

API Яндекс, Google и других популярных веб-сервисов

Готовые решения для вашего сайта



API Яндекс.Карт

ISPmanager API

API Google-сервисов

API Twitter

API Wikipedia

Готовые к размещению
в сети сайты

PRO
ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ

+ 
Материалы
на www.bhv.ru

УДК 681.3.06
ББК 32.973.26-018.2
П29

Петин В. А.

П29 API Яндекс, Google и других популярных веб-сервисов. Готовые решения для вашего сайта. — СПб.: БХВ-Петербург, 2012. — 480 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-0743-1

Рассмотрены возможности, предоставляемые API Яндекс, Google, Twitter, ISPmanager, Wikipedia. Показано, как повысить функциональность и привлекательность веб-проектов, интегрировав в них возможности, предоставляемые API этих популярных веб-сервисов. Описано создание 4-х больших готовых к размещению в сети проектов (личного кабинета для сайта хостинговой компании, каталога предприятий, сайта учета заказов для фирмы такси, интерактивной карты местности региона), а также ряда небольших практических решений. Во всех случаях использованы современные технологии создания сайтов без перезагрузки страницы, в том числе подробно рассмотренные в книге фреймворки хаях и jQuery. Исходные коды описанных в книге и готовых к размещению в сети проектов можно скачать по ссылке: <ftp://85.249.45.166/9785977507431.zip>.

Для веб-разработчиков

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Наталья Перишакова</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.08.11.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 38,7.

Тираж 1500 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0743-1

© Петин В. А., 2011
© Оформление, издательство "БХВ-Петербург", 2011

Оглавление

Введение.....	1
Для кого и о чем эта книга	1
Структура книги.....	1
Благодарности	2
Глава 1. API веб-сервисов и технологии использования.....	3
1.1. Использование возможностей общедоступных Web API в асинхронных приложениях.....	3
1.2. Библиотека xAjax	4
1.2.1. Как работает xAjax	5
1.2.2. Возможности xAjax	5
1.2.3. Подключение xAjax	6
1.2.4. Методы объекта <i>xajaxResponse</i>	8
Метод <i>assign()</i>	8
Метод <i>append()</i>	8
Метод <i>prepend()</i>	9
Метод <i>replace()</i>	9
Метод <i>remove()</i>	9
Метод <i>create()</i>	9
Метод <i>insert()</i>	10
Метод <i>insertAfter()</i>	10
Метод <i>clear()</i>	10
Метод <i>createInput()</i>	10
Метод <i>insertInput()</i>	11
Метод <i>insertInputAfter()</i>	11
Метод <i>removeHandler()</i>	11
Метод <i>includeScript()</i>	11
Метод <i>script()</i>	12
Метод <i>addEvent()</i>	12
Метод <i>call()</i>	12
Метод <i>alert()</i>	13
Метод <i>redirect()</i>	13

1.2.5. Сайт — тренировочный стенд для изучения xAjax	13
1.2.6. Глобальные переменные xAjax	17
Глобальные константы	17
Методы объекта <i>xajax</i>	18
1.3. Примеры использования xAjax	21
1.3.1. Форма регистрации с проверкой правильности заполнения полей "на лету"	21
1.3.2. Динамически подгружаемые <i>select</i> -элементы	24
1.3.3. Многоуровневый неоднородный каталог	30
1.3.4. Динамическое управление количеством полей формы	34
1.4. Библиотека jQuery	40
1.4.1. Возможности jQuery	41
1.4.2. Использование jQuery	41
Функция <i>\$</i>	42
Селекторы	42
Методы jQuery	46
Обработка событий в jQuery	47
Эффекты в jQuery	47
1.4.3. PHP и jQuery	48
Динамическая подгрузка jQuery и плагина <i>jQuery</i> <i>UI</i>	48
Совместное использование jQuery <i>UI</i> -виджетов <i>Tabs</i> и <i>Accordion</i>	51
Глава 2. API Яндекса	59
2.1. API Яндекс.Бара	59
2.1.1. Создание описания	60
2.1.2. Подготовка необходимых ресурсов	61
2.1.3. Создание пакета	61
2.1.4. Создание манифеста	61
2.1.5. Создание сборки	62
2.2. Виджетная платформа	63
2.3. API Яндекс.Спеллера	66
2.3.1. Web Service API	67
2.3.2. JavaScript API	69
2.4. API Поиска по блогам	71
2.5. API Яндекс.Фоток	77
Глава 3. API Яндекс.Карт	83
3.1. Как установить Яндекс.Карты на сайт	83
3.1.1. Получение API-ключа	83
3.1.2. Загрузка API	85
3.1.3. Создание контейнера для размещения карты	86
3.1.4. Создание карты	86
3.1.5. Удаление карты	87
3.2. Управление картой	87
3.2.1. Встроенные элементы	88
Перемещение	88

