ClassAction

Gasmi Salim

COLLABORATORS			
	TITLE : ClassAction		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Gasmi Salim	August 22, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ClassAction

1.1 ClassAction 2.8 Guide

```
                    ClassAction
                    Version 2.8


~Was~ist~ClassAction~~~~~~~~   Dies zuerst lesen
~Systemanforderungen~~~~~~~~   Was du brauchst

~Installation~~~~~~~~~~~~~~~   Wie installiert man ClassAction
~Nach~der~Installation~~~~~~   Installation Teil II

~Benutzung~von~ClassAction~~   Wie benutzt man dieses Programm

~Konfiguration~~~~~~~~~~~~~~   Ändern der ToolTypes
~ClassAction~Prefs~~~~~~~~~~   Konfigurieren der Klasses und Aktionen

~Die~Arexx~Kommandos~~~~~~~~   Der ARexx Port und Kommandos

~Zukünftige~Verbesserungen~~   Was sollte in der Zukunft noch kommen
~Geschichte~&~Eigenschaften~   Seit dem Anfang...
~Registrierung~~~~~~~~~~~~~~   Warum und wie registrieren

~Rechtliches~~~~~~~~~~~~~~~~
~Der~Autor~~~~~~~~~~~~~~~~~~
~Grüße~~~~~~~~~~~~~~~~~~~~~~
~Bekannte~Fehler~~~~~~~~~~~~
```

1.2 Was ist ClassAction ?

```
ClassAction ist ein kleines Programm, welches das Leben aller
Festplattennutzer vereinfacht.

Wenn du eine Festplatte besitzt, hast du auch eine Menge Dateien drauf :
ausführbare, Module, Bilder, Quellcodes, Animationen, Sounddateien...

ClassAction stellt für dich die Art der gewählten Datei fest und gibt
```

eine Liste der möglichen Aktionen mit dieser Datei an.

Zum Beispiel, wenn du ein GIF Bild wählst, wird es als solches erkannt
und eine Liste der möglichen Aktionen, wie 'Anzeigen' oder 'Ändern'
erscheint.

ClassAction ist sehr gut konfigurierbar, man kann seine eigenen Klassen
und Aktionen hinzufügen.
Aktionen benutzen externe Programme, so kann man seinen bevorzugten
GIF-Anzeiger einbinden (oder ähnliches).

ClassAction hat ein AppIcon, einen Arexx Port, ist lokalisiert und ein
Commodity.

ClassAction nutzt die xfdmaster.library, um Dateien zu entpacken. Durch
diese Eigenschaft können sogar gepackte Klassen erkannt werden.

ClassAction nutzt sehr wenig Speicher und CPU Zeit.

Wenn du dir mit der Konfiguration von ClassAction Zeit nimmst, kannst du
alles damit tun !!! Es ist ein einfacher Weg, mit Dateien umzugehen.

Bevor du es löschst, probier' es aus !!!!!!

Falls du mehr über ClassAction wissen möchtest, versuche ~Mehr~.

1.3 Mehr über ClassAction

Dieses Paket ist Shareware. Du kannst es frei probieren !!! und verteil'
es, solange du nichts änderst. ABER wenn du es regelmäßig nutzt, mußt du
dich registrieren lassen !!!

PD-Anbieter dürfen das ClassAction Paket in ihre Sammlung aufnehmen, wenn
sie mich davon unterrichten.

Dies ist die Version 2.8 von ClassAction
Wenn es dir gefällt und du es nutzt, MUßT du dich ~registrieren~.

Ich hoffe, du findest es nützlich.
(denn ich schon :])

Zu technischen Infos über ClassAction lies ~Technische~Infos~.

1.4 Systemanforderungen

Um ClassAction zu nutzen, brauchst du folgendes :

o Amiga OS 3.0 oder höher
(Vielleicht werde ich eine OS 2.0 Version veröffentlichen)

o eine Festplatte
(ClassAction ist ohne nutzlos)

Das ist alles, Kumpels !!!

1.5 Konfiguration von ClassAction

Die ClassAction Schnittstelle und Eigenschaften sind über ToolTypes einstellbar.

Um ein Tooltype zu ändern, wähle das Piktogramm von ClassAction und wähle den Punkt Information aus dem Menü Piktogramme von der Workbench.

Hier eine Liste aller Tooltypes :

Wichtig : Wenn ein ToolType nicht gefunden wird, wird der voreingestellte Wert genutzt.

==========

DONOTWAIT

Diesen Tooltype NICHT entfernen, denn er ist notwendig, wenn ClassAction aus WBStartup geladen wird.

==========

CX_PRIORITY

Dies stellt die Priorität von ClassAction ein.

Vorgabewert ist : 0

==========

CX_HOTKEY

Mit dieser Taste kann man ClassAction anzeigen/verbergen.

wenn man sie drückt, während ClassAction verborgen oder ein AppIcon ist, öffnet sich das Fenster, wenn man sie drückt, während das Fenster offen ist, wird ClassAction verborgen.

Voreingestellt ist : LALT C (Linke Alt + Shift + c)

==========

REQBUG

Wenn du Programme wie MagicMenu (oder andere) benutzt, können
Fehler mit den REQ Kommandos im AppIcon-Modus auftreten, das
System kann womöglich abstürzen...
wenn du diesen Tooltype auf YES setzt (d.h. REQBUG=YES)
transformiert ClassAction die REQV,REQF,REQD Kommandos in ein
REQT Kommando, falls die Aktion mittels des AppIcons ausgelöst
wird.

Vorgabe ist : NO

==========

APPSTART

Setze diesen ToolType auf YES, NO oder HIDE

NO   : ClassAction startet als Fenster
YES : ClassAction startet ikonifiziert
HIDE : ClassAction startet verborgen

Vorgabe ist : NO

==========

STARTDIR

Dieser ToolType gibt das Verzeichnis vor, welches beim Start
angezeigt werden soll.

