

TweakEdge

*Stand 09.10.2007 12:12
Autor: TeamWiSE GmbH*

Leerseite

Inhaltsverzeichnis

Inhaltsverzeichnis	3
1. Zielsetzung	5
2. Problemstellung.....	6
3. Lösungsansatz	7
3.1. Allgemein	7
3.2. Aufgaben.....	7
3.3. Wizards.....	7
4. Aufgabe: Debugmodus für einen Baustein anpassen.....	8
4.4. Allgemein	8
4.5. Einstellungen aktivieren.....	8
4.6. Deaktivieren von Einstellungen	10
5. Aufgabe: Einen Baustein ausführen.....	11
5.7. Allgemein	11
5.8. Schritt 2: Kontext bearbeiten	11
5.9. Schritt 3: Aufzeichnung festlegen	11
5.10. Schritt 4: Protokollierung steuern	11
5.11. Schritt 5: CTV-Einstellungen vornehmen	12
5.12. Schritt 6: Übersicht	12
6. Vorlagen	13
7. Dokumentation und Hilfe.....	14
8. Übersichten	15
8.13. Übersicht "Aktuell konfigurierte Anwendungen".....	15
8.14. Übersicht „Bausteine mit aktiven Debug-Einstellungen“.....	15
8.15. Übersicht „Bausteine mit Post- bzw-Preview aktiviert“.....	16
8.16. Aktionen aus Übersichten heraus ausführen.....	16

Leerseite

1. Zielsetzung

Dieses Dokument beschreibt das mit TAA-Release 7.07 verfügbare Werkzeug TwkEdge (Tweak-Edge). Dieses Werkzeug wird mit diesem Release in seiner ersten Version ausgeliefert und diese Beschreibung soll den ersten Kontakt mit diesem Werkzeug erleichtern.

Außerdem sollen Anwendungsentwickler die Grundprinzipien des Arbeitens mit dem Werkzeug verstehen um eigene Anregungen für den weiteren Ausbau des Werkzeugs geben zu können.

Das Dokument erhebt nicht den Anspruch, eine vollständige Dokumentation der Bedienung und Funktionalität des Werkzeugs zu sein. Zum Thema Dokumentation siehe den gleichnamigen Abschnitt in diesem Dokument.

2. Problemstellung

Die Aufgabe des Werkzeugs Tweak-Edge besteht darin, Anwendungsentwickler bei ihrer alltäglichen Arbeit mit der TAA zu unterstützen.

In der ersten Version des Werkzeugs liegt der Schwerpunkt auf der Unterstützung des Debuggings von TAA-Anwendungsbausteinen. In zukünftigen Releases sollen weitere Aufgaben unterstützt werden.

Im Rahmen des Debuggings von Anwendungen hat es der Anwendungsentwickler insbesondere mit einer Vielzahl von Registry-Einstellungen zu tun, welche (zum Teil) auch noch untereinander in Beziehung stehen. Daneben gibt es eine Unmenge von Optionen bei der Ausführung von TAA-Anwendungsbausteinen, welche den Verlauf des Debuggings beeinflussen.

Bei den manuell vorgenommenen Registry-Einstellungen kommt es immer wieder zu Problemen. Hier einige der populärsten Beispiele:

- Tippfehler im Schlüsselnamen (z.B. hinter dem Schlüsselnamen „ityp „ steht noch ein blank)
- Tippfehler im Bausteinnamen in der Debug-Section
- DebugAllowed an unterschiedlichen Stellen gesetzt (projektspezifisch, übergreifend, HKLM)
- EndUser auf falschen Wert (01 statt 00)
- DebugFledge bei GeVos nicht gesetzt (kein Debuggen von GeVos trotz DebugAllowed)
- FledgeArguments nicht gesetzt (kein Trace trotz anderer Registry-Einstellungen)
- Geänderte Schreibweise von Registry-Einträgen (z.B. TaaTrace**Supress**MVS statt TaaTrace**Suppress**MVS)
- Bessere Lösungen für Registry-Einträge (z.B. IsaEnterOptionsNet sollte nur in noch in Ausnahmen verwendet werden)

3. Lösungsansatz

3.1. *Allgemein*

Mit Tweak-Edge wird eine Oberfläche zur Verfügung gestellt, welche von den konkreten Registry-Einträgen abstrahiert und das eigentlich vom Anwendungsentwickler verfolgte Ziel in den Vordergrund stellt.