Масштабирование двойным щелчком мыши	88
Масштабирование колесиком мыши	89
Лупа	89
"Горячие" клавиши	89
Линейка.....	89
3.2.2. Пример со встроенными элементами управления	90
3.2.3. Внешние элементы управления	93
Панель инструментов	93
Элемент масштабирования	94
Компактный элемент масштабирования.....	95
Обзорная карта.....	96
Переключатель типа карты	97
Масштабная линейка	98
Поиск по карте	98
3.2.4. Пример с внешними элементами управления	99
3.2.5. Пользовательские кнопки для панели инструментов	102
Обычная кнопка	103
Переключатель	107
Флажок	111
Разделитель на панели инструментов	112
3.2.6. Создание пользовательских элементов управления	112
3.3. События	115
3.3.1. Обработчики событий	115
3.3.2. Подключение обработчика событий	115
3.3.3. Удаление обработчика событий	116
3.3.4. Включение/выключение обработчика событий	116
3.3.5. Инициирование события	116
3.4. Объекты-оверлеи на карте	117
3.4.1. Балун	117
Параметры балуна	118
Установка содержимого балуна	119
Задание стиля для содержимого балуна	120
3.4.2. Метки	120
Добавление метки на карту.....	120
Содержимое метки	121
Перетаскивание метки.....	121
Задание стиля метки	122
Пример динамического управления свойствами метки	123
Создание пользовательского значка метки	129
3.4.3. Ломаная	131
Добавление ломаной на карту	131
Задание стиля ломаной.....	132
Методы объекта <i>YMaps.Polyline</i>	132
3.4.4. Многоугольник	137
Добавление многоугольника на карту	137

Задание стиля многоугольника.....	137
Методы объекта <i>YMaps.Polygon</i>	138
3.4.5. Всплывающая подсказка.....	141
3.4.6. Группировка объектов.....	142
3.5. Сервисы	143
3.5.1. Геокодирование	143
3.5.2. Геотаргетинг	144
3.5.3. Маршрутизация.....	145
Точки маршрута.....	145
События построения маршрута	146
Отрезки маршрута	147
Отображение маршрута на карте	148
3.5.4. Визуализация YMapsML	148
3.5.5. Карта пробок	149
3.6. Пользовательские карты.....	151
3.6.1. Создание пользовательского слоя карты	151
3.6.2. Подготовка тайлов для пользовательского слоя карты	152
Глава 4. Примеры использования в проектах API Яндекс.Карт	155
4.1. Каталог предприятий	155
4.1.1. Проектирование базы данных сайта	156
4.1.2. Программирование сайта	160
Программирование дерева категорий видов деятельности.....	162
Вывод списка предприятий категории.....	165
Форма поиска предприятий	168
Вывод результатов поиска	171
Программа начальной загрузки.....	172
4.1.3. Использование API Яндекс.Карт.....	173
4.2. Сайт учета заказов такси	176
4.2.1. Проектирование базы данных	176
4.2.2. Программирование сайта	180
Программирование блока <i>Водители</i>	180
Программирование блока <i>Автомобили</i>	187
Получение заказа и создание маршрута с API Яндекс.Карт	195
Программирование блока <i>Заказы</i>	204
4.3. Создание карты местности с несколькими слоями пользовательских карт.....	212
4.3.1. Создание пользовательских карт городов	213
4.3.2. Размещение пользовательских слоев на Яндекс.Карте	213
4.3.3. Создание переключателя выбора городов.....	216
4.3.4. Размещение на картах меток	219
4.3.5. Скрытие/показ меток при изменении масштаба	223
4.3.6. Передача параметров в скрипт и возврат значений из скрипта.....	225
4.4. Создание, редактирование меток для карты местности с несколькими слоями пользовательских карт.....	229
4.4.1. Проектирование базы данных	230

4.4.2. Авторизация администратора	232
4.4.3. Вывод карты	234
4.4.4. Добавление новой метки	234
4.4.5. Редактирование содержимого метки	238
4.4.6. Изменение местоположения метки	241
4.4.7. Удаление метки	243
4.4.8. Загрузка на сервер файлов через форму без перезагрузки страницы	245
4.4.9. Форма поиска меток	246
4.4.10. Варианты изменения скрипта	252
Глава 5. ISPmanager API	253
5.1. ISPmanager API	253
5.1.1. Методы авторизации	255
Авторизация с использованием уникального номера сессии	256
Авторизация с использованием параметра <i>authinfo</i>	256
Авторизация с использованием доверенных IP-адресов	257
Авторизация при локальном вызове функций ISPmanager	257
HTTP или HTTPS?	257
Вызов функций ISPmanager с правами другого пользователя	257
5.1.2. Вызов функций ISPmanager из PHP	258
5.1.3. Администратор сервера	258
Параметры администратора, создание, изменение	258
Удаление администраторов	259
Включение администратора	260
Выключение администратора	260
5.1.4. Реселлер	260
Создание, изменение, параметры реселлера	261
Удаление реселлеров	262
Включение реселлера и его пользователей	262
Отключение реселлера и его пользователей	263
Доступ к функциям	263
Сообщение в центр поддержки	263
5.1.5. Пользователь	264
Создание, изменение, параметры пользователя	264
Удаление пользователей	266
Включение пользователя и всех его WWW-доменов	266
Отключение пользователя и всех его WWW-доменов	266
Доступ к функциям	266
Разрешение доступа к выбранным функциям	267
Запрещение доступа к выбранным функциям	267
5.1.6. Почтовые ящики	267
Создание, изменение, почтовый ящик	268
Автоответчик (vacation), просмотр, изменение	269
Сортировка почты	269
Удаление почтовых ящиков	272