Vorgabe ist : das aktuelle Verzeichnis

==========

WBFONT

Setze diesen ToolType auf YES (d.h. WBFONT=YES), wenn du
ClassAction mit dem aktuellen Workbench-Zeichensatz nutzen willst.
Wenn du ihn auf YES setzt, muß die wirkliche Zeichensatzgröße
kleiner als 15 sein, ansonsten wird Topaz 8 benutzt.

Wenn dieser ToolType auf NO steht, benutzt ClassAction Topaz 8.

Vorgabe ist : YES

==========

DECRUNCH

Wenn dieser Tooltype gesetzt ist (d.h. DECRUNCH=YES), versucht
ClassAction gepackte Dateien mittels der xfdmaster.library zu
entpacken.

Er ist notwendig, wenn du solche Dateien erkennen willst.

Natürlich werden die entpackten Daten im Speicher gehalten (ungefähr
2.5 x soviel, wie die Dateigröße). Falls du nicht genug Speicher

hast, solltest du den Tooltype deaktivieren.

Voreingestellt ist : NO

==========

HEIGHT

Dieser ToolType gibt die Fenstergröße für ClassAction vor.

Die minimale Größe hängt vom Zeichensatz ab. Falls du eine zu kleine Größe angibst, wird ClassAction die kleinste mögliche Höhe nehmen.

Vorgabe ist : 0 (Zwingt das kleinste mögliche Fenster auf)

==========

WINX and WINY

Diese ToolTypes geben die Koordinaten des Fensters an.

Vorgaben sind : 10

==========

ICONX and ICONY

Diese ToolTypes geben die Koordinaten des AppIcons an.

Beachte, daß dies nur ein Vorschlag ist, falls die Koordinaten schon von einem anderen Piktogramm belegt sind, werden sie ignoriert, und das Appicon so nahe wie möglich daneben plaziert.

Vorgaben sind : Keine Position

==========

PUBSCREEN

Wenn du diesen ToolType auf YES setzt (PUBSCREEN=YES), öffnet ClassAction sein Fenster auf dem vordersten öffentlichen Bildschirm.

Vorgabe ist : NO

==========

ICONNAME

Dieser ToolType bestimmt den Text des AppIcons.
Zum Beispiel kannst du ICONNAME=Drop einstellen und der Text unter dem AppIcon ist : 'Drop'.
Der Text kann leer sein (d.h. ICONNAME=), wenn unter dem AppIcon nichts stehen soll.

```
    Vorgabe ist : ClassAction

=========================================================================

ICONFILE

    Diese ToolType ist die AppIcon Datei.
    D.h., daß man sein eigenes AppIcon wählen kann.
    Wenn man diesen Tooltype auf eine Piktogrammdatei setzt, benutzt
    ClassAction dieses als AppIcon.

    Wichtig : Benutze nicht die .info Endung für den Dateinamen.

    Beispiel : ICONFILE=Sys:icons/head
               nutzt das Piktogramm head.info aus dem Verzeichnis
               Sys:Icons/ als AppIcon.

    Wenn das Laden des Piktogramms fehlschlägt, wird das voreingestellte
    benutzt (ClassAction.info).

    Voreingestellt ist : "" (Nutzt das voreingestellte AppIcon)

=========================================================================

CLISIZE

    Das ist die Größe des CLI Ausgabefensters, wenn die SystemTags()
    benutzt werden.

    Die Syntax ist : GERÄT:ObenX/ObenY/Breite/Höhe/Titel

    Dies kann auch zu einem anderen gerät als CON: gesetzt werden oder
    eine andere Fensterposition und Dimension festsetzen.


    WARNUNG :

    Wenn diese Tooltype auf einen falschen Wert gesetzt wird, öffnet
    ClassAction kein CLI.
    Ändere NICHTS, wenn du nicht genau weßt, was du tust.

    Benutze ** NIEMALS ** Kommandos, wie AUTO, CLOSE, WAIT, da dies schon
    ClassAction tut.

    Benutze ** NIEMALS ** Leerzeichen im Titel.


    In Version 2.0 wurde ein ähnlicher ToolType OUTPUT definiert, dieser
    ist nicht mehr nötig.


    Voreingestellt ist :
             CON:0/0/640/100/ClassAction_Output_Window

=========================================================================
```

DRIVE1 bis DRIVE1

Wenn dieser Tooltype auf einen gültigen Pfad zeigt, so erscheint dieser als Knopf im Requester. Das ist sinnvoll, um schnell in ein Verzeichnis zu gelangen.

Die Syntax ist :

DRIVEx=<Knopftext>,<Pfad>

z.B. DRIVE9=JPEG,dh0:gfx/pictures/jpeg

Jetzt steht im Knopf9 zwar 'JPEG', aber der Pfad ist dh0:gfx/pictures/jpeg.

Man kann auch nur den <Knopftext> ohne den <Pfad> angeben
z.B. DRIVE3=dh0:libs

In diesem Fall ist der Knopftext gleich dem Pfad. Aber nutze nicht zu lange Texte, diese können nicht in den kleinen Knöpfen dargestellt werden.

Voreingestellt ist : "" (nichts)

1.6 ClassAction Prefs

Neue Klassen und Aktionen zu definieren ist der Hauptzweck des Programms. Dazu mußt du das Programm ClassActionPrefs nutzen.

Die Benutzung sollte einfach sein... also los !!

Das Fenster ist in zwei Teile geteilt : Klassen & Aktionen.

Wenn eine Klasse gewählt ist, werden die zugehörigen Aktionen im Aktionsteil dargestellt.

Um eine Klasse hinzuzufügen oder zu löschen, einfach die entsprechenden Knöpfe drücken.

Genauso geht's bei den Aktionen...

~1.~Was~ist~eine~Klasse~~~~~~~~~

~2.~Eine~neue~Klasse~erstellen~~

~3.~Eine~neue~Aktion~definieren~

~4.~Die~Lernfunktion~~~~~~~~~~~~

1.7 Benutzung der Lernfunktion

Die Lernfunktion soll beim Erstellen neuer Klassen helfen.

Wenn man eine neue Klasse definiert, muß man deren Offset setzen, und es ist langwierig und langweilig, diesen herauszufinden.

