Глава 2

PHP и объекты

Объекты не всегда были основной частью PHP-проекта. Более того, идея реализовать объекты пришла в голову разработчикам PHP "потом".

Но впоследствии эта идея доказала свою жизнеспособность. В данной главе я познакомлю вас с объектами и опишу процесс разработки объектно-ориентированных приложений на PHP.

Мы рассмотрим следующее.

- *PHP/FI 2.0:* это PHP, но не такой, каким мы его знаем.
- *PHP 3:* первое появление объектов.
- *PHP 4:* объектно-ориентированное программирование развивается.
- *PHP 5:* объекты — в самом сердце языка.
- *PHP 6:* отблеск будущего.

Неожиданный успех PHP-объектов

При таком количестве объектно-ориентированных библиотек и приложений на PHP, не говоря уже об улучшенной поддержке объектов в PHP 5, развитие объектов в PHP может показаться просто кульминацией естественного и неизбежного процесса. На самом деле все это очень далеко от истины.

Вначале — PHP/FI

Своим происхождением PHP, каким мы его знаем сегодня, обязан двум инструментам, которые разработал Расмус Лердорф с помощью Perl. Аббревиатура PHP обозначала Personal Homepage Tools (инструменты для персональной домашней страницы), а FI — Form Interpreter (интерпретатор форм). Вместе они составляли набор макросов для отправки SQL-запросов в базу данных, обработки форм и управления процессом обмена данными.

Затем эти инструменты были переписаны на языке C и объединены под именем PHP/FI 2.0. На этом этапе синтаксис языка PHP отличался от того, который мы знаем сегодня, но не *слишком* значительно. Была поддержка переменных, ассоциативных массивов и функций. Но объектов не было еще и в помине.

Изменение синтаксиса: PHP 3

На самом деле, даже когда PHP был еще на этапе планирования, объекты не стояли на повестке дня. Как и сегодня, главными архитекторами PHP 3 были Зив Сураски и Энди Гутманс. PHP 3 представлял собой полностью переписанный PHP/FI 2.0, но объекты не считались необходимой частью нового синтаксиса.

По словам Зива Сураски, поддержка классов была добавлена почти в самом конце (а точнее, 27 августа 1997 года). Классы и объекты на самом деле представляли собой просто другой способ определения и доступа к ассоциативным массивам.

Конечно, добавление методов и наследования существенно расширило возможности классов по сравнению с хвалеными ассоциативными массивами. Однако все еще существовали жесткие ограничения в отношении операций с классами. В частности, нельзя было получить доступ к переопределенным методам родительского класса (если вы еще не знаете, что это такое, не волнуйтесь; я объясню позже). Еще одним недостатком, о котором мы поговорим в следующем разделе, был не самый оптимальный способ передачи объектов в PHP-сценариях.

В то время объекты считались второстепенным вопросом. Об этом говорило и то, что им придавалось мало значения в официальной документации. В руководстве было дано одно предложение о них и один пример кода. В примере не иллюстрировалось использование наследования или свойств.

PHP 4 и тихая революция

Хотя PHP 4 был еще одним революционным шагом в развитии языка, большинство основных изменений произошло незаметно для пользователя на нижнем уровне. Для расширения возможностей языка PHP был заново переписан движок Zend, название которого происходит от имен **Ze**ev and A**nd**i. Zend — один из основных компонентов, положенных в основу работы PHP. Любая вызываемая PHP-функция на самом деле является частью яруса высокоуровневых расширений. Эти функции выполняют всю возложенную на них работу, например взаимодействие с API системы управления базами данных или манипуляции со строками. А на нижнем уровне движок Zend управляет памятью, передает управление другим компонентам и преобразует знакомый вам PHP-синтаксис, с которым вы работаете каждый день, в исполняемый байт-код. Именно движок Zend мы должны "благодарить" за поддержку ключевых возможностей языка, таких как классы.

С точки зрения рассматриваемых нами *объектов*, большим преимуществом PHP 4 стала возможность переопределения родительских методов и получения к ним доступа из дочерних классов.

Но остался и большой недостаток. Присвоение объекта переменной, передача его функции или возвращение его из метода приводило к появлению копий этого объекта. Поэтому присвоение, подобное следующему:

```
$my_obj = new User('bob');
$other = $my_obj;
```

приводило к появлению двух копий объекта `User`, а не двух ссылок на один и тот же объект `User`. В большинстве объектно-ориентированных языков программирования используется более естественное присвоение по ссылке, а не по значению, как здесь. Это означает, что вы передаете и присваиваете указатели на объекты, а не копируете сами объекты. Принятая по умолчанию стратегия передачи по значению приводила к появлению в сценариях множества скрытых ошибок, когда программисты модифицировали объект в одной части сценария и ожидали, что эти изменения

отразятся на всех его копиях. В этой книге вы увидите много примеров, в которых будет использоваться несколько ссылок на один и тот же объект.