Очистка почтовых ящиков	272
Включение почтовых ящиков	272
Отключение почтовых ящиков	273
5.1.7. WWW-домены	273
Создание, изменение, параметры WWW-домена	273
Удаление WWW-доменов	275
Ротация логов, просмотр, изменение	275
5.1.8. Почтовые домены	275
Создание, изменение, настройки почтового домена	276
Удаление почтового домена	277
5.1.9. Доменные имена (DNS)	277
Создание, изменение, параметры домена	277
Управление записями	278
Подтверждение удаления домена	279
Обновление домена на внешнем сервере имен	280
Настройки доменов по умолчанию, просмотр, изменение	280
Внешние серверы имен	280
5.1.10. Базы данных	282
Создание, изменение, параметры базы данных	282
Удаление выбранных баз	283
Управление пользователями базы данных	283
Проверка выбранных баз	285
5.1.11. Брандмауэр (firewall)	285
Настройка фильтрации для порта, просмотр, изменение	285
5.1.12. Сервисы	286
Создание, изменение, настройка сервиса	286
Удаление сервиса	287
Остановка сервисов	287
Запуск сервисов	287
Перезапуск сервисов	287
Глобальные настройки сервисов, просмотр, изменение	288
5.1.13. Задания резервного копирования	289
Создание, изменение, задание	289
Удаление задания	290
Включение задания	290
Отключение задания	290
Сделать резервную копию сейчас	290
Данные для резервного копирования	291
5.1.14. Перенос пользователя	292
5.1.15. Списки блокировки dnsbl	293
Создание, изменение, просмотр параметров	293
Удаление списков блокировки dnsbl	293
5.1.16. "Серый" список (greylisting)	294
Создание, изменение, правило для "серого" списка	294
Удаление правила серого списка	295

5.1.17. "Белый" список	295
Создание, изменение, параметры записи	295
Удаление	296
5.1.18. "Черный" список	296
Создание, изменение, параметры записи	296
Удаление	297
5.1.19. Используемые ресурсы	297
5.1.20. Информация о системе	297
5.1.21. Параметры сервера	297
5.1.22. IP-адреса	298
Создание, изменение, параметры IP-адреса	298
Удаление IP-адреса	299
5.1.23. Настройки RНР	299
5.1.24. Расширения RНР	300
Включение выбранных расширений RНР	300
Отключение выбранных расширений RНР	300
Установка других расширений RНР, просмотр, изменение	301
5.1.25. Модули Perl	301
Добавление модуля Perl, просмотр, изменение	301
5.1.26. Возможности	302
Просмотр, изменение	302
Удаление	302
Включение	303
Выключение	303
5.1.27. Шаблоны пользователей	303
Создание, изменение, параметры шаблона	303
Удаление шаблонов	305
Доступ к функциям	305
Импорт шаблонов, просмотр, изменение	306
5.1.28. Настройки доменов по умолчанию	306
5.1.29. Ротация журналов WWW-домена	307
5.1.30. FTP-аккаунты	307
Создание, изменение	308
Удаление FTP-аккаунтов	309
Включение FTP-аккаунтов	309
Временное отключение FTP-аккаунтов	309
5.1.31. Редиректы (перенаправление URL)	309
Создание, изменение, параметры перенаправления	309
Удаление перенаправления	310
5.1.32. Страницы ошибок	310
Создание, изменение, параметры страницы ошибки	311
Удаление страницы ошибки	312
5.1.33. Ограничение доступа к каталогу	312
Просмотр, изменение	312
Снятие защиты с каталога	313
Пользователи защищенного каталога	313

5.1.34. Почтовые группы.....	314
Создание, изменение, параметры почтовой группы.....	314
Удаление почтовых групп	315
5.1.35. Почтовые редиректы	315
Создание, изменение, параметры почтового редиректа	315
Удаление почтовых редиректов	316
5.1.36. Почтовые автоответчики.....	316
Создание, изменение, параметры автоответчика.....	316
Удаление почтовых автоответчиков	317
5.2. Сайт-тренажер для изучения запросов к API ISPmanager	317
5.2.1. Получение доступа к демо-серверу с ISPmanager.....	318
5.2.2. Создание формы получения данных ISPmanager.....	318
5.2.3. Получение списка шаблонов (тарифных планов).....	321
5.2.4. Добавление нового шаблона.....	323
5.2.5. Редактирование шаблона	327
5.2.6. Удаление шаблона	331
5.2.7. Получение списка пользователей.....	333
5.2.8. Добавление нового пользователя	334
5.2.9. Редактирование параметров пользователя	338
5.2.10. Удаление пользователя	340
Глава 6. Создание личного кабинета для сайта хостинговой компании.....	343
6.1. Необходимый функционал сайта.....	343
6.2. Проектирование баз данных	344
6.3. Главная страница	351
6.4. Регистрация пользователей	353
6.5. Вход в систему, восстановление пароля	364
6.6. Выбор тарифного плана	371
6.7. Заказ тарифного плана. Формирование счета	373
6.8. Счета пользователя	378
6.9. Просмотр, изменение тарифных планов	381
6.10. Меню администратора.....	385
6.11. Просмотр счетов	386
6.12. Подтверждение оплаты счета администратором	391
6.13. Просмотр и редактирование профилей пользователей и их тарифных планов.....	394
6.14. Функция активации тарифа с использованием API ISPmanager	399
6.15. Скрипты, запускаемые по cron. Деактивация аккаунта	402
Глава 7. Google, Twitter и другие сервисы	405
7.1. API сервисов Google	405
7.2. Google Ajax API	406
7.3. Ajax API для Google Переводчика.....	410
7.4. Ajax API поиска Google.....	414
7.5. API Google Chart	419
7.6. API визуализаций Google	428

