Clean ABAP

Sauberer Code, smarter Erfolg: Tools für nachhaltige ABAP-Entwicklung

Webinar, 31.1.2025, 10:00 Uhr

Sören Schlegel

SAP Development Architect

Twitter: @SoSchlegel87
LinkedIn: https://www.linkedin.com/in/soschlegel

Themenschwerpunkte

- Konzeption & Management von Entwicklungsprojekten
- SAP Architektur moderner Anwendungen
- ABAP RAP und S/4HANA Entwicklung
- SAP CDS, OData und Cloud

- **SAP Champion** 🏅

Johann Fößleitner

Development Architect

eMail: johann.foessleitner@cadaxo.com
LinkedIn: https://www.linkedin.com/in/foessleitnerj/

Beratungsschwerpunkte

- Konzeption & Management von Entwicklungsprojekten
- Fiori, Fiori Elements & ABAP RESTful
- S/4HANA Custom Development
- ABAP Trainer, ABAP Coach

- Clean Code

- **SAP Champion**

Domi Bigl

Senior Consultant at Cadaxo

eMail: dominik.bigl@cadaxo.com
LinkedIn: https://www.linkedin.com/in/domi-bigl

Beratungsschwerpunkte

- Konzeption & Management von Entwicklungsprojekten
- Qualitymanagement & Performanceoptimierung
- ABAP Units
- SAP Fiori, SAP UI5

- **ABAP, UI5, … Champion**

cadaxo

- **Consulting, Workshops, Coaching, Produkte**
- ABAP, CDS/RAP, S/4 Extension, Custom Code Migration, SQL Cockpit, …
- https://www.cadaxo.com

S/4 FIT

Brandeis Consulting

- **Consulting, Schulungen, Workshops**
- ABAP, CDS/RAP, SQLScript, …
- https://www.brandeis.de

- **S/4FIT Modern ABAP Workshop**
  - 10./11. März 2025
  - Salzburg
  - Instructor: **Johann Fößleitner**
  - Themen:
    - Expressions, ABAP SQL,
      SQL Expressions, String Templates, …

Community News

ABAP Webinare 2025 - Termine

Freitag,	31. Jänner 2025
Freitag,	28. März 2025
Mittwoch,	28. Mai 2025
Freitag,	26. September 2025
Freitag,	28. November 2025

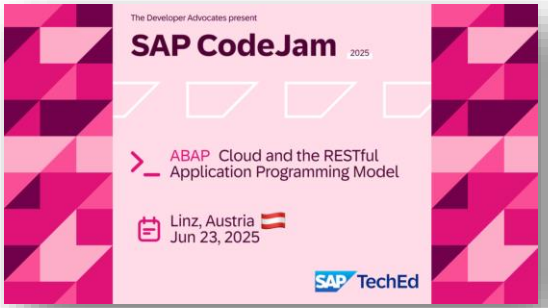
Immer um 10:00 Uhr

https://www.cadaxo.com/webinare/

- **ABAPConf 2025**
  - 4. und 5. Juni 2025
  - Free | Live | Onsite | Online
    - Mafinex Mannheim
    - ???? Wien
    - …
  - https://abapconf.org/abapconf2025/
  - https://www.linkedin.com/groups/13033358/