К счастью, в сценарии можно было принудительно осуществить передачу объекта по ссылке, но для этого нужно было использовать довольно "неуклюжую" конструкцию.

Выполним присвоение по ссылке следующим образом.

```
$other =& $my_obj;
// $other и $my_obj указывают на один и тот же объект
```

Выполним передачу по ссылке следующим образом.

```
function setSchool( & $school ) {
   // $school теперь ссылается на сам объект, а не на его копию
}
```

И осуществим возврат по ссылке.

```
function & getSchool( ) {
   // Возврат ссылки на объект, а не его копии
   return $this->school;
}
```

Хотя эта конструкция работала нормально, программисты легко забывали добавить символ амперсанда, и вероятность появления ошибок в объектно-ориентированном коде была очень высока. Причем такие ошибки очень трудно было обнаружить, потому что они редко вызывали сообщения об ошибках; только программа работала не совсем так, как нужно.

Описание синтаксиса в целом и объектов в частности было расширено в руководстве по PHP, и объектно-ориентированное программирование стало превращаться в основное направление, главную тенденцию. Объекты в PHP не были приняты сообществом программистов без споров, и сообщения типа "Зачем мне нужны эти объекты?" часто раздували флеймы на форумах и в списках рассылки. На сайте Zend размещались статьи, которые поощряли объектно-ориентированное программирование, наряду со статьями, в которых звучали предостережения.

Но несмотря на проблемы передачи по ссылке и споры, многие программисты просто приняли новые возможности и "усеяли" свои коды символами амперсандов. Популярность объектно-ориентированного PHP стала расти. Вот что Зив Сураски написал в статье для сайта DevX.com (`http://www.devx.com/webdev/Article/10007/0/page/1`).

> *Одним из величайших неожиданных поворотов в истории PHP было то, что, несмотря на очень ограниченную функциональность, на множество проблем и ограничений, объектно-ориентированное программирование в PHP процветало и стало самой популярной парадигмой для растущего числа стандартных PHP-приложений. Эта тенденция, которая по большей части была неожиданной, застала PHP не в самой выигрышной ситуации. Стало очевидным, что объекты ведут себя не так, как объекты в других объектно-ориентированных языках, а как [ассоциативные] массивы.*

Как отмечалось в предыдущей главе, интерес к объектно-ориентированному проектированию стал очевиден из растущего числа публикаций статей на сайтах и в форумах в Интернете. В официальном хранилище программного обеспечения PHP, PEAR, тоже была принята концепция объектно-ориентированного программирования. Некоторые из лучших примеров использования шаблонов объектно-ориенти-

рованного проектирования можно найти в пакетах PEAR, созданных для увеличения функциональности PHP.

Оглядываясь в прошлое, можно подумать, что введение в PHP поддержки средств объектно-ориентированного программирования стало результатом вынужденной капитуляции перед лицом неизбежности. Но важно помнить, что хотя концепция объектно-ориентированного программирования существует с 60-х годов прошлого века, широкое распространение она получила только в средине 90-х годов. Язык Java, этот "великий популяризатор" методологии объектно-ориентированного программирования, был выпущен только в 1995 году. Язык C++, расширение процедурного языка C, появился в 1979 году. И после длительного этапа эволюции он совершил большой скачок только в 90-х годах. В 1994 году вышел язык Perl 5. Это была еще одна революция для бывшего процедурного языка, что дало возможность пользователям перейти к концепции объектов (хотя некоторые утверждают, что поддержка объектно-ориентированных возможностей в Perl напоминает добавления, сделанные задним числом). Для небольшого процедурного языка поддержка объектов в PHP была разработана удивительно быстро, что продемонстрировало оперативный отклик на требования пользователей.

Изменения приняты: PHP 5

В PHP 5 уже была явно выражена поддержка объектов и объектно-ориентированного программирования. Это не означает, что теперь объекты — единственный способ работы с PHP (кстати, настоящая книга тоже этого не утверждает). Но объекты теперь считаются мощным и важным средством для разработки корпоративных приложений, и в PHP предусмотрена их полная и всесторонняя поддержка.

Объекты из дополнительной возможности превратились в "двигатель" языка. И, вероятно, самое важное изменение — это принятый по умолчанию тип передачи объектов по ссылке, а не по значению. Но это только начало. По всей книге, и особенно в этой части, будет описано гораздо больше изменений, которые расширяют и усиливают поддержку объектов в PHP, включая уточнения типов аргументов (hints), закрытые (private) и защищенные (protected) методы и свойства, ключевое слово `static`, пространства имен, исключения и многое другое.