7.7. API Wikipedia	435
7.8. API Twitter	440
7.9. API Loginza	453
Заключение	459
Приложение. Описание компакт-диска	461
Предметный указатель	463

ГЛАВА 1



API веб-сервисов и технологии использования

1.1. Использование возможностей общедоступных Web API в асинхронных приложениях

Создание веб-сайтов требует от разработчиков немало опыта и навыков. При этом они должны постоянно следить за новшествами и быть в курсе современных технологий. Развитие технологий программирования привело к тому, что многие популярные интернет-сервисы принесли на просторы сети такую функциональность, о которой еще несколько лет назад никто не мог и мечтать. Сделан громадный шаг от простых страниц для отображения статической информации до сервисов, не уступающих своим десктопным собратьям. Кто мог несколько лет назад предположить, что мы сможем путешествовать по миру с Google Street View или Яндексом? Панорамы, с помощью которых можно бродить по улицам любого города мира, сидя дома в уютном кресле! Или пользоваться графическим редактором прямо на страницах сайта. Как хочется разместить такие же сервисы на своих сайтах для привлечения клиентов. И сейчас это становится реальностью! Очень многие популярные Web-сервисы дают возможность использования своего функционала предоставлением API своих сервисов для публичного пользования.

Что же такое API? API — интерфейс прикладного программирования (или интерфейс программирования приложений), т. е. набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована. Если программу (модуль, библиотеку) рассматривать как черный ящик, то API — это множество "ручек", которые доступны пользователю данного ящика, которые он может вертеть и дергать.

Вообще, API — довольно широкое понятие. Мы же здесь будем рассматривать только API для сайтов — Web API. Web API (или API веб-приложений) — это интерфейс программирования приложений, использующих данные определенных веб-

сервисов. Web API представляют собой инструмент, позволяющий получать данные от определенного стороннего сервиса, обрабатывать их в собственном приложении и при необходимости передавать какие-либо данные или команды обратно в сторонний сервис. Web API — это внешние приложения, которые могут легко встраиваться в любой сайт. Обычно используют HTML, XML, JavaScript и Flash в различных комбинациях.

Количество сервисов, предоставляющих публичный доступ к своему API, очень велико. Это и платежные системы (например, PayPal, WebMoney), торговые площадки (Amazon, "Молоток"), социальные сети (Facebook, Twitter, "ВКонтакте") и др.

Использование API популярных сайтов позволяет многократно увеличить полезность собственного ресурса, создать интересный и неповторимый контент, удовлетворить требования всевозможных бизнес-задач — устроить их не составляет труда. Использование API очень выгодно — они потребляют ресурсы своего сервера, а не вашего, при этом работают на вашем сайте. Это очень удобно: часто для установки API на сайт, как правило, достаточно встроить предложенный производителем код в нужное место на странице сайта, хотя можно создавать и собственные неповторимые проекты.

Использование API весьма рекомендуется владельцам небольших сайтов, а также сайтов с небольшой функциональностью — блок мировых новостей, красивые часы, котировки валют или прогноз погоды (смотря что подойдет вашему сайту) и многое другое может расширить функциональность сайта, украсить его, а значит, сделать более привлекательным для посетителей. Не следует пренебрегать возможностями API и крупным проектам. Например, полезными будут встроенная проверка орфографии, переводчик, поисковик, интерактивная карта и многое другое.

Для веб-программистов изучение Web API поможет не просто повысить свой профессиональный уровень знаний в веб-разработке, но и получить бесценные знания в области интеграционных технологий и стать широко востребованным специалистом.

Применение API популярных сервисов и внедрение их возможностей в своих проектах мы будем рассматривать в связке с современными технологиями сайтостроения, предполагающего применение технологии Ajax. Поэтому далее мы рассмотрим библиотеки xAjax и jQuery, позволяющие создавать современные сайты без перезагрузки страницы.

1.2. Библиотека xAjax

xAjax — это библиотека Open Source классов PHP, которая позволяет легко создавать мощные веб-ориентированные Ajax-приложения, использующие HTML, CSS, JavaScript и PHP. Приложения, разработанные при помощи библиотеки xAjax, могут асинхронно вызывать расположенные на сервере PHP-функции и обновлять содержание без перезагрузки страницы. xAjax предоставляет простую реализацию технологии Ajax, а начиная с версии 0.5, еще и PHP-инструменты для формирования HTML-форм и документов. В отличие от многих других подобных библиотек,

хАјах позволяет разрабатывать Ајах-приложения, не требуя от разработчика знания JavaScript. Библиотека хАјах распространяется по лицензии GNU Lesser General Public License (LGPL) и может быть использована для написания платного программного обеспечения. Сайт проекта — <http://xajaxproject.org>. На момент написания книги доступна версия 0.6.

