

Содержание

Введение	26
Почему вы должны читать эту книгу?	26
Что вы узнаете из этой книги?	27
Что такое PHP?	27
Что такое MySQL?	28
Почему используются PHP и MySQL?	28
Некоторые достоинства PHP	29
Производительность	29
Масштабируемость	29
Интеграция с базами данных	30
Встроенные библиотеки	30
Стоимость	30
Легкость изучения PHP	30
Поддержка объектно-ориентированного подхода	30
Переносимость	30
Гибкость подхода к разработке	31
Исходный код	31
Доступность поддержки и документации	31
Основные возможности PHP 7	31
Некоторые достоинства MySQL	32
Производительность	32
Низкая стоимость	32
Легкость использования	32
Переносимость	32
Исходный код	32
Доступность поддержки	33
Что нового в MySQL (5.x)?	33
Как организована эта книга?	34
Заключение	34
Часть I. Использование PHP	35
Глава 1. Ускоренный курс по PHP	36
Предварительное условие: доступ к PHP	37
Пример приложения: “Автозапчасти от Вовки”	37
Создание формы заказа	37
Обработка формы	39
Встраивание PHP в HTML	39
Дескрипторы PHP	41
Операторы PHP	41
Пробельные символы	42
Комментарии	42

Добавление динамического содержимого	43
Вызов функций	44
Использование функции <code>date()</code>	44
Доступ к переменным формы	45
Переменные формы	45
Конкатенация строк	46
Переменные и литералы	47
Идентификаторы	48
Типы переменных	49
Типы данных PHP	49
Прочность типов	50
Приведение типов	50
Переменные переменных	50
Объявление и использование констант	51
Области видимости переменных	52
Использование операций	53
Арифметические операции	53
Строковые операции	54
Операции присваивания	54
Операции сравнения	56
Логические операции	58
Поразрядные операции	58
Прочие операции	59
Вычисление итоговых сумм для формы	61
Приоритет и ассоциативность	62
Использование функций для обработки переменных	63
Проверка и установка типов переменных	63
Проверка состояния переменных	65
Повторная интерпретация переменных	65
Принятие решений с помощью условных структур	66
Операторы <code>if</code>	66
Блоки кода	66
Операторы <code>else</code>	67
Операторы <code>elseif</code>	67
Операторы <code>switch</code>	68
Сравнение разных условных конструкций	70
Повторение действий с помощью итераций	70
Циклы <code>while</code>	71
Циклы <code>for</code> и <code>foreach</code>	73
Циклы <code>do...while</code>	74
Выход из управляющей структуры или сценария	74
Использование альтернативного синтаксиса управляющих структур	75
Использование <code>declare</code>	75
Что дальше	75

Глава 2. Хранение и извлечение данных	77
Сохранение данных для более позднего использования	78
Сохранение и извлечение заказов в компании “Автозапчасти от Вовки”	78
Обработка файлов	79
Открытие файла	79
Выбор режима файла	79
Использование <code>fopen()</code> для открытия файла	80
Открытие файлов через FTP или HTTP	82
Решение проблем при открытии файлов	82
Запись в файл	85
Параметры функции <code>fwrite()</code>	85
Форматы файлов	85
Закрытие файла	86
Чтение из файла	88
Открытие файла для чтения: <code>fopen()</code>	89
Выяснение, когда останавливать чтение: <code>feof()</code>	89
Построчное чтение: <code>fgets()</code> , <code>fgetss()</code> и <code>fgetcsv()</code>	90
Чтение файла целиком: <code>readfile()</code> , <code>fpasssthru()</code> , <code>file()</code> и <code>file_get_contents()</code>	91
Чтение символа: <code>fgetc()</code>	92
Чтение произвольного количества байтов: <code>fread()</code>	92
Использование других файловых функций	93
Проверка, существует ли файл: <code>file_exists()</code>	93
Определение размера файла: <code>filesize()</code>	93
Удаление файла: <code>unlink()</code>	93
Перемещение внутри файла: <code>rewind()</code> , <code>fseek()</code> и <code>ftell()</code>	93
Блокирование файлов	94
Лучший способ: базы данных	96
Проблемы, связанные с использованием плоских файлов	96
Как СУРБД решают проблемы, связанные с использованием плоских файлов	97
Дополнительные источники информации	98
Что дальше	98
Глава 3. Использование массивов	99
Что такое массив	100
Массивы с числовыми индексами	100
Инициализация массивов с числовыми индексами	101
Доступ к содержимому массива	101
Использование циклов для доступа к содержимому массива	102
Массивы с другими индексами	103
Инициализация массива	103
Доступ к элементам массива	103
Использование циклов	104
Операции для обработки массивов	105
Многомерные массивы	106

Сортировка массивов	109
Использование <code>sort()</code>	109
Использование <code>asort()</code> и <code>ksort()</code> для сортировки массивов	110
Сортировка в обратном порядке	110
Сортировка многомерных массивов	111
Использование <code>array_multisort()</code>	111
Сортировка, определяемая пользователем	112
Сортировка, определяемая пользователем, в обратном порядке	113
Переупорядочение массивов	114
Использование <code>shuffle()</code>	114
Изменение порядка следования элементов в массивах на обратный	115
Загрузка массивов из файлов	116
Выполнение других манипуляций с массивами	120
Перемещение внутри массива: <code>each()</code> , <code>current()</code> , <code>reset()</code> , <code>end()</code> , <code>next()</code> , <code>pos()</code> и <code>prev()</code>	120
Применение любой функции к каждому элементу в массиве: <code>array_walk()</code>	121
Подсчет элементов в массиве: <code>count()</code> , <code>sizeof()</code> и <code>array_count_values()</code>	122
Преобразование массивов в скалярные переменные: <code>extract()</code>	122
Дополнительные источники информации	124
Что дальше	124
Глава 4. Манипулирование строками и регулярные выражения	125
Пример приложения: интеллектуальная форма отправки электронной почты	126
Форматирование строк	128
Усечение строк: <code>chop()</code> , <code>ltrim()</code> и <code>trim()</code>	128
Форматирование строк для вывода	129
Фильтрация строк для других форм вывода	130
Объединение и разбиение строк с помощью строковых функций	136
Использование <code>explode()</code> , <code>implode()</code> и <code>join()</code>	136
Использование <code>strtok()</code>	137
Использование <code>substr()</code>	138
Сравнение строк	138
Упорядочение строк: <code>strcmp()</code> , <code>strcasecmp()</code> и <code>strnatcmp()</code>	139
Проверка длины строки с помощью <code>strlen()</code>	139
Сопоставление и замена подстрок с помощью строковых функций	140
Поиск строк внутри строк: <code>strstr()</code> , <code>strchr()</code> , <code>strrchr()</code> и <code>stristr()</code>	140
Нахождение позиции подстроки: <code>strpos()</code> и <code>strrpos()</code>	141
Замена подстрок: <code>str_replace()</code> и <code>substr_replace()</code>	142
Введение в регулярные выражения	143
Основы	143
Ограничители	144
Классы и типы символов	144
Повторение	146
Подвыражения	146
Подвыражения с подсчетом	146

