgen sowie das komplette Löschen der Liste erlauben.



Listet alle abgelegten Dateien auf, obwohl diese nicht mehr geöffnet sein müssen. Wird auf einen Eintrag doppelt geklickt, dann öffnet sich ie Datei oder das Projekt. Das funktioniert auch mit einer alternativ im Projektexplorer im Register "Config" eingestellten Editorkomponente.

Möchte man einen oder mehrere Einträge entfernen, müssen diese auf folgende Art und Weise markiert werden:

Zusammenhängende Einträge
1) bei gedrückter linker Maustaste über die Einträge ziehen
2) ersten Eintrag markieren und bei gedrückter UMSCHALT-Taste mit der aus auf letzten Eintrag klicken

Nichtzusammenhängende Einträge
Einträge nacheiander mit gedrückter STRG-Taste mit der Maus markieren.

Code-Editor

Der Code-Editor ist die Codierwerkstatt von AT HTML Editor 32. Er ist kompakt gehalten und bündelt wesentliche Funktionalität zur Entwicklung von HTML-, PHP- oder sonstigen Dokumenten. Luxus ist nicht Trumpf aber schnelle und intuitive Bedienbarkeit. Darüber hinaus bietet er noch weitere unterstützende Funktionen.

Navigation
Jeder Code-Editor besitzt schnelle und einfache Navigationsfunktionen, die es ermöglichen unkompliziert zwischen den Code-Editoren sowie dem Projektexplorer und Sitzungs-Manager zu wechseln.

Hierüber kann in jedes Register des Code-Editors gewechselt werden. Dabei werden die Arbeitszustände in den Registern gespeichert bzw. Manager ein- und ausgeblendet.

Dies kleinen Navigationspfeile ermöglichen den schnellen Wechsel zwischen den Code-Editoren. Mittels den Tastenkombinationen **ALT+1** und **ALT+2** kann blitzschnell zwischen den geöffneten Code-Editoren zurück und vor geblättert werden.

Für gezielten Zugriff auf geöffnete Dateien (und damit auf einen geöffneten Code-Editor), kann dieses Kombinationsfeld genutzt werden.

Neben oben genannten Navigationsmöglichkeiten kann auch die Dateiliste des Projektexplorers genutzt werden.

Register "Code"
Register "Vorschau"
Register "Bildviewer"
Register "Internet"
Register "Diverses"

Register "Code"

Hier wird der Code erzeugt!

Jeder Code-Editor kann in bis zu 4 Anzeigebereichen aufgeteilt werden. Dazu befinden sich oben bzw. links an den Bildlaufleisten kleine Anfasser, die in den Codebereich gezogen werden können. Wird doppelt darauf geklickt, dann wird der Codebereich jeweils zur Hälfte geteilt bzw. wieder vollständig hergestellt.

Wird in den Codebereich ein Rechtsklick mit der Maus gemacht, dann erscheint ein Kontextmenü, über das weitere Funktionen (umwandeln in Groß-und Kleinbuchstaben, Suchen etc.) sowie auch der Eigenschaften-Dialog erreicht werden kann.

Bedienelemente
Farben-Manager
Syntax-Manager

Bedienelemente

Die Bedienelemente von AT HTML Editor 32 sind auf einfache, schnelle und vor allem sinnvolle Nutzung ausgelegt, so daß man schnell zu verwertbaren Ergebnissen kommt. Alle Bedienelemente sind auch per Tastenkombination zugänglich, die jeweils in der dritten Spalte angezeigt werden.

Schließt den aktuellen Code-Editor. Es erfolgt eine Speicherabfrage, sofern noch nicht gespeichert wurde.	ALT+F4
Schaltet in den Vollbild-Modus um. Alle geöffneten Manager und Fenster bleiben erhalten.	ALT+O
Lädt einen neuen leeren Code-Editor.	ALT+N
Speichert die aktuelle im Code-Editor geöffnete Datei. Ist kein Name vorhanden, wird der Dialog "Datei speichern unter" eingeblendet.	ALT+S
Speichert die aktuell im Code-Editor geöffnete Datei unter einem anderen Namen. Dabei wird der Dialog "Datei speichern unter" eingeblendet.	ALT+L
Druckt den gesamten Text im Code-Editor schwarz/weiß oder farbig (je nach Drucker) aus.	ALT+K
Markiert den gesamten Text im Code-Editor.	ALT+A
Schneidet markierten Text aus dem Code-Editor aus.	STR+X
Kopiert markierten Text aus dem Code-Editor.	STRG+C
Fügt Text an der Cursorposition in den Code-Editor ein.	STRG+V
Macht alle Bearbeitungsschritte im Text seit dem Öffnen der Datei rückgängig. Die Anzahl der Rückgängig-Schritte kann im "Eigenschaften"-Dialog (Kontextmenü|"Properties"| Register "Misc"|"Max undoable actions") auch begrenzt werden, ist aber nicht erforderlich.	STRG+Z
Stellt rückgängig gemachte Arbeitschritte wieder her.	STRG+A
Blendet den Farben-Manager ein/aus.	ALT+F
Blendet den Syntax-Manager ein/aus.	ALT+E
Blendet den Code-Manager ein/aus.	ALT+T
Holt den Projektexplorer in den Vordergrund.	ALT+X
Zeigt einen Suchen-Dialog an, über den ein Begriff im Text gesucht werden kann.	STRG+F
Sucht das nächste Vorkommen des im Suchen-Dialog eingegebenen Begriffes.	F3
Zeigt einen Suchen-Ersetzen-Dialog an, über den ein Begriffe mit einem anderen Begriff ersetzt werden kann.	STRG+ALT+F3
Blendet den Code-Browser ein/aus. Dieser ist ein wirksamens Mittel, um schnell und gezielt auf verschiedene Codeteile zugreifen zu können. Im Register "Config" des Projektexplorers unter "Suchbegriffe für den Code-Browser" können individuelle Suchbegriffe definiert werden.	
Wann sich der Code-Browser aktualisiert - sobald der Code-Browser angezeigt wird - wenn der Code-Editor den Fokus erhält - wenn der Text geändert wurde: bei Eingabetaste, bei Pfeiltasten oben/unten	ALT+W
Das kann daran erkannt werden, daß der Code-Browser kurz flackert.	
Ordnet den Code-Editor in der oberen Bildschirmhälfte an.	ALT+3
Ordnet den Code-Editor in der unteren Bildschirmhälfte an.	ALT+4
Ordnet den Code-Editor in der linken Bildschirmhälfte an.	ALT+5
Ordnet den Code-Editor in der rechten Bildschirmhälfte an.	ALT+6
Führt die Datei mit benutzerdefinierter Hostadresse, die im Projektexplorer im Register "Config" eingestellt ist, in der Vorschau aus. Damit können z.B. Active-Server-Pages (ASP) oder Java-Server-Pages (JSP) entwickelt und getestet werden.	ALT+Y
Interpretiert, kompiliert oder führt die Datei in der Vorschau aus und zeigt gleichzeitig die Ausgabekonsole im Code-Editor an. Ob die Vorschau angezeigt wird oder nicht ist abhängig von der eingestellten Syntax im Code-Editor (Syntax-Manager).	
Vorschau wird angezeigt bei folgenden Syntaxen HTML, PHP. Wird im Register "Run" des Projektexplorers ein Häkchen bei "Run" gesetzt, dann werden PHP-Dateien nur auf Syntax überprüft (sofern der PHP-Interpreter angebunden ist) und nicht in der Vorschau angezeigt.	ALT+G
Vorschau wird nicht angezeigt bei folgenden Syntaxen Purebasic (pb-Dateien), Rapid-Q (rq-Dateien), benutzerdefinierte Syntax. Ist die benutzerdefinierte Syntax eine Websprache wie ASP oder JSP, dann muß für die Vorschau "Run für benutzerdefinierte	

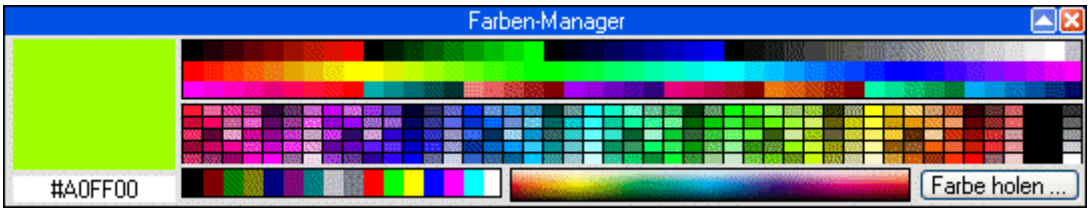
Syntax" genutzt werden. Soll dagegen eine Java-Klasse für JSP-Seiten kompiliert werden, dann muß der Java-Compiler im Register "Run" im Projektexplorer als benutzerdefinierter Compiler/Interpreter eingestellt sein. Gleiches gilt auch für andere Compiler/Interpreter wie z.B. bei Tcl/Tk.

- Hier gilt Gleiches wie bei "Run mit Ausgabekonsole", nur daß hier die Ausgabekonsole nicht angezeigt wird. ALT+R

Farben-Manager

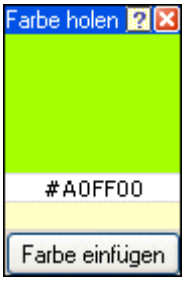
Der Farben-Manager erlaubt das schnelle und flexible Ermitteln, Konvertieren und Einfügen von Farben im HTML-Farbcode in den Code-Editor. Zudem können systemweit Farben geholt und in den Code eingefügt werden.

Farben-Manager



Einfach nur die Maus über die entsprechende Farbe halten und doppelt darauf klicken. Dadurch wird der HTML-Farbcode an der Cursorposition im Codebereich eingefügt.

Systemweit Farben holen

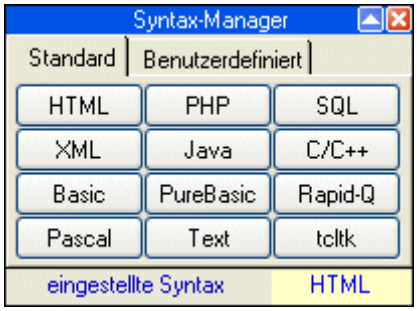


Dieser Dialog läßt sich aus dem Farben-Manager mittels des Buttons "Farbe holen" aufrufen. Möchte man nun systemweit eine Farbe holen, dann nur die Maus über die Farbe halten und die Eingabetaste drücken. Jetzt befindet sich der ermittelte und konvertierte Farbwert im unteren gelben Feld. Jetzt kann er mittels dem Button "Farbe einfügen" in den Codebereich an der Cursorposition eingefügt werden.