Darum wurde diese wunderbare Funktion geschaffen !

Um sie zu nutzen, tue folgendes :

1 : Definiere deine Klasse normal, setze Namen und Klassennamen.

2 : Klicke den Knopf 'Lernen'.

3 : Wähle in dem folgenden Auswahlfenster soviele Dateien, wie du möchtest und sie der neuen Klasse angehören. Je mehr du wählst, desto besser wird das Resultat. ClassAction versucht nun, die Offsetdefinition zu finden. (um mehrere Dateien auszuwählen, nutze die Shift-Taste.)

4: Nach der Analyse bekommst du ein Fenster mit den gefundenen Offsets.

5: Ändere dies, wenn nötig, von Hand.

6: Klicke auf 'Akzeptieren', um diese Offsets für deine Klasse zu nutzen, oder 'Abbruch', um abzubrechen.

WARNUNG :

Wenn du die Lernfunktion nutzt, mußt du sicher sein, daß :

* ALLE GEWÄHLTEN DATEIEN DERSELBEN KLASSE ANGEHÖREN.

* ALLE GEWÄHLTEN DATEIEN NICHT GEPACKT SIND.

Das ist alles !!! Einfach, nicht ? .

1.8 Was ist eine Klasse

Eine Klasse ist eine Familie von Dateien. Zum Beispiel können C Dateien als eine Klasse bezeichnet werden, als C Klasse.

Mit ClassActionPrefs kann man soviele Klassen definieren, wie man will, solange man diese unterscheiden kann.

Es gibt zwei Methoden, eine Klasse zu erkennen : Die Kennung und den Inhalt der Datei.

Mittels der Kennung kann man eine Datei anhand des Namens erkennen. Offsets werden benutzt, um eine Datei anhand ihres Inhalts zu erkennen.

Es gibt zwei eingebaute, nicht entfernbare, Klassen, die weiß in der Klassenliste auftauchen.

Die erste heißt "Unbekannte Klasse", aber du kannst sie umbenennen. Diese Klasse enthält alle Dateien, die ClassAction nicht erkennen kann.

Die zweite heißt "Allgemeine Aktionen" und sie kann nicht umbenannt werden. Diese Klasse enthält Aktionen, die in ALLEN anderen Klassen sind...

Interessant : falls du eine Aktion 'Kopieren' in allen Klassen hast, so könntest du diese in allen Klassen definieren, was aber lange dauert und anstrengt.
Besser ist es, diese in der Klasse "Allgemeine Aktionen" zu definieren, und sie gilt für alle Klassen.

Die allgemeinen Aktionen werden in der Aktionsliste weiß dargestellt und sind nur im ClassAction Fenster sichtbar, nicht im AppIcon oder im ARexx-Modus.

1.9 Eine neue Klasse erstellen

Nur auf den Knopf 'Hinzufügen' im Klassenteil drücken und schon wird eine neue Klasse erstellt.

Eine Klasse hat 3 Merkmale :

- einen Namen
- eine Kennung
- Offsets

o Der Name ist einfach der Klassenname, wie du ihn wählst.
WARNUNG : ein Klassenname darf nur einmal vergeben werden.

o Die Kennung ist jede reguläre AmigaDos-Endung, wie :

#?.c , mod.#? , #?.c|#?.h , #?b[a|c] , #?toto? .....

(lies im AmigaDOS-Handbuch zum Thema Wildcards)
WARNUNG : Benutze nicht * sondern #? als Wildcard.

Die Kennung ist nicht einmalig, z.B. könnte toto.c in mehreren Klassen auftauchen.

Wenn du eine Klasse anhand der Kennung definierst, mußt du dieses gut überlegen.

Beispiel : wenn du die GIF Klasse mit der Kennung #?.gif definierst, werden alle Dateien mit der Endung .gif als GIF Dateien interpretiert.

Aber bist du sicher, daß alle deine GIF Dateien diese Endung haben und das ALLE Dateien mit der Endung .gif auch GIF Dateien sind ?

Darum solltest du die Kennung nur in zwei Fällen nutzen :

- die Kennung ist unabhängig von Groß- und Kleinschreibung (z.B.: #?.info ist eine gute Kennung für die Klasse Piktogramm)

- wenn du keine Wahl hast (z.B.: wie kann man einen C-Quellcode erkennen, wenn nicht mit #?.c)

o Andernfalls solltest du Offsets nutzen.

Ein Offset ist eine Stelle in einer Datei, wo man etwas Erkennbares finden kann.

Zum Beispiel beginnen GIF Bilder immmer mit 'GIF' am Offset 0.

Es gibt drei Syntaxe, um Offsets zu definieren :

==========

Syntax #1 : Offset,HexString

Offset ist eine DEZIMALE Zahl, die den Offset enthält.
HexString ist ein HEX-Sstring, der bei diesem Offset steht.

Beispiel 1 : 0,4f4a heißt, daß die Datei mit den Bytes $4f und $4a an Position 0 beginnen muß.

Beispiel 2 : 9,448b3c heißt, daß bei Byte #9 $44 $8b $3c gefunden werden sollten.

==========

Syntax #2 : Offset,'String'

Offset ist eine DEZIMALE Zahl, die den Offset enthält
String ist eine ASCII-Zeichenkette, die bei dem Offset steht.

Beispiel 1 : 0,'GIF' heißt, daß die datei mit 'GIF' beginnen muß.

Beispiel 2 : 9,'FuBar' heißt, daß man bei Byte #9 die Zeichenkette 'FuBar' finden muß.

