PHP CLASS MANAGER

Оглавление

[Цель работы 3](#_Toc478152673)

[Обзор предметной области 4](#_Toc478152674)

[Описание применимости 7](#_Toc478152675)

[описание результатов проекта 8](#_Toc478152676)

[ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА 9](#_Toc478152677)

# Цель работы

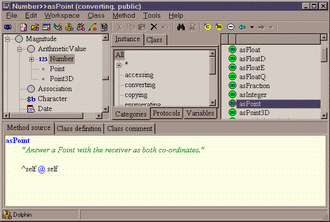
Создать удобную среду разработки ООП систем с возможностью генерации исходного кода на его основе.

За основу взяты идеи среды Smalltalk, в которой так же происходит создание приложений.

# Обзор предметной области

Рассматриваемая мною предметная область относиться к разработке веб-приложений с использованием языка PHP.

**Основная идея и цель проекта** – это создать среду разработки по аналогии среды Smalltalk, для создания, изменения и проектирования ООП системы будущего веб-приложения.



Проект не несёт цели создания полного аналога/аналогов для Web`а, а берёт идеи, в соответствии с которой манипулирование с классами ведётся с помощью графического интерфейса.

Аналогами являются:

* Dolphin Smalltalk
* VisualWorks
* Visual Smalltalk
* И другие реализации Smalltalk

Язык Smalltalk и его интерактивная среда разработки неразрывно связанны. Сам язык используется для решения следующих задач:

|  |  |
| --- | --- |
| * Автоматизация, робототехника * Диспетчеризация, планирование * Коммуникация, связь * Медицина | * Обработка коммерческой информации * Системы управления * Тренажёры, моделирования * Обучение программированию |

Сам подход к написанию программ в языке Smalltalk кардинально отличается от подхода современных языков.

В таких языках как C++, Java, C#, Python, Ruby и т.п. мы разрабатываем программу путём композиции библиотек/модулей/пакетов в связке с особенностями парадигмы и самого языка, а в конечном итоге мы получаем программу, которая решает поставленную задачу.

В случае с языком Smalltalk, который неразрывно связан с графической средой, разработка идёт путём декомпозиции, то есть мы из "коробки" имеем библиотеки/модули/пакеты и затем в процессе разработки мы убираем лишнее. Так же мы занимаемся реализацией своих решений и создания своих классов, объекто и т.д.

Как правило такая среда разработки предоставляет разработчику следующие инструменты:

* **Workspace** – рабочее пространство, в котором можно писать исходный код и выполнять его
* **Inspector** – инструмент для просмотра содержимого инспектируемого объекта
* **Class** **Browser** – основной инструмент, в котором происходит создание/удаление и изменения классов
* **Debugger** – отладчик
* **Pool Dictionaries** – инструмент для работы со словарями. Своего рода константы

Помимо этого, вся система является единым целым и её можно изменять "на лету". Так же, среда разработки Smalltalk является самой продуктивной с точки зрения программирования.

В качестве основного языка для реализации проекта, был выбран PHP, так как он имеет общие черты с языком Smalltalk:

* Интерпретируемость
* Динамическая типизация
* Объекто-ориентированная парадигма
* Имеет виртуальную машину (Zend Engine)

Ниже приведены рисунки, которые изображают устройство виртуальных машин Smalltalk и PHP.