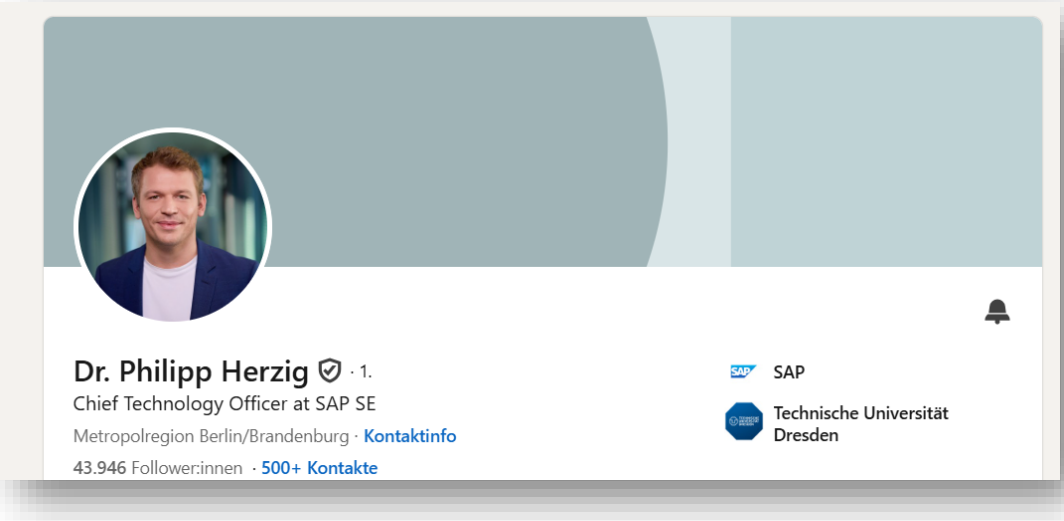
- **SAP Inside Track Vienna 2025**
  - 21. März 2025 09:00 – 17:30
  - 1130 Wien, Mantlergasse 30-32
  - Christian Punz, SCC

- **SAP CodeJam – ABAP Cloud & RAP**
  - 23. Juni, Linz 09:00 – 16:00 Uhr
  - Host: **Develite IT Solutions** aus Linz
  - Instructor: **Andre Fischer**
  - Adresse: TechCenter Hafenstraße 47-51
    4020 Linz AUSTRIA



- **SAP Stammtisch Wien**
  - Monatliches Treffen irgendwo in Wien
  - Organisiert über die SAP Location Community Group
  - Nächster Termin: 13. Februar 2025

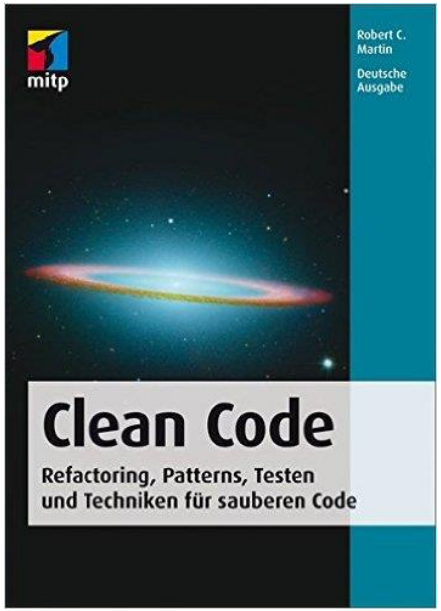
- **SAP CTO**
  - **Philipp Herzig** ist der neue CTO der SAP
  - https://www.linkedin.com/in/philipp-herzig/

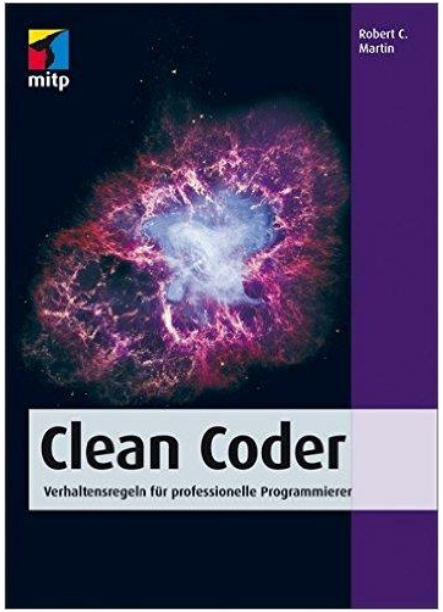


Heute kein RAP

Unsere bisherigen RAP Webinare

Clean ABAP





Clean Code – von Robert C. Martin „Uncle Bob“

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

***Martin Fowler**, Pionier der agilen Softwareentwicklung*

"The proper use of comments is to compensate for our failure to express ourself in code. Note that I used the word failure. I meant it. Comments are always failures. We must have them because we cannot always figure out how to express ourselves without them, but their use is not a cause for celebration.

Robert C. Martin, *Clean Code*

Gibt's da was von…

SAP Code Style Guides

REUSE compliant

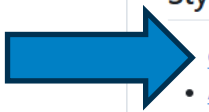
Continuous Release · Open Source · Grassroots Project · Optional · Ecosystem

This repository provides SAP's style guides for coding.