Beispiel: Der Entwickler will für seine Anwendung Trace-Sätze für die von dieser Anwendung gerufenen Host-Bausteine erzeugen. Welche Eintragungen in der Registry sind vorzunehmen?

- TAATraceSuppressMVS?
- IsaEnterOptions.Net?
- TaaTraceHost?
- Optionen beim Starten der Anwendung?
- TraceDir?
- ...?

Mit der Benutzung von TwkEdge soll der Anwendungsentwickler diese Details nicht alle selber bedenken müssen. Das Werkzeug sorgt dafür, dass für das verfolgte Ziel die richtigen Eintragungen an den richtigen Stellen gemacht werden.

3.2. *Aufgaben*

TwkEdge arbeitet mit sogenannten Aufgaben. Eine Aufgabe umfasst in der Regel eine Menge von Einstellungen für die Arbeitsweise der TAA-Infrastruktur, welche für das Erreichen eines bestimmten Ziels notwendig sind.

Derzeit sind die Aufgaben:

- Debugmodus für einen Baustein anpassen
- Einen Baustein ausführen
- Trace-Einstellungen ändern
- Uhrzeit-Einstellungen ändern

umgesetzt.

3.3. *Wizards*

Die Durchführung der Aufgaben wird durch Wizards unterstützt. Ein Wizard führt den Entwickler durch die einzelnen Schritte der Aufgabe, d.h. der Anwender kann sich über eine „Weiter“-Funktion immer zum richtigen nächsten Schritt führen lassen. Das Werkzeug erkennt überflüssige Schritte und lässt diese aus.

Die Funktion „Fertigstellen“ führt immer direkt zur Übersicht, außer man befindet sich bereits dort. Wird die Funktion „Fertigstellen“ aus der Übersicht gewählt, wird die Aufgabe ausgeführt und das Ergebnis (kurz) dokumentiert.

Das Grundprinzip des Wizards ist für alle Aufgaben immer wieder gleich. Die einzelnen Schritte sollten für einen Anwendungsentwickler weitgehend selbsterklärend sein. In diesem Dokument werden die beiden Aufgaben: „Debugmodus für einen Baustein anpassen“ und „Einen Baustein ausführen“ detaillierter beschrieben.

Einmal mit Hilfe des Wizards vorgenommene Einstellungen können in Form von sogenannten Vorlagen gesichert werden. Somit können einmal vorgenommene Einstellungen jederzeit wiederverwendet und modifiziert werden. Das Arbeiten mit Vorlagen ist bei allen Aufgabentypen möglich.

4. Aufgabe: Debugmodus für einen Baustein anpassen

4.4. Allgemein

Das Ziel dieser Aufgabe ist es, verschiedene Einstellungen in der Registry vorzunehmen, welche für das Debugging eines TAA-Moduls erforderlich sind.

4.5. Einstellungen aktivieren

Einstellungen zu dieser Aufgabe können über die Oberfläche von Twk-Edge aktiviert und deaktiviert werden. Die nachfolgenden Schritte beschreiben die Aktivierung von Einstellungen. Auf der Einstiegsmaske wird die Option „Die Debug-Einstellungen sollen für das Modul aktiviert werden“ ausgewählt.

4.5.1. Schritt 1: Auswahl des Bausteins

Es wird das Modul ausgewählt, für welches die Debug-Einstellungen gemacht werden sollen. Dies kann direkt durch Eintippen oder Kopieren über die Zwischenablage erfolgen. Weitere Möglichkeiten bestehen darin, sich eine Liste der Bausteine anzuzeigen, welche bereits in der eigenen Registry eingetragen sind oder sich die Liste aller Bausteine aus den Ressourcen geben zu lassen. Die nachfolgende Abbildung zeigt diese Auswahl.