1.2.1. Как работает хАјах

Библиотека хАјах создает функции JavaScript, являющиеся оболочкой для PHP-функций, которые вы можете вызывать с сервера из вашего приложения. Когда вызывается функция JavaScript, являющаяся оболочкой для функции PHP, то она использует объект XMLHttpRequest для асинхронного соединения с хАјах-объектом на сервере, который вызывает соответствующую функцию PHP. После завершения этого действия возвращается хАјах XML-ответ от вызванной PHP-функции. Возвращенный XML-код содержит инструкции и данные, которые будут проанализированы специальными функциями JavaScript-части хАјах и использованы для обновления содержания вашего приложения.

1.2.2. Возможности хАјах

Библиотека хАјах предлагает следующие возможности, которые вместе делают ее уникальным и мощным инструментом.

- хАјах — уникальная библиотека на JavaScript, которая может анализировать возвращенный XML-код и автоматически его обрабатывать согласно инструкциям, находящимся в этом ответе. Так как хАјах обрабатывает все это, то вам не нужно писать отдельные функции на JavaScript для того, чтобы обрабатывать возвращенный XML-код.
- хАјах — это объект, ориентированный на создание отношений между программным кодом и данными для хранения хАјах-кода отдельно от другого программного кода. Так как это объектно-ориентированный код, то вы всегда можете добавлять свои функции в класс xajaxResponse, используя метод script().
- хАјах работает в браузерах Firefox, Mozilla, Internet Explorer и Safari. Помимо обновления значений элементов (имеется в виду DOM) и innerHTML, хАјах также может быть использован для обновления стилей, CSS-классов, значений флажков и раскрывающихся списков или каких-либо других свойств элемента.
- хАјах может использовать одномерные и многомерные массивы, а также ассоциативные массивы из JavaScript в PHP как параметры ваших хАјах-функций. В дополнение, если вы вводите JavaScript-объект в хАјах-функцию, функция PHP будет получать ассоциативный массив, определяющий свойства этого объекта.
- хАјах предоставляет легкую асинхронную обработку форм. Используя JavaScript-метод xajax.getFormValues(), в форме вы можете легко передать массив данных, как параметры для асинхронной хАјах-функции:

```
xajax_processForm(xajax.getFormValues('formId')).
```

Используя xAjax, вы можете динамически подгружать дополнительный JavaScript-код для вашего приложения для того, чтобы при его исполнении менялись свойства элемента DOM.

- ❑ xAjax автоматически сравнивает данные, возвращенные из PHP-функций, с текущими значениями свойства элемента, который вы хотите изменить. Свойство изменяется только в том случае, если это изменение актуально на текущий момент. Это позволяет устранить мерцание, которое происходит, если элемент обновляется через определенные промежутки времени. Каждая функция регистрируется для того, чтобы быть доступной через xAjax, который имеет различные типы запросов. Все функции по умолчанию используют метод передачи POST, за малым исключением — метод GET. Это сделано для большей безопасности запросов.
- ❑ Если не определен запрашиваемый URI, xAjax пытается автоматически определить запрашиваемый URL скрипта. Алгоритм автоопределения xAjax достаточно универсален, так что он будет работать как на безопасном протоколе `https://`, так и на `http://` и на нестандартных портах.
- ❑ xAjax перекодирует все свои запросы и ответы в кодировку UTF-8. Таким образом, он поддерживает большой спектр различных знаков и языков.
- ❑ xAjax был протестирован на различных языках в кодировке Unicode, включая испанский, русский, арабский. Почти весь JavaScript-код динамически подгружается через JavaScript-расширения.
- ❑ xAjax может быть использован в шаблонном движке Smarty. Для создания переменной в Smarty должен быть следующий код:

```
$smarty->assign('xajax_JavaScript', $xajax->getJavascript());
```

При использовании xAjax подставляйте в заголовок тег `{xajax_JavaScript}`.

1.2.3. Подключение xAjax

xAjax — это PHP-библиотека, которая отличается тем, что позволяет исполнять JavaScript-код на основе PHP-кода. Весь процесс состоит из двух PHP-классов и обработчика XML на JavaScript. В общем, на PHP сначала инициализируется объект и объявляются функции, которые будут отвечать на Ajax-запрос. В этих функциях необходимо использовать объект, который и будет генерировать XML-ответ.

Для скачивания библиотеки xAjax следует обратиться по адресу <http://xajax-project.org/en/download/> и нажать на ссылку **Xajax 0.6 beta1**. Начнется скачивание архива с библиотекой (рис. 1.1).