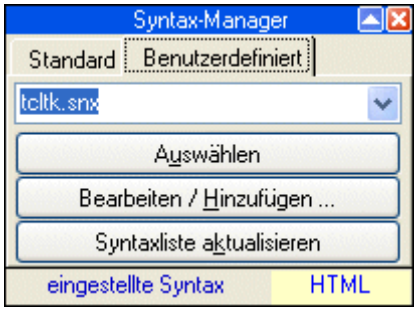
Beide Dialog sind ein- und ausklappbar per Doppelklick auf die Titelleiste bzw. den Pfeilicons oben rechts. Auch werden beide bei Aktivierung in der Titelleiste in den Vordergund geholt.

Syntax-Manager

Über den Syntax-Manager wird die Syntax für den aktuellen Code-Editor eingestellt. Es hat aber keine Auswirkungen auf neue Code-Editoren, wie das in früheren Versionen der Fall war. Im Register "Config" im Projektexplorer kann eine Standardsyntax eingestellt werden. Ist diese nicht vorhanden, ist HTML voreingestellt.



Das sind die in AT HTML Editor 32 zur Auswahl stehenden Standardsyntaxmodi. Der Button unten rechts beinhaltet jedoch eine ausgewählte benutzerdefinierte Syntax. Das ist sinnvoll, wenn man zwischen einer Standardsyntax und der benutzerdefinierten Syntax hin und her schalten möchte. Ist keine benutzerdefinierte Syntax ausgewählt, ist der Button leer. Ist dagegen eine benutzerdefinierte Syntax als Standardsyntax eingestellt, erscheint diese hier ebenfalls automatisch.



Über dieses Register kann eine benutzerdefinierte Syntax ausgewählt werden. Mittels des Syntax-Editors kann diese bearbeitet werden oder es kann eine neue Syntax-Datei erzeugt werden.

Der Syntax-Manager ist per Doppelklick auf die Titelleiste bzw. mittels des Pfeilicons ein- und ausklappbar. Auch kann er per Klick in die Titelleiste in den Vordergrund geholt werden.

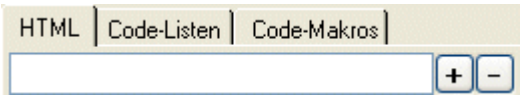
Code-Manager

Der Code-Manager ist die Schaltzentrale in Sachen Codierhilfen. Er enthält die altbekannte HTML-Tag-Liste aber auch eine Code-Listen-Auswahl, in der beliebige Codelisten definiert und bearbeitet werden können. Eine einfache und effektive Suche rundet diese beiden Teile ab. Zusätzlich ist ein kleiner aber leistungstarker Makrorecorder enthalten, mit dem Code-Makros aufgezeichnet und einfach abgerufen werden können.

Der Code-Manager ist per Doppelklick auf die Titelleiste bzw. mittels des Pfeilicons ein- und ausklappbar. Auch kann er per Klick in die Titelleiste in den Vordergrund geholt werden.

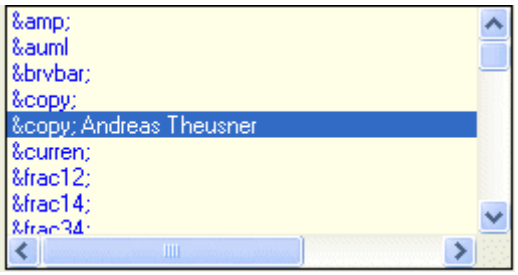
Register "HTML"

In diesem Register können benutzerdefinierte HTML-Tags gespeichert und auch wieder in den Codebereich eingefügt werden. Aus Kompatibilitätsgründen befindet sich die entsperchende Datei (tags.edt) im Plugin-Verzeichnis (plugins), denn der WYSIWYG-Editor ist nunmehr ein externes Programm (Plugin-) Programm, das ebenfalls darauf zugreift.



Hierüber werden neue Tags eingegeben und hinzugefügt, aber auch der in der Liste markierte HTML-Tags aus der Liste gelöscht.

Mittels dieses Suchfeldes kann ein Tag kontextsensitive gesucht werden. Das bedeutet, daß z.B. bei Eingabe vo "<" alle Tags gefunden werden die damit beginnen und bei "<T" alle Tags mit diesen Zeichen gefunden werden wie "<TABLE>", "<TR>" oder "<TD>".



Liste der HTML-Tags. Per Doppelklick wird der markierte Tag in den Codebereich eingefügt.

Vorschau (wenn möglich) des markierten HTML-Tags.

Das sind die sogenannten Schnellbuttons, die per Klick ganz schnell häufig gebrauchten HTML-Code in den Codebereich einfügen. Eine überaus praktische und von vielen Anwendern gern genutzte Einrichtung.

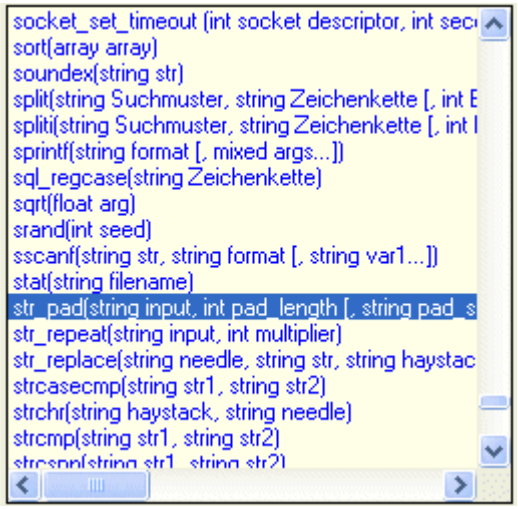
Register "Code-Listen"

Code-Listen sind nichts weiter als Code-Bausteine, Codesnippets oder Code-Vorlagen. Insbesondere Listen von Funktionen, Kontrollstrukturen und sonstigen Elementen, die eine Programmier- oder Skriptsprache ausmachen, können auf diese Weise schnell, einfach und individuell zugänglich und bearbeitet werden. Zwar ersetzt das nicht die oft angenehme Möglichkeit der Codevervollständigung, aber in vielen Programmierwerkzeugen ist diese oft lästig oder unausgereift. Deshalb bevorzuge ich persönlich diese Variante und schalte bei anderen Werkzeugen, wenn es geht diese ab.

Auswahl der Code-Liste, die angezeigt werden soll. Auch kann eine neue Codeliste angelegt und eine vorhandene gelöscht werden. Wurde eine neue angelegt und soll diese sofort zugänglich sein, dann muß auf den Button "Aktualisieren" geklickt werden.

In diesem Feld können neue Einträge gemacht und zur Liste hinzugefügt aber auch ein markierter Eintrag aus der Liste entfernt werden.

Mittels dieses Suchfeldes kann ein Tag kontextsensitive gesucht werden. Das bedeutet, daß z.B. bei Eingabe vo "str_" alle PHP-Funktionen gefunden werden können, die "str_" beginnen, wie z.B. "str_replace".

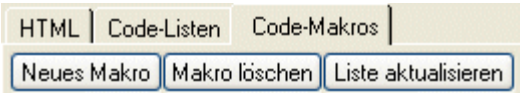


Liste der Codesnippets. Per Doppelklick wird der markierte Eintrag in den Codebereich eingefügt.

Auch hier gibt es die bewährten Schnellbuttons zum schnellen Einfügen von Kontrollstrukturen.

Register "Code-Makros"

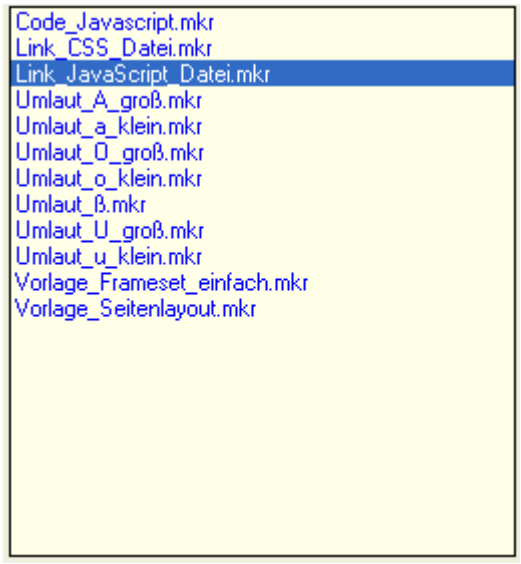
Dieses Werkzeug ist zweifellos eines der stärksten im AT HTML Editor 32, aber auch eines, vor dem sich viele "fürchten" werden. Dennoch ist es so einfach zu bedienen wie z.B. in Microsoft-Office-Produkten, ja vielleicht sogar noch einfacher. Eine interne (nicht sichtbare) Engine regelt das Handling der Makros, damit sich diese nicht ins Gehege kommen. **Wichtig** ist zu wissen, daß gleichzeitig bis zu 10 ungespeicherte Makros pro Sitzung vorgehalten werden können, aber es kann immer nur das zuletzt erstellte gespeichert werden. Alle 10 Makros sind (sofern sie erstellt worden sind) mit den Tastenkombination STRG+1 bis STRG+0 ausführbar, aber in der Regel reichen STRG+1 und STRG+2, weil man selten 10 Makros gleichzeitig vorhält. Auch gibt der Makrozähler dann Auskunft, welche Tastenkombination nötig ist. Wird entgegen dieser Konvention ein elftes Makro erzeugt, wird wieder von vorn (1) begonnen zu zählen. Dieses ist global in AT HTML Editor 32 gültig.



Erstellt ein neues Makro. Jetzt können alle Aktionen, die den Codierbereich betreffen, wie z.B. Texteingabe, ausschneiden, kopieren, einfügen etc. durchgeführt werden. Auf diese Weise lassen sich schnell kleine und große Code-Vorlagen zum wiederkehrenden Gebrauch erstellen.
Auch kann ein in der Liste markiertes Makro wieder gelöscht werden.
Wurde ein Makro in einem anderen Code-Editor erstellt, dann kann die Liste der Makros per Button aktualisiert werden.

Wurde ein neues Makro erstellt, ändert sich die Farbe dieses Feldes in ein helles rot zum Zeichen, daß ein erstelltes Makro noch nicht gespeichert wurde. Wenn ohne zu speichern erneut ein neues Makro ausgeführt wurde, dann wird die zuletzt erstellte Information im Makro gespeichert. Die davor erstellte geht verloren, obwohl sich der Makrozähler nicht erhöht.