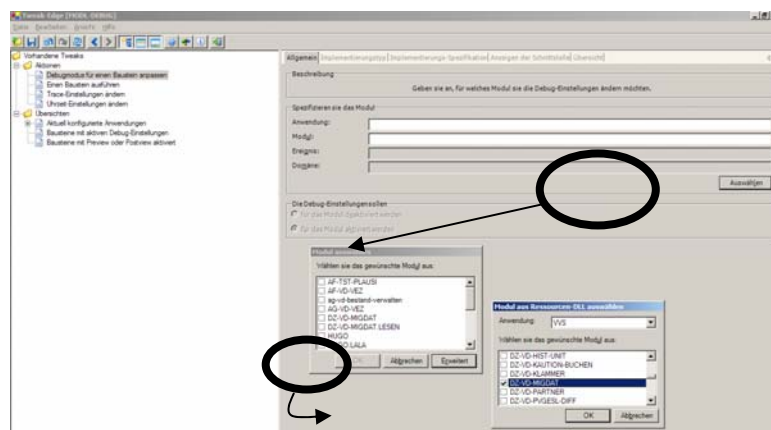


Abbildung 1: Baustein für Debugmodus auswählen

Die Angabe des Bausteins kann weiter eingeschränkt werden durch die Angabe eines Ereignisses für das die Debugging-Einstellungen gelten sollen.

4.5.2. Schritt 2: Implementierungstyp auswählen

Hier wird der für das Debugging verwendete Implementierungstyp ausgewählt. Derzeit werden keine Standards aus der bereits vorhandenen Konfigurationsbeschreibung des Moduls abgeleitet (also z.B. NetExpress als Default für COBOL-Module).

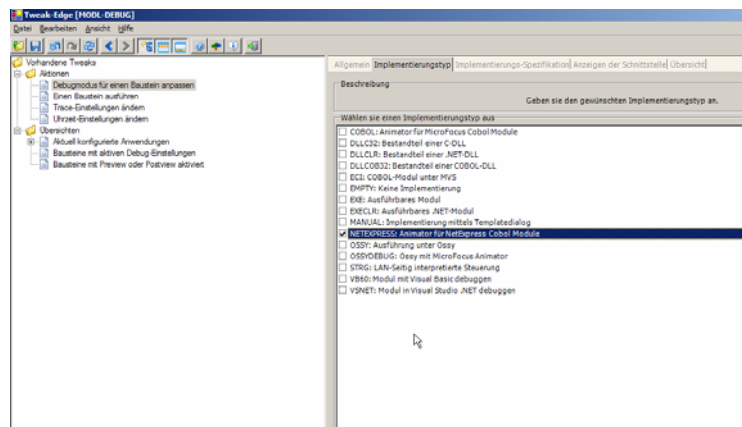


Abbildung 2: Implementierungstyp auswählen

4.5.3. Schritt 3: Implementierungsspezifikation auswählen

Als nächstes wird die Implementierungsspezifikation ausgewählt.

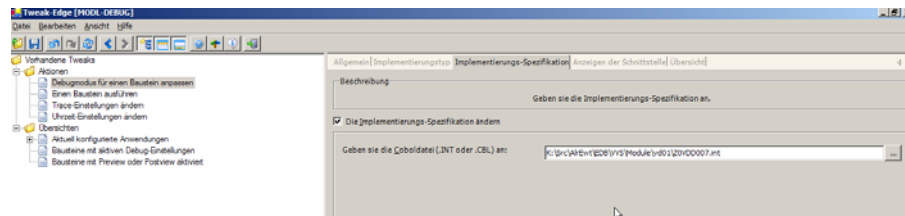


Abbildung 3: Auswahl der Implementierungsspezifikation

4.5.4. Schritt 4: Anzeigen der Schnittstelle angeben

Als nächstes werden die Einstellungen zu Pre- und Postview für das Modul gemacht.