Programming languages enable us to say the same thing in different ways. While all of them may be correct, some may be more efficient, easier to understand, and more robust than others.

Our style guides want to show up differences and guide you towards code that has a healthy balance between all of these qualities.

Style Guides



- Clean ABAP
- ABAP Code Reviews

Continuous Release

These guides are updated **continuously**, meaning any change is reviewed and immediately put "live", without special publication versions.

As programming languages and our understanding of them evolve, we believe that these guides are "work in progress" and will probably never see a status "finished"; as agile developers, we welcome this.

Was verstehst du unter „Clean Code“?

- Variablen, Methoden und Klassen sollten so benannt werden, dass man ihren Zweck sofort erkennt.

- Vermeiden von Abkürzungen oder kryptischen Bezeichnungen, die spätere Wartung erschweren.

Use plural

> Clean ABAP > Content > Names > This section

There is a legacy practice at SAP to name tables of things in singular, for example `country` for a "table of countries". Common tendency in the outside world is to use the plural for lists of things. We therefore recommend to prefer `countries` instead.

Use snake_case

> Clean ABAP > Content > Names > This section

ABAP is case insensitive which is why we recommend following the convention to use `snake_case` consistently.

There's a character limit for names, e.g. 30 characters for methods. When you reach the maximum length of an object, don't fall back to using `flatcase` or `UPPERCASE`. Try to conscientiously use abbreviations instead (see Use same abbreviations everywhere).

```
" a variable which contains the maximum reponse time measured in milliseconds
DATA max_response_time_in_millisec TYPE i.
```

is better than

```
" anti-pattern
DATA maxresponsetimeinmilliseconds TYPE i.
```

Avoid abbreviations

> Clean ABAP > Content > Names > This section

If you have enough space, write out names in full. Start abbreviating only if you exceed length limitations.

If you do have to abbreviate, start with the *unimportant* words.

Abbreviating things may appear efficient at first glance, but becomes ambiguous very fast. For example, does the "cust" in `cust` mean "customizing", "customer", or "custom"? All three are common in SAP applications.

Wunsch an die SAP

Methoden, Views und Klassen auf 30 Charakter begrenzt ➔ Ich glaube, dass war NICHT gemeint mit „kurzen Methoden“

Wunsch an die SAP

Unterstützung von CamelCase im Eclipse-Tree!

- Code sollte so formatiert sein, dass er leicht überblickbar ist

ABAP Code Cleaner

- Kurze Methoden und Funktionen, die genau eine Aufgabe erfüllen (Single-Responsibility-Prinzip).

- Möglichst wenig verschachtelte Strukturen (z. B. if-else, Schleifen), damit der Ablauf klar nachvollziehbar bleibt.

Consider decomposing complex conditions

> Clean ABAP > Content > Conditions > This section

Conditions can become easier when decomposing them into the elementary parts that make them up:

```
DATA(example_provided) = xsdbool( example_a IS NOT INITIAL OR
                                  example_b IS NOT INITIAL ).

DATA(one_example_fits) = xsdbool( applies( example_a ) = abap_true OR
                                  applies( example_b ) = abap_true OR
                                  fits( example_b ) = abap_true ).

IF example_provided = abap_true AND
   one_example_fits = abap_true.
```

instead of leaving everything in-place:

```
" anti-pattern
IF ( example_a IS NOT INITIAL OR
     example_b IS NOT INITIAL ) AND
   ( applies( example_a ) = abap_true OR
     applies( example_b ) = abap_true OR
     fits( example_b ) = abap_true ).
```

> Use the ABAP Development Tools quick fixes to quickly extract conditions and create variables as shown above.