Der Makrozähler zeigt die Anzahl der bisher erzeugten Makros an. Und mit dem Button "Makro ausführen" führt man natürlich ein in der Liste ausgewähltes Makro an der Cursorposition des Codebereiches aus. Gleiches wir auch durch einen Doppelklick erreicht.



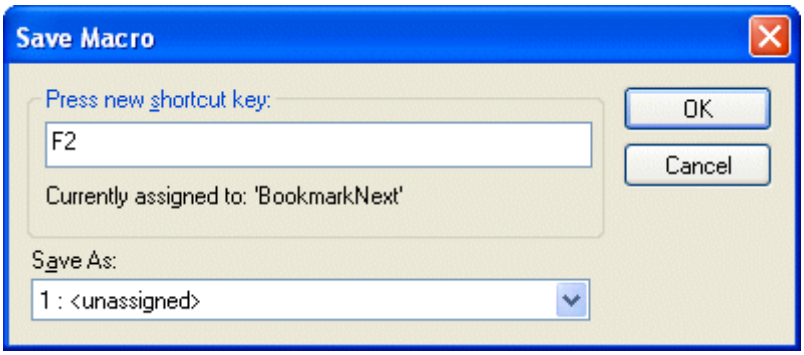
Listet alle verfügbaren Makros auf.

Code-Makros erstellen

Das ist ganz einfach!

1) auf den Button "Neues Makro" klicken
2) nachfolgender Dialog erscheint und der Mauscursor ändert sich in eine typische "Filmklappe"

3) jetzt können die gewünschten Aktionen im Codebereich durchgeführt werden
4) es erscheint (proforma) nachfolgender Dialog für die Zuordnung einer weiteren Tastenkombination



5) dort einfach nur ein Zeichen oder Zahl wie z.B. F2 eingeben
6) dem Makro einen Namen geben und auf den Button "Speichern" klicken
7) jetzt ist das erstellte Makro in der Liste aufgetaucht und kann beliebig genutzt werden

War das schwer? Nicht wirklich oder?

Verschiedene Modi

Das Register "Code" kennt verschiedene Modi bei der Bearbeitung und Verarbeitung von Programm- und Skriptcode. Diese Modi dienen der Übersicht und dem Ausführen sowie Checken der Syntax.

Vollbild-Modus

Der Vollbildmodus ermöglicht die maximale Nutzung des Codierbereiches des Code-Editors. Es werden am linken Rand Icons für die wesentlichsten und (wie ich finde) nützlichsten bereitgestellt. Dennoch sind alle anderen Funktionen weiterhin per Tastenkombination erreichbar.

 Damit wird der Vollbildmodus verlassen und schaltet wieder in den normalen Modus.	ALT+O
 Speichert die aktuelle im Code-Editor geöffnete Datei. Ist kein Name vorhanden, wird der Dialog "Datei speichern unter" eingeblendet.	ALT+S
 Blendet die Fensterliste in der Auswahlbox zum Wechseln in andere geöffnete Code-Editoren und die Wechselicons ein/aus.	ALT+1, ALT+2
 Blendet den Code-Manager ein/aus.	ALT+T
 Zeigt einen Suchen-Dialog an, über den ein Begriff im Text gesucht werden kann.	STRG+F
Blendet den Code-Browser ein/aus. Dieser ist ein wirksamens Mittel, um schnell und gezielt auf verschiedene Codeteile zugreifen zu können. Im Register "Config" des Projektexplorers unter "Suchbegriffe für den Code-Browser" können individuelle Suchbegriffe definiert werden.	
 **Wann sich der Code-Browser aktualisiert** - sobald der Code-Browser angezeigt wird - wenn der Code-Editor den Fokus erhält - wenn der Text geändert wurde: bei Eingabetaste, bei Pfeiltasten oben/unten	ALT+W
Das kann daran erkannt werden, daß der Code-Browser kurz flackert.	
Interpretiert, kompiliert oder führt die Datei in der Vorschau aus ohne Ausgabekonsole. Ob die Vorschau angezeigt wird oder nicht ist abhängig von der eingestellten Syntax im Code-Editor (Syntax-Manager).	
Vorschau wird angezeigt bei folgenden Syntaxen HTML, PHP. Wird im Register "Run" des Projektexplorers ein Häkchen bei "Run" gesetzt, dann werden PHP-Dateien nur auf Syntax überprüft (sofern der PHP-Interpreter angebunden ist) und nicht in der Vorschau angezeigt.	
	ALT+G
Vorschau wird nicht angezeigt bei folgenden Syntaxen Purebasic (pb-Dateien), Rapid-Q (rq-Dateien), benutzerdefinierte Syntax. Ist die benutzerdefinierte Syntax eine Websprache wie ASP oder JSP, dann muß für die Vorschau "Run für benutzerdefinierte Syntax" genutzt werden. Soll dagegen eine Java-Klasse für JSP-Seiten kompiliert werden, dann muß der Java-Compiler im Register "Run" im Projektexplorer als benutzerdefinierter Compiler/Interpreter eingestellt sein. Gleiches gilt auch für andere Compiler/Interpreter wie z.B. bei Tcl/Tk.	
 Blendet alle Zeichen (Whitespaces) wie Leerzeichen, Tabulatoren etc. ein/aus. Das ist besonders dann von Vorteil, wenn auf richtige Codestruktur und korrekte Klammerschließung geachtet werden soll.	In diesem Modus keine.

Run-Modus

AT HTML Editor 32 kennt 3 verschiedene Arten von Run-Modus plus HTML-Vorschau.

Run-Modus ohne Ausgabekonsole **ALT+R**
Dieser Run-Modus interpretiert, kompiliert oder führt die Datei in der Vorschau aus. Ob die Vorschau angezeigt wird oder nicht ist abhängig von der eingestellten Syntax im Code-Editor (Syntax-Manager).

Vorschau wird angezeigt bei folgenden Syntaxen
HTML, PHP. Wird im Register "Run" des Projektexplorers ein Häkchen bei "Run" gesetzt, dann werden PHP-Dateien nur auf Syntax überprüft (sofern der PHP-Interpreter angebunden ist) und nicht in der Vorschau angezeigt.

Vorschau wird nicht angezeigt bei folgenden Syntaxen
Purebasic (pb-Dateien), Rapid-Q (rq-Dateien), benutzerdefinierte Syntax. Ist die benutzerdefinierte Syntax eine Websprache wie ASP oder JSP, dann muß für die Vorschau "Run für benutzerdefinierte Syntax" genutzt werden. Soll dagegen eine Java-Klasse für JSP-Seiten kompiliert werden, dann muß der Java-Compiler im Register "Run" im Projektexplorer als benutzerdefinierter Compiler/Interpreter eingestellt sein. Gleiches gilt auch für andere Compiler/Interpreter wie z.B. bei Tcl/Tk.
Wird z.B. ein PureBasic oder Rapid-Q-Programm kompiliert und ausgeführt ist es sinnvoll, im Register "Run" des Projektexplorers die Option "benutz. Compiler-Interpreterprozess abwarten" zu aktivieren. Damit wird AT HTML Editor 32 angwiesen, solange mit der weiteren Abarbeitung zu warten, bis der Compiler-/Interpreterprozeß beendet wurde.

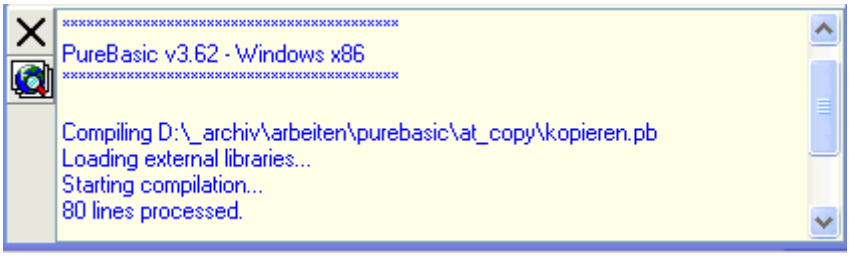
Run-Modus mit Ausgabekonsole **ALT+G**
Gleiches gilt auch für diesen Run-Modus, nur wird in diesem Falle zusätzlich die Ausgabekonsole im Codebereich angezeigt. Darin finden sich dann (je nach Programmier-/Skriptsprache) zusätzliche Informationen z.B. zur Fehlerart oder Fehlerstelle.

Run-Modus mit benutzerdefiniertem Host **ALT+Y**
Mit diesem Run-Modus kann eine Datei mit benutzerdefinierter Hostadresse, die im Projektexplorer im Register "Config" eingestellt ist, in der Vorschau ausgeführt werden. Damit können z.B. Active-Server-Pages (ASP) oder Java-Server-Pages (JSP) entwickelt und getestet werden.

HTML-Vorschau Vorschau **ALT+V**
In der Vorschau können alle Webseiten, die mit AT HTML Editor 32 entwickelt werden, schnell und unkompliziert zur Kontrolle angezeigt werden. Insbesondere ist diese Vorschau für die Entwicklung von HTML-Seiten gedacht. Aber auch von Tidy kontrollierte HTML-Seiten, die in der Ausgabekonsole ausgegeben werden, können mittels des entsprechendem Icons in der Ausgabekonsole in der Vorschau angezeigt werden.

Bei allen Run-Modi gilt: Ist im Register "Config" des Projektexplorers die Option "Speichern vor Run" aktiviert, wird die Datei im Code-Editor vor dem Ausführen gespeichert.

Ausgabekonsole



Hier werden weitere Informationen zu z.B. Fehlerart oder Fehlerstelle ausgegeben. Mit dem oberen Icon wird die Ausgabekonsole geschlossen und mit dem unteren wird der Inhalt der Ausgabekonsole in der Vorschau angezeigt.

Register "Vorschau"

In der Vorschau können schnell und unkompliziert HTML-Dateien aus dem Codebereich sowie aus der Ausgabkonsole angezeigt werden. Sofern für PHP oder benutzerdefiniert eine Hostadresse eingetragen ist, dann wird nach dem Ausführen des jeweiligen Skriptes ebenfalls die Vorschau mit der fertig erzeugten Seite angezeigt.

HTML-Seiten können auch angezeigt werden, ohne daß diese zuvor gespeichert wurden. Das ermöglicht z.B., etwas mal eben schnell auszuprobieren ohne gleich speichern zu müssen.