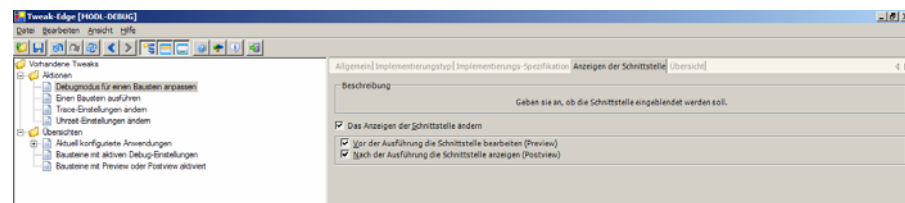


Abbildung 4: Einstellungen zu Pre- und Postview

4.5.5. Schritt 5: Übersicht

In der Übersicht werden die gemachten Einstellungen noch mal aufgelistet. Es kann jederzeit zurückgegangen werden und Einstellungen geändert oder verworfen werden.

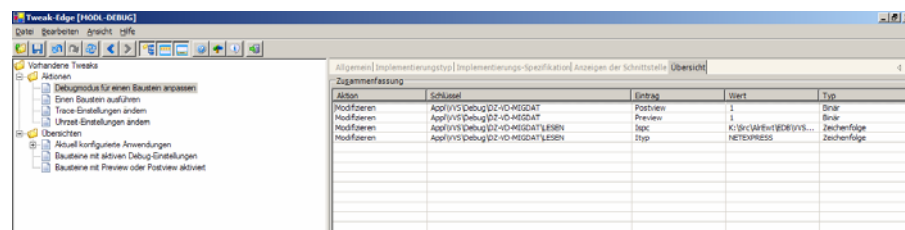


Abbildung 5: Übersicht der Einstellungen

4.5.6. Schritt 6: Fertigstellen

Die Funktion Fertigstellen aus der Übersicht heraus gewählt, führt die Aufgabe aus und dokumentiert sie.

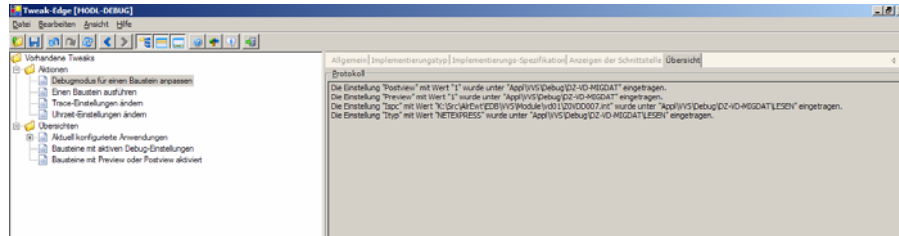


Abbildung 6: Funktion Fertigstellen

4.6. Deaktivieren von Einstellungen

Die Option „Die Debug-Einstellungen sollen für das Modul deaktiviert werden“ auf dem Tab „Allgemein“ kann dazu verwendet werden, um die gemachten Einstellungen für einen Baustein vollständig zu deaktivieren.

5. Aufgabe: Einen Baustein ausführen

5.7. Allgemein

Das Ziel dieser Aufgabe ist es, die Ausführung eines Bausteins mit tstdo vorzubereiten und auszuführen. Der Schwerpunkt liegt hier auf der richtigen Auswahl von Optionen für die Durchführung.

5.7.1. Schritt 1: Auswahl des Bausteins

Zunächst wird der Baustein ausgewählt, für den die Debug-Einstellungen gemacht werden sollen. Dies kann direkt durch Eintippen oder Kopieren über die Zwischenablage erfolgen. Weitere Möglichkeiten bestehen darin, sich eine Liste der Bausteine anzuzeigen, welche bereits in der eigenen Registry eingetragen sind oder sich die Liste aller Bausteine aus den Ressourcen geben zu lassen. Die nachfolgende Abbildung zeigt diese Auswahl.