10 Содержание

Привязка к началу или концу строки	146
Ветвление	147
Сопоставление с буквальными специальными символами	147
Обзор метасимволов	148
Управляющие последовательности	148
Обратные ссылки	149
Утверждения	150
Использование регулярных выражений в приложении интеллектуальной формы отправки электронной почты	150
Поиск подстрок с помощью регулярных выражений	151
Замена подстрок с помощью регулярных выражений	152
Разбиение строк с помощью регулярных выражений	153
Дополнительные источники информации	154
Что дальше	154
Глава 5. Повторное использование кода и написание функций	155
Преимущества повторного использования кода	156
Затраты	156
Надежность	156
Согласованность	156
Использование <code>require()</code> и <code>include()</code>	157
Использование <code>require()</code> для включения кода	157
Использование <code>require()</code> для шаблонов веб-сайта	159
Использование <code>auto_prepend_file</code> и <code>auto_append_file</code>	163
Использование функций в PHP	164
Вызов функций	165
Вызов неопределенной функции	166
Регистр символов и имена функций	167
Определение собственных функций	168
Базовая структура функции	168
Назначение имени функции	169
Параметры	170
Область видимости	172
Передача по ссылке и передача по значению	175
Ключевое слово <code>return</code>	176
Возвращение значений из функций	177
Реализация рекурсии	178
Реализация анонимных функций (или замыканий)	180
Дополнительные источники информации	181
Что дальше	181
Глава 6. Объектно-ориентированная разработка с помощью PHP	183
Концепции объектно-ориентированной разработки	184
Классы и объекты	184
Полиморфизм	185
Наследование	186

Создание классов, атрибутов и операций в PHP	186
Структура класса	186
Конструкторы	187
Деструкторы	187
Создание экземпляра класса	188
Использование атрибутов класса	189
Вызов операций класса	189
Управление доступом с помощью <code>private</code> и <code>public</code>	190
Написание функций доступа	191
Реализация наследования в PHP	192
Управление видимостью при наследовании с помощью <code>private</code> и <code>protected</code>	193
Переопределение	194
Предотвращение наследования и переопределения с помощью <code>final</code>	196
Множественное наследование	196
Реализация интерфейсов	197
Использование трейтов	198
Проектирование классов	200
Написание кода класса	201
Расширенная объектно-ориентированная функциональность в PHP	208
Использование констант класса	209
Реализация статических методов	209
Проверка типа объекта и подсказка типа	209
Позднее статическое связывание	210
Клонирование объектов	211
Использование абстрактных классов	211
Перегрузка методов с помощью <code>__call()</code>	212
Использование <code>__autoload()</code>	213
Реализация итераторов и итерации	213
Генераторы	215
Преобразование классов в строки	217
Использование API-интерфейса рефлексии	217
Пространства имен	218
Использование подпространств имен	220
Глобальное пространство имен	220
Импортирование и создание псевдонимов пространств имен	221
Что дальше	221
Глава 7. Обработка ошибок и исключений	223
Концепции обработки исключений	223
Класс <code>Exception</code>	225
Исключения, определяемые пользователем	226
Исключения в приложении “Автозапчасти от Вовки”	228
Исключения и другие механизмы обработки ошибок PHP	232
Дополнительные источники информации	232
Что дальше	232

Часть II. Использование MySQL	233
Глава 8. Проектирование базы данных для веб-приложения	234
Концепции реляционных баз данных	235
Таблицы	235
Столбцы	236
Строки	236
Значения	236
Ключи	236
Схемы	237
Отношения	238
Проектирование базы данных для веб-приложения	238
Думайте о реальных объектах, которые вы моделируете	239
Избегайте хранения избыточных данных	240
Используйте атомарные значения столбцов	241
Выбирайте практичные ключи	242
Обдумайте, что вы хотите запрашивать у базы данных	242
Избегайте проектных решений с многочисленными пустыми атрибутами	242
Сводка по типам таблиц	243
Архитектура баз данных для веб-приложений	244
Дополнительные источники информации	245
Что дальше	245
Глава 9. Создание базы данных для веб-приложения	247
Использование монитора MySQL	249
Вход в систему MySQL	249
Создание баз данных и пользователей	251
Настройка пользователей и привилегий	251
Введение в систему привилегий MySQL	251
Принцип наименьшего уровня привилегий	252
Настройка пользователей: команды CREATE USER и GRANT	252
Типы и уровни привилегий	254
Команда REVOKE	257
Примеры использования GRANT и REVOKE	257
Настройка пользователя для веб-приложения	258
Использование правильной базы данных	259
Создание таблиц базы данных	259
Смысл других ключевых слов	261
Типы столбцов	262
Просмотр базы данных с помощью SHOW и DESCRIBE	264
Создание индексов	265
Идентификаторы MySQL	265
Выбор типов данных для столбцов	267
Числовые типы	268
Типы даты и времени	268
Строковые типы	270
Дополнительные источники информации	272
Что дальше	272

Глава 10. Работа с базой данных MySQL	273
Что такое SQL?	274
Вставка данных в базу данных	274
Извлечение из базы данных	276
Извлечение данных по определенному критерию	278
Извлечение данных из множества таблиц	280
Группирование и агрегирование данных	286
Выбор строк для возвращения	288
Использование подзапросов	288
Обновление записей в базе данных	291
Изменение таблиц после создания	291
Удаление записей из базы данных	294
Удаление таблиц	294
Удаление целой базы данных	294
Дополнительные источники информации	295
Что дальше	295
Глава 11. Доступ к базе данных MySQL из веб-приложения с помощью PHP	297
Как работает архитектура баз данных при доступе через Интернет	298
Выполнение запросов к базе данных через Интернет	300
Проверка и фильтрация входных данных	301
Настройка подключения	302
Выбор базы данных для использования	303
Выполнение запроса к базе данных	304
Использование подготовленных операторов	305
Извлечение результатов запроса	305
Отключение от базы данных	307
Внесение новой информации в базу данных	308
Использование других интерфейсов PHP для работы с базами данных	311
Использование обобщенного интерфейса доступа к базам данных: PDO	312
Дополнительные источники информации	314
Что дальше	314
Глава 12. Расширенное администрирование в MySQL	315
Детальные исследования системы привилегий	315
Таблица user	317
Таблица db	319
Таблицы tables_priv, columns_priv, procs_priv и proxies_priv	320
Управление доступом: как MySQL использует таблицы привилегий	321
Обновление привилегий: когда изменения вступают в силу?	322
Защита базы данных MySQL	323
MySQL с точки зрения операционной системы	323
Пароли	323
Привилегии пользователей	324
Проблемы, связанные с веб-доступом	325

