



MCT Debugger

Analyse

Erstellt durch:

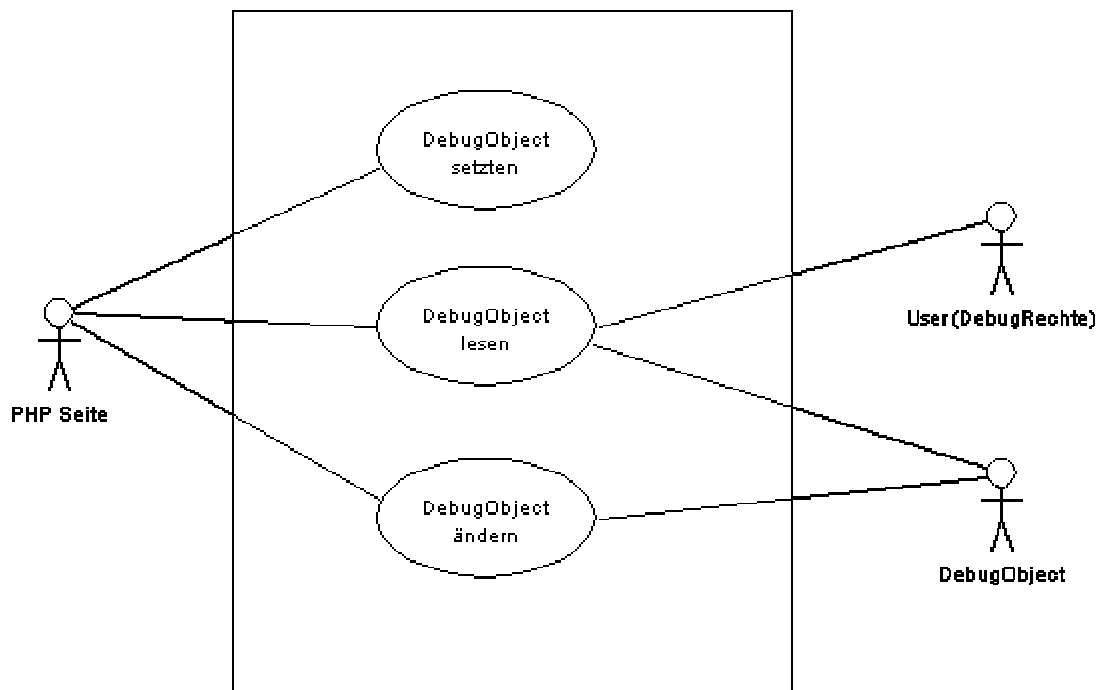
- Markus Bättig, Schulstrasse 13, 6037 Root
- Jürg Zraggen, Ottigenbühlring 11, 6030 Ebikon

Problematik

Die Entwicklung und Fehlersuche von MCT nimmt einen grossen Zeitaufwand ein. PHP bietet zwar komplette Entwicklungsumgebungen mit eingebautem Debugger, doch diese ist aus Kostengründen nicht erschwinglich. Auch gibt es ein paar Free- und Shareware Tools, die so was ähnliches anbieten. Doch unseren Ansprüchen genügen sie nicht. Zudem sind die meisten sehr fehleranfällig und haben noch grosse Bugs."