Далее надо распаковать архив в корневой каталог сайта. Теперь покажем, какой код необходимо внести в файл для подключения xAjax:

```
<html>
<head>
<?php
```

```

// подключение библиотеки
require_once ("xajax_core/xajax.inc.php");
// создать новый xAjaх-объект
$xajax = new xajax();
// регистрация функций
$xajax->register(XAJAX_FUNCTION, "Function1");
// разрешение обрабатывать асинхронные xAjaх-запросы
$xajax->processRequest();
function Function1()
{
    $objResponse = new xajaxResponse();
    // код
    return $objResponse;
}
?>
<?php
    echo $xajax->getJavascript("");
?>
</head>
<body>

</body>
</html>

```

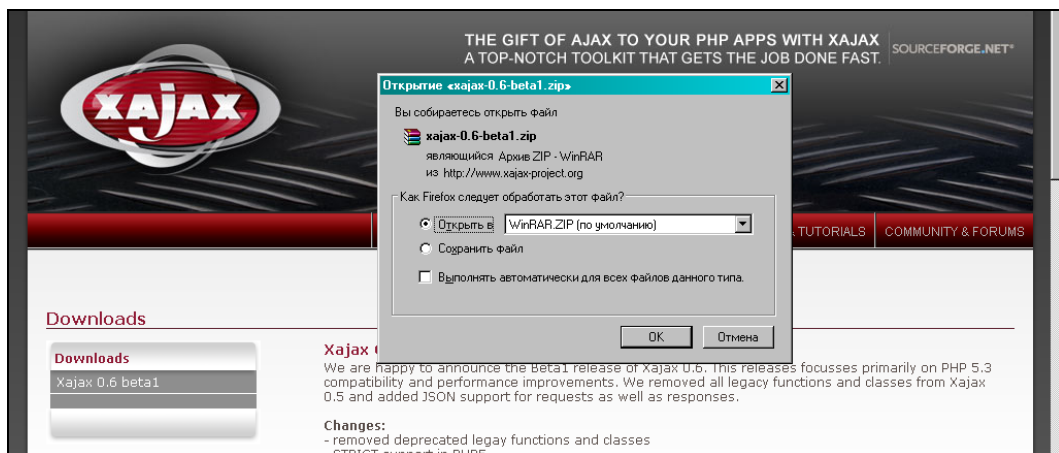


Рис. 1.1. Страница скачивания библиотеки xAjaх

Ясно видно, что xAjaх-объект — основной, в него регистрируются функции обработки, а xajaxResponse — вспомогательный, он генерирует XML-код, который потом распознается на уровне JavaScript и выполняет соответствующие действия. При вызове xAjaх-функций из JavaScript-кода добавляется префикс xajax_:

```
<a href=JavaScript:void();' onclick='xajax_Function1();'></a>
```


1.2.4. Методы объекта *ajaxResponse*

Объект *ajaxResponse* имеет следующие методы:

- | | |
|--|---|
| <input type="checkbox"/> <code>assign();</code> | <input type="checkbox"/> <code>insertInput();</code> |
| <input type="checkbox"/> <code>append();</code> | <input type="checkbox"/> <code>insertInputAfter();</code> |
| <input type="checkbox"/> <code>prepend();</code> | <input type="checkbox"/> <code>removeHandler();</code> |
| <input type="checkbox"/> <code>replace();</code> | <input type="checkbox"/> <code>includeScript();</code> |
| <input type="checkbox"/> <code>remove();</code> | <input type="checkbox"/> <code>script();</code> |
| <input type="checkbox"/> <code>create();</code> | <input type="checkbox"/> <code>addEvent();</code> |
| <input type="checkbox"/> <code>insert();</code> | <input type="checkbox"/> <code>call();</code> |
| <input type="checkbox"/> <code>insertafter();</code> | <input type="checkbox"/> <code>alert();</code> |
| <input type="checkbox"/> <code>clear();</code> | <input type="checkbox"/> <code>redirect();</code> |
| <input type="checkbox"/> <code>createInput();</code> | |

Рассмотрим подробнее эти методы.

ПРИМЕЧАНИЕ

На компакт-диске, прилагаемом к данной книге, в каталоге `glava_01\1` представлен Web-сайт — тренажер для изучения этих методов с просмотром результатов в визуальном режиме. Подробно работа тренажера рассмотрена в *разд. 1.2.5*.

Метод *assign()*

Изменяет параметры HTML-элементов, будь то `innerHTML`, `style` и т. п.

```
$objResponse->assign($sTarget, $sAttribute, $sData)
```

Заменяет значение `$sAttribute` элемента `$sTarget` на `$sData`.

Примеры:

```
// установить новое содержимое элемента с id=div1 — <b>New</b>
$objResponse->assign("div1", "innerHTML", "<b>New</b>");
// установить красный цвет текста в элементе с id=div1
$objResponse->assign("div1", "style.color", "red");
// скрыть элемент с id=div1
$objResponse->assign("div1", "style.display", "block");
```

Метод *append()*

Изменяет параметры элементов, добавляя данные в конец.

```
$objResponse->append($sTarget, $sAttribute, $sData)
```

Добавляет `$sData` в конец значения атрибута `$sAttribute` элемента `$sTarget`.

Примеры:

```
// добавить в конец содержимого элемента с id=div1 код HTML — <U>element</U>
$objResponse->append("div1", "innerHTML", "<U>element</U>");
// если HTML-код был <b>New</b>, то теперь будет <b>New</b><U>element</U>
```

Метод *prepend()*

Добавляет данные в начало.

```
$objResponse->prepend($sTarget, $sAttribute, $sData)
```

Добавляет `$sData` в начало значения атрибута `$sAttribute` элемента `$sTarget`.