Active

[redacted] Purchase

[redacted] (Herbert [redacted]

October 5, 2022

[redacted] (Herbert [redacted]

Monday, January 20, 2025, 3:51:03 PM

EN

[redacted]

t: -

```
5623
5624                    COMMIT WORK AND WAIT.
5625                  ENDIF.
5626                ENDIF.
5627              ENDLOOP.
5628            ENDLOOP.
5629          ENDIF.
5630        ENDLOOP.
5631      ENDIF.
5632    ENDMETHOD.
5633  ENDCLASS.
```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Pro... Tem... Tran... Boo... ABA... Fee... Sear... ABA... Quic... ATC ... Hist... ABA... Lo

References fo[redacted] - ? matches in 513 objects

Einfache Struktur und Lesbarkeit

Clean ABAP Tools

Wunsch an die SAP

BRF+ oder „schönen Nachfolger“ bereitstellen → gerade auch in der Cloud

Composition over Inheritance

- Klassen und Module sollten jeweils eine klar umrissene Aufgabe haben (Single-Responsibility-Prinzip).

- Gutes, durchdachtes Design trägt zu einer verständlichen Architektur bei, in der man Code leichter anpassen und erweitern kann.

- „Don’t repeat yourself“ (DRY): Logik soll nicht mehrfach kopiert werden, sondern an einer Stelle zentral verwaltet werden.

- Dadurch sinkt das Risiko von Inkonsistenzen und der Wartungsaufwand wird geringer.

Beispiel aus der SAP-Welt:

- ZSD_SEND_EMAIL
- ZMM_SEND_EMAIL

- Testbarer Code ist meist auch „cleaner“ Code, da man Schnittstellen und Abhängigkeiten klar definiert.

- Automatisierte Tests (Unit- und Integrationstests) helfen, Fehler früh zu entdecken und neue Funktionen leichter hinzuzufügen.

- Keine Programme oder Klassen anlegen „für Tests“

➔ Vorteil von Interfaces: Anstelle von Objekten werden Interfaces in eine Methode übergeben

➔ Der Aufrufer kann somit für seinen Test eine eigene Klasse anlegen, welche das Interface implementiert und hier die „Mock-Daten“ einbauen

Problem in der SAP-Welt:

- Wir rufen einen BADI auf, der einfach stabile 1283 Parameter hat und ebenso viele Tabellen aufruft!

- Wie mockt man das?

- Exceptions sollten an sinnvollen Stellen geworfen bzw. abgefangen werden.

- Fehlerfälle sollten nachvollziehbar dokumentiert sein, damit andere Entwickler*innen wissen, wie das System damit umgeht.

Prefer RAISE EXCEPTION NEW to RAISE EXCEPTION TYPE

> Clean ABAP > Content > Error Handling > Throwing > This section

Note: Available from NW 7.52 onwards.

```
RAISE EXCEPTION NEW cx_generation_error( previous = exception ).
```

in general is shorter than the needlessly longer

```
RAISE EXCEPTION TYPE cx_generation_error
  EXPORTING
    previous = exception.
```

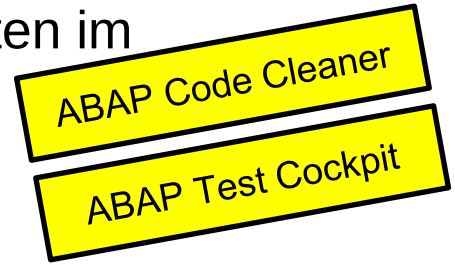
However, if you make massive use of the addition `MESSAGE`, you may want to stick with the `TYPE` variant:

```
RAISE EXCEPTION TYPE cx_generation_error
  MESSAGE e136(messages)
  EXPORTING
    previous = exception.
```

- „Clean Code“ entsteht selten im ersten Wurf, sondern wird durch regelmäßiges Refactoring verbessert.

- Kontinuierliche Refactorings vermeiden technische Schulden und sorgen für eine stetig hohe Codequalität.

- Ein konsistenter Stil (z. B. Code Conventions zu Einrückung, Benennungen, Kommentarformatierung) erleichtert das Verständnis für alle Beteiligten im Team.



- Projektweite oder unternehmensweite Standards dienen als Leitfaden und sorgen für eine einheitliche Codebasis.

