УДК 004.4’22

А. И. Факторович – магистрант кафедры моделирования вычислительных и электронных систем
В. С. Павлов (канд. техн. наук, доц.) – научный руководитель

ПРОГРАММНЫЕ СРЕДСТВА АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ УРОВНЯ ПРЕДСТАВЛЕНИЯ ИС

Автоматизация проектирования подразумевает получение качественной информационной системы при минимизации затрат на разработку. Для этих целей используется концепция разделения проектирования системы на 3 составляющих: модель, представление, контроллер. Данный подход позволяет понизить вероятность возникновения коллизий между разработчиками системы и сделать разрабатываемую систему более понятной и структурированной [3].

Уровень представления имеет преимущество в связи с расширением круга неквалифицированных пользователей. Меняется поколение вычислительной техники и парадигма ее использования. Компания apple, которая несколько лет назад терпела убытки, сейчас по капитализации является среди конкурентов, фирмой номер один. Из-за того, что в свои сложные времена они сделали ставку именно на технику 5 поколения, в которой уже не требуются специальные навыки у пользователя, так как большое внимание уделено юзабилити, а работа с устройством осуществляется через графический интерфейс.

Уровень представления может проектироваться с нуля или с помощью готовых решений, называемых «умным заимствованием».

Для разработки уровня представления предлагается шаблонно-ориентированное проектирование. Шаблонизатор (компилирующий обработчик шаблонов) – это программное обеспечение, позволяющее использовать html-шаблоны для генерации конечных html-страниц. Основная цель использования шаблонизаторов – отделение представления данных от исполняемого кода. Часто это необходимо для обеспечения возможности параллельной работы программиста и дизайнера-верстальщика. Использование шаблонизаторов часто улучшает читаемость кода и внесение изменений во внешний вид, когда проект целиком выполняет один человек [1].

Актуальной задачей является сравнение, выявление специфики и оценки качества шаблонизаторов.

В настоящее время существует множество средств управления шаблонами, которые позволяют использовать подход MVC. Все они имеют как достоинства, так и недостатки и могут использоваться лишь с определёнными Framework и языками программирования.

Все шаблонизаторы можно условно разделить на 4 типа:

- классические (Нативный, ZendView, Smarty, Quicky);
- атрибутивные (ZPT, PHPTAL, Prado);
- блочные (XTemplate);
- XSLT (MSXML) [2].

В исследовании рассматриваются PHP шаблонизаторы Smarty и Twig. Smarty и Twig являются классическими компилирующими обработчиками шаблонов для PHP. Они позволяют добиться отделения прикладной логики и данных от представления [4].

Для того, чтобы оценить PHP шаблонизаторы Smarty и Twig по скорости компилирования, выполнения и общей эффективности производятся три операции сравнения: «получение значений переменных», «обход массивов и вывод значений полей», «наследование».

Перед проведением теста важно отметить, что используются достаточно сложные шаблоны, чтобы разница во времени обработки была существенна. Оценка времени производится с помощью стандартной функции получения времени PHP microtime(); которая возвращает текущую метку времени UNIX с микросекундами. Оба шаблонизатора настроены похожим образом: выключено автоматическое экранирование, отключена автоматическая перекомпиляция шаблонов при изменениях. Объектом исследования будут являться компилирующие обработчики шаблонов Smarty версии 3.1.8 и Twig вер-

сии 1.2.0. Для исследования скорости компилирования используются веб-сервер Apache2 в стандартной конфигурации и PHP 5.3 без использования дополнительных акселераторов.

Получение значений переменных, одна из наиболее активно используемых операций. В сложных шаблонах она может использоваться по нескольку сотен раз. Для оценки производительности этой операции написан шаблон, в котором выводится значение 10000 переменных.

Таблица 1.

Результат операции получения значений переменных

Обработчик	Компиляция, сек.	Выполнение, сек.
Smarty 3.1.8	16.320	0.058
Twig 1.2.0	9.757	0.083

В таблице 1 приведены средние значения из 5 последовательных тестов. Десять тысяч синтаксических конструкций долго компилировалось обработчиками. Разница между Smarty и Twig чуть более 7.5 секунд. Однако, компиляция выполняется всего один раз, а в дальнейшем работает уже скомпилированная версия шаблона, поэтому время работы последнего намного важнее. Обработчик Smarty выполняет эту операцию примерно на 30% быстрее, чем аналог Twig.

Обход массивов и вывод значений полей. Практически в каждом шаблоне используется цикл foreach. Конструкция foreach предоставляет простой способ перебора массивов. Для тестирования написан шаблон, который выводит по 10 полей из 1000 элементов массива.