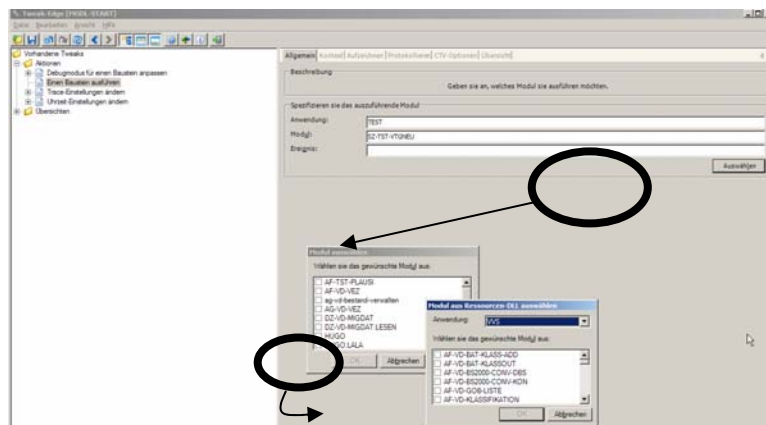


Abbildung 7: Baustein für Ausführung auswählen

Die Angabe des Bausteins kann weiter eingeschränkt werden durch die Angabe eines Ereignisses für das die Debugging-Einstellungen gelten sollen.

5.8. Schritt 2: Kontext bearbeiten

In diesem Schritt können Pre- und Postview-Einstellungen vorgenommen werden. Außerdem kann festgelegt werden, ob die Bausteinschnittstelle zu Beginn der Ausführung mit Daten aus einer Datei versorgt werden soll, bzw. die Schnittstellendaten am Ende der Ausführung gespeichert werden sollen.

5.9. Schritt 3: Aufzeichnung festlegen

In diesem Schritt können Einstellungen zum Aufzeichnen des Bausteins festgelegt werden. Über Optionen kann gesteuert werden, was mit den aufgezeichneten Daten nach dem Ende der Durchführung geschehen soll.

5.10. Schritt 4: Protokollierung steuern

In diesem Schritt können Einstellungen zum Protokollieren der Bausteinausführung festgelegt werden.

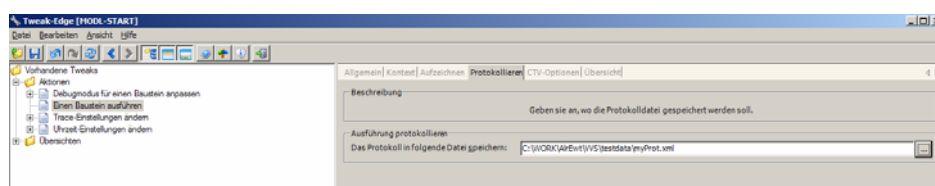


Abbildung 8: Protokollierung steuern

Ein Protokoll der Ausführung enthält u. a. einen Verweis auf eine während der Ausführung erstellte Tracedatei. Um diese Tracedatei zu öffnen, kann die Protokolldatei einfach auf eine geöffnete Instanz des Test-Edge gezogen werden. Somit ist es nicht notwendig zu wissen, wo und unter welchen Namen die Trace-Datei tatsächlich abgelegt wurde.

5.11. Schritt 5: CTV-Einstellungen vornehmen

Wenn es sich um einen CTV-Baustein handelt, können in diesem Schritt Einstellungen vorgenommen werden, ob

- das Schriftgut im CTV-Dialog bearbeitet werden kann
- im Wordviewer angezeigt werden soll
- und/oder auf einen Drucker ausgegeben werden soll

Wenn es sich bei dem auszuführenden Baustein nicht um einen CTV-Baustein handelt, wird dieser Schritt ausgelassen.

5.12. Schritt 6: Übersicht

Die Übersicht zeigt vor der Ausführung noch einmal das vollständige, vom Wizard zusammengestellte tstdo start Kommando.

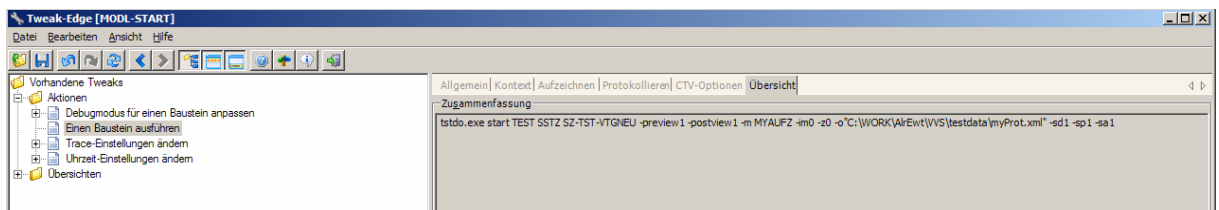


Abbildung 9: Übersicht zur Ausführung eines Bausteins

Schritt 7: Ausführung

Die Ausführung des Bausteins wird aus dem Werkzeug heraus angestoßen und nach Abschluss dokumentiert.