PHP остается языком, который поддерживает объектно-ориентированную разработку, а не языком для объектно-ориентированного программирования. Но поддержка в нем объектов развита достаточно для того, чтобы стало оправданным появление книг, подобных этой, посвященных проектированию исключительно с объектно-ориентированной точки зрения.

Взгляд в будущее

Когда я пишу эти строки, PHP 6 находится еще где-то на пути к нам, но тем не менее он активно разрабатывается. Уже известно, что он будет построен на совершенно новом поколении движка Zend (ZE3) и будет обеспечивать встроенную поддержку и обработку строк, заданных в формате Unicode, что позволит улучшить поддержку интернациональных возможностей в приложениях. Это означает, что вы сможете использовать все функции PHP для операций со строками, не беспокоясь о том, могут ли они работать с текущим набором символов. В прошлом разработчикам приходилось использовать многобайтовые эквиваленты во многих распространенных функциях. Это была неприятная работа, и, к тому же, она приводила к появлению многочисленных ошибок. Поскольку интернациональный фактор приоб-

ретает все большее значение, данная возможность быстро становится совершенно необходимой в любом серьезном языке программирования.

В некотором роде будущее уже не за горами. Многие функциональные возможности, объявленные в PHP 6, уже реализованы в PHP 5 (точнее, PHP 5.3). С объектно-ориентированной точки зрения, PHP 6 не даст такого скачка функциональности, какой мы наблюдали в предыдущей версии. Но поддержка пространств имен (namespaces) уже является большим шагом вперед. Они позволяют ограничить область видимости имен классов и функций, чтобы уменьшить вероятность дублирования имен при включении библиотек и расширении системы с помощью пакетов. Благодаря пространствам имен вы также избавитесь от уродливых, но необходимых правил именования объектов, подобных следующему.

```
class megaquiz_util_Conf {
}
```

Подобные имена классов — способ предотвращения конфликтов между пакетами, но это еще больше запутывает код.

Когда я писал эти строки, в PHP 6 снова было объявлено о поддержке уточнений для возвращаемых типов данных. Это позволит определять в методе или объявлении функции тип возвращаемого объекта. Данное обязательство должно быть со временем реализовано в движке PHP. Уточнение типов возвращаемых объектов позволило бы еще больше усовершенствовать в PHP поддержку шаблонных принципов программирования, наподобие программирования на основе интерфейса, а не его реализации. Как только эта возможность будет реализована, я надеюсь, что включу ее описание в новое издание книги.

Сторонники и противники: дебаты об объектах

Объекты и объектно-ориентированное проектирование, похоже, “разжигают” страсти среди программистов. Многие прекрасные программисты годами писали отличные программы, не пользуясь объектами, и PHP продолжает быть великолепной платформой для процедурного веб-программирования.

В данной книге повсюду демонстрируется пристрастие к объектно-ориентированному программированию, поскольку это является отражением моего мировоззрения, “зараженного” любовью к объектам. Поскольку эта книга посвящена объектам и является введением в объектно-ориентированное проектирование, ничего удивительного, что главное внимание, в основном, уделяется объектно-ориентированным методам. Но в то же время в книге нигде не утверждается, что объекты — это единственно правильный путь к успеху в программировании на PHP.

Во время чтения книги стоит иметь в виду знаменитый девиз Perl: “Любую задачу можно решить несколькими способами”. Это особенно верно для небольших сценариев, когда важнее быстро получить рабочий код и запустить его, чем создать структуру, которая сможет эффективно и безболезненно вырасти в большую систему (в мире экстремального программирования наспех разрабатываемые проекты такого рода называют “пробными решениями”).

Код — это гибкая среда. Самое главное — понять, когда быстрое испытание идеи станет корнем дальнейшего развития, и вовремя остановиться, прежде чем решения по вопросам проектирования вам будет диктовать громадный объем кода. И если вы примете решение использовать для развивающегося проекта процедурно-ориентированный подход, то найдете множество книг, в которых приведены примеры процедурного проектирования для разнообразных проектов. В данной книге

предлагаются некоторые идеи проектирования с помощью объектов. Надеюсь, она станет для вас хорошей отправной точкой.

Резюме

В этой короткой главе объекты рассмотрены в контексте языка PHP. Будущее PHP во многом связано с объектно-ориентированным проектированием. В следующих нескольких главах мы рассмотрим текущее состояние поддержки объектов в PHP и обсудим некоторые вопросы проектирования.