14 Содержание

Получение дополнительной информации о базах данных	325
Получение информации с помощью SHOW	325
Получение информации о столбцах с помощью DESCRIBE	328
Выяснение, как работают запросы, с помощью EXPLAIN	328
Оптимизация базы данных	333
Оптимизация проектного решения	333
Разрешения	333
Оптимизация таблиц	333
Использование индексов	334
Использование стандартных значений	334
Дополнительные советы	334
Резервное копирование базы данных MySQL	334
Восстановление базы данных MySQL	335
Реализация репликации	335
Настройка ведущего сервера	336
Выполнение первоначальной передачи данных	337
Создание одного или нескольких ведомых серверов	337
Дополнительные источники информации	338
Что дальше	338
Глава 13. Расширенное программирование в MySQL	339
Оператор LOAD DATA INFILE	339
Механизмы хранения	340
Транзакции	341
Определения транзакций	341
Использование транзакций в InnoDB	342
Внешние ключи	343
Хранимые процедуры	344
Простой пример	344
Локальные переменные	347
Курсоры и управляющие структуры	348
Триггеры	351
Дополнительные источники информации	353
Что дальше	353
Часть III. Безопасность веб-приложения	355
Глава 14. Угрозы безопасности веб-приложения	356
Идентификация угроз, с которыми мы сталкиваемся	356
Доступ к чувствительным данным	356
Модификация данных	359
Потеря или разрушение данных	360
Отказ в обслуживании	361
Внедрение вредоносного кода	363
Скомпрометированный сервер	363
Отказ от обязательств	364

Понимание, с кем мы имеем дело	365
Атакующие и взломщики	365
Неосведомленные пользователи инфицированных машин	365
Недовольные сотрудники	365
Похищение оборудования	366
Мы сами	366
Дополнительные источники информации	366
Что дальше	366
Глава 15. Построение защищенного веб-приложения	367
Стратегии обращения с безопасностью	367
Принятие правильного образа мыслей	368
Соблюдение баланса между безопасностью и удобством эксплуатации	368
Слежение за безопасностью	369
Наш основной подход	369
Защита кода	370
Фильтрация пользовательского ввода	370
Отмена управляющих символов в выводе	374
Организация кода	376
Что входит в ваш код	377
Соображения по поводу файловой системы	378
Стабильность кода и дефекты	379
Выполнение команд	379
Защита веб-сервера и интерпретатора PHP	381
Поддержание программного обеспечения в актуальном состоянии	381
Просмотр файла <code>php.ini</code>	382
Конфигурирование веб-сервера	382
Совместный хостинг веб-приложений	383
Безопасность сервера баз данных	384
Пользователи и система разрешений	385
Отправка данных серверу	386
Подключение к серверу	386
Запуск сервера	387
Защита сети	387
Брандмауэры	387
Использование демилитаризованной зоны	388
Подготовка к противодействию атакам DoS и DDoS	388
Безопасность компьютера и операционной системы	389
Поддержание операционной системы в актуальном состоянии	389
Запуск только необходимого программного обеспечения	389
Физическая защита сервера	390
Разработка планов на случай аварийных ситуаций	390
Что дальше	391
Глава 16. Реализация методов аутентификации с помощью PHP	393
Идентификация посетителей	393
Реализация контроля доступа	394

16 Содержание

Хранение паролей	397
Защита паролей	397
Защита множества страниц	399
Использование базовой аутентификации	400
Использование базовой аутентификации в PHP	400
Использование базовой аутентификации с помощью файлов <code>.htaccess</code> в Apache	402
Реализация специальной аутентификации	405
Дополнительные источники информации	405
Что дальше	405

Часть IV. Расширенные приемы в PHP 407

Глава 17. Взаимодействие с файловой системой и сервером 408

Загрузка файлов	409
Создание HTML-разметки для загрузки файлов	409
Написание сценария PHP для работы с файлом	411
Отслеживание продвижения загрузки с помощью сеансов	415
Избегание распространенных проблем, связанных с загрузкой	417
Использование функций для работы с каталогами	418
Чтение из каталогов	418
Получение информации о текущем каталоге	422
Создание и удаление каталогов	422
Взаимодействие с файловой системой	423
Получение информации о файлах	423
Изменение свойств файла	426
Создание, удаление и перемещение файлов	426
Использование функций выполнения программ	427
Взаимодействие со средой: <code>getenv()</code> и <code>putenv()</code>	429
Дополнительные источники информации	430
Что дальше	430

Глава 18. Использование функций для работы с сетью и протоколами 431

Исследование доступных протоколов	431
Отправка и чтение электронной почты	432
Использование данных из других веб-сайтов	432
Использование функций просмотра сети	435
Резервное копирование или зеркальное отображение файла	440
Использование FTP для резервного копирования или зеркального отображения файла	440
Выгрузка файлов	447
Избегание тайм-аутов	447
Использование других функций FTP	447
Дополнительные источники информации	448
Что дальше	448

Глава 19. Работа с датой и временем	449
Получение даты и времени в PHP	449
Часовые пояса	449
Использование функции <code>date()</code>	450
Работа с отметками времени Unix	452
Использование функции <code>getdate()</code>	453
Проверка допустимости дат с помощью <code>checkdate()</code>	455
Форматирование отметок времени	455
Преобразование между форматами даты PHP и MySQL	457
Вычисления с датами в PHP	459
Вычисления с датами в MySQL	460
Использование микросекунд	461
Использование календарных функций	461
Дополнительные источники информации	462
Что дальше	462
Глава 20. Интернационализация и локализация	463
Локализация — это больше, чем перевод	463
Наборы символов	464
Последствия для безопасности от наборов символов	466
Использование функций PHP для работы с многобайтовыми строками	466
Создание базовой страничной структуры, допускающей локализацию	467
Использование <code>gettext()</code> в интернационализированном приложении	470
Конфигурирование системы для использования <code>gettext()</code>	470
Создание файлов перевода	472
Реализация локализованного содержимого в PHP с использованием <code>gettext()</code>	473
Дополнительные источники информации	474
Что дальше	474
Глава 21. Генерация изображений	475
Настройка поддержки изображений в PHP	476
Форматы изображений	476
JPEG	476
PNG	477
GIF	477
Создание изображений	477
Создание холста изображения	479
Рисование или вывод текста на изображении	479
Вывод финальной графики	481
Очистка ресурсов	482
Использование автоматически генерируемых изображений на других страницах	482
Использование текста и шрифтов для создания изображений	482
Настройка базового холста	486
Подгонка текста к кнопке	487

18 Содержание

Позиционирование текста	490
Вывод текста на кнопку	490
Заключительные действия	490
Рисование фигур и построение диаграмм для данных	491
Использование других функций работы с изображениями	498
Что дальше	498
Глава 22. Управление сеансами в PHP	499
Понятие управления сеансами	499
Базовая функциональность сеансов	500
Понятие cookie-набора	500
Установка cookie-наборов в коде PHP	501
Использование cookie-наборов с сеансами	501
Хранение идентификатора сеанса	501
Реализация простых сеансов	502
Начало сеанса	502
Регистрация переменных сеанса	503
Использование переменных сеанса	503
Сброс переменных и уничтожение сеанса	503
Простой пример сеанса	504
Конфигурирование управления сеансами	506
Реализация аутентификации с помощью механизма управления сеансами	507
Что дальше	514
Глава 23. Интеграция JavaScript и PHP	515
Понятие AJAX	515
Краткое введение в jQuery	516
Использование jQuery в веб-приложениях	516
Использование jQuery и AJAX в PHP	526
Сценарий для беседы с поддержкой AJAX/сервер	526
Методы AJAX в jQuery	529
Клиент для беседы/приложение jQuery	532
Дополнительные источники информации	538
Что дальше	538
Глава 24. Другие полезные средства	539
Интерпретация строк: <code>eval()</code>	539
Прекращение выполнения: <code>die()</code> и <code>exit()</code>	540
Сериализация переменных и объектов	541
Получение информации о переменных среды PHP	542
Выяснение, какие расширения загружены	542
Идентификация владельца сценария	543
Выяснение, когда сценарий был модифицирован	543
Внесение временных изменений в исполняющую среду	544
Выделение цветом синтаксиса в исходном коде	545
Использование PHP в командной строке	545
Что дальше	547

Часть V. Разработка практических проектов на PHP и MySQL 549

Глава 25. Использование PHP и MySQL в крупных проектах	550
Применение практик конструирования ПО к разработке веб-приложений	551
Планирование и запуск проекта веб-приложения	552
Множественное использование кода	553
Написание сопровождаемого кода	554
Стандарты написания кода	554
Разбиение кода	557
Использование стандартной структуры каталогов	558
Документирование и совместное использование внутренних функций	558
Внедрение контроля версий	559
Выбор среды разработки	560
Документирование проектов	560
Прототипирование	561
Отделение логики от содержимого	562
Оптимизация кода	563
Использование простых оптимизаций	563
Тестирование	563
Дополнительные источники информации	565
Что дальше	565
Глава 26. Отладка и ведение журналов	567
Ошибки при программировании	567
Синтаксические ошибки	568
Ошибки времени выполнения	569
Логические ошибки	574
Вспомогательное средство отладки переменных	576
Уровни выдачи сообщений об ошибках	578
Изменение настроек выдачи сообщений об ошибках	579
Инициирование собственных ошибок	581
Элегантная обработка ошибок	581
Регистрация сведений об ошибках в журнальном файле	584
Что дальше	584
Глава 27. Реализация аутентификации и персонализации пользователей	585
Компоненты решения	586
Идентификация и персонализация пользователей	586
Хранение закладок	587
Рекомендация закладок	587
Обзор решения	587
Создание базы данных	589
Реализация базового сайта	590
Реализация аутентификации пользователей	593
Регистрация пользователей	593

20 Содержание

Вход	599
Выход	603
Изменение паролей	603
Переустановка забытых паролей	605
Реализация сохранения и извлечения закладок	610
Добавление закладок	610
Отображение закладок	612
Удаление закладок	613
Реализация выдачи рекомендаций	615
Возможные расширения	618
Глава 28. Построение клиента веб-почты с помощью инфраструктуры Laravel, часть I	619
Введение в Laravel 5	619
Создание нового проекта Laravel	619
Структура приложения Laravel	620
Цикл запросов Laravel и паттерн MVC	622
Классы моделей, представлений и контроллеров в Laravel	624
Глава 29. Построение клиента веб-почты с помощью инфраструктуры Laravel, часть II	641
Построение простого клиента IMAP с использованием Laravel	641
Функции PHP для работы с IMAP	642
Построение библиотеки IMAP для приложения Laravel	650
Объединение всех компонентов для построения клиента веб-почты	667
Реализация класса <code>ImapServiceProvider</code>	668
Страница аутентификации клиента веб-почты	669
Реализация главного представления	674
Реализация представления для вывода сообщения	679
Реализация действий удаления и отправки сообщений	682
Заключение	687
Глава 30. Интеграция с социальными сетями и аутентификация	689
Аутентификация веб-служб с помощью OAuth	689
Предоставление кода авторизации	691
Неявное предоставление	692
Построение веб-клиента Instagram	693
Отметка фотографий как понравившихся в Instagram	702
Заклучение	703
Глава 31. Построение корзины для покупок	705
Компоненты решения	706
Построение онлайн-каталога	706
Отслеживание покупок пользователей во время их нахождения в магазине	706
Реализация системы платежей	707
Построение административного интерфейса	707

Обзор решения	708
Реализация базы данных	712
Реализация онлайн-каталога	714
Вывод списка категорий	716
Вывод списка книг в категории	718
Отображение сведений о книге	719
Реализация корзины для покупок	721
Использование сценария <code>show_cart.php</code>	721
Просмотр корзины	724
Добавление книг в корзину	726
Сохранение обновленной корзины	727
Вывод сводки в панели заголовка	728
Оформление заказа	729
Реализация платежа	734
Реализация интерфейса администрирования	736
Расширение проекта	743
Приложение А. Установка Apache, PHP и MySQL	744
Установка Apache, PHP и MySQL под Unix	745
Двоичная установка	745
Установка из исходного кода	746
Модификация базовой конфигурации Apache	753
Работает ли поддержка PHP?	755
Работает ли поддержка SSL?	755
Установка Apache, PHP и MySQL для Windows и Mac OS X с использованием установочных пакетов “все в одном”	757
Установка PEAR	758
Установка PHP с другими веб-серверами	759
Предметный указатель	760

Интернационализация и локализация

В этой главе мы обсудим основы интернационализации веб-приложений в плане подготовки к локализации их содержимого. Создавать интернационализированные веб-приложения с помощью РНР несложно, а преимуществом интернациональной аудитории является ее многочисленность. Начните с осознания того, что интернационализация и локализация – разные, но связанные концепции, и вы будете готовы обрести по-настоящему всемирную аудиторию.

В главе рассматриваются следующие основные темы.

- Подготовка к разным наборам символов.
- Структурирование приложения для генерации локализованного содержимого.
- Использование `gettext()` для интернационализации и локализации.

Локализация — это больше, чем перевод

Распространенное заблуждение заключается в том, что локализацию веб-сайта, веб-приложения или вообще чего угодно сводят просто к переводу содержимого на язык, применяемый в целевом местоположении. Однако важно понимать, что ни интернационализация, ни локализация не являются тем же самым, чем является перевод содержимого. В действительности вы можете иметь веб-сайт или веб-приложение, наполненное содержимым, которое переведено, скажем, на немецкий, английский или японский язык, и все же такой веб-сайт или веб-приложение вообще может не считаться интернационализированным или даже локализованным. Это будет просто переведенный веб-сайт или веб-приложение, но не более того.

Чтобы создать локализованное программное обеспечение (охватывающее веб-сайты и веб-приложения, а также все другие типы), вы должны сначала его интернационализировать. Базовые аспекты интернационализованного ПО включают:

- вынесенные вонне строки, значки и графика;
- изменяемые способы отображения результатов, возвращаемых функциями форматирования (дат, денежных значений, чисел и т.д.).

Процесс локализации имеет смысл начинать только после того, как ПО сконструировано так, что строки вынесены вонне (т.е. все строки, используемые в функциях, классах и других местах кода, поддерживаются в одном месте и включаются либо по-другому задействуются в виде константных переменных), а функции форматирования можно изменять в зависимости от локали. Перевод содержимого является частью локализации, или нацеливанию ПО на конкретную локаль.

Локаль — это место, где что-то установлено, такое как место проживания людей, которые разговаривают на американском английском и пишут слово “color” без буквы “u”; на эту локаль можно ссылаться как на “США”. Но в мире вычислений понятие локали относится к набору параметров, идентифицирующих язык пользователя, географический регион и любые основанные на местоположении предпочтения, которые могут быть важны для представления внутри пользовательского интерфейса.

Стандартные идентификаторы локали состоят из индикаторов языка и региона, например, `en_US` для “английского языка в США” и `en_GB` для “английского языка в Великобритании”.

Вас может интересовать, зачем нужно устанавливать региональные различия, но подумайте всего лишь об отличиях в орфографии американского и британского английского, не говоря уже о контекстных отличиях, которые могут быть применены ко всему ПО для пользователей из США, но не для пользователей из Великобритании. В качестве другого примера представьте, что вы поддерживаете веб-сайт с текстом на немецком языке, который ориентирован на использование только посетителями из Германии; это будет локаль `de_DE` (“немецкий язык в Германии”). Хотя австрийцы, разговаривающие на немецком, прекрасно смогут работать с веб-сайтом в том виде, как он есть, если только вы не локализовали его для локали `de_AT` (“немецкий язык в Австрии”), этот веб-сайт никогда не будет для таких пользователей полностью правильным.

Наборы символов

Наборы символов обычно бывают *однобайтовыми* или *многобайтовыми*, что имеет отношение к количеству байтов, которое необходимо для представления символа в языке. Английский, немецкий и французский языки (среди многих других) относятся к однобайтовым наборам символов, потому что для представления символа, такого как буква *a* или цифра *9*, достаточно только одного байта. Однобайтовые кодовые наборы имеют максимум 256 символов, включая полный набор символов ASCII, символы с ударением и другие символы, необходимые для форматирования.

Многобайтовые наборы символов имеют более 256 символов, в том числе од-нобайтовые символы как подмножество. Многобайтовые языки среди прочих включают традиционный и упрощенный китайский, японский, корейский, тайс-кий, арабский и иврит. Для представления символа такие языки требуют более од-ного байта. Хорошим примером может служить слово *Токуо* (Токио). В английском языке оно записывается с помощью четырех разных символов, используя всего 5 байтов. Тем не менее, в японском языке это слово представляется двумя слога-ми, *тоу* и *кюу*, каждое из которых занимает 2 байта, в итоге давая 4 байта.

Чтобы правильно интерпретировать и отображать текст веб-страниц на языке, для которого они предназначены, вы должны сообщить веб-браузеру, какой набор символов необходимо применять. Цель достигается отправкой перед всем содер-жимым подходящих заголовков.

Таковыми заголовками являются Content-type и Content-language; они так-же могут быть установлены как атрибуты дескрипторов HTML5. Поскольку PHP снабжает вас возможностью создания динамической среды, охватите все свое приложение за счет отправки соответствующих заголовков перед текстом и выво-да корректных атрибутов в дескрипторах HTML5 внутри выходного документа.

В следующем примере функция header () выводит нужные заголовки для сайта на английском языке:

```
header("Content-Type: text/html;charset=ISO-8859-1");
header("Content-Language: en");
```

Дескрипторы HTML5, соответствующие показанным выше заголовкам, выгля-дят так:

```
<html lang="en">
<meta charset="ISO-8859-1">
```

Сайт на японском языке использует другой набор символов и другой код языка:

```
header("Content-Type: text/html;charset=UTF-8");
header("Content-Language: ja");
```

Вот дескрипторы HTML5, которые соответствуют этим заголовкам:

```
<html lang="ja">
<meta charset="UTF-8">
```

Важно правильно устанавливать заголовки. Например, если есть множество страниц, которые содержат текст на японском языке, и вы не отправили подхо-дящие заголовки относительно языка и набора символов, тогда такие страницы будут визуализироваться некорректно в браузерах, где в качестве главного языка установлен не японский. Другими словами, из-за того, что не была включена ин-формация о наборе символов, браузер предположит, что текст должен визуализи-роваться с применением стандартного набора символов.

Аналогично, если ваши страницы на японском языке используют набор сим-волов UTF-8, а браузер настроен на ISO-8859-1, то браузер будет пытаться визу-ализировать японский текст с применением однобайтового набора символов ISO-8859-1. Браузер потерпит неудачу, если только заголовки не предупредят его об использовании UTF-8 и в системе не будут установлены необходимые библиотеки и языковые пакеты для отображения текста так, как было задумано.

Последствия для безопасности от наборов символов

На протяжении всего руководства по PHP (особенно в разделах, которые сосредоточены на взаимодействии с базами данных вроде MySQL) вы встретите предупреждения о последствиях для безопасности от наборов символов. Речь не идет о том, что наборы символов в своей основе небезопасны. Предупреждения предназначены для того, чтобы заставить разработчиков чуть больше вникать в сущность применяемых наборов символов. Разработчики должны быть осведомлены о том, что могут возникнуть проблемы с безопасностью, если не предпринять основные меры предосторожности при работе со строками, которые содержат такие символы — например, в операторах SQL.

Классическим примером проблем с безопасностью, связанных с наборами символов, является несоответствие кодировки между сервером, PHP и MySQL, особенно в случае многобайтовых языков. Давайте предположим, что PHP считает, что отправляет чистый текст ASCII в MySQL, а стандартный набор символов базы данных установлен в Big5. В такой ситуации при использовании функции `mysql_real_escape_string()` (или ее объектно-ориентированного эквивалента) для очистки строк от управляющих символов перед их отправкой в базу данных интерпретатор PHP потеряет замыкающий символ двухбайтового набора символов, поскольку ему попросту не известно о том, что его нужно искать.

Описанное недоразумение будет причиной того, что закодированный символ станет по существу ненужными данными, а это само по себе уже достаточно плохо, если вы намереваетесь отображать его пользователю. Однако гораздо худший исход произойдет, когда кто-то воспользуется таким несоответствием в своих интересах и внедрит код SQL в то, что теперь является открытой строкой, стремясь добраться до вашей базы данных с целью возможного выполнения в ней вредоносного кода.

Использование функций PHP для работы с многобайтовыми строками

Упомянув ранее в главе о многобайтовых схемах кодирования символов, было бы упущением не указать, что в состав PHP входит набор функций, специально предназначенных для манипулирования многобайтовыми строками. Если вы попытаетесь манипулировать многобайтовой строкой с помощью строковой функции, не осведомленной о многобайтовых схемах кодирования символов, то такая функция, скорее всего, не сможет корректно проанализировать строку. Вообще говоря, это имеет смысл, т.к. функция, которая ожидает однобайтовые символы, и не должна знать, что делать (или привносить беспорядок), когда она сталкивается с многобайтовыми строками.

Чтобы можно было применять функции PHP для работы с многобайтовыми строками, их потребуется включить во время процесса конфигурирования, используя параметр `--enable-mbstring` при настройке и сборке PHP. Пользователям Windows понадобится включить расширение `php_mbstring.dll` в файле `php.ini`. После успешного конфигурирования вы можете применять любую из свыше 40 функций, относящихся к `mbstring`, для обработки многобайтового ввода в PHP.

Исчерпывающие сведения о функциях, работающих с многобайтовыми строками, приведены в руководстве по PHP: <http://php.net/manual/ru/book.mbstring.php>. В качестве общего правила при работе с многобайтовыми строками ищите функцию с именем, похожим на имя однобайтового эквивалента (например, когда нужно найти первое вхождение строки внутри другой строки, ищите функцию `mb_stripos()` вместо просто `stripos()`).

Создание базовой страничной структуры, допускающей локализацию

Теперь, когда вы ознакомились с основными сведениями об интернационализации, локализации и наборах символов, рассмотрим процесс создания для веб-сайта базовой страничной структуры, допускающей локализацию. Показанные здесь фрагменты позволяют пользователю выбрать целевой язык и затем получить приглашающее сообщение на выбранном языке.

Цель настоящего раздела — предоставить простой пример вынесения вонне строк (одна из характеристик интернационализации) и отображения локализованного текста на основе пользовательских предпочтений. Рабочий поток этого набора сценариев таков, что пользователь оказывается на вашем русскоязычном веб-сайте, но также имеет возможность просматривать его внутри выбранной локали (в данном примере только русской или японской).

В процесс вовлечены три действия.

- Создание и использование мастер-файла для отправки заголовка, специфичного к локали.
- Создание и применение мастер-файла для отображения информации на основе выбранной локали.
- Использование самого сценария.

В листинге 20.1 приведено содержимое мастер-файла, применяемого для отправки заголовка, специфичного к локали.

Листинг 20.1. `define_lang.php` — файл определения языка

```
<?php
if (!isset($_SESSION['lang'])) || (!isset($_GET['lang'])) {
    $_SESSION['lang'] = "ru";
    $currLang = "ru";
} else {
    $currLang = $_GET['lang'];
    $_SESSION['lang'] = $currLang;
}
switch($currLang) {
    case "ru":
        define("CHARSET", "ISO-8859-1");
        define("LANGCODE", "ru");
        break;
    case "ja":
        define("CHARSET", "UTF-8");
        define("LANGCODE", "ja");
        break;
```

```

    default:
        define("CHARSET", "ISO-8859-1");
        define("LANGCODE", "ru");
        break;
}
header("Content-Type: text/html;charset=".CHARSET);
header("Content-Language: ".LANGCODE);
?>

```

В листинге 20.1 можно заметить, что если переменная сеанса не существует, тогда используются настройки русской локали. Если бы сайт был по умолчанию японским, то данный файл понадобилось бы изменить для применения японской локали как стандартной. Этот сценарий рассчитан на использование вместе со сценарием из листинга 20.2, который содержит механизм выбора языка, устанавливая значение `$currLang` согласно результату выбора в строке 6 листинга 20.1.

Оператор `switch`, начинающийся в строке 10 листинга 20.1, имеет несколько операторов `case`, которые предназначены для присваивания подходящих значений константным переменным `CHARSET` и `LANGCODE`. В строках 27 и 28 листинга 20.1 эти переменные применяются в первый раз, когда динамически создаются и отправляются заголовки `Content-type` и `Content-language`.

Листинг 20.2. lang_strings.php — файл определения строк

```

<?php
function defineStrings() {
    switch($_SESSION['lang']) {
        case "ru":
            define("WELCOME_TXT", "Добро пожаловать!");
            define("CHOOSE_TXT", "Выберите язык");
            break;
        case "ja":
            define("WELCOME_TXT", "ようこそ!");
            define("CHOOSE_TXT", "言語を選択");
            break;
        default:
            define("WELCOME_TXT", "Добро пожаловать!");
            define("CHOOSE_TXT", "Выберите язык");
            break;
    }
}
?>

```

Внутри блоков `case` оператора `switch` в листинге 20.1 определяются две константы для каждого языка. Константами являются `CHARSET` и `LANGCODE`, которые соответствуют набору символов и коду языка для каждой локали. Эти константы используются в сценарии отображения из листинга 20.3 при создании подходящих дескрипторов `META` для набора символов и кода языка.

В листинге 20.2 создается функция, которая будет применяться в листинге 20.3 для передачи локализованных строк браузеру. Подобно листингу 20.1 в данном коде с использованием оператора `switch` определяются строки, применяемые для двух констант, `WELCOME_TXT` и `CHOOSE_TXT`, которые будут отображаться в сценарии, показанном в листинге 20.3.

Последний фрагмент головоломки связан с самим сценарием отображения, приведенным в листинге 20.3. Сценарий просто начинает сеанс, так что он может прочитать переменную сеанса, которая хранит результат выбора пользователем языка (и устанавливается через щелчок на ссылке, отображаемой на странице), и затем заполняет пустые места с помощью констант, определенных для выбранного языка.

Листинг 20.3. lang_selector.php — сценарий выбора и отображения языка

```
<?php
session_start();
include 'define_lang.php';
include 'lang_strings.php';
defineStrings();
?>
<!DOCTYPE html>
<html lang="<?php echo LANGCODE; ?>">
<title><?php echo WELCOME_TXT; ?></title>
<meta charset="<?php echo CHARSET; ?>" />
<body>
  <h1><?php echo WELCOME_TXT; ?></h1>
  <h2><?php echo CHOOSE_TXT; ?></h2>
  <ul>
    <li><a href="<?php echo $_SERVER['PHP_SELF']."?lang=ru"; ?>">ru</a>
    </li>
    <li><a href="<?php echo $_SERVER['PHP_SELF']."?lang=ja"; ?>">ja</a>
    </li>
  </ul>
</body>
</html>
```

Когда страница выбора языка (lang_selector.php) посещается впервые, она должна выглядеть примерно так, как показано на рис. 20.1, т.к. выбор языка не делался и потому по умолчанию отображается текст на русском языке.