Wird in die Vorschau ein Rechtsklick mit der Maus gemacht, erscheint das Kontextmenü des Internet Explorers.

Bedienelemente

Bedienelemente

Die Bedienelemente dienen hier ausschließlich den Anforderungen einer einfachen Vorschau.

	Geht zur vorherigen Seite zurück, sofern man zur nächsten Seite gewechselt.	ALT+<
	Geht zur wieder nächsten Seite, sofern eine Seite zurückgegangen ist.	ALT+SHIFT+>
	Lädt die aktuell angezeigte Seite nochmal neu.	ALT+N
	Startet die im Register "Config" im Projektexplorer unter "HTML-Startdatei für (Projekt-)Website" eingestellte Startseite. Dadurch kann die gesamte Website, so wie der Nutzer sie beim erstmaligen Anzeigen sehen würde, geladen werden. Das ist besonders für eine schnelle Kontrolle der gesamten Website nützlich.	ALT+R
PHP-/Benutzer-Hostadresse:	Zeigt die komplette Hostadresse samt der aktuellen Datei bei PHP- und benutzerdefinierter Hostadresse an.	

Register "Bildviewer"

Der Bildviewer eröffnet eine einfache Möglichkeit, nicht nur Bilder und Grafiken anzuschauen, sondern auch mittels der mitlaufenden Positionsangaben Imagemaps zu erstellen. Diese können ganz einfach per Linksklick in den Codebereich eingefügt werden.

Es werden die Windows-Standardformate angezeigt: BMP, GIF, JPG, WMF, ICO.

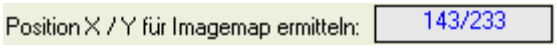
Bedienelemente

Bedienelemente

Die Bedienelemente von AT HTML Editor 32 sind auf einfache, schnelle und vor allem sinnvolle Nutzung ausgelegt, so daß man schnell zu verwertbaren Ergebnissen kommt. Alle Bedienelemente sind auch per Tastenkombination zugänglich, die jeweils in der dritten Spalte angezeigt werden.

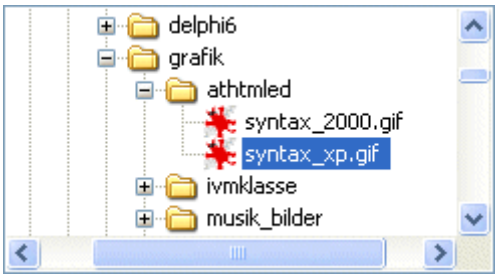
Setzt wie im Projektexplorer im Register "Dateien" einen Verzeichnisroot. ALT+L

Blendet den Dialog "Farbe holen" ein/aus. Dieser Dialog erlaubt systemweites Holen von beliebigen Farben, die in HTML-Farben konvertiert werden und zum Einfügen in den Codebereich zur Verfügung stehen. ALT+F



Lädt einen neuen leeren Code-Editor.Wenn mit der Maus über das Bild gefahren wird, dann erscheinen hier die Positionsangaben in Pixel. Dabei bedeutet X von links nach rechts und Y von oben nach unten.

Wird mit der linken oder rechten Maustaste in das Bild geklickt, erscheint hier die entsprechende Position. Gleichzeitig wird diese Position in den Codebereich an der Cursorposition eingefügt.



Über diesen praktischen Dateibaum erfolgt die Auswahl der Bilder.

Register "Internet"

Wer das Ergebnis seiner Arbeit auch Online schnell überprüfen möchte, kann das mit diesem kleinen Webbrowser tun. Ohne Schnickschnack lädt der sofort die gewünschte Seite und berücksichtig dabei auch Javascript. Aber auch zum Browsen in anderen Sites ist er geeignet.

Ist eine gültige Internetadresse eingegeben, wird zu dieser gewechselt, wenn wieder auf dieses Register gewechselt wird.

Bedienelemente

Bedienelemente

Die Bedienelemente von AT HTML Editor 32 sind auf einfache, schnelle und vor allem sinnvolle Nutzung ausgelegt, so daß man schnell zu verwertbaren Ergebnissen kommt. Alle Bedienelemente sind auch per Tastenkombination zugänglich, die jeweils in der dritten Spalte angezeigt werden.

	Hier wird die gewünschte Internetadresse eingegeben. Dabei muß nicht immer "http://" zuerst eigegeben werden, um zu einer Adresse zu springen. Per Eingabetaste oder nebenstehenden Icon wird die Seite geladen. Alle eingegebenen Adressen werden in der Liste nur des aktiven Code-Editors gespeichert.	ALT+A
	Sofern eine gültige Internetadresse eingegeben wurde, wird per Klick auf dieses Icon dahin gewechselt.	ALT+A
	Stopt das Laden einer Internetseite.	ALT+S
	Geht zur vorherigen Seite zurück, sofern man zur nächsten Seite gewechselt.	ALT+<
	Geht zur wieder nächsten Seite, sofern eine Seite zurückgegangen ist.	ALT+SHIFT+>
	Lädt die aktuell angezeigte Seite nochmal neu.	ALT+N

Register "Diverses"

Hier finden sich verschiedene Funktionen und Optionen, die alle Register bestreffen bzw. zusätzlich zur Verfügung stehen.

Bereitgestellte Funktionen

	Lädt den Sitzungs-Manager.	ALT+T
	Legt die aktuelle Datei in den "Dateispeicher" des Sitzung-Managers.	ALT+E
	Blendet die Zeilennummerierung ein/aus. Dadurch kann der Codierbereich nochmals vergrößert werden.	ALT+Z
¶	Blendet alle Zeichen wie Leerzeichen, Tabulatoren etc. (Whitspaces) ein/aus. Das ist besonders dann nützlich, wenn man den Code auf korrekte Klammersetzung oder Tabulatorstruktur überprüfen möchte. Erreichbar ist diese Funktion auch über das Kontextmenü des Codebereiches.	ALT+W
	Lädt den Eigenschaften-Dialog des Codebereiches, auch erreichbar über das Kontextmenü desselbigen.	ALT+O

Editor-Programme

Ab dieser Version von AH HTML Editor 32 sind alle bisherigen integrierten Editoren in externe Editor-Programme ausgelagert. Diese sind eigenständig als ausführbares Programm lauffähig. Auch besteht so die Möglichkeit, Editor-Programme unabhängig von AT HTML Editor 32 weiterzuentwickeln oder diese komplett durch neue zu ersetzen. Alle Editorprogramme sind zugänglich über das Register "Plugins" des Projektexplorers und dort im Register "Programme" unter "Editor-Programme".

Zusätzlich zu diesen Editor-Programmen können dort noch weitere Editor-Programme erscheinen. Wenn ein Programm in das Unterverzeichnis "plugins" kopiert wird, dann wird es dort automatisch aufgelistet. Dabei ist zu beachten, daß es nicht zusätzliche Komponenten benötigt, sondern als eine EXE-Datei vorliegt.

WYSIWYG-Editor

Der WYSIWYG-Editor ist eine ganz nette Einrichtung. Er ist gedacht zum visuellen nachbearbeiten von HTML-Dokumenten. Aber es können damit auch ganz schnell neue Dokumente erstellt werden.
Der WYSIWYG - Editor arbeitet in Echtzeit. Sobald eine Änderung im oberen WYSIWYG-Bereich oder im unteren Code-Bereich gemacht wurde, wird diese in beiden Bereichen sofort sichtbar.

Funktionen der Buttons

Schließen	Schließt den WYSIWYG - Editor.
Neu	Legt ein neues Dokument an.
Öffnen ...	Öffnet ein vorhandenes Dokument.
Speichern als ...	Speichert das Dokument unter einem Namen.
Speichern	Speichert Änderungen am aktuellen Dokument.
Browsen ein/aus	Schaltet für den oberen WYSIWYG-Bereich den Browsemode ein und aus. Im Browsemode kann sich wie im Browser durch die Links geklickt werden - Bearbeitung ist dann nicht möglich.
Bild ...	Lädt einen Dialog, über den Bilder in den WYSIWYG-Bereich eingefügt werden können.
Tabelle ...	Lädt einen Dialog, über den mal schnell eine Tabelle in den WYSIWYG-Bereich eingefügt werden kann. Zusätzlich erlaubt dieser Dialog, in den Code-Bereich Tabellen- und Zellenattribute einzufügen.
Listpunkt	Fügt einen Listpunkt in den WYSIWYG-Bereich ein.
Listpunkt - Zahl	Fügt einen Listpunkt mit Zahl (Nummerierung) in den WYSIWYG-Bereich ein.
Einrücken	Rückt einen Absatz im WYSIWYG-Bereich ein.
Ausrücken	Rückt einen Absatz im WYSIWYG-Bereich aus.
Code anzeigen	Psychoknopf 1: Refresht den Code aus den WYSIWYG-Bereich in den Code-Bereich - geht automatisch.
Code speichern	Psychoknopf 2: Speichert den Code des aktuellen Dokumentes - geht automatisch.
HTML - Tags ...	Blendet exakt den gleichen Dialog ein, der in Code - Editor auch zur Verfügung steht. Wird ein Tag hier zugefügt, ist er woanders auch sichtbar und umgekehrt.

Struktur-Editor

Wie schon im AT HTML Editor 16 gibt es hier auch den Struktur-Editor. Damit kann man ganz einfach und schnell eine Struktur der zu erstellenden Website entwerfen. Der Struktur-Editor hat sich als ganz hilfreich erwiesen. Ab der Version von AT HTML Editor 32.3 können Strukturdateien in ein Projekt aufgenommen werden. Der Struktur-Editor wird jeweils automatisch gestartet. Die Strukturdatei wird automatisch im Projektverzeichnis angelegt und heißt wie das Projekt, jedoch mit der Endung *.str.
Damit das Ganze überschaubarer ist, stellt er alle Zeichen farbig dar.

Eine Struktur kann z.B. so aufgebaut werden:

```
index.htm
  |
  |- welcome.htm
  |- produkt.htm
  |      |
  |      |- hardware.htm
  |      |      |
  |      |      |- monitor.htm
  |      |      |- drucker.htm
  |      |
  |      |- software.htm
  |      |      |
  |      |      |- betriebssystem.htm
  |      |      |- office.htm
  |      |
  |- kontakt.htm
  |
  usw.
```

Menü "Datei"

Neu	Legt eine neues Struktur-Dokument an.
Öffnen ...	Öffnet ein vorhandenes Struktur-Dokument.
Speichern als ...	Speichert das Dokument unter einem Namen.
Speichern	Speichert Änderungen am aktuellen Dokument.
Drucken ...	Druckt das Struktur-Dokument per Dialog aus.
Schließen	Schließt den Struktur - Editor.