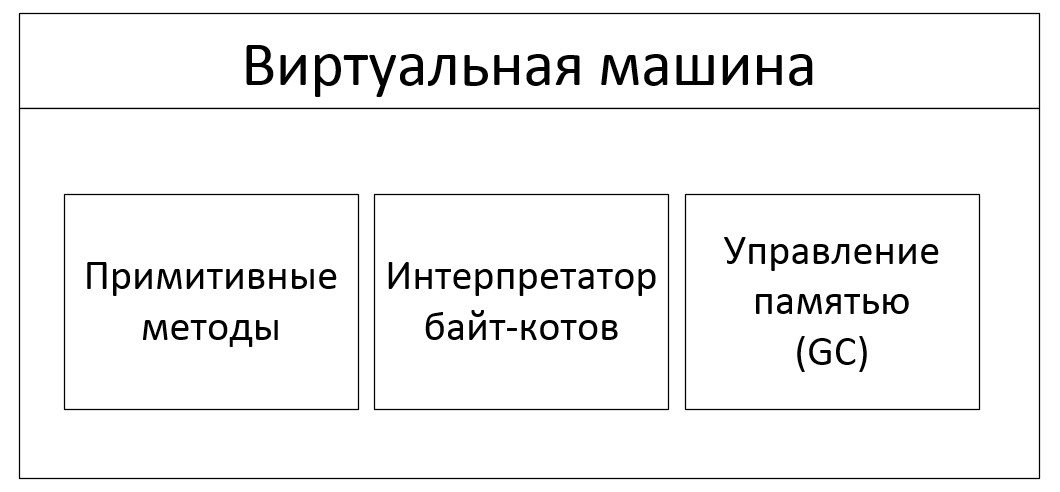


Рисунок . Устройство VM Smalltalk

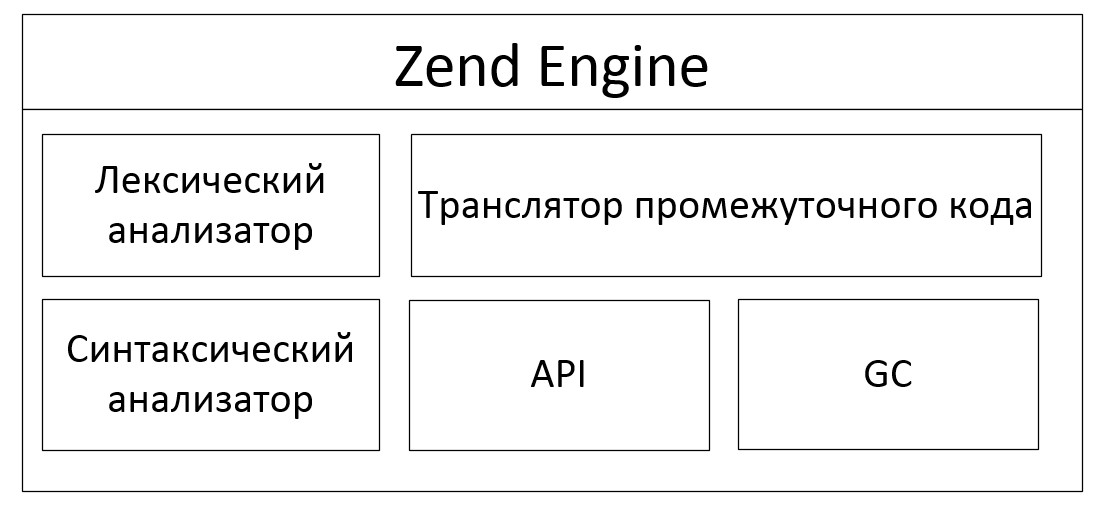


Рисунок . Устройство VM Zend Engine

Так же есть и недостатки, касательно PHP:

* Язык не является чисто объекто-ориентированным
* Язык не предназначен для реализации огромных комплексных систем и приложений

# Описание применимости

Разрабатываемая среда должна решать следующие задачи:

* Управления ООП системой (образом)
  + Создание классов
  + Изменения реализации классов
* Оперативный доступ к частям системы
* Тестирование отдельных классов
* Возможность корректировки генерируемого исходного кода
* Экспорт/импорт образов в виде упакованных объектов

Лично я считаю, что данная среда позволит больше сосредоточиться на следующих этапах разработки:

1. Проектирование
2. Быстрое тестирование отдельных классов
3. Исправлении ошибок в классах

Во время работы со средой, все выполняемые действия сохраняются во временном образе. После создания основного образа уже можно будет по нему сгенерировать код или сразу испытать работу отдельных классов прямо в среде. Таким образом мы не работает напрямую с файлами, а работаем с целым образом, что позволяет нам легко получить доступ к реализации кого-либо класса и при необходимости внести корректировки в код.

Таким образом, с помощью данной среды можно будет создавать ооп систему, которая, например, описывает BL веб-приложения или фреймворка и т.д.

# описание результатов проекта

В результате было разработано веб-приложение, которое в конечном итоге не изменило подход к написанию веб-приложений, но лично я считаю, что оно коснулось нескольких важных вещей:

1. Цикл разработки
2. Больший акцент на тестировании и проектировании
3. Единство разрабатываемой системы

Цикл разработки теперь выглядит выглядит следующим образом:



В любой момент можно вернуться на любую стадию и внеси изменения. То есть в начале мы анализируем предметную область, затем разрабатываем UML диаграммы нашего веб-приложения, после чего с помощью среды мы переносим всё это в среду и разрабатываем функционал. Затем мы производим тестирование нашей системы по отдельности или в целом, наблюдая, какие коррективы необходимо внести на предыдущих этапах и выполняем их. Когда мы убедились, что всё работоспособно, то мы собираем образ и генерируем код. В конечном итоге мы должны получить работоспособный набор классов, который можно будет уже использовать в боевых условиях.

Так как вся система находиться в едином образе, то его можно будет экспортировать или импортировать, чтобы продолжить работу там, где есть PCM.

Проблемы и вопросы, которые возникают:

1. Размер образа. На сколько он может быть большим и как это повлияет на работу PCM с ним
2. Сложность архитектуры разрабатываемого веб-приложения. Сможет ли PCM оставаться таким же удобным в использовании
3. Паттерны проектирования. Сложно сказать, возможно ли будет их реализовать на данный момент, в текущей версии PCM

На данный момент PCM может:

Сравнение с Dolphin Smalltak:

|  |  |  |
| --- | --- | --- |
|  | Dolphin Smalltalk | PCM |
| Class Browser | + | + |
| Inspector | + | + |
| Workspace | + | + |
| Debugger | + | - |
| Pool Dictionaries | + | - |

Перспективы дальнейшего развития:

* Поддержка XML для описания иерархии классов
* Интеграция стандартных классов в PCM

# дополнительные материалы

В этом разделе я оставлю ссылки на некоторые статьи о языке Smalltalk и PHP.

* Smalltalk
  + [История языка](http://www.smalltalk.ru/articles/smalltalk.html)
  + [Почему Smalltalk?](http://www.smalltalk.ru/articles/why.html)
* PHP
  + [Устройство PHP и Zend Engine](https://books.google.ru/books?id=red4i4BtuowC&pg=PA476&lpg=PA476&dq=%D0%B2%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F+%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0+zend&source=bl&ots=yVxsX9pPXy&sig=-49Wu0CeiBM9IqNA5rfzOOzYXWc&hl=ru&sa=X&ved=0ahUKEwi5i6XUjvTSAhWqF5oKHdhoA54Q6AEIIjAB#v=onepage&q=%D0%B2%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0%20zend&f=false)

# ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА

1. История языка Smalltalk (статья) URL: <https://goo.gl/uc7v3q>
2. Неизвестный Smalltalk (статья) URL: <https://goo.gl/X8QaX2>
3. Smalltalk: Концепция объекто-ориентированного программирования (статья) URL: <https://goo.gl/L8A38Q>
4. «Профессиональное программирование на PHP» (книга) URL: <https://goo.gl/7mCwsM>