- „Lokale Klassen“ waren ein Anti-Pattern – jetzt „aufgeweicht“
- Hungarian-Notation: Bitte erst dann entfernen, wenn das Coding sonst „clean“ ist!
- Verwendung KTDs: Dokumentation was warum gemacht wird und nicht was gemacht wird!

code pal for ABAP

- **Was ist *code pal for ABAP*?**
  - Sammlung von Clean ABAP ATC Checks
  - Verbesserung der Codequalität durch die Überprüfung von Clean ABAP-Richtlinien
  - Open Source via Github
  - Installation via abapGit

- ABAP 7.40 SP8
- ~55 Checks
- Nur OnPremise
- Nicht remote fähig

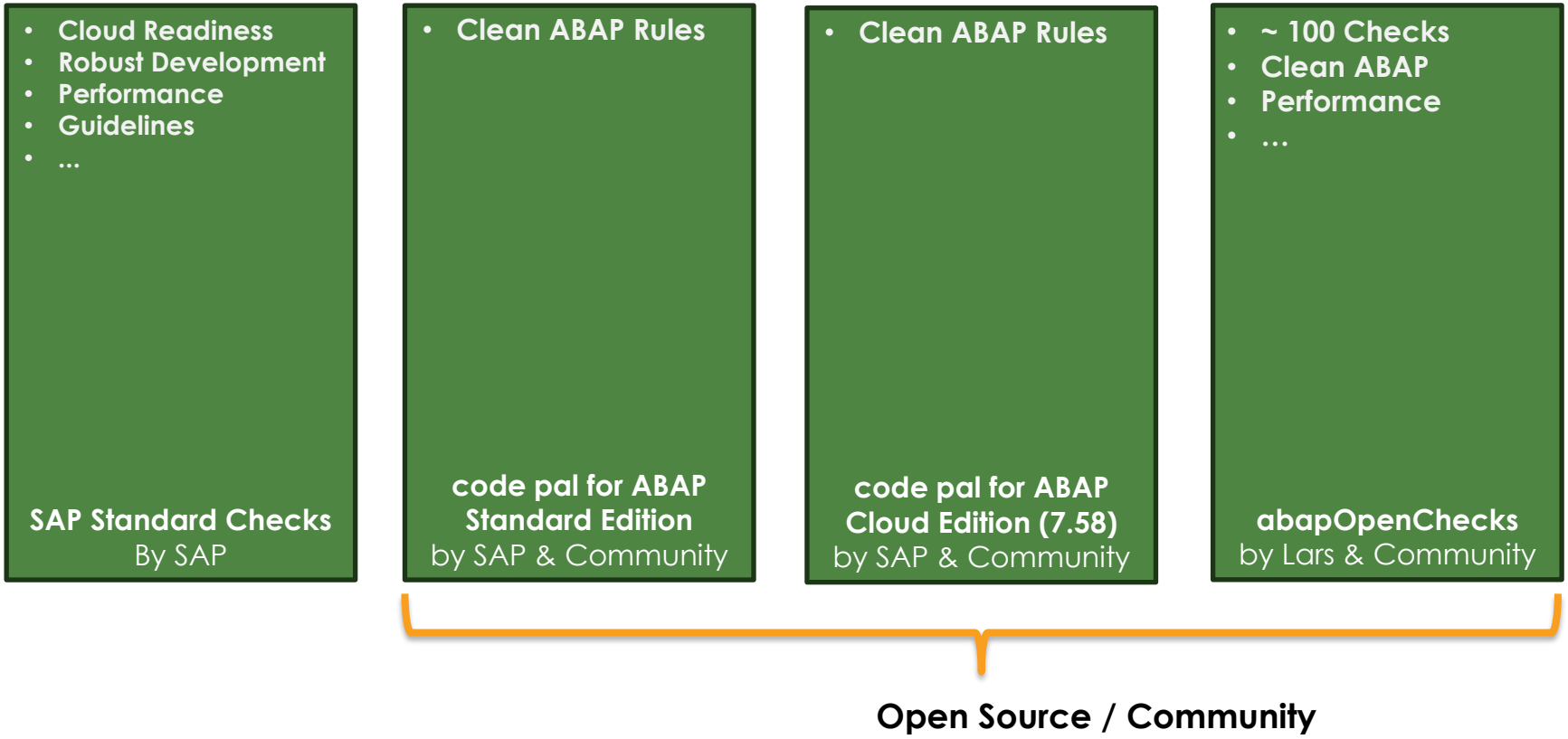
Classic Edition

https://github.com/SAP/code-pal-for-abap

- ABAP 7.58 (2023)
- ~15 Checks
- Cloud + OnPremise
- Remote fähig
- Quick fixes (ADT)