==========

Syntax #3 : Offset,"String"

```
    Offset ist eine DEZIMALE Zahl, welche den Offset enthält.
    String ist eine ASCII-Zeichenkette, die bei diesem Offset stehen
    muß.

    Beachte den Unterschied zur vorherigen Syntax, hier nutzen wir "
    statt '.

    Es ist dasselbe Konzept, wie bei Syntax #2, aber der Vergleich
    ist UNABHÄNGIG VON GROß- ODER KLEINSCHREIBUNG.

    Zum Beispiel beginnt eine AmigaGuide datei immer mit @database
    mit großen und kleinen Buchstaben.

    Wenn du Methode #2 benutzt mit 0,'@database' wird ClassAction
    eine Datei mit @DATABASE nicht als AmigaGuide datei erkennen.

    Es funktioniert aber mit Syntax #3 : 0,"@database"


==========================================================================


Man kann bis zu 5 Offsets in einer Klasse definieren. Eine Datei wird
als der Klasse zugehörig erkannt, wenn alle Offsets übereinstimmen.

Bsp. : wenn Klasse X so definiert ist :

Offset #1 :  0,4a8b6c
Offset #2 : 58,14

Werden alle Dateien, di mit 4a8b6c beginnen UND bei Byte #58 $14
haben als X deklariert.

Um mehrere Offsets zu deklarieren, nur den Knopf 'Offset #' drücken.


Anmerkung : Die ASCII Klasse

Es gibt ein eingebautes Offset Kommando : ASCII[]

Wenn du dieses Kommando in Offset #1 (d.h. Offset#1=ASCII[])
einschreibst, werden ASCII Dateien ausgewertet.
ClassAction probiert das, nachdem alles andere fehlschlägt. Darum
werden Amigaguide Dateien im ASCII-Format trotzdem als AmigaGuide
Dateien erkannt, wenn die AmigaGuide-Klasse definiert wurde.

Normalerweise sollte dich das nicht kümmern, da ich in der
Standardeinstellungsdatei dieses schon vordefiniert habe.
```

1.10 Eine neue Aktion definieren

```
Wenn eine Klasse definiert ist, solltest du Aktionen dafür bereitstellen.
Jede Klasse kann so viele Aktionen haben, wie du möchtest.
```

Klicke einfach auf den Knopf 'Hinzufügen' im Aktionsteil, um eine neue Aktion zu schaffen.

Eine Aktion hat 6 Merkmale:

- einen Namen
- einen Laufmodus
- eine Stapelspeichergröße (StackSize) (nur im CLI-Modus)
- eine Verzögerung (nur im CLI-Modus)
- ein Exec Kommando
- ein RescanDir Zeichen (Verzeichnis wiedereinlesen)

o Name ist der Name der Aktion.

o Laufmodus kann sein : ~CLI~, ~WB~, ~No~CLI~ oder ~ARexx~.

o Exec Kommando ist eine gültige AmigaDOS Kommandozeile und kann verschiedene Parameter enthalten.
DU SOLLTEST immer den vollen Pfad für Programme nutzen.

Beispiel : benutze C:Copy statt Copy in der Zeile.

In Kommandozeilen kannst du in ClassAction eingebaute ~Argumente~ und ~Kommandos~ benutzen.

o Setze den RescanDir Zeiger, wenn du das aktuelle Verzeichnis wiedereinlesen lassen willst, nützlich, wenn die Aktion das Verzeichnis geändert hat.

Knöpfe

o Die beiden Knöpfe 'H' und 'R' sind zum Sortieren der Aktionen gedacht.

o Der Knopf 'Laden' läßt dich ein Kommando auswählen.

o Der Knopf 'Komm.' öffnet ein Auswahlfenster mit allen möglichen Kommandos und Argumenten für die Kommandozeile.

o Mit dem Knopf 'Kop.' kann man Aktionen in die aktuelle Klasse kopieren.
Den Knopf anklicken und die Quellklasse wählen.
Das ist nützlich, wenn mehrere Klassen dieselben Aktionen haben sollen.

1.11 CLI Modus

**** CLI Modus ****

Wenn 'CLI' gewählt ist, wird die Aktion vom CLI gestartet und der eingegebene Stapelspeicher benutzt (vorgegeben ist 4096).

Ein CLI-Fenster wird nur geöffnet, wenn's nötig ist (falls irgendetwas dargestellt werden muß).

Man kann eine Verzögerung für das CLI einstellen:

Wenn die Verzögerung negativ ist (d.h. Verzögerung = -1), wartet das CLI, bis du es mittels des Schließsymbols oben links schließt.

Wenn die Verzögerung null ist (d.h. Verzögerung = 0), schließt das CLI selbsttätig nach Beendigung der Aktion.

Wenn die Verzögerung positiv ist (d.h. Verzögerung =n mit n>0), wartet das CLI n Sekunden, bevor es schließt, aber man auch das Schließsymbol benutzen.

Die Dimension des benutzten CLI kann im Tooltype CLISIZE eingestellt werden (der alte Tooltype OUTPUT ist nicht mehr notwendig).

1.12 WB Modus

*** WB Modus ***

Wenn 'WB' gewählt ist, dann wird kein CLI geöffnet und ClassAction simuliert einen Workbenchstart der Aktion. Das gewählte Programm kann mit den Tooltypes in seinem Piktogramm gesteuert werden.

WARNUNG : dieser Modus ist nur mit Programmen, die Piktogramme besitzen, auszuführen.

1.13 NO CLI Modus

*** NO CLI Modus ***

Wenn 'No CLI' gewählt ist, wird kein CLI-Fenster geöffnet, auch wenn das Programm etwas darstellen möchte, doch es läuft vom CLI.

1.14 ARexx Modus

*** AREXX Modus ***

Wenn 'Arexx' gewählt ist, wird rx mit dem gegebenen Kommando ausgeführt.
Das Kommando MUß ein Arexx-Skript sein.
Natürlich sollte RexxMaster aktiv sein und Rx sich im Verzeichnis Sys:rexxc/ befinden.