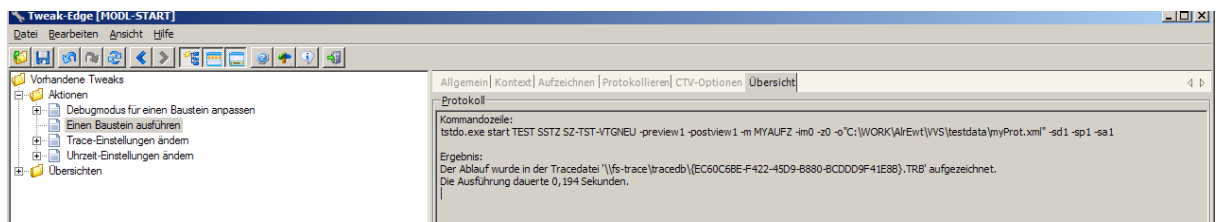


Abbildung 10: Protokoll der Ausführung eines Bausteins

6. Vorlagen

Die im vorherigen Kapitel beschriebenen Schritte sollen vom Anwendungsentwickler natürlich nicht immer wieder durchlaufen werden, sondern nur einmalig. Danach sollen die Einstellungen wieder verwendet werden können.

Die Wiederverwendung wird ermöglicht durch die Verwendung von sogenannten Vorlagen. In diesen werden die für einen Baustein gemachten Einstellungen abgelegt und können zu einem beliebigen Zeitpunkt wieder geladen werden.

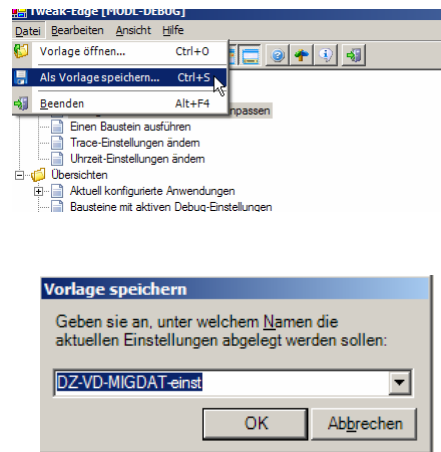


Abbildung 11: Vorlage für Einstellungen erstellen

Über die Menü-Funktion „Vorlage öffnen“ kann eine Vorlage dann erneut geladen werden. Da alle Einstellungen nun in dem Werkzeug geladen sind, ändert sich die Abfolge der zu durchlaufenden Schritte für die Aufgabe. Der Wizard berücksichtigt dies und die „Fertigstellen“-Funktion führt zur vollständig ausgefüllten Übersicht, da ja alle Angaben bereits vollständig sind.

Die „Weiter“-Funktion führt auch beim Arbeiten mit Vorlagen immer zum nächsten Schritt, in dem etwas geändert werden könnte.

7. Dokumentation und Hilfe

Für die einzelnen Aufgaben soll kontextsensitive Hilfe bereitgestellt werden, welche über die Hilfe-Funktion (F1) angesteuert werden kann. Derzeit verzweigt die Hilfe auf eine Teamwise-Internet-Seite. Das Ziel für die Hilfe kann aber auf eine beliebige URL eingestellt werden.



Abbildung 12: Kontextsensitive Hilfe

8. Übersichten

Das Werkzeug bietet die Möglichkeit bestimmte Einstellungen im Überblick darzustellen. Insbesondere solche die über die direkten Ansichten der Registry nicht ohne weiteres zu erstellen sind, z. B. weil die Informationen an unterschiedlichen Stellen in der Registry untergebracht sind (z. B. unterschiedliche HKLM wie LOCAL_MACHINE und CURRENT_USER, unterschiedliche Abschnitte wie Config und Debug)

8.13. Übersicht "Aktuell konfigurierte Anwendungen"

Diese Übersicht gibt insbesondere eine Antwort auf die Frage: Mit welcher Version eines beliebigen Projektes arbeite ich derzeit?