Cloud Edition

https://github.com/SAP/code-pal-for-abap-cloud


• Cloud Readiness
• Robust Development
• Performance
• Guidelines
• ...
SAP Standard Checks
By SAP
• Clean ABAP Rules
code pal for ABAP
Standard Edition
by SAP & Community
• Clean ABAP Rules
code pal for ABAP
Cloud Edition (7.58)
by SAP & Community
• ~ 100 Checks
• Clean ABAP
• Performance
• ...
abapOpenChecks
by Lars & Community
Open Source / Community

- **Ausgewählte Beispiele**
  - Avoid DEFAULT KEY
    - Problem bei SORT, DELETE ADJACENT, BINARY SEARCH
  - Prefer INSERT INTO vs. APPEND
    - APPEND in sortierte Tabelle problematisch
  - Number of Methods, Attributes, …
    - Ähnlich auch im Standard in den OO Size Matrics vorhanden
  - Method Name Misleading
    - Wenn Returning type ABAP_BOOL – Name sollte is_, has_, …

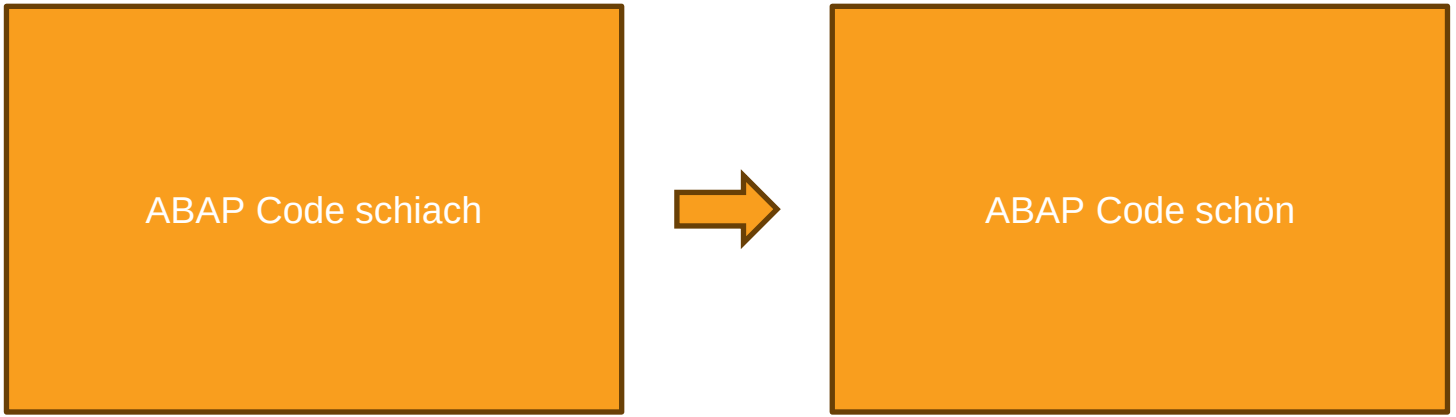
- **Demo**
  - Gui: ABAPGit herzeigen
  - Gui: Checks aktivieren (SCI)
  - Gui: Checks pflegen – Variante Z_FOESS
  - Gui: Check durchführen
    - Hinweis: Ergebnisse Prio 3, kann übersteuert werden
  - Gui: Clean Code Profiler
  - ADT: Check
  - ADT: Check Quick Fix

- **Code pal for ABAP**
  - Vorteile
    - Steigerung der Code Qualität
  - Anwendungstipps
    - ABAP Cleaner first
    - Nicht Freigaben blockieren bzw. nur ausgewählte wichtige Checks
  - Ausblick
    - Erweiten/Verbessern der Cloud Edition