1.15 Argumente

```
    Zur Zeit sind 8 Argumente möglich :

die ersten 4 Kommandos sind kleingeschrieben : [f] [s] [b] [x] und sie
enthalten das Resultat in Anführungszeichen


    [f] : voller Pfad der gewählten Datei in Anführungszeichen
    [s] : voller Pfad der gewählten Datei ohne Endung in Anführungszeichen
    [b] : Dateiname in Anführungszeichen
    [x] : Dateiname ohne Endung in Anführungszeichen


die anderen 4 Kommandos bewirken dasselbe, bloß ohne Anführungszeichen


    [F] : voller Pfad der gewählten Datei
    [S] : voller Pfad der gewählten Datei ohne Endung
    [B] : Dateiname
    [X] : Dateiname ohne Endung


Beispiel : sagen wir, du möchtest die Datei ram:env/sys.prefs auswählen


    [f] = "ram:env/sys.prefs"
    [s] = "ram:env/sys"
    [b] = "sys.prefs"
    [x] = "sys"

    [F] = ram:env/sys.prefs
    [S] = ram:env/sys
    [B] = sys.prefs
    [X] = sys



    Beispiel : sagen wir, du wählst die Datei ram:main.c

    * Die Kommandozeile
                c:copy [f] [F].bak
      wird ersetzt durch :
                c:copy "ram:main.c" ram:main.c.bak

    * Die Kommandozeile
                c:copy [f] [S].bak
      wird ersetzt durch :
                c:copy "ram:main.c" ram:main.bak
```

1.16 Kommandos

```
    Zur Zeit sind 5 Abfragekommandos verfügbar :


    REQD[Text] : Fragt nach einem Verzeichnis.
```

```
REQF[Text] : Fragt nach einer Datei.
REQV[Text] : Fragt nach einem Laufwerk.
REQT[Text] : Fragt nach einem Text.
SURE[Text] : Fragt nach der Zustimmung des Nutzers.


=====================================================================

REQ Kommandos


    REQ Kommandos öffnen ein Reqtoolsrequester mit einem kleinen Titel
    [Text].
    Dies könnte man in interaktiven Kommandozeilen benötigen.

    Beispiel :
        bin:lha x [f] to REQD[Wähle ein Verzeichnis zum Entpacken]

    Dies wird ein Auswahlfenster öffnen, um das Zielverzeichnis zu
    bestimmen, und die gewählte Datei [f] wird dann in dieses entpackt.


    REQF[] macht dasselbe, außer daß es nach einer Datei fragt.

    Beispiel :
        c:dir [f] > REQF[Wähle eine Datei]


    REQV[] fragt nach einem Laufwerk.


    REQT[] fragt nach einem Text, falls man Argumente benötigt, zum
           Beispiel :

        c:cpu REQT[Gib Argumente für CPU an]


ACHTUNG :

    Die Kommandos REQD, REQF und REQV könnten mit dem AppIcon-Modus bei
    Benutzung von Programmen wie MagicMenu inkompatibel sein.

    Wenn du regelmäßig 'Gurus' mit diesen Aktionen erhälst, prüfe deine
    Konfiguration dieser Aktionen oder versuch' den Tooltype REQBUG mit
    YES.

    Wenn du diesen Tooltype auf YES setzt, werden diese Kommandos im
    AppIcon-Modus durch das REQT-Kommando ersetzt.

=====================================================================

SURE Kommando:

    Das SURE[Text] Kommando öffnet ein Requester mit dem Text [Text] und
    zwei Knöpfen : Ja / Nein.
    Wenn man Nein wählt, wird die Kommandozeile abgebrochen.
    Wenn man Ja wählt, wird ClassAction die Kommandozeile im RECHTEN
```

```
        Teil des Sure-Kommandos ausführen.

        Beispiel :
            SURE[Diese Datei wirklich löschen ?]C:delete [f]

        Es wird ein requester geöffnet, das den Nutzer fragt "Diese Datei
        wirklich löschen ?".
        Wenn der Nutzer mit Nein antwortet, passiert nichts; wenn er aber mit
        Ja antwortet, dann wird C:delete [f] ausgeführt.

    ========================================================================


    Natürlich kann man jede Anzahl von Argumenten / Kommandos in einer
    Kommandozeile kombinieren.

Beispiel :
 SURE[Diese Datei wirklich umbenennen]c:rename [f] REQF[Nenne den neuen Namen]
```

1.17 Zukünftige Verbesserungen

```
    Was werde ich in der nächsten Version in das Programm einbauen :


    o   ein Konzept von Oberklassen :
        Zum Beispiel die Oberklasse Bilder enthält GIF, IFF, TARGA, JPEG...
        und die Möglichkeit einen dementsprechenden Filter zu wählen. Dann
        kann man Bilder, Sounds... filtern.

    o   eine unbegrenzte Anzahl von Pfadknöpfen.

    o   das ClassActionPrefs den WB-Zeichensatz nutzt.

    o   Und natürlich alles, wonach ich gefragt werde :)
```

1.18 Über

```
    Ich möchte den folgenden Leuten danken :


    o   Mireille (für ihre Nachsicht...)

    o   Philippe Thomas (für Vorschläge, Hilfe, Betatesten, das französische
        Guide und UTT, welches für die Installation benutzt wurde)

        Hey Phil, if you read this, click ~here~

    o   Richier Pierre (für die MagicWB Piktogramme)

    o   Jean Michel und George (für's Betatesten auf einem A4000/40)

    o   Obvious Implementations Corp (für Dice C Pro)
```

- Nico Francois für die ReqTools.library
- Georg Hörmann für die *GROßARTIGE* xfdmaster.library
- Allen Nutzern, die mir Fehlermeldungen und Vorschläge schickten
- Allen registrierten Nutzern

1.19 Der Autor

Du kannst mich unter folgender Adresse erreichen :

Gasmi Salim
4b rue des petits champs

67300 Shiltigheim
France

Irc: Dr_Unix (#amigafr, #amiga)

Internet: http://www.gasmi.net

E-Mail: salim@gasmi.net oder salim@sdv.fr

1.20 Benutzung von ClassAction

Die Benutzung von ClassAction ist WIRKLICH einfach.

Wähle eine Datei im Auswahlfenster. Du kannst mit dem entsprechenden Knopf ins Mutterverzeichnis springen.
Die rechte Maustaste zeigt Laufwerke und Zuweisungen.

Nach der Auswahl sieht man im rechten Fenster die Klasse der Datei und die möglichen Aktionen. Nun wähle die gewünschte Aktion...

Wenn man einen Doppelklick auf einer Datei ausführt, wird die erste definierte Aktion ausgelöst.