Lösungsansatz

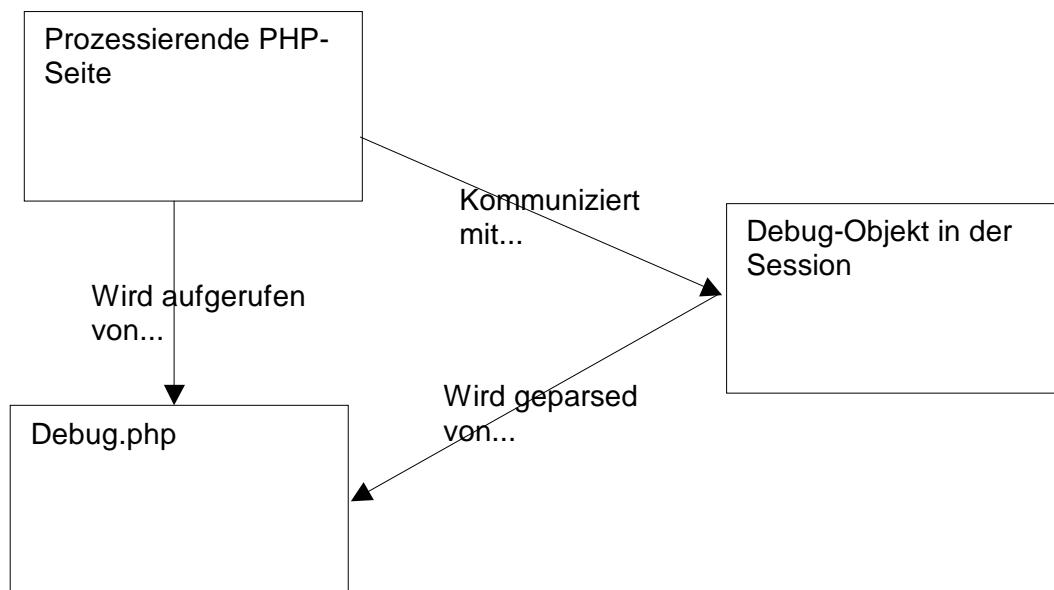
Wir werden einen in die Webseite integrierten Debugger erstellen. Die Aktivierung geschieht über den Userlevel. Ist ein Benutzer als Debug-User erfasst, hat er die gleichen Rechte, wie der Administrator. Zudem wird die Webseite in einen Debug-Modus gewechselt, die es dem Benutzer erlauben, sämtliche zusätzliche Seiteninformationen zu bekommen. Das Konzept ist somit ähnlich wie bei Webseiten von ASP.net und ColdFusion. Man soll Stackinformationen über die Seite erhalten. Zudem was für Queries mit welchen Resultaten ausgeführt wurden. Sämtliche aktuellen Session- und Request-Variablen sollen ausgelesen werden können. Auch allgemeine wichtige Informationen über die aktuellen geladenen Module, Konstanten, Funktionen und Klassen sollen eingesehen werden können. Um den Overhead im Griff zu behalten, kann im MCT.ini definiert werden, welche Teile des Debuggers gebraucht und welche abgeschaltet werden sollen.



Implementation

Allgemein

Ohne ein zusätzliches Programm, welches die Zend-Engine überwacht, können keine Informationen der aktuell parsenden Seite gewonnen werden. Somit muss die Seite unserem Debugger selber sagen, was sie gerade am machen ist.



Verhalten des Debug-Objektes

Es gibt 3 wichtige Punkte im Leben des Debug-Objektes.

Instanziierung

Die PHP Seite, über welche Debug-Informationen gewonnen werden soll, instanziert das Debug-Objekt. Mit der Instanziierung wird der Konstruktor des Debug-Objektes aufgerufen. Dieser bewirkt, dass ein altes vorheriges Debug-Objekt in der Session zerstört wird. Zudem holt er sich die nötigen Informationen aus dem MCT.ini, um zu wissen, welche Informationen gewünscht sind und welche er ignorieren resp. übergehen soll.

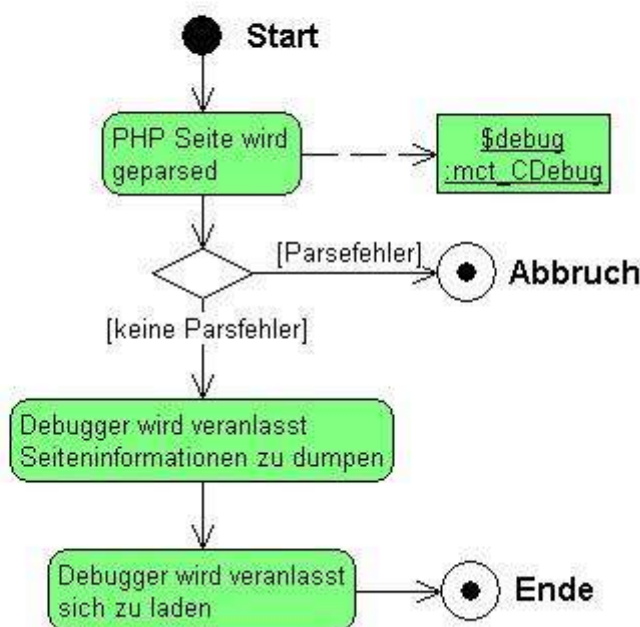
Dumping

Das Dumping wird erst am Ende der PHP-Seite aufgerufen. Dieser Aufruf weist das Objekt an, nun sämtliche relevanten Variablen des Server, Requests usw. ins Objekt zu schreiben. Zudem setzt sich das Debug-Objekt am Methoden-Ende selber in die Session, um die gewonnenen Informationen über die Lebenszeit der PHP-Seite behalten zu können.

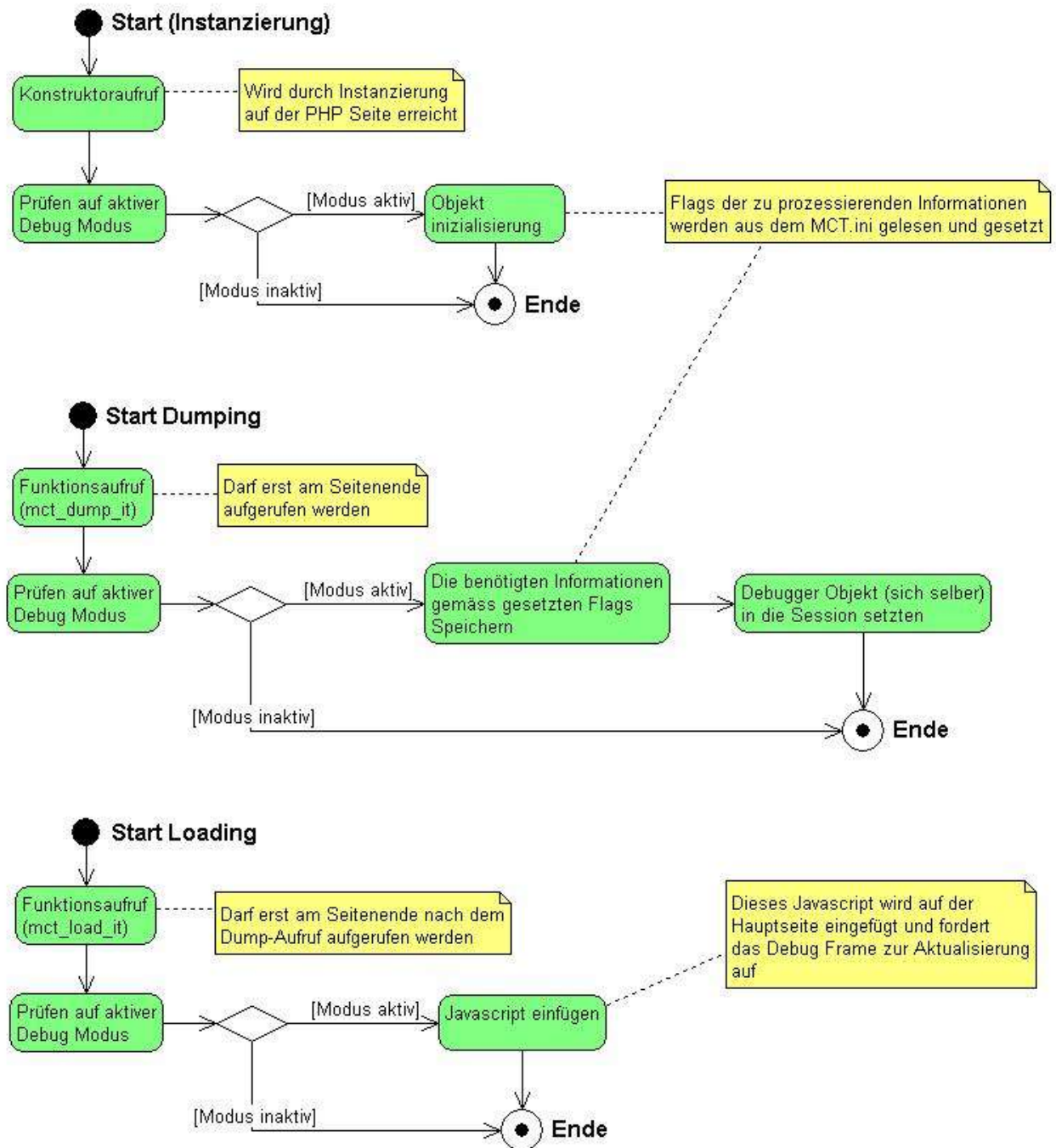
Loading

Diese Methode wird nach dem Dumpingaufruf als letzte Funktion der PHP-Seite aufgerufen. Sie weist das Debug-Objekt an, sich nun zu laden. Das Debug-Objekt setzt hierfür ein Javascript ab, welches das Debug-Frame der Webapplikation auffordert einen Reload der Seite durchzuführen. Diese Seite im Debug-Frame wiederum wird das Debug-Objekt aus der aktuellen PHP Session parsen.

Aktivitätsdiagramm Debugger benutzen



Aktivitätsdiagramm Verhalten Debugger Objekt



Inhaltsverzeichnis

Analyse	1
Problematik	2
Lösungsansatz	2
Implementation.....	3
Allgemein.....	3
Verhalten des Debug-Objektes	4
Instanzierung.....	4
Dumping.....	4
Loading	4
Inhaltsverzeichnis.....	6