ABAP Cleaner

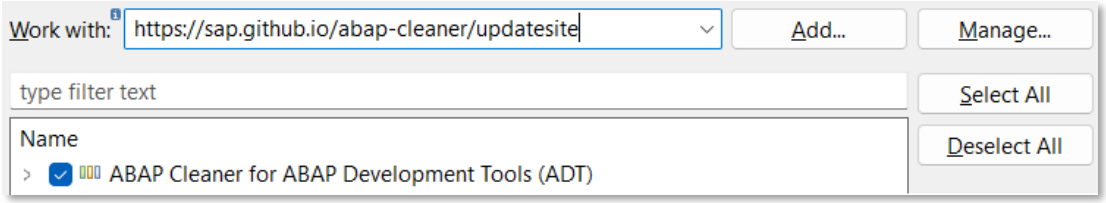
„ABAP cleaner is an configurable tool with the ambition to automate whatever can be automated with respect to ABAP code style.“

- **ABAP Cleaner**
  - Mehr als 75 Cleanup Regeln für ABAP
  - Direkt in ADT, am Frontend

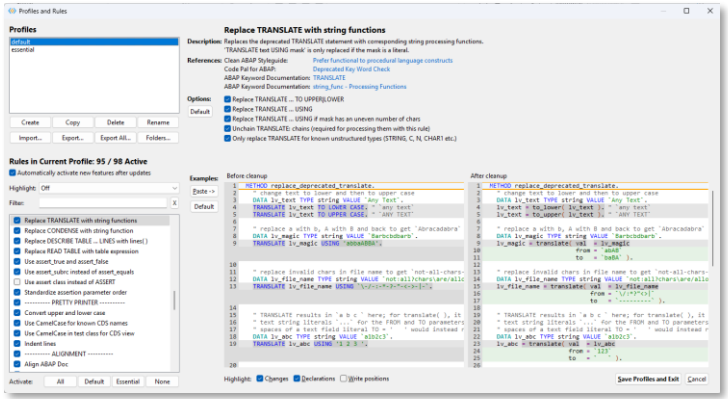


- **ABAP Cleaner**
  - 2020 Geburt als stand-alone Anwendung
    - Zuerst C#, dann Java
  - 2021 ADT Plugin
    - Sorry -> SAPGui wird (vermutlich) NIE kommen
  - 2023 Open Source

https://github.com/SAP/abap-cleaner?tab=readme-ov-file#requirements-and-installation



- **Konfigurationsmöglichkeiten**
  - Profile
    - default, essential, eigene
    - Import/Export, Team-Folder
  - Feineinstellungen je Regel



- **ABAP Cleaner – Open Source / Mitmachen**
  - Fehler melden
  - Ideen einbringen
  - Neue Prüfungen implementieren
  - …

- **ABAP Cleaner - Wie geht‘s weiter**
  - Nahe Zukunft
    - Neue Bereinigungsregeln
    - Neue Konfigurationsmöglichkeiten
    - Neue UI Features
  - Fernere Zukunft
    - EML
    - SQLScript / AMDP

ABAP Tools for Clean ABAP

Björn Jüliger & Jörg-Michael Grassau

6. Juni 2024 | ABAPConf 2024 Europe

Open Source Project ABAP Cleaner

Jörg-Michael Grassau

5. März 2024 | openSAP

Tools for Clean Code

Björn Jüliger & Jörg-Michael Grassau

30. September 2024 | Devtoberfest

Code Pal – ABAP test cockpit checks for Clean ABAP

Thomas Fiedler, Björn Jüliger

21. Februar 2024 | openSAP

Danke!