Таблица 2.

Результат операции обход массивов и вывод значений полей

Обработчик	Компиляция, сек.	Выполнение, сек.
Smarty 3.1.8	0.065	0.009
Twig 1.2.0	0.131	0.082

Из таблицы 2 видно, что скомпилированный шаблон Smarty выполняется почти в 10 раз быстрее, чем Twig. Более того, суммарное время компиляции и выполнения шаблона в Smarty, быстрее, выполнения готового скомпилированного шаблона в Twig. Из этого можно сделать вывод, что компилятор шаблонов Smarty инициализируется быстрее Twig, но судя по предыдущему тесту работает медленнее, что на маленьких шаблонах практически незаметно.

Наследование шаблонов – это удобный и применяемый повсеместно механизм, позволяющий предотвращать дублирование участков кода. В следующем тесте проведён анализ, какие временные расходы появляются при использовании наследования в Smarty и Twig. Для этого создан один родительский шаблон с 500 блоками, и ещё 500 практически пустых шаблонов, каждый из которых будет наследоваться друг от друга, заполняя статическими данными один из 500 блоков родительского шаблона. Шаблонизатору выводится на компиляцию последний в цепочке.

Таблица 3.

Результат операции наследование шаблонов

Обработчик	Компиляция, сек.	Выполнение, сек.
Smarty 3.1.8	1.329	0.002
Twig 1.2.0	2.641	0.121

Из таблицы 3 следует, что время выполнения операции наследования обработчиком Smarty в 60 раз быстрее, чем в Twig. Исходя из скомпилированного кода, становится понятно, что это не предел возможностей быстродействия. Smarty объединил всю цепочку из наследуемых шаблонов в один большой файл, таким образом в результате при использовании наследования вообще нет потерь производительности. Twig же создал для каждого шаблона по классу и файлу, и при каждой генерации страницы загружает всё последовательно.

Так же важно оценить преимущества использования шаблонизаторов относительно разработки с помощью стандартных средств PHP. В качестве примера приведены два основных параметра присущих шаблонизаторам.

1. Краткость и лаконичность синтаксиса. Вывод переменной стандартным синтаксисом PHP занимает порядка 20 символов: «<? php echo $var; ?>», а при более сложных конструкциях, например, экранирование выходных данных, количество символов на порядок больше: «<? php echo htmlspecialchars($var, ENT_QUOTES,'UTF-8') ?>», в отличие от синтакиса Smarty «{$var}», а при экранировании синтаксис Smarty выглядит так «{$var|escape}»

2. Возможность автоматического экранирования данных вывода стандартными средствами. Для предотвращения XSS-атак на веб-ресурсы повсеместно используется экранирование выходных параметров для вывода данных в URL, в значения переменных JavaScript, для подстановки данных в html-форму. В связи с этим, последующее тестирование и производство веб-ресурсов гораздо более безопасно с включённым автоматическим экранированием данных, особенно в случае любительской разработки в среде web.

Единственным сильным контраргументом использования шаблонизаторов является то, что на данный момент веб-разработчик должен хорошо разбираться в Html и CSS, JavaScript, SQL и определённом серверном языке помимо знания синтаксиса шаблонизатора.

Таким образом, использование шаблонизаторов, основанных на MVC-концепции, существенно ускоряет скорость производства веб-ресурсов, архитектурно позволяя вести одновременно разработку логики и дизайна. Помимо этого, разрабатываемый с использованием шаблонизатора web-ресурс менее уязвим к различным атакам, так как код неоднократно протестирован и присутствует «нативная» возможность автоматического экранирования данных, что отсутствует в PHP.

Компилирование шаблонов в обработчике Smarty выполняется быстрее, чем в Twig. Хотя компиляция больших шаблонов занимает больше времени, но в результате получается лучшая производительность при более широких возможностях обработчика шаблонов, в том числе создания дополнительных плагинов.

В связи с этим можно сделать вывод, что полезность применения MVC-концепции и компилирующих обработчиков шаблонов высока, особенно в среде непрофессиональных разработчиков, а так же в организациях, ведущих регулярную разработку веб-ресурсов.

Библиографический список

1. http://ru.wikipedia.org/wiki/%D8%E0%E1%EB%EE%ED%E8%E7%E0%F2%EE%F0
2. http://www.alabor22.ru/klassifikaciya-shablonizatorov
3. *Дейв Крейн, Эрик Паскарелло* Ajax in action Вильямс 2007.640 стр.
4. http://www.smarty.net/docsv2/ru/