Anwendung	Kürzel	Version	Komponententyp	Zusatzanwendungen	Debuggen erlaubt
AARCH	AA	029			Ja
ALDEX	AX	001			Ja
ALGE	AU	001			Ja
BB5	BB	001			Ja
CHANCE	BM	001			Ja
CTV	CT	029			Ja
DMS	DO	027			Ja
GAUS	GA	001			Ja
IFPOPO	IP	001			Ja
PADEP	AS	001			Ja
INQUEL	IQ	001			Ja
LEFRM	LF	029			Ja
LEFRM	LI	029			Ja
LEPS	PS	029			Ja
LEVERT	EL	029			Ja
LUMIG	LV	001			Ja
MEDWAR	MW	001			Ja
PADEP	PA	001			Ja
PAVIS	AD	029		BB5	Ja
PRIVAT	SH	029			Ja
REBE	RE	001			Ja
REICH	R2	001			Ja
REICHU	RU	001			Ja
RVLISE	RL	001			Ja
RVSACH	RS	001			Ja
SACHAL	SA	001			Ja
SCHACN	SC	029			Ja
SWANG	SV	029			Ja
TAA	AZ	007			Ja
TEST	ST	001			Ja
TIANG	TI	001			Ja
TLAB	CA	001			Ja
UDEM_C	ZS	001			Ja
VO	VO	007			Ja
VRM-VS	VV	029			Ja
VTRB2	VB	001			Ja
VTRB3	VT	029			Ja
VVC	VC	001			Ja
VVS	VO	029			Ja
ZENTWD	ZK	029			Ja

Abbildung 13: Übersicht "Aktuell konfigurierte Einheiten"

8.14. Übersicht „Bausteine mit aktiven Debug-Einstellungen“

Diese Übersicht gibt insbesondere eine Antwort auf die Frage: Welche Bausteine sind aktuell für das Debugging konfiguriert.

Dabei zeigt die Übersicht nur die Bausteine, bei denen die Debugging-Einträge auch wirklich „ziehen“, d.h. wenn für ein bestimmtes Projekt Debugging grundsätzlich unterdrückt ist (projektspezifisches DebugAllowed=0) werden die Einträge dieser Bausteine auch nicht angezeigt. Damit zeigt diese Übersicht mehr als die direkte Ansicht in der Registry.

Wenn bestimmte Einstellungen nicht den Erwartungen des Anwenders entsprechen, kann aus der Übersicht über die Funktion „Aktion / Bearbeiten“ direkt auf die Aktion zur Anpassung der Einstellungen verzweigt werden. Alle vorhandenen Einstellungen werden dabei übernommen.

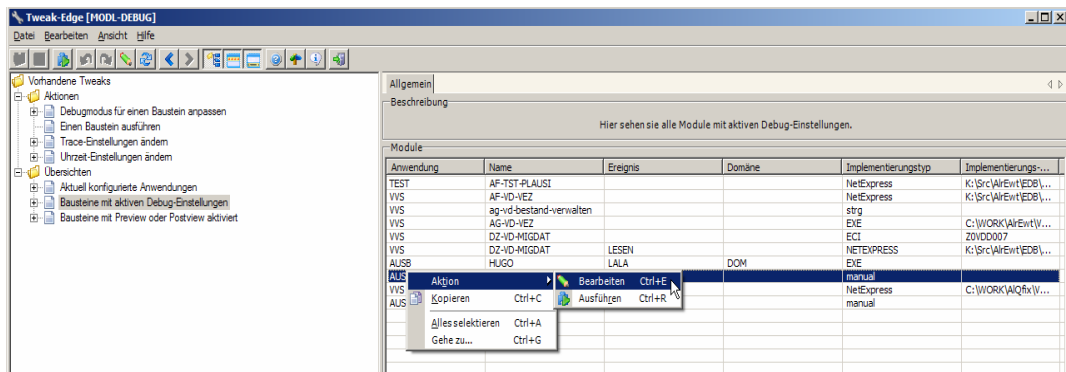


Abbildung 14: Übersicht „Bausteine mit aktiven Debug-Einstellungen“

8.15. Übersicht „Bausteine mit Post- bzw-Preview aktiviert“

Diese Übersicht beantwortet die Frage, für welche Bausteine entweder Pre- oder Postview oder beide Einstellungen gesetzt sind.