Mit dem Zoomgadget rechts oben kann man die Größe des Fensters ändern.

Zum Beenden einfach auf den Knopf 'Ende' klicken.

Um das Fenster in ein AppIcon zu verwandeln, einfach das Schließsymbol betätigen. Ein Doppelklick auf das AppIcon öffnet das Fenster wieder.

Wenn ClassAction ein AppIcon ist, kann man Piktogramme raufwerfen und der Dateityp wird erkannt.
Wenn die Klasse nur eine definierte Aktion hat, wird diese ausgeführt. Ansonsten öffnet sich ein Fenster mit den möglichen Aktionen, daß man eine auswählen kann.

Das ist alles, einfach, nicht ???

1.21 Technische Infos

ClassAction ist 100 % mit DICE C 3.0 geschrieben

Weitere Infos :

Die Einstellungsdatei ist eine ASCII-Datei namens :
ENVARC:ClassAction.prefs
Die allgemeinen Aktionen werden gespeichert in
ENVARC:ClassAction_Gen.prefs

ClassAction erstellt ein ausführbares Programm namens ClassAction_RunTask,
gespeichert in T:
Dieses Programm wird für WB Tasks benötigt.

Die Geschwindigkeit der Dateiauswahlfenster hängt vom gewählten
Sortieralgorithmus ab (rekursive Baumsortierung)

Informationen über Bibliotheken :

Benutzte ROM libraries :

exec.library	V37+
dos.library	V37+
intuition.library	V37+
graphics.library	V37+
gadtools.library	V39+
workbench.library	V37+
utility.library	V39+

Benötigte DISK libraries :

rexxsyslib.library	V39+
commodities.library	V37+
asl.library	V39+
icon.library	V37+
reqtools.library	V38+

DISK libraries, die benutzt werden, falls vorhanden :

locale.library	V38+
datatypes.library	V39+
xfdmaster.library	V30+

Wie erkennt ClassAction eine Klasse :

1- Prüfung, ob der Dateiname mit einer definierten Kennung
   übereinstimmt.
2- Prüfung, ob die Offsets mit denen definierter Klassen
   übereinstimmen.
3- Entpacken der Datei mittels der xfdmaster.library.
4- Prüfung, ob der entpackte Puffer übereinstimmende Offsets mit
   definierten Klassen hat.
5- Prüfung, ob die Datei eine ASCII-Datei ist (wenn das ASCII Offset
   Kommando besteht)

Wenn alles fehlschlägt, wird die Datei der 'Unbekannten Klasse'
zugeordnet.

1.22 Installation

Um dieses Zeug zu installieren :

BENUTZE das Installationsskript in diesem Archiv.
Klicke dazu auf des entsprechende Piktogramm.

Aber falls du das nicht möchtest, hier der andere Weg :

- Kopiere ClassAction, ClassActionPrefs und deren Piktogramme wohin du
  willst.

- Kopiere die .prefs Dateien nach ENVARC:

- Kopiere das ClassAction.guide wohin du willst.

Das ist alles...

1.23 Registrierung

Wenn du diese Zeilen liest, wirst du dich wundern, ein registrierter
Nutzer zu werden.

Laß mich erklären, warum du dich registrieren lassen solltest :

Zuerst, um den besten Computer allerzeiten zu unterstützen, weil die
Zukunft des Amigas von der verfügbaren Software abhängt; wenn du eine
Programmierer in seiner Arbeit unterstützt, unterstützt du deinen
Computer und dessen Zukunft !!!!

Auch, da ich all meine freie Zeit dafür verwende, um Shareware zu
entwickeln. Wenn du diese nutzt, warum solltest du dich dann nicht auch
bei mir registrieren lassen. Dies wird mich beflügeln, weitere Proggies

zu entwickeln.

Die MailWare Version ist 100% nutzbar, nichts ist abgestellt, nur ein Requester erscheint : falls es dir wirklich gefällt, REGISTRIEREN !!

Die Registration ist FREI !!! ALLES was du tun mußt ist :

- Gehe im Netz auf meinen Seite http://www.gasmi.net
- Folge dem Link 'CA registration' und beantworte die Fragen nach deinem Namen und deiner E-Mail Adresse.
- Du bekommst dann deinen Keyfile über Email

oder über den Postweg an meine Adresse
(in diesem Fall STECKE $2 in Briefmarken dazu und vergiß deine vollständige Anschrift nicht)

Danke im Voraus für deine Unterstützung !

Euer,

Salim

1.24 Rechtliches

Copyright

ClassAction ist eine Shareware Programm. Das Paket darf in keiner Weise verändert werden und nicht für kommerzielle Zwecke benutzt werden ohne Zustimmung des Autors. Die Copyright Mitteilung ist geschützt.

Garantie

Für Schäden durch Benutzung dieses Programms wird keinerlei Haftung übernommen. Die Benutzung erfolgt auf eigenes Risiko. Die Software wird "so wie sie ist" bereitgestellt, ohne jegliche Garantieerklärungen oder anderweitige Versicherungen auf Lauffähigkeit und Stabilität der Software und Dokumentation. Die Dokumentation ist nach bestem Wissen und Gewissen erstellt, aber der Autor behält sich das Recht vor, die Software und/oder Dokumentation ohne Bemerkung zu erneuern.

1.25 Geschichte

ClassAction Geschichte V 2.8 (c) Salim Gasmi

25.09.95 : V2.8

- ClassAction hat nun VOLLE Commodtity Unterstützung und einen Hotkey.

- Tooltype APPSTART kann auf HIDE (APPSTART=HIDE) gesetzt werden, wenn ClassAction verdeckt gestartet werden soll.

- Tooltype CX_HOTKEY hinzugefügt

- Tooltype PUBSCREEN hinzugefügt für Arbeit auf öffentlichen Bildschirmen

- Die Interne Dateiauswahl von ClassAction gibt nun wieder allen Speicher frei.

11.09.95 : V2.75

- Die REQD,REQF,REQV Kommandos waren nicht inkompatibel mit dem AppIcon Modus, aber mit einigen Programmen, wie MagicMenu (So ein Pech, ich benutze MagicMenu...)