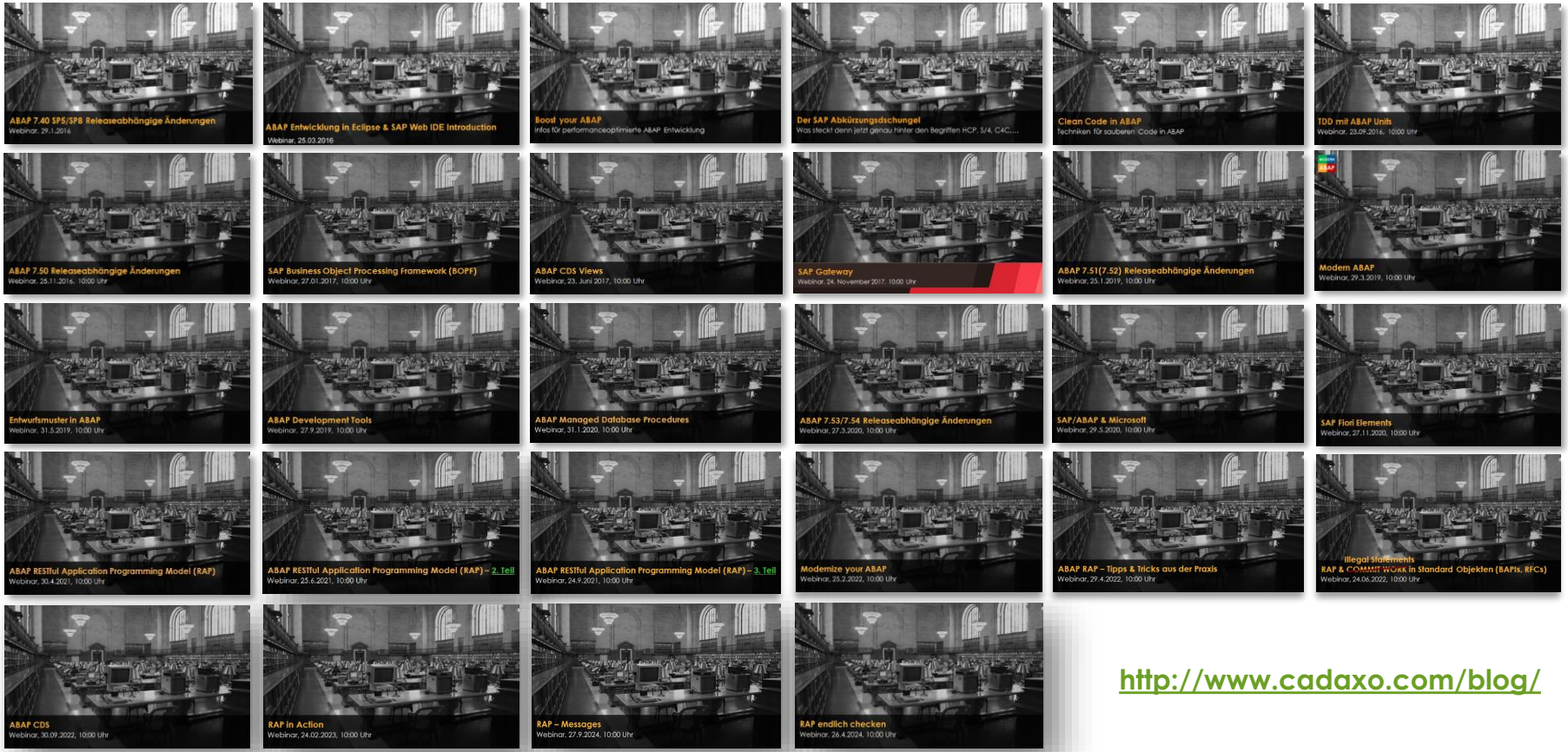
@SoSchlegel87

johann.foessleitner@cadaxo.com

@foessleitnerj

dominik.bigl@cadaxo.com

@DomiBiglSAP



http://www.cadaxo.com/blog/

Unsere vergangenen Webinare

- **Unterdrücken der Meldungen**
  - Meldungen können mit **Pseudo Kommentaren** unterdrückt werden
    - `"#EC DEFAULT_KEY` , `"#EC CI_NESTING`, …
  - Wo findet man mögliche Pseudo Kommentare
    - Prüfergebnissen
    - Dokumentation
    - Check-klasse - Constructor

- **Prefer Pragmas to Pseudo Comments**
  - Wenn möglich, sollen Pragmas anstatt Pseudo Kommentare verwendet werden
  - Syntaxprüfung und erweitere Syntaxprüfung
  - Report ABAP_SLIN_PRAGMAS

```
DATA: a TYPE string,    "#EC NEEDED
      b TYPE string.

DATA: c TYPE string.    "#EC UNUSED
```

```
DATA: a TYPE string ##NEEDED,
      b TYPE string.

DATA: c TYPE string ##UNUSED.
```