Syntax-Editor

Der Syntax-Editor dient zur Bearbeitung vorhandener und Erstellung neuer Syntaxdateien für das Syntaxhighlighting von AT HTML Editor 32. Jeder Abschnitt der Syntaxdatei ist mit genauen Erklärungen und Anweisungen versehen.

Funktionen der Buttons

Schließen	Schließt den Syntax - Editor.
Neu	Legt eine neue Syntaxdatei an.
Speichern	Speichert eine bearbeitete Syntaxdatei. Wurde eine neue Syntaxdatei angelegt, erscheint ein Dialog mit der Aufforderung, der Datei einen Namen zu geben. Dabei sollte darauf geachtet werden, einen Namen ohne Leerzeichen, Umlaute oder Sonderzeichen zu wählen.
Liste der Syntax-Dateien	Zeigt alle verfügbaren benutzerdefinierten Syntaxdateien an, aber keine Standardsyntaxmodi. Wird eine Syntaxdatei ausgewählt, wird diese in den Editor geladen.
Datei laden	Lädt die ausgewählte Syntaxdatei in den Editor.
Font +	Vergrößert die Schrift im Editor.
Font -	Verkleinert die Schrift im Editor.
fett	Formatiert die Schrift im Editor fett.
Statuszeile	Enthält verschiedene nützliche und unnützliche Informationenm bereit :-).

Tastenkombinationen und Kontextmenüs

Die Tastenkombinationen sind immer nur im sichtbaren Register sowie den angezeigten Dialogen gültig. Zu den einzelnen Controls kann auch mit der TAB-Taste gewechselt werden. Das Auslösen der Aktion erfolgt dann mit der Eingabetaste oder den Pfeiltasten.

Tastenkombinationen und Kontextmenüs des Projektexplorers

Die Tastenkombinationen sind immer nur im sichtbaren Menü, Register sowie den angezeigten Dialogen gültig. Zu den einzelnen Controls kann auch mit der TAB-Taste gewechselt werden. Das Auslösen der Aktion erfolgt dann mit der Eingabetaste oder den Pfeiltasten.

Menü "Datei"

Sitzungs-Manager - Zuletzt geöffnete Dateien und Projekte ...	ALT + Z
Neues Code-Editorfenster ...	ALT + N
Alle Code-Editorfenster speichern	ALT + A
Alle Code-Editorfenster schließen	ALT + C
Verzeichniswurzel setzen ...	ALT + V
Verzeichnisbaum aktualisieren	ALT + E
Beenden	ALT + B

Menü "Projekte"

Menü öffnen	ALT + P
Neu	ALT + N
Öffnen ...	ALT + Ö
Speichern	ALT + S
Speichern als ...	ALT + A
Gruppen hinzufügen ...	ALT + G
In der Liste markierte Datei hinzufügen	ALT + L
Dateien hinzufügen ...	ALT + D
Laufwerk/Verzeichnis/Netzwerkpfad hinzufügen ...	ALT + V
Strukturdatei hinzufügen	ALT + R
Markierten Eintrag aus dem Projekt entfernen	ALT + E
Projektnotizen anzeigen	ALT + P

Menü "Hilfe"

Menü öffnen	ALT + H
Hilfe ...	ALT + H
HTML - Referenz ...	ALT + R
Homepage AT HTML Editor 32 .: Forum :.	ALT + A
Supportseite von AT HTML Editor 32	ALT + V
Direkt zum Forum	ALT + F
Homepage Autor	ALT + U
Homepage AT Contenator	ALT + C
Homepage Projekt Proxis	ALT + P
Homepage SELFHTML	ALT + S
Homepage SELFPHP	ALT + H
Homepage JSP-DEVELOP	ALT + J
Info ...	ALT + I

Register "Dateien"

In das Register wechseln	ALT + N
Kontextmenü	Rechtsklick mit der Maus auf Eintrag

Register "Projekte"

In das Register wechseln	ALT + P
Kontextmenü Projektdateien	Rechtsklick mit der Maus in die Liste oder auf Eintrag
Kontextmenü Projektnotizen	Rechtsklick mit der Maus in das Editierfeld

Register "Config"

In das Register wechseln	ALT + C
Einstellungen speichern	ALT + E

Register "Run"

In das Register wechseln	ALT + R
Einstellungen speichern	ALT + E

Register "Plugins"

In das Register wechseln	ALT + L
In das Register "Programme" wechseln	ALT + M
In das Register "Plugins" wechseln	ALT + S

In allen Listen funktionieren die Pfeiltasten analog den Windows-Konventionen.

Tastenkombinationen und Kontextmenüs des Code-Editors

Die Tastenkombinationen sind immer nur im sichtbaren Register sowie den angezeigten Dialogen gültig. Zu den einzelnen Controls kann auch mit der TAB-Taste gewechselt werden. Das Auslösen der Aktion erfolgt dann mit der Eingabetaste oder den Pfeiltasten.

Navigationspfeile neben dem Register

Linkspfeil	ALT + 1
Rechtspfeil	ALT + 2

Register "Code"

In das Register wechseln	ALT + C
Schließen	ALT + F4
Vollbild/Normalbild	ALT + O
Neues Code-Editorfenster	ALT + N
Datei speichern	ALT + S
Datei speichern als	ALT + L
Drucken	ALT + K
gesamten Text markieren	ALT + A
markierten Text ausschneiden	STRG + X
markierten Text kopieren	STRG + C
Text am Cursor einfügen	STRG + V
Rückgängig	STRG + Z
Wiederherstellen	STRG + A
Farben-Manager ein/aus	ALT + F
Syntax-Manager ein/aus	ALT + E
Code-Manager ein/aus	ALT + T
Projektexplorer	ALT + X
Dialog "Text suchen"	STRG + F
Weitersuchen	F3
Dialog "Text suchen und erstezen"	STRG + ALT + F3
Fenster oben anordnen	ALT + 3
Fenster unten anordnen	ALT + 4
Fenster links anordnen	ALT + 5
Fenster rechts anordnen	ALT + 6
Run-Modus mit benutzerdefiniertem Host	ALT + Y
Run-Modus mit Ausgabekonsole	ALT + G
Run-Modus ohne Ausgabekonsole	ALT + R
Code-Manager	In entsprechenden Feldern (Hinzufügen) Eingabetaste
Markierten Text per Drag&Drop kopieren	STRG + gedrückte linke Maustaste
Markierten Text per Drag&Drop verschieben	gedrückte linke Maustaste
Fensterteiler	jeweils Doppelklick
Kontextmenü	Rechtsklick mit der Maus in den Codebereich

Register "Vorschau"

In das Register wechseln	ALT + V
Seite zurück blättern	ALT + Pfeiltaste links oder ALT + <
Seite vorwärts blättern	ALT + Pfeiltaste rechts oder ALT + SHIFT + >
Seite neu laden	ALT + N
Website starten	ALT + R
Kontextmenü	Rechtsklick mit der Maus in die Vorschau

Register "Bildviewer"

In das Register wechseln	ALT + B
Neue Verzeichniswurzel setzen	ALT + L
Farbe holen ...	ALT + F
Dialog Farbe holen / Farbe holen	Eingabetaste

Dialog Farbe holen / Farbe einfügen	ALT + Ü
Kontextmenü Dateisystem	Rechtsklick mit der Maus auf Eintrag

Register "Internet"

In das Register wechseln	ALT + I
Webseite anzeigen	ALT + A / Eingabetaste in Adressfeld
Anzeige stoppen	ALT + S
Seite zurück blättern	ALT + Pfeiltaste links oder ALT + <
Seite vorwärts blättern	ALT + Pfeiltaste rechts oder ALT + SHIFT + >
Seite neu laden	ALT + N

Register "Diverses"

In das Register wechseln	ALT + D
Sitzungs-Manager laden	ALT + T
Datei in den Dateispeicher legen	ALT + E
Zeilennummern ein/aus	ALT + Z
Alle Zeichen (Whitespaces) ein/aus	ALT + W
Optionen für Codebereich	ALT + O

Tastenkombinationen und Kontextmenüs des WYSIWYG-Editors

Schließen	ALT + H
Neu	ALT + N
Öffnen	ALT + Ö
Speichern als ...	ALT + A
Speichern	ALT + S
Browsen ein/aus	ALT + W
Bild ...	ALT + B
Tabelle ...	ALT + E
Listpunkt	ALT + L
Listpunkt - Zahl	ALT + Z
Einrücken	ALT + R
Ausrücken	ALT + U
Code anzeigen	ALT + C
Code speichern	ALT + O
HTML - Tags ...	ALT + T
WYSIWYG - Bereich: Alles markieren	STRG + A
WYSIWYG - Bereich: Text fett ein/aus	STRG + B
WYSIWYG - Bereich: markierten Text kopieren	STRG + C
WYSIWYG - Bereich: Suchen-Dialog	STRG + F
WYSIWYG - Bereich: Text kursiv ein/aus	STRG + I
WYSIWYG - Bereich: setzt über den markierten Text einen Hyperlink	STRG + L
WYSIWYG - Bereich: Absatz einfügen	STRG + M
WYSIWYG - Bereich: Dokument per Dialog ausdrucken	STRG + P
WYSIWYG - Bereich: Blockqote setzen (Absatz einrücken)	STRG + T
WYSIWYG - Bereich: Text unterstreichen ein/aus	STRG + T
WYSIWYG - Bereich: Text an Cursorposition einfügen	STRG + V
WYSIWYG - Bereich: markierten Text ausschneiden	STRG + X
WYSIWYG - Bereich: Rückgängig	STRG + Y
WYSIWYG - Bereich: Wiederherstellen	STRG + Z
Vertikale Größenänderung der Bereiche	Mit gedrückter linker Maustaste Trennbalken zwischen den Bereichen ziehen
Dialog HTML - Tags / Tag hinzufügen	Eingabetaste im Hinzufügefeld
Markierten Text per Drag&Drop kopieren	STRG + gedrückte linke Maustaste
Markierten Text per Drag&Drop verschieben	gedrückte linke Maustaste
Kontextmenü im Code - Bereich	Rechtsklick mit der Maus in diesen Bereich

Alle Tastenkombinationen im WYSIWYG - Bereich stehen nur dann zur Verfügung, wenn der Browsemode aus ist.

Weitere undokumentierte Tastenkombinationen können vorhanden sein.

Tastenkombinationen und Kontextmenüs des Struktur-Editors

Menü Datei öffnen	ALT + D
Menü Datei: Neu	ALT + N
Menü Datei: Öffnen	ALT + Ö
Menü Datei: Speichern als ...	ALT + A
Menü Datei: Speichern	ALT + S
Menü Datei: Drucken ...	ALT + D
Menü Datei: Schließen	ALT + C
Markierten Text ausschneiden	STRG + X
Markierten Text kopieren	STRG + C
Text an Cursorposition einfügen	STRG + V
Markierten Text per Drag&Drop kopieren	STRG + gedrückte linke Maustaste
Markierten Text per Drag&Drop verschieben	gedrückte linke Maustaste
Kontextmenü	Rechtsklick mit der Maus in den Editorbereich

Tastenkombinationen und Kontextmenüs des Syntax-Editors

Schließen	ALT + S
Neu	ALT + N
Speichern	ALT + E
Datei laden	ALT + D

Beschreibung der Plugin-Schnittstelle

Einleitung

Mit dieser Plugin-Schnittstelle habe ich etwas fabriziert, was eigentlich so gar nicht geht (wenn man den vielen Newsgroups und "Experten" glauben schenken will). AT HTML Editor 32 ist ein reines in Visual Basic programmiertes Programm. Dennoch dachte ich mir, daß eine Erweiterung mit selbstprogrammierten Komponenten durch den Benutzer sinnvoll wäre, so, wie es auch andere Programme ermöglichen. Ich hatte verschiedene Konzepte entwickelt, so u.a. auch eine Variante reiner textbasierender Plugins, die auch der erstellen kann, der nicht der DLL-Programmierung mächtig wäre. Dennoch habe ich mich für ein DLL-basierendes Interface entschieden, weil zum einen u.a. auch ein textbasierendes Plugin aufgesetzt werden kann und zum anderen die Implementierbarkeit von weiterer (beliebiger) Funktionalität DLL-basierend wesentlich besser möglich und flexibler ist. Also habe ich meinem VB-Programm ein flexibles Plug-Interface spendiert, daß ganz "normale" DLLs dynamisch (!) zur Laufzeit lädt und entlädt. Und somit habe ich Visual Basic mit Delphi- und/oder C++-DLLs verheiratet. Dagegen funktionieren ActiveX-DLLs mit diesem Interface nicht. PureBasic-DLLs funktionieren nur eingeschränkt bzw. erlauben nur den Aufruf ohne Datenübergabe.

Mögliche Probleme

Die Tatsache, daß ich hier ein Visual-Basic-Programm mit einer Plugin-Schnittstelle zum dynamischen Anbinden von DLLs versehen habe, birgt einige Probleme in sich. Insbesondere, was die Übergabe von Text aus einem Plugin heraus in einen Code-Editor betrifft. VB hat nicht ein so ausgefeiltes Speicher-Management wie z.B. Delphi oder C++. Werden Zeichenmengen von mehr als 1200 Zeichen (1200 Byte) vom Plugin zurückgegeben, kann das zum Absturz führen. Aber es ist auch möglich, daß das erst bei Mengen von mehr als 10000 Zeichen (10000 Byte) passiert. Das scheint von System zu System unterschiedlich zu sein, deshalb habe ich mich entschlossen, die Zeichenmenge nicht zu begrenzen.
Desweiteren kann es zu Schwierigkeiten bei der Übergabe von Dateinamen aus einem Plugin kommen, die dann von AT HTML Editor 32 geöffnet werden soll. Kurze Dateinamen führen öfter zu diesen Schwierigkeiten als lange Dateinamen. Aber auch hier war das von System zu System unterschiedlich.
Wohlgemerkt, diese Probleme sind nicht in erster Linie fehlerhafte Programmierung, sondern auf das Verhalten von VB-Programmen zu Delphi- oder anderen Programmen zurückzuführen, weil ich hier ein VB-Programm mit Delphi- und anderen Programmen quasi "verheiratet" habe.

Architektur

Jeder wird dafür Verständnis haben, daß ich hier nicht alle technische Einzelheiten darlegen werde. Das würde auch den Rahmen sprengen, zumal diese Informationen auch gar nicht notwendig wären zum Schreiben eigener Plugins. Dennoch soll hier kurz das Konzept der Architektur angesprochen werden, um Entwicklern das Verständnis zu erleichtern.

Zwei Dinge sind wichtig für das Funktionieren:
1. AT HTML Editor sucht alle Plugins (ob EXE oder DLL) im Unterverzeichnis "plugins".
2. Das Geheimnis der dynamischen DLL-Anbindung steckt zum Teil in der pluginwrapper.dll im Programmverzeichnis.

Dadurch, daß die Plugin-Schnittstelle teilweise selbst ein Plugin ist, kann sie jederzeit geupdatet werden, ohne AT HTML Editor 32 erneut anpassen zu müssen. Wenn man so will, ist auch hier das seit AT HTML Editor 32.3 umgesetzte modulare Konzept der Komponenten konsequent angewandt.

Rückgabetypen

Die Plugin-Schnittstelle kennt 4 Rückgabetypen.

1. Markierter Text zum Plugin
Hiermit wird der im Code-Editor markierte Text an das Plugin zur weiteren Bearbeitung übergeben ohne daß das Plugin diesen Text wieder zurück gibt. Das ist z.B. dann sinnvoll, wenn der Text anderweitig weiterverarbeitet werden soll.

2. Markierter Text zum/vom Plugin
Hiermit wird der im Code-Editor markierte Text an das Plugin zur weiteren Bearbeitung übergeben. Nach Bearbeitung durch das Plugin erfolgt eine Rückgabe des bearbeiteten Textes. Dieser ersetzt dann den im Code-Editor markierten Text. Ist kein Text markiert, wird der Text aus dem Plugin an der Cursorposition zusätzlich im Code-Editor eingefügt. Das ist dann nützlich, wenn z.B. per Plugin eine Tabelle oder Frameset generiert werden soll.

3. Dateiname zum Plugin
Hier wird lediglich der Dateiname des aktuellen Code-Editors an das Plugin übergeben. Das ist z.B. dann nützlich, wenn man eine bearbeitete Datei mit einem kleinen FTP-Uploader automatisch schnell mal in das FTP-Verzeichnis "schieben" möchte. Auch wäre denkbar, hiermit einen kleinen externen Debugger oder Syntaxchecker zu realisieren.

4. Dateiname vom Plugin
Hiermit ergibt sich die Möglichkeit, z.B. aus einem externen Dateimanager oder FTP-Downloader schnell die übergebene Datei in AT HTML Editor 32 zur Bearbeitung zu öffnen.

Ein Plugin für AT HTML Editor 32.3 schreiben

Am besten läßt sich das anhand von praktischen Beispielen erklären. Im nachfolgenden wird jeder Rückgabtyp als Delphi-Code gezeigt. Im Code selber sind genügend ausreichend Erklärungen, die jedem versierten Programmierer genügen sollten. Besteht trotzdem noch weiterer Erklärungsbedarf, dann entweder auf die **Homepage von AT HTML Editor 32** schauen oder mich einfach per **E-Mail** kontaktieren.

1. **Markierter Text zum Plugin**
2. **Markierter Text zum/vom Plugin**
3. **Dateiname zum Plugin**
4. **Dateiname vom Plugin**

Damit Plugins auch ganz gefahrlos außerhalb von AT HTML Editor 32.3 getestet werden können, gibt es auf der **Homepage von AT HTML Editor 32** einen Plugin-Tester, mit dem das selbstentwickelte Plugin eben getestet werden kann. Der Plugin-Tester ist ebenfalls ein VB-Programm mit zum AT HTML Editor 32.3 identischer Plugin-Schnittstelle.