  Viele Nutzer meldeten einen automatischen Sprung ins REQT Kommando, sogar ohne inkompatible Programme zu benutzen.

  Ich fügte einen Tooltype (REQBUG) hinzu, damit der Nutzer den Auto-Sprung in REQT überwachen kann.

03.09.95 : V2.7

- Kommandos [b],[x],[B],[X],[F],[S] hinzugefügt.

- Das Aktion Arexx Kommando hinzugefügt.

- Einige Fehler behoben

- Einen Fehler behoben, wenn mehrere Piktogramme auf’s AppIcon geworfen werden.

- Die REQD,REQF,REQV Kommandos springen nun in REQT, wenn sie vom AppIcon oder im Arexx-Modus benutzt werden, sie sind nur mit dem Fenstermodus kompatibel.

28.08.95 : V2.6

- Das Fenster von ClassAction ist nun in der Größe veränderbar.

- ClassAction und ClassActionPrefs nutzen nun die ReqTools.library.

- REQV Kommando zur Frage nach einem Laufwerk hinzugefügt.

- REQT Kommando zur Frage nach einem Text hinzugefügt.

- WINX und WINY Tooltypes hinzugefügt.

- Das Lernrequester hat jetzt einen ALLES Knopf.

- Die gewählte datei ist jetzt in allen REQ Requestern.

- ClassAction nutzt nun für registrierte Nutzer einen Keyfile in S:.

17.07.95 : V2.5

- ClassAction und ClassActionPrefs sind jetzt lokalisiert und ein französischer Katalog ist beigelegt.

- Die Lernfunktion ist zugefügt.

- Um die erste Aktion einer datei zu starten, reicht nun ein Doppelklick.

- Allgemeine Aktionen sind nun synchron und lesen das aktuelle Verzeichnis wieder ein.

- AUTOSELECT ToolType wird nicht mehr benutzt.

- REQ Requester öffnen jetzt das aktuelle Verzeichnis.

- Kleinere Verbesserungen gemacht.

12.06.95 : V2.1

- Die eingebaute Klasse 'Allgemeine Aktionen' hinzugefügt.

- SURE[] Kommando hinzugefügt.

- ASCII[] Offset Kommando zum Erkennen von ASCII Dateien hinzugefügt.

- Arexx Kommandos hinzugefügt: AppIconify, Show, Status, GetClass.

- Tasksystem geändert, jetzt nutze ich Systemtags().
  Wir brauchen keine tmp Dateien mehr.

- Man kann nun im CLI Modus eine Verzögerung definieren.

- 'string' und "string" für Offsetdefinition hinzugefügt.

- ToolType OUTPUT ist nun überflüssig und wird nicht mehr benutzt, man kann dafür den Tooltype CLISIZE nutzen.

- Aktionsrequester sind nun richtig proportioniert, erscheinen unter'm Mauszeiger und nutzen den vorderen Bildschirm.

- Knöpfe 'Benutzen' und 'Speichern" vertauscht und 'Abbruch' hinzugefügt in ClassActionPrefs, um dem Amigaaussehen zu folgen, den Knopf 'Über' als '?' in die rechte obere Ecke verschoben.

- ClassActionPrefs Cycle-Gadgets waren nicht 100% systemkonform und einige Patches, wie CycletoMenu verursachten Fehler mit ClassActionPrefs; das ist nun behoben.

- Erkennungscode und Inforoutine verbessert: diese sind nun bis zu 400% schneller.

- Farbfehler bei Highlight entfernt.

- ClassAction schließt nun beim Ikonfizieren den WB-Screen nicht mehr.

- Einen gemeinen Fehler in ClassActionPrefs gefunden und beseitigt.

- Die rechte Maustaste zeigt Zuweisungen nur, wenn die Maus im Requester ist und rechte Maustaste bringt auch das Verzeichnis zurück.

- ClassAction speichert nun die Fensterposition.

23.05.95 : V2.00 (Großes Update)

- ClassAction hat jetzt ein AppIcon.

- ClassAction ist jetzt ein Commodity.

- ClassAction hat jetzt einen Arexx Port.

- ClassAction nutzt jetzt den eingestellten WB-Zeichensatz.

- Exec Modus 'Arexx' hinzugefügt.

- Verschiedene Farben für Verzeichnisse/Dateien.

- Hoch/Runter Symbole in ClassActionPrefs.

- 'Benutzen' Knopf in ClassActionPrefs.

- Klassen werden nun in ClassActionPrefs sortiert.

- APPSTART, ICONNAME, ICONX, ICONY, CX_PRIORITY, WBFONT, OUTPUT, ICONFILE ToolTypes hinzugefügt.

- Wenn das Fenster ikonifiziert ist, hat es die richtige Höhe für den eingestellten Zeichensatz.

- Neues Speicherformat (CASF20).

- Endungs/Vorsilben Knopf entfernt. Ersetzt durch Erkennungsnamen Knopf, der alle Wildcards akzeptiert.

- Konfigurationsdatei in ENVARC: verschoben.

- einen Installer hinzugefügt.

- Code optimiert.

05.05.95 : V1.43

- Requestercode bereinigt, jetzt 5% schneller.

- ClassAction schaut jetzt nach seinem Namen in der WBStartup-Struktur und kann somit umbenannt werden.

02.05.95 : V1.42

- 'No Cli' Exec Modus in ClassActionPrefs eingefügt.

- "" werden immer bei Dateinamen benutzt, das ist einfacher für ARexx-Skripts.

06.04.95 : V 1.4

- ToolType HEIGHT hinzugefügt.

- Code optimiert.

22.02.95 : V 1.31

- Bei Doppelklick auf dieselbe Datei wird die erste Aktion ausgeführt (ist schneller, als erst die Aktion zu wählen).

- diese Version ist jetzt ShareWare und man muß sich registrieren lassen.

15.01.95 : V 1.3

- REQD[] und REQF[] als interaktive Kommandos hinzugefügt.