Примеры:

```
// добавить в начало содержимого элемента с id=div1 код HTML - <U>element</U>
$objResponse->append("div1", "innerHTML", "<U>element</U>");
// если HTML-код был <b>New</b>, то теперь будет <U>element</U><b>New</b>
```

Метод *replace()*

Заменяет в элементе одни значения другими.

```
$objResponse->replace($sTarget, $sAttribute, $sSearch, $sData)
```

Заменяет найденное по маске `$sSearch` значение атрибута `$sAttribute` элемента `$sTarget` на `$sData`.

Примеры:

```
// заменить в содержимом элемента с id=div1 код HTML - 1 на код 2<br>
$objResponse->replace("div1", "innerHTML", "1", "2<br>");
// если HTML-код был <b>1234512345</b>, то теперь будет
// <b>2<br>23452<br>2345</b>
// напоминает PHP-функцию str_replace
```

Метод *remove()*

Удаляет элемент.

```
$objResponse->remove($sTarget)
```

Уничтожает элемент `$sTarget`.

Примеры:

```
// удалить элемент с id=div1
$objResponse->remove("div1");
```

Метод *create()*

Создает элемент.

```
$objResponse->create($sParent, $sTag, $sId, $sType = "")
```

Создает элемент `$sTag` с идентификатором `$sId`, как потомка от `$sParent` и типом `$sType`.

Примеры:

```
// создать элемент span с id=span1 в элементе с id=div1
$objResponse->create("div1", "span", "span1");
// если на странице был элемент div с id=div1: <div id=div1></div>,
// то теперь будет <div id=div1><span id=span1></span></div>
```

Метод *insert()*

Вставляет новый элемент.

```
$objResponse->insert($sBefore, $sTag, $sId)
```

Вставляет элемент `$sTag` с идентификатором `$sId` перед элементом `$sBefore`.

Примеры:

```
// создать элемент span с id=span1 перед элементом с id=div1
$objResponse->create("div2", "span", "span1");
// если на странице был div-элемент с id=div1,
// а внутри него div-элемент с id=div2
// <div id=div1><div id=div2></div></div>, то теперь будет
// <div id=div1><span id=span1></span><div id=div2></div></div>
```

Метод *insertAfter()*

Добавляет элемент после заданного элемента.

```
$objResponse->insertAfter($sAfter, $sTag, $sId)
```

Вставляет элемент `$sTag` с идентификатором `$sId` после элемента `$sAfter`.

Примеры:

```
// создать span-элемент с id=span1 после элемента с id=div1
$objResponse->create("div2", "span", "span1");
// если на странице был div-элемент с id=div1,
// а внутри него div-элемент с id=div2
// <div id=div1><div id=div2></div></div>, то теперь будет
// <div id=div1><div id=div2><span id=span1></span></div></div>
```

Метод *clear()*

Очищает содержимое элемента.

```
$objResponse->clear($sTarget, $sAttribute)
```

Очищает значение атрибута `$sAttribute` элемента `$sTarget`.

Примеры:

```
// очищает содержимое элемента с id=div1
$objResponse->clear("div2", "innerHTML");
// очищает содержимое свойства color элемента с id=div1
$objResponse->clear("div2", "color");
```

Метод *createInput()*

Создает элемент формы.

```
$objResponse->createInput($sParent, $sType, $sName, $sId)
```

Создает элемент HTML-формы как дочерний элемент от элемента `$sParent` с типом `$sType`, именем `$sName` и идентификатором, равным `$sId`.

Примеры:

```
// создает в форме с id=form1 элемент input
// с id=input2 и name=input2
$objResponse->createInput("form1","input","input2","input2");
```

Метод *insertInput()*

Создает элемент формы.

```
$objResponse->insertInput($sBefore, $sType, $sName, $sId)
```

Создает элемент HTML-формы перед элементом `$sBefore` с типом `$sType`, именем `$sName` и идентификатором, равным `$sId`.

Примеры:

```
// создает в форме input-элемент с id=input2 и name=input2
// перед элементом с id=input1
$objResponse->insertInput("input1","input","input2","input2");
```

Метод *insertInputAfter()*

Создает элемент формы.

```
$objResponse->insertInputAfter($sAfter, $sType, $sName, $sId)
```

Создает элемент HTML-формы после элемента `$sAfter` с типом `$sType`, именем `$sName` и идентификатором, равным `$sId`. Примеры:

```
// создает в форме input-элемент с id=input2 и name=input2
// после элемента с id=input1
$objResponse->insertInputAfter("input1","input","input2","input2");
```

Метод *removeHandler()*

Удаляет функцию обработки событий.

```
$objResponse->removeHandler($sTarget, $sEvent, $sHandler)
```

Удаляет js-функцию `$sHandler` для обработки события `$sEvent` элемента `$sTarget`.

Примеры:

```
// удаляет функцию fun_over для события onmouseover элемента с id=div1
$objResponse->removeHandler("div1","onmouseover","fun_over");
```

Метод *includeScript()*

Подключает внешний js-файл.

```
$objResponse->includeScript($sPath)
```

Подключает внешний js-файл, путь к которому задан в параметре `$sPath`.