Plugin-Funktionen

```
library formdll;
uses
  SysUtils,
  Classes,
  Forms,
  Windows,
  Dialogs,
  DLLForm in 'DLLForm.pas' {frmMain};
{$R *.res}

{###############################################################################################
{###############################################################################################

{                       GÜLTIG AB AT HTML EDITOR 32.3

{###############################################################################################
{###############################################################################################

//---------------------------------------------------------------------
//  DIE MAX. GRÖßE EINES STRINGS, DER ZU VB OHNE FEHLER ZURÜCKGEGEBEN
//  WERDEN KANN, IST 1200 ZEICHEN (1200 BYTE - SCHON MAL 10000 BYTE) -
//  SPEICHERHANDLING VON VB
//  DAGEGEN KANN EIN STRING VON MAX. 64000 ZEICHEN PROBLEMLOS
//  ÜBERNOMMEN WERDEN
//---------------------------------------------------------------------
function pluginrun(text: PChar) : PChar; stdcall;
var
  Form : TfrmMain;      //FORM
  runresult : Integer;  //DUMMYRESULT

begin
  Form := TfrmMain.Create(Application);  //FORM AUFBAUEN
  Form.Memo.Text := text;                //AUS HAUPTANWENDUNG ÜBER DEN WRAPPER ÜBERGEBENEN STRI
  runresult := Form.ShowModal;           //FORM MODAL! ANZEIGEN

//*************************************************************************************
//  RÜCKGABE-TYPEN -> UNBEDINGT BEACHTEN -> MUß MIT RÜCKGABETYPEN IN DER FUNKTION pluginbackval
//  ÜBEREINSTIMMEN !!!
//*************************************************************************************
//-------------------------------------------------------------------------------------
//WENN EINE WERTERÜCKGABE AN AT HTML EDITOR ERFOLGEN SOLL, MUß DIESE result-ZEILE AKTIVIERT
//WERDEN UND DIE NACHFOLGENDE AUSKOMENTIERT
//BETRIFFT result := 1; UND result := 2; IN DER FUNKTION pluginbackvalue
//-------------------------------------------------------------------------------------
  //result := Pchar(backstring);           //INHALT DER GLOBALEN VARIABLE ZURÜCKGEBEN
                                           //backstring SOLLTE IMMER EINE VARIABLE VOM TYP stri
//-------------------------------------------------------------------------------------
//WENN KEINE RÜCKGABE ERFOLGEN SOLL, MUß DIESE result-ZEILE AKTIVIERT WERDEN UND DIE OBER
//AUSKOMMENTIERT
//BETRIFFT result := 0; UND result := 3; IN DER FUNKTION pluginbackvalue
//-------------------------------------------------------------------------------------
  result := nil;                        //NICHTS ZURÜCKGEBEN
//*************************************************************************************
  Form.Free;                            //FORM FREI GEBEN - ENTLADEN
end;

{###############################################################################################
{###############################################################################################

//---------------------------------------------------------------------
//  VARIABLE, DIE FESTLEGT, OB EINE WERTERÜCKGABE ERFOLGEN SOLL ODER
//  NICHT - ALSO NUR TEXT WIRD ZUR WEITERVEARBEITUNG ÜBERNOMMEN
//---------------------------------------------------------------------
function pluginbackvalue : Integer; stdcall;
begin

//*************************************************************************
//MITTELS DER RÜCKGABETYPEN KANN DAS VERHALTEN VON AT HTML EDITOR BEEINFLUßT WERDEN
//*************************************************************************
```

```
//  RÜCKGABETYP  |        ERKLÄRUNG                  | EINSTELLUNG IN ANDERER FUNKTION
//-------------------------------------------------------------------------------------------
//  result := 0; | ES WIRD KEIN WERT ZURÜCKGEGEBEN   | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 1; | TEXT WIRD ZURÜCKGEGEBEN           | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 2; | DATEINAME WIRD ZURÜCKGEGEBEN      | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 3; | DATEINAME WIRD AN PLUGIN ÜBERGEBEN| FUNKTION: pluginrun, RÜCKGABE: result :=

//  ZU BEACHTEN IST, DAß BEI result := 2; DIE DATEI MIT DEM KOMPLETTEN PFAD ZURÜCKGEGEBEN WERDE
//  WENN TEXT AN DAS PLUGIN ÜBERGEBEN WERDEN SOLL, MUß DIESER ZUVOR MARKIERT WERDEN. WIRD KEIN
//  MARKIERT, WIRD AUCH KEIN TEXT AN DAS PLUGIN ÜBERGEBEN !

  result := 0;
  //result := 1;
  //result := 2;
  //result := 3;

end;
{#############################################################################################
{#############################################################################################
//--------------------------------------------------------------------
//  NAME DES PLUGINS
//--------------------------------------------------------------------
function pluginname:PChar; stdcall;
begin
  result := PChar('Text-Editor!');
end;

//--------------------------------------------------------------------
//  KURZE BESCHREIBUNG DER FUNKTIONALITÄT DES PLUGINS
//--------------------------------------------------------------------
function plugindescription : PChar; stdcall;
begin
  result := PChar('Übernimmt den selektierten Text in einen kleinen Editor.');
end;

{#############################################################################################
{#############################################################################################

//--------------------------------------------------------------------
//  EXPORTIERT DIE FUNKTION, DIE DURCH DAS EXTERNE PROGRAMM
//  ANGESPROCHEN WERDEN SOLLEN
//--------------------------------------------------------------------
exports
  pluginrun,
  pluginname,
  plugindescription,
  pluginbackvalue;

{#############################################################################################
{#############################################################################################

begin

end.
```

Dazugehörige Form

```
unit DLLForm;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TfrmMain = class(TForm)
    Memo: TMemo;
    btnOk: TButton;
    procedure btnOkClick(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;
var
```

```
  frmMain: TfrmMain;   //FORM DEKLARIEREN
  backstring : string;  //GLOBALEN RÜCKGABESTRING DEKLARIEREN, FALLS ER AUAS VERSCHIEDENEN
                       //TEXTFELDERN ZUSAMMENGESETZT WERDEN SOLL
implementation
{$R *.dfm}
procedure TfrmMain.btnOkClick(Sender: TObject);
begin
  backstring := Memo.Text; //IN GLOABLE VARIABLE SETZEN
                                //SO KÖNNEN DER GLOBALEN VARIABLE WEITERE WERT HINZUGEFÜGT WE
  Close;                        //FORMULAR SCHLIESSEN
end;
end.
```

Plugin-Funktionen

```
library textdll;
uses
  SysUtils,
  Classes,
  Types,
  Dialogs;
{$R *.res}
var
  strPuffer : string;  //DIE STRINGVARIABLE MUß GLOBAL SEIN, UM EINE GRÖßERE TEXTMENGE ZURÜCKGE
{#############################################################################################
{#############################################################################################
{                       GÜLTIG AB AT HTML EDITOR 32.3
{#############################################################################################
{#############################################################################################
//---------------------------------------------------------------
//  BEARBEITET DEN STRING OHNE FORMULAR UND GIBT IHN ZURÜCK
//  DIE MAX. GRÖßE EINES STRINGS, DER ZU VB OHNE FEHLER ZURÜCKGEGEBEN
//  WERDEN KANN, IST 1200 ZEICHEN (1200 BYTE - SCHON MAL 10000 BYTE) -
//  SPEICHERHANDLING VON VB
//  DAGEGEN KANN EIN STRING VON MAX. 64000 ZEICHEN PROBLEMLOS
//  ÜBERNOMMEN WERDEN
//---------------------------------------------------------------
function pluginrun(text:PChar):PChar; stdcall;
begin
//*************************************************************************************************
//  RÜCKGABE-TYPEN -> UNBEDINGT BEACHTEN -> MUß MIT RÜCKGABETYPEN IN DER FUNKTION pluginbackval
//  ÜBEREINSTIMMEN !!!
//*************************************************************************************************
//-------------------------------------------------------------------------------------------------
//WENN EINE WERTERÜCKGABE AN AT HTML EDITOR ERFOLGEN SOLL, MUß DIESE result-ZEILE AKTIVIERT
//WERDEN UND DIE NACHFOLGENDE AUSKOMENTIERT
//BETRIFFT result := 1; UND result := 2; IN DER FUNKTION pluginbackvalue
//-------------------------------------------------------------------------------------------------
  strPuffer := '<html><body>' + #13#10 + text + #13#10 + '</body></html>'; //VORHER IN EINE GLO
                                                                           //SONST ABSTURZ BEI
  result := PChar(strPuffer);           //WERT ZURÜCKGEBEN
//-------------------------------------------------------------------------------------------------
//WENN KEINE RÜCKGABE ERFOLGEN SOLL, MUß DIESE result-ZEILE AKTIVIERT WERDEN UND DIE OBER
//AUSKOMMENTIERT
//BETRIFFT result := 0; UND result := 3; IN DER FUNKTION pluginbackvalue
//-------------------------------------------------------------------------------------------------
  //result := nil;                       //NICHTS ZURÜCKGEBEN
//*************************************************************************************************
end;
{#############################################################################################
{#############################################################################################
//---------------------------------------------------------------
//  VARIABLE, DIE FESTLEGT, OB EINE WERTERÜCKGABE ERFOLGEN SOLL ODER
//  NICHT - ALSO NUR TEXT WIRD ZUR WEITERVEARBEITUNG ÜBERNOMMEN
//---------------------------------------------------------------
function pluginbackvalue : Integer; stdcall;
begin
//*************************************************************************
//MITTELS DER RÜCKGABETYPEN KANN DAS VERHALTEN VON AT HTML EDITOR BEEINFLUßT WERDEN
//*************************************************************************
//  RÜCKGABETYP  |        ERKLÄRUNG                  | EINSTELLUNG IN ANDERER FUNKTION
//-------------------------------------------------------------------------------------------------
//  result := 0; | ES WIRD KEIN WERT ZURÜCKGEGEBEN   | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 1; | TEXT WIRD ZURÜCKGEGEBEN           | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 2; | DATEINAME WIRD ZURÜCKGEGEBEN      | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 3; | DATEINAME WIRD AN PLUGIN ÜBERGEBEN| FUNKTION: pluginrun, RÜCKGABE: result :=
//  ZU BEACHTEN IST, DAß BEI result := 2; DIE DATEI MIT DEM KOMPLETTEN PFAD ZURÜCKGEGEBEN WERDE
//  WENN TEXT AN DAS PLUGIN ÜBERGEBEN WERDEN SOLL, MUß DIESER ZUVOR MARKIERT WERDEN. WIRD KEIN
//  MARKIERT, WIRD AUCH KEIN TEXT AN DAS PLUGIN ÜBERGEBEN !
  //result := 0;
  result := 1;
  //result := 2;
  //result := 3;
end;
{#############################################################################################
{#############################################################################################
//---------------------------------------------------------------
//  NAME DES PLUGINS
```

```
//-----------------------------------------------------------------------
function pluginname : PChar; stdcall;
begin
  result := PChar('HTML-Body');
end;
//-----------------------------------------------------------------------
//  KURZE BESCHREIBUNG DER FUNKTIONALITÄT DES PLUGINS
//-----------------------------------------------------------------------
function plugindescription : PChar; stdcall;
begin
  result := PChar('Bettet den selektierten Text in einen HTML-Body ein, so daß sie als HTML-Sei
end;
{#######################################################################################################
{#######################################################################################################
//-----------------------------------------------------------------------
//  EXPORTIERT DIE FUNKTION, DIE DURCH DAS EXTERNE PROGRAMM
//  ANGESPROCHEN WERDEN SOLLEN
//-----------------------------------------------------------------------
exports
  pluginrun,
  pluginname,
  plugindescription,
  pluginbackvalue;
{#######################################################################################################
{#######################################################################################################
begin
end.
```