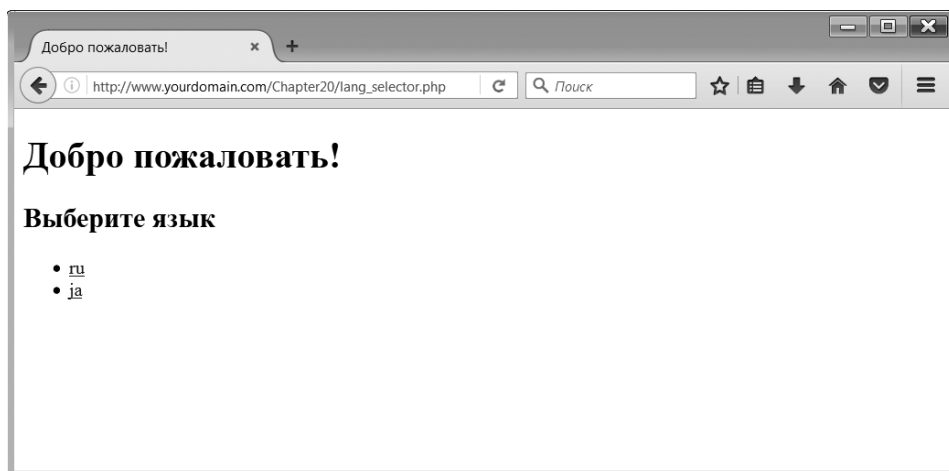


Рис. 20.1. По умолчанию отображается текст на русском языке

Тем не менее, после выбора другой локали (японской в данном примере) отображение изменится для вывода локализованных строк (рис. 20.2).

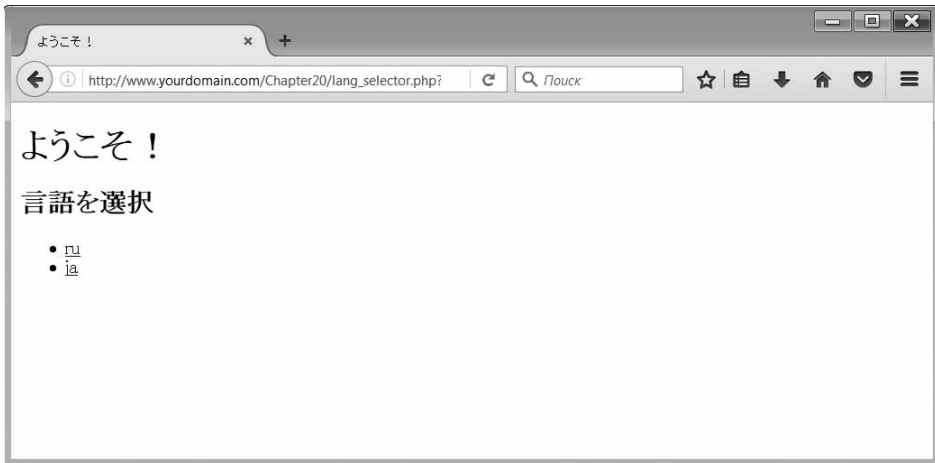


Рис. 20.2. При выборе японской локали отображается текст на японском языке

Использование `gettext()` в интернационализированном приложении

В предыдущем разделе был представлен базовый подход к интернационализации и локализации набора веб-страниц. Более передовой подход, применяемый для крупных сайтов с большим объемом содержимого или взаимодействия с пользователем, предусматривает использование встроенной функции PHP по имени `gettext()`, которая предлагает уровень API-интерфейса к GNU-пакету `gettext`.

Несмотря на то что применение функции `gettext()` вполне очевидно специфично для PHP, многие другие языки программирования обладают встроенной поддержкой GNU-пакета `gettext`. Концептуальное понимание того, что делает GNU-пакет `gettext`, важно для общего понимания интернационализации и локализации независимо от используемого языка программирования. В своей основе GNU-пакет `gettext` ищет специфически представленную строку, указанную в определенной локали, и заменяет такую строку в подходящем месте. Это очень похоже на процесс, который применялся для изменения строк, ассоциированных с константами, в листинге 20.2, но в более широких масштабах.

Конфигурирование системы для использования `gettext()`

Конфигурирование PHP для применения `gettext()` и связанных функций требует установки GNU-пакета `gettext`, изменения конфигурации PHP и привязки к некоторому каталогу внутри дерева документов веб-сервера.

Если вы используете сервер Linux или Mac OS X для разработки или развертывания, тогда с высокой вероятностью имеете установленный GNU-пакет `gettext`. В случае применения для разработки или развертывания сервера Windows, скорее всего, это не так.

Чтобы установить GNU-пакет `gettext` в любой системе, перейдите в раздел **Downloading gettext** (Загрузить `gettext`) страницы <http://www.gnu.org/software/gettext/> и загрузите файл, подходящий для вашей системы. После установки GNU-пакета `gettext` вы можете сконфигурировать PHP для его распознавания и использования.

Включение `gettext()` и связанных функций в PHP после установки GNU-пакета `gettext` на сервере требует изменения конфигурации и повторной компиляции PHP в системе Linux или Mac OS X и разрешения готового расширения в системе Windows. В случае конфигурирования PHP для компиляции в системе Linux или Mac OS X добавьте следующий флаг компиляции:

```
--with-gettext
```

Добавив такой флаг компиляции, продолжайте компилировать и устанавливать PHP как обычно. Не забудьте перезапустить Apache после установки новой сборки модуля PHP.

В системе Windows отредактируйте файл `php.ini`, чтобы включить `php_gettext.dll`, удалив символ точки с запятой в начале строки вида:

```
;extension=php_gettext.dll
```

Внеся такое изменение и сохранив файл, перезапустите веб-сервер Apache. Благодаря описанным выше изменениям вы должны увидеть в выводе функции `phpinfo()` раздел, который указывает, что поддержка GNU-пакета `gettext` включена.

При включенной поддержке GNU-пакета `gettext` далее нужно создать в корне документов веб-сервера каталоги для содержимого, специфичного для локали. Первым делом вам понадобится создать в корне документов каталог, который будет содержать все каталоги локали (для всех поддерживаемых локалей).

Каталоги локали внутри родительского каталога должны иметь имена, которые содержат аббревиатуру из двух букв нижнего регистра для языка согласно спецификации ISO-639-1 (например, “en”, “ja”, “ru”), символ подчеркивания и код страны из двух букв верхнего регистра в соответствии со спецификацией ISO-3166-1 (скажем, “US”, “JP”, “RU”). Внутри каталога, специфичного для локали, должен присутствовать каталог по имени `LC_MESSAGES`.

Таким образом, для поддержки на веб-сайте трех локалей с применением GNU-пакета `gettext` и PHP-функции `gettext()` структура каталогов может выглядеть примерно так:

```
/htdocs
  /locale
    /ru_RU
      /LC_MESSAGES
    /en_US
      /LC_MESSAGES
    /ja_JP
      /LC_MESSAGES
```

Далее вы узнаете, какие виды файлов помещаются в указанные каталоги — это не файлы PHP, а файлы, которые содержат переведенные строки, предназначенные для использования по всему локализованному веб-сайту.

Создание файлов перевода

Файлы перевода, хранящиеся в файловой системе, как указано выше, и применяемые GNU-пакетом `gettext`, являются специфическим файловым типом, который называется файлом переносимого объекта (Portable Object – PO). Такие файлы содержат простой текст, но имеют расширение `*.po`. Вообще говоря, для создания файлов PO какой-то специальный редактор не требуется – в конце концов, в них находится всего лишь простой текст. Однако многие обнаруживают, что использование определенного редактора либо инструмента управления содержимым значительно сокращает накладные расходы, связанные с сопровождением файлов подобного рода (которые включают сотрудничество с переводчиками и другими создателями содержимого).