- Knopf 'Kopieren' in ClassActionPrefs eingefügt.

- kleinere Verbesserungen gemacht.

- Betatester berichten, daß diese Version sehr stabil läuft.

25.11.94 : V 1.22

- Erste Veröffentlichung

- Knopf 'Info' hinzugefügt.

- ClassAction hatte Locking() die Verzeichnisse, wenn es sie mit UnLocking() gelesen hat .... *BEREINIGT*

- Codeoptimierung

- Kleinere Fehler entfernt.

08.11.94 : V 1.21

- Programm stürzte mit leeren Diskettenlaufwerken ab .. *BEHOBEN*

- Laufwerks/Namen Fehler behoben

- <..> entfernt im Hauptverzeichnis.

07.11.94 : V 1.2

- Neue Schnittstelle, Ich habe meine eigenen Dateiauswahlfenster.

- STARTDIR & DRIVE1 to DRIVE11 ToolTypes hinzugefügt.

01.11.94 : V 1.1

- Benutzt jetzt die xfdmaster library zum Erkennen und Entpacken von Dateien.

- Konfiguration mit ToolTypes hinzugefügt (DECRUNCH,AUTOSELECT).

- 'Unbekannte Klasse' ist jetzt eine eingebaute Klasse mit unbegrenzter Anzahl von Aktionen.

- Neues Speicherformat (CASF11).

16.10.94 : V 1.0

- Neues Speicherformat (CASF10).

- Fenster hat jetzt ein Größensymbol.

- eine Menge Klassendefinitionen hinzugefügt.

10.10.94 : Beta Version

- tmp Dateifehler entfernt.

- Offset Zählfehler behoben.

- benutzt asl.library für die Dateiauswahlfenster.

01.10.94 : Alpha Version

1.26 Die Arexx Kommandos

ClassAction hat einen Arexx Port namens : ClassAction.01

Hier ist eine Liste der Arexx Kommandos :

==========

Quit

```
    Beendet ClassAction...

=====================================================================

Use

    Zwingt ClassAction, die Einstellungsdatei erneut zu lesen.

=====================================================================

Ver

    Gibt die Version von ClassAction zurück.

=====================================================================

Status

    Gibt den aktuellen Staus von ClassAction zurück.

    Return 0 : ClassAction ist Ikonifiziert.
    Return 1 : ClassAction ist im Fenstermodus.

=====================================================================

AppIconify

    ZwingtClassAction in den Ikonifizierungsmodus.

    Wenn ClassAction schon ikonifiziert ist, tut dieses Kommando
    gar nichts.

=====================================================================

Show

    ClassAction zeigt das Hauptfenster.

    Wenn ClassAction schon im Fenstermodus ist, tut das Kommando nichts.

=====================================================================

Load <Dateiname>

    ClassAction versucht die Datei <Dateiname> zu laden, und öffnet ein
    Aktionsfenster zur Auswahl.

    Wenn die Datei nicht existiert, tut das Kommando nichts.

=====================================================================

GetClass <Dateiname>

    ClassAction versucht die Datei <Dateiname> zu laden und ihre Klasse
    zurückzugeben.

    Wenn die Datei nicht existiert, tut das Kommando nichts.
```

```
================================================================

Action <Dateiname> <Aktionsmuster>


    ClassAction führt die erste zutreffende Aktion für die Datei
    <Dateiname> aus.

    Beispiel : Action ram:toto.lha extr

    startet die erste Aktion, deren Name 'extr' enthält mit der Datei
    ram:toto.lha

    Das Muster ist nützlich, damit man nicht den exakten Namen der Aktion
    angeben muß.
```

1.27 Was sollte man nach der Installation tun

```
Nach der Installation solltest du ein funktionierendes ClassAction
besitzen, mit einigen Klassendefinitionen, aber sehr wenigen Aktionen
(viele Klassen haben nicht mal welche).

Es liegt an dir, Aktionen zu definieren für dein System und deine
Programme.

Lade ClassActionPrefs und konfiguriere nach deinen Wünschen...

Wenn du nicht weißt wie, dann lies dieses Guide :)).

Nach der Konfiguration der Klassen und Aktionen konfiguriere ClassAction
mit seinen Tooltypes.

Es mag lange dauern, bis eine 'feine' Konfiguration erstellt ist. Aber
wenn sie da ist, ist's GROßARTIG !!!

Okay, jetzt lies das Guide und Viel Glück.
```

1.28 Mitteilung für Philippe THOMAS

```
Salut Philippe !!!

Je voulais simplement te remercier pour toute l'aide que
tu m'as apporté à la création de ce programme.

Quasiment toutes les améliorations de la version 2.0, c'est
toi qui me les a proposées, parfois même avec insistance,
style le resize de la fenêtre que je n'ai toujours pas fait.. :(

Encore merçi pour tous les appels téléphoniques ; parfois plus
d'une heure à discuter des Hooks, SystemTagList, de bugs etc...
et ce, même en periode d'exams.
```

Franchement si ClassAction commence à être cool, c'est beaucoup
grâce à toi, je crois que j'aurais eu la flemme de le paufinner
autant si tu n'étais pas là.

Bref, ce programme est aussi un peu le tien.

Okay Phil, à la prochaine.

Salim.

PS : Non, non ton processeur il est bien, il est pas buggé... :)).

1.29 Bekannte Fehler

Viele Nutzer fragten nach einer in der Größe veränderbaren GUI,
und ich habe mein Bestes getan, um dies in V2.6+ zu unterstützen,
aber mit GadTools ist das so eine Sache.

Manchmal wird der Wiederaufbau des Fensters nach einer
Größenveränderung nicht richtig durchgeführt, dann mußt du das
Fenster nochmal schließen.

Ich weiß nicht warum, da dieser Fehler bei mir nicht auftritt.
Die Betatester berichteten davon, bei ihnen passiert das zweimal
im Monat.

Einige Programme, wie MagicMenu, verursachen Abstürze mit den REQ
Kommandos im AppIcon-Modus.
Du kannst den Tooltype REQBUG benutzen, um diese Kommandos
auszuschalten.