Dabei zeigt die Übersicht nur die Bausteine, bei denen die Debugging-Einträge auch wirklich „ziehen“, d.h. wenn für ein bestimmtes Projekt Debugging grundsätzlich unterdrückt ist (projektspezifisches DebugAllowed=0) werden die Einträge dieser Bausteine auch nicht angezeigt. Damit zeigt diese Übersicht mehr als die direkte Ansicht in der Registry.

Wenn die Einstellungen nicht den Erwartungen des Anwenders entsprechen, kann aus der Übersicht über die Funktion „Aktion / Bearbeiten“ direkt auf die Aktion zur Anpassung der Einstellungen verzweigt werden. Alle vorhandenen Einstellungen werden dabei übernommen.

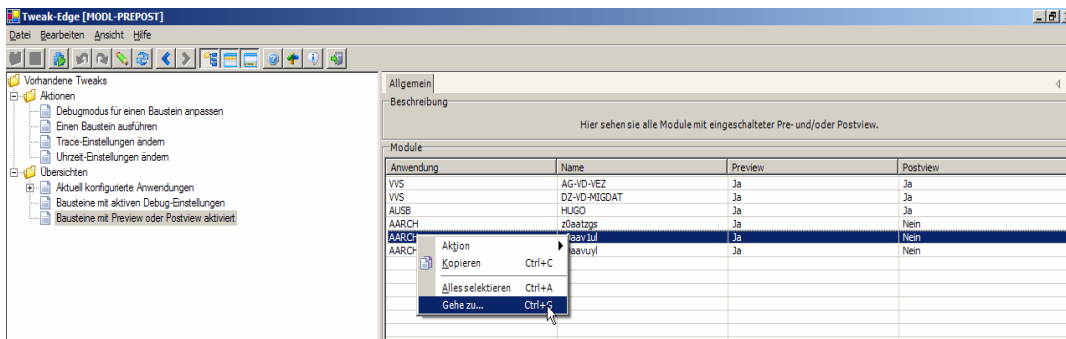


Abbildung 15: Übersicht „Bausteine mit Post- bzw-Preview aktiviert“

8.16. Aktionen aus Übersichten heraus ausführen

Aus Übersichten heraus können für die einzelnen Instanzen oft direkt Aktionen angestoßen werden. So kann z.B. in der Übersicht, welche die Debug-Einstellungen für Module darstellt, ein Modul ausgewählt werden und für dieses z.B. die Aktion „Ausführen“ gewählt werden.

Alle bekannten und für die Aktion relevanten Angaben zu diesem Modul werden dann bereitgestellt.

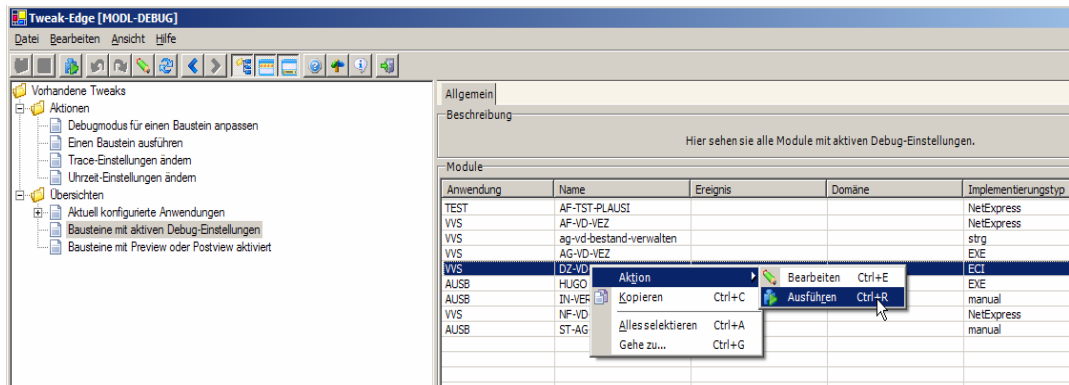


Abbildung 16: Aktion für ein Modul in einer Übersicht auswählen