Мы рекомендуем опробовать для создания и сопровождения файлов локализованного содержимого инструменты вроде Poedit (<https://poedit.net/>) или POEditor (<https://poeditor.com/>), но пока просто продемонстрируем примеры файлов PO, введенные как простой текст. Для каждой локали необходим один файл PO по имени `messages.po`.

Файлы PO начинаются с идентифицирующей заголовочной информации и продолжаются списком идентификаторов и строк сообщений. В листинге 20.4 приведен пример содержимого файла PO, устанавливающего две строки для применения в локали `ru_RU`.

Листинг 20.4. `messages.po` – файл PO для локали `ru_RU`

```
# требуются пустые msgid и msgstr
msgid ""
msgstr ""

"Project-Id-Version: 0.1\n"
"POT-Creation-Date: 2017-04-05 14:00+0500\n"
"Last-Translator: Иван Петров <ivan@petrov.com>\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Language: ru_RU\n"

# приветственное сообщение
msgid "WELCOME_TEXT"
msgstr "Добро пожаловать!"

# приглашение выбрать язык
msgid "CHOOSE_LANGUAGE"
msgstr "Выберите язык"
```

В начале файла указан пустой идентификатор сообщения (`msgid`) и пустая строка сообщения (`msgstr`). За ними следует идентифицирующая информация о файле и его создателях: файл имеет версию 0.1, создан 5 апреля 2017 года Иваном Петровым и закодирован в наборе символов UTF-8 для предполагаемого использования с локалью `ru_RU`.

После всей заголовочной информации приводятся сообщения и их переводы. В этом примере имеются два сообщения, идентифицируемые собственными ключами: `WELCOME_TEXT` и `CHOOSE_LANGUAGE`. Ключи сообщений похожи на константы, которые применялись в простой версии локализации ранее в главе, но именованы по-другому, чтобы не путаться в двух примерах.

За каждым ключом сообщения следует строка сообщения для использования вместо этого ключа. Комментарии поясняют назначение каждого набора ключа и строки, а между парами строк предусмотрена одна пустая строка.

Выглядит просто, что и есть на самом деле, но с созданием и сопровождением файлов PO также связана скрытая сложность: длинный список доступных для применения параметров, которые перечислены в спецификации по адресу <http://www.gnu.org/software/gettext/manual/gettext.html#PO-Files>.

Спецификацию рекомендуется прочитать. Редактор файлов PO используется еще и потому, что он берет на себя заботу об еще одном действии — преобразовании файлов PO в файлы MO. Файл MO представляет собой файл машинного объекта (Machine Object), или файл, который содержит двоичный объект, в итоге читаемый GNU-пакетом `gettext`. Легкие для восприятия и сопровождения файлы PO не применяются GNU-пакетом `gettext` напрямую, а взамен используются файлы MO.

Благодаря GNU-пакету `gettext` и связанным с ним инструментам, установленным в системе, файлы PO можно преобразовывать в файлы MO с помощью служебной программы. В среде Linux соответствующая команда будет такой:

```
msgfmt messages.po -o messages.mo
```

Команда создает файл MO по имени `messages.mo` из содержащего простой текст файла PO по имени `messages.po`. Теперь все готово к применению PHP для выполнения остальной работы.

Реализация локализованного содержимого в PHP с использованием `gettext()`

После всех объяснений способа установки и применения GNU-пакета `gettext` реализация в PHP может выглядеть слегка разочаровывающей. Ниже перечислены основные шаги реализации.

- Использовать `putenv()` для установки переменной среды `LC_ALL` для локали.
- Применить `setlocale()` для установки значения `LC_ALL`.
- Использовать `bindtextdomain()` для установки местоположения каталога переводов для заданной области (область в этом случае означает имя, идентифицирующее файл, который хранит строки сообщений, а не домен, подобный `www.mydomain.com`).
- Применить `textdomain()` для установки стандартной области, подлежащей использованию с `gettext()`.
- Применить `gettext("какой-то msgid")` или `_("какой-то msgid")` для обращения к переводу GNU-пакетом `gettext`, который относится к указанному идентификатору сообщения.

В результате сбора всех фрагментов вместе получится сценарий, похожий на приведенный в листинге 20.5.

Листинг 20.5. `use_gettext.php` — чтение файлов MO с помощью PHP

```
<?php
$locale="ru_RU";
putenv("LC_ALL=".$locale);
setlocale(LC_ALL, $locale);
```

```
$domain='messages';
bindtextdomain($domain, "./locale");
textdomain($domain);
?>
<!DOCTYPE html>
<html>
<title><?php echo gettext("WELCOME_TEXT"); ?></title>
<body>
  <h1><?php echo gettext("WELCOME_TEXT"); ?></h1>
  <h2><?php echo gettext("CHOOSE_LANGUAGE"); ?></h2>
  <ul>
    <li><a href="<?php echo $_SERVER['PHP_SELF']. "?lang=ru_RU"; ?>">ru_RU
    </a></li>
    <li><a href="<?php echo $_SERVER['PHP_SELF']. "?lang =ja_JP"; ?>">ja_JP
    </a></li>
  </ul>
</body>
</html>
```

Если после освоения основ применения интернационализации и локализации вы собираетесь заняться разработкой приложения, используемого носителями множества разных языков, тогда мы рекомендуем обратить внимание на инфраструктуру локализации, базирующуюся на GNU-пакете `gettext`, и краудсорсинговые службы перевода для поддержки создания файлов PO (на тот случай, если вы не располагаете группой носителей языка или же не готовы тратить крупные суммы на услуги переводчиков).

Дополнительные источники информации

Интернационализация и локализация — это обширные темы, и в настоящей главе были раскрыты только самые основные концепции. Например, мы не обсуждали локализацию чисел, дат и денежных значений с применением PHP. Тем не менее, все упомянутые задачи можно решить с использованием встроенной функциональности, такой как функция `strftime()` для отображения времени с учетом локали, или за счет расширения функциональности для создания вспомогательных классов, которые удовлетворяют имеющиеся потребности. Можете также взглянуть на инфраструктуру PHP, которые помогают все это обработать автоматически, либо с целью оценки их пригодности к вашему проекту, либо для ознакомления с тем, как разработчики построили такую функциональность. Обилие примеров можно найти на веб-сайтах Zend Framework (<http://framework.zend.com/manual/current/en/modules/zend.i18n.translating.html>) и Symfony (<http://symfony.com/doc/current/book/translation.html>).

Что дальше

Одним из многочисленных полезных действий, которые можно выполнять с помощью PHP, является создание изображений на лету. В следующей главе будет показано, как применять функции библиотеки обработки изображений для достижения ряда интересных и полезных эффектов.