Примеры:

```
// Подключаем внешний js-файл, чтобы первая страница
// загружалась не слишком долго.
// Внешний js-файл загружаем не в начале, а только тогда,
// когда он будет использоваться
$.objResponse->includeScript("js/jquery.fancybox-1.3.0.js");
```

Метод *script()*

Добавляет прописанную вручную js-обработку.

```
$.objResponse->script($sScript)
```

Выполняет JavaScript-код, содержащийся в \$sScript.

Примеры:

```
// выполнить js-код — выдать alert-окно с сообщением "Сообщение!!!"
$.objResponse->script("alert('Сообщение!!!')");
// элемент с id=d1 в зону видимости страницы
$.objResponse->script("document.getElementById('d1').scrollIntoView()");
// установить красный цвет текста в элементе с id=div1
// (аналогично $.objResponse->assign("d1","style.color","red");)
$.objResponse->script("document.getElementById('d1').style.color='red'");
```

Метод *addEvent()*

Создает новое событие.

```
$.objResponse->event($sTarget, $sEvent, $sScript)
```

Создает событие \$sEvent, привязывая его к элементу \$sTarget и связывая с ним код \$sScript.

Примеры:

```
// создаем новые события для элемента с id=div1:
// onmouseover — при наведении мыши на объект цвет элемента красный;
// onmouseout — при наведении мыши на объект цвет элемента синий
$.objResponse->event("div1","onmouseover","this.style.color='red'");
$.objResponse->event("div1","onmouseout","this.style.color='blue'");
```

Метод *call()*

Вызывает заданную js-функцию с указанными параметрами.

```
$.objResponse->call($sFunc, $args, ...)
```

Вызывает js-функцию \$sFunc с заданными параметрами \$args.

Примеры:

```
// вызывает js-функцию my_function("div1",4,10) с тремя аргументами
$.objResponse->call("my_function", "div1",4, 10))
```

Метод *alert()*

Создает окно-оповещение.

```
$objResponse->alert($sMsg)
```

Отображает предупреждение JavaScript с текстом, заданным параметром `$sMsg`.

Примеры:

```
// выдать alert-окно с сообщением "Сообщение!!!"  
$objResponse->script("alert('Сообщение!!!');");
```

Метод *redirect()*

Осуществляет перенаправление на другую страницу, возможно, через некоторое время.

```
$objResponse->redirect($sURL)
```

Перенаправляет на страницу `$sURL`.

Примеры:

```
// перенаправляет на другую страницу  
$objResponse->redirect("http://www.site.ru/register.php");
```

1.2.5. Сайт — тренировочный стенд для изучения xAjax

Для лучшего усвоения материала я создал небольшое веб-приложение для изучения Response-методов библиотеки xAjax. Сайт позволяет изучать асинхронно вызываемые функции xAjax (Response-методы): в режиме тренировочного стенда можно выбрать параметры и сразу же увидеть результат на странице. Сайт сделан с использованием библиотеки xAjax без перезагрузки страницы. Сайт находится на компакт-диске в каталоге `glava_01\1`. Данный пример доступен в Интернете по адресу http://examples-api.bazakatalogov.ru/glava_01/1. Вид главной страницы представлен на рис. 1.2.

Для удобства восприятия методы `xajaxResponse` объекта разбиты на 3 группы:

1. Работа с элементами:

- метод `assign()` изменяет параметры HTML-элементов, будь то `innerHTML`, `style` и т. п.;
- метод `append()` также изменяет параметры элементов, добавляя данные в конец;
- метод `prepend()` добавляет данные в начало;
- метод `replace()`, как вы догадались, заменяет в элементе одни значения другими, как функция `str_replace()`;
- метод `remove()` удаляет элемент;
- метод `create()` создает элемент;

Пособие по изучению xajax к книге

Xajax

xAJax – это php библиотека, которая отличается тем, что позволяет исполнять javascript на основе php-кода. Весь процесс состоит из двух php классов и обработчика xml на javascript. В общем – на php сначала инициализируется объект и объявляются функции, которые будут отвечать на ajax-запрос. В этих функциях необходимо использовать объект, который и будет генерировать xml-ответ.

```
require_once ("xajax_core/xajax.inc.php");
$xajax = new xajax("index.server.php");
$xajax->register(XAJAX_FUNCTION,"F1");
$xajax->processRequest();
function F1()
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("data","innerHTML",$content);
    return $objResponse;
}
```

Ясно видно что xajax объект основной, в него регистрируются функции обработки, а xajaxResponse вспомогательный, который генерирует XML, который потом распознаётся на уровне javascript и выполняет соответствующие действия.

Возможности – Класс xajaxResponse имеет следующие наиболее используемые методы:

- * **Работу с элементами:**
 - o **assign** изменяет параметры html-элементов, будь то innerHTML, style и тп.
 - o **append** также изменяет параметры элементов, добавляя в конец данные
 - o **prepend** добавляет данные в начало
 - o **replace** как вы догадались – заменяет в элементе одни на другие, как str_replace
 - o **remove** удаляет элемент
 - o **create** создаёт элемент
 - o **insert** вставляет новый элемент

Готово

Рис. 1.2. Страница сайта — тренировочного стенда

- МЕТОД insert() вставляет новый элемент;
- МЕТОД insertafter() добавляет элемент после заданного элемента;
- МