Reflection API

Stefan Marr

I. Extended Reflection API

(Einführung auf den Rest des Papers abstimmen)

Die in PHP5 integrierte Reflection API bietet die grundlegenden Möglichkeiten, um zur Laufzeit Informationen zur Struktur der verfügbaren programmiersprachlichen Konstrukte, insbesondere der verfügbaren Klassen, Erweiterungen und Funktionen, zu erhalten. Darüber hinaus bietet sie ebenso die Möglichkeit auf Exemplare dieser Konstrukte zuzugreifen, um den aktuellen Zustand zur Laufzeit zu ermitteln oder zu modifizieren.

In Abb. 1 ist einmal die grobe Struktur dargestellt, welche einem Metamodel für die Definition einer Klasse im Sinne der objektorientierten Programmierung entspricht. Da PHP5 selbst eine schwach typisierte und teils sogar dynamische Sprache ist, biete die API nur sehr wenige Informationen zur Typisierung der einzelnen Sprachkonstrukte an. Um jedoch nicht nur zur Laufzeit Informationen über die Typisierung erhalten zu können, wurde es erforderlich, die vorhandene API zu erweitern. Ziel dieser Erweiterung war es, zusätzliche Informationen in der Kommentierung der Sprachkonstrukte nutzen zu können. Dabei wurde auf PHPDoc, eine in der PHP-Welt verbreitete Variation des JavaDoc-Standards, gesetzt. Ursprünglich waren diese Informationen dazu gedacht, dem Entwickler unter die Arme zu greifen, in dem der Quellcode durch zusätzliche Informationen angereichert wird.

Da PHPDoc in der PHP-Welt überwiegend als Standard akzeptiert ist und in viele bestehenden Projekten davon gebrauch machen wird, bietet es eine gute Basis um die nötigen Informationen aus den Quelltext-Kommentaren zu

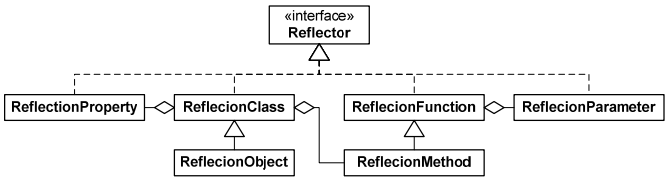

Abb. 1. UML-Darstellung der PHP5 Reflection API

extrahieren. Weiterhin kommt die sehr formale Notation der PHPDoc-Tags dem Vorhaben zugute.

Die erweiterte Reflection API nutzt zur Gewinnung der Typinformationen einen einfachen PHPDoc-Parser, welcher

Manuscript received October 9, 2001. (Write the date on which you submitted your paper for review.) This work was supported in part by the U.S..go.jp).

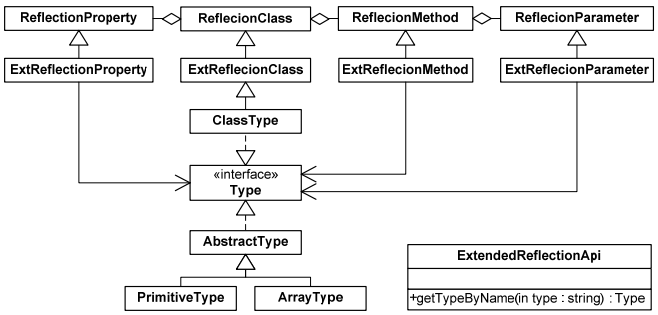

Abb. 2. UML-Darstellung der erweiterten Reflection API

die Quellcode-Kommentare untersucht und interpretiert. Um diese Informationen geeignet nutzen zu können, wurde eine programmiersprachliche Repräsentation für ein einfaches Typsystem entworfen.

In Abb. 2 ist das zentrale Interface `Type` dargestellt. Diese wird letztendlich von drei wesentlichen Klassen implementiert. `ClassType` repräsentiert alle Klassen, `ArrayType` alle Arrays und `PrimitiveType` die einfachen Datentypen wie `Integer`, `Float` und `String`. Damit ist es möglich aus einem programmiersprachlichen Konstrukt nicht nur zur eigenen Laufzeit Typeninformationen zugewinnen.

Dieses Typsystem erlaut zusätzlich vielfältige eigene Erweiterungen. Durch die Verwendung einer `TypeFactory` ist es möglich für verschiedene Anwendungsaspekte angepasste Typenklassen zu entwickeln, die weitere Zusatzinformationen zur Verfügung stellen. Aktuell implementiert ist unter anderem die Möglichkeit, direkt aus den Typen ihre Definition in XML-Schema zu erhalten, um Exemplare so ohne Umwege nach XML serialisieren zu können.

Ein weitere wesentliche Erweiterung ist die Implementierung eines Annotation-Mechanismus, welcher in PHP bis dahin nicht verfügbar war, sich jedoch in anderen gängigen Programmiersprachen zunehmender Beliebtheit erfreut. Für dieses Projekt stand die statische Verwendung von Zusatzinformationen im Vordergrund. So ist es möglich die programmiersprachlichen Konstrukte mit zusätzlichen Eigenschaften zu versehen, unter anderem die Kennzeichnung als Web Service. Zusätzlich denkbar wäre es, diesen Annotation-Mechanismus durch eine geeignete Laufzeitumgebung zu nutzen, um aspektorientierte Programmierung unterstützen zu können. Dies ist in diesem Projekt jedoch nicht realisiert.