Plugin-Funktionen

```
library filefromeditor;
uses
  SysUtils,
  Classes,
  Types,
  Dialogs;
{$R *.res}
{#############################################################################################
{#############################################################################################
{                       GÜLTIG AB AT HTML EDITOR 32.3
{#############################################################################################
{#############################################################################################
//---------------------------------------------------------------
//  BEARBEITET DEN STRING OHNE FORMULAR UND GIBT IHN ZURÜCK
//  DIE MAX. GRÖßE EINES STRINGS, DER ZU VB OHNE FEHLER ZURÜCKGEGEBEN
//  WERDEN KANN, IST 1200 ZEICHEN (1200 BYTE - SCHON MAL 10000 BYTE) -
//  SPEICHERHANDLING VON VB
//  DAGEGEN KANN EIN STRING VON MAX. 64000 ZEICHEN PROBLEMLOS
//  ÜBERNOMMEN WERDEN
//---------------------------------------------------------------
function pluginrun(text:PChar):PChar; stdcall;
begin
//*************************************************************************************
//  RÜCKGABE-TYPEN -> UNBEDINGT BEACHTEN -> MUß MIT RÜCKGABETYPEN IN DER FUNKTION pluginbackval
//  ÜBEREINSTIMMEN !!!
//*************************************************************************************
//-------------------------------------------------------------------------------------
//WENN EINE WERTERÜCKGABE AN AT HTML EDITOR ERFOLGEN SOLL, MUß DIESE result-ZEILE AKTIVIERT
//WERDEN UND DIE NACHFOLGENDE AUSKOMENTIERT
//BETRIFFT result := 1; UND result := 2; IN DER FUNKTION pluginbackvalue
//-------------------------------------------------------------------------------------
  //result := PChar('Ein bearbeiteter Text!');           //WERT ZURÜCKGEBEN
                                                         //SOLLTE IMMER EINE ZEICHENFOLGE VOM T
//-------------------------------------------------------------------------------------
//WENN KEINE RÜCKGABE ERFOLGEN SOLL, MUß DIESE result-ZEILE AKTIVIERT WERDEN UND DIE OBER
//AUSKOMMENTIERT
//BETRIFFT result := 0; UND result := 3; IN DER FUNKTION pluginbackvalue
//-------------------------------------------------------------------------------------
  ShowMessage('Dateiname ... '#13#10#13#10 + text + #13#10#13#10' ... erfolgreich an das Plugir
  result := nil;                        //NICHTS ZURÜCKGEBEN
//*************************************************************************************
end;
{#############################################################################################
{#############################################################################################
//---------------------------------------------------------------
//  VARIABLE, DIE FESTLEGT, OB EINE WERTERÜCKGABE ERFOLGEN SOLL ODER
//  NICHT - ALSO NUR TEXT WIRD ZUR WEITERVEARBEITUNG ÜBERNOMMEN
//---------------------------------------------------------------
function pluginbackvalue : Integer; stdcall;
begin
//*************************************************************************
//MITTELS DER RÜCKGABETYPEN KANN DAS VERHALTEN VON AT HTML EDITOR BEEINFLUßT WERDEN
//*************************************************************************
//  RÜCKGABETYP  |        ERKLÄRUNG                 | EINSTELLUNG IN ANDERER FUNKTION
//-------------------------------------------------------------------------------------
//  result := 0; | ES WIRD KEIN WERT ZURÜCKGEGEBEN   | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 1; | TEXT WIRD ZURÜCKGEGEBEN           | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 2; | DATEINAME WIRD ZURÜCKGEGEBEN      | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 3; | DATEINAME WIRD AN PLUGIN ÜBERGEBEN| FUNKTION: pluginrun, RÜCKGABE: result :=
//  ZU BEACHTEN IST, DAß BEI result := 2; DIE DATEI MIT DEM KOMPLETTEN PFAD ZURÜCKGEGEBEN WERDE
//  WENN TEXT AN DAS PLUGIN ÜBERGEBEN WERDEN SOLL, MUß DIESER ZUVOR MARKIERT WERDEN. WIRD KEIN
//  MARKIERT, WIRD AUCH KEIN TEXT AN DAS PLUGIN ÜBERGEBEN !
  //result := 0;
  //result := 1;
  //result := 2;
  result := 3;
end;
{#############################################################################################
{#############################################################################################
//---------------------------------------------------------------
//  NAME DES PLUGINS
//---------------------------------------------------------------
function pluginname : PChar; stdcall;
```

```
begin
  result := PChar('Dateiname');
end;
//---------------------------------------------------------------------
//  KURZE BESCHREIBUNG DER FUNKTIONALITÄT DES PLUGINS
//---------------------------------------------------------------------
function plugindescription : PChar; stdcall;
begin
  result := PChar('Übernimmt einen von AT HTML Editor übergebenen Dateinamen, der im Plugin wei
end;
{#############################################################################################
{#############################################################################################
//---------------------------------------------------------------------
//  EXPORTIERT DIE FUNKTION, DIE DURCH DAS EXTERNE PROGRAMM
//  ANGESPROCHEN WERDEN SOLLEN
//---------------------------------------------------------------------
exports
  pluginrun,
  pluginname,
  plugindescription,
  pluginbackvalue;
{#############################################################################################
{#############################################################################################
begin
end.
```

Plugin-Funktionen

```
library filedll;
uses
  SysUtils,
  Classes,
  Forms,
  Windows,
  Dialogs,
  DLLForm in 'DLLForm.pas' {frmMain};

{$R *.res}

{#############################################################################################
{#############################################################################################

{                       GÜLTIG AB AT HTML EDITOR 32.3

{#############################################################################################
{#############################################################################################
//---------------------------------------------------------------------
//  BEARBEITET DEN STRING OHNE FORMULAR UND GIBT IHN ZURÜCK
//  DIE MAX. GRÖßE EINES STRINGS, DER ZU VB OHNE FEHLER ZURÜCKGEGEBEN
//  WERDEN KANN, IST 1200 ZEICHEN (1200 BYTE - SCHON MAL 10000 BYTE) -
//  SPEICHERHANDLING VON VB
//  DAGEGEN KANN EIN STRING VON MAX. 64000 ZEICHEN PROBLEMLOS
//  ÜBERNOMMEN WERDEN
//---------------------------------------------------------------------
function pluginrun(text: PChar) : PChar; stdcall;
var
  Form : TfrmMain;      //FORM
  runresult : Integer;  //DUMMYRESULT

begin
  Form := TfrmMain.Create(Application);  //FORM AUFBAUEN
  //Form.Memo.Text := text;                //AUS HAUPTANWENDUNG ÜBER DEN WRAPPER ÜBERGEBENEN ST
  runresult := Form.ShowModal;           //FORM MODAL! ANZEIGEN

//*************************************************************************************
//  RÜCKGABE-TYPEN -> UNBEDINGT BEACHTEN -> MUß MIT RÜCKGABETYPEN IN DER FUNKTION pluginbackval
//  ÜBEREINSTIMMEN !!!
//*************************************************************************************
//-------------------------------------------------------------------------------------
//WENN EINE WERTERÜCKGABE AN AT HTML EDITOR ERFOLGEN SOLL, MUß DIESE result-ZEILE AKTIVIERT
//WERDEN UND DIE NACHFOLGENDE AUSKOMENTIERT
//BETRIFFT result := 1; UND result := 2; IN DER FUNKTION pluginbackvalue
//-------------------------------------------------------------------------------------
  result := Pchar(backstring);           //INHALT DER GLOBALEN VARIABLE ZURÜCKGEBEN
                                         //backstring SOLLTE IMMER EINE VARIABLE VOM TYP string
//-------------------------------------------------------------------------------------
//WENN KEINE RÜCKGABE ERFOLGEN SOLL, MUß DIESE result-ZEILE AKTIVIERT WERDEN UND DIE OBER
//AUSKOMMENTIERT
//BETRIFFT result := 0; UND result := 3; IN DER FUNKTION pluginbackvalue
//-------------------------------------------------------------------------------------
  //result := nil;                       //NICHTS ZURÜCKGEBEN
//*************************************************************************************
  Form.Free;                             //FORM FREI GEBEN - ENTLADEN
end;

{#############################################################################################
{#############################################################################################


//---------------------------------------------------------------------
//  VARIABLE, DIE FESTLEGT, OB EINE WERTERÜCKGABE ERFOLGEN SOLL ODER
//  NICHT - ALSO NUR TEXT WIRD ZUR WEITERVEARBEITUNG ÜBERNOMMEN
//---------------------------------------------------------------------
function pluginbackvalue : Integer; stdcall;
begin

//*************************************************************************
//MITTELS DER RÜCKGABETYPEN KANN DAS VERHALTEN VON AT HTML EDITOR BEEINFLUßT WERDEN
//*************************************************************************
```

```
//  RÜCKGABETYP  |        ERKLÄRUNG                  | EINSTELLUNG IN ANDERER FUNKTION
//-------------------------------------------------------------------------------------------
//  result := 0; | ES WIRD KEIN WERT ZURÜCKGEGEBEN   | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 1; | TEXT WIRD ZURÜCKGEGEBEN           | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 2; | DATEINAME WIRD ZURÜCKGEGEBEN      | FUNKTION: pluginrun, RÜCKGABE: result :=
//  result := 3; | DATEINAME WIRD AN PLUGIN ÜBERGEBEN| FUNKTION: pluginrun, RÜCKGABE: result :=

//  ZU BEACHTEN IST, DAß BEI result := 2; DIE DATEI MIT DEM KOMPLETTEN PFAD ZURÜCKGEGEBEN WERDE
//  WENN TEXT AN DAS PLUGIN ÜBERGEBEN WERDEN SOLL, MUß DIESER ZUVOR MARKIERT WERDEN. WIRD KEIN
//  MARKIERT, WIRD AUCH KEIN TEXT AN DAS PLUGIN ÜBERGEBEN !

  //result := 0;
  //result := 1;
  result := 2;
  //result := 3;
end;

{#############################################################################################
{#############################################################################################
//--------------------------------------------------------------------
//  NAME DES PLUGINS
//--------------------------------------------------------------------
function pluginname:PChar; stdcall;
begin
  result := PChar('Datei-Manager!');
end;

//--------------------------------------------------------------------
//  KURZE BESCHREIBUNG DER FUNKTIONALITÄT DES PLUGINS
//--------------------------------------------------------------------
function plugindescription : PChar; stdcall;
begin
  result := PChar('Gibt einen ausgwählten Dateinamen zurück.');
end;

{#############################################################################################
{#############################################################################################

//--------------------------------------------------------------------
//  EXPORTIERT DIE FUNKTION, DIE DURCH DAS EXTERNE PROGRAMM
//  ANGESPROCHEN WERDEN SOLLEN
//--------------------------------------------------------------------
exports
  pluginrun,
  pluginname,
  plugindescription,
  pluginbackvalue;

{#############################################################################################
{#############################################################################################

begin

end.
```

Dazugehörige Form

```
unit DLLForm;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, FileCtrl;
type
  TfrmMain = class(TForm)
    Button1: TButton;
    FileListBox1: TFileListBox;
    procedure Button1Click(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;
var
```

```
  frmMain: TfrmMain;
  backstring : string; //DIE STRINGVARIABLE MUß GLOBAL SEIN, UM EINE GRÖßERE TEXTMENGE ZURÜCKGE
implementation
{$R *.dfm}
procedure TfrmMain.Button1Click(Sender: TObject);
begin
  backstring := FileListBox1.FileName;  //DATEINAME ZURÜCKGEGEBEN
  Close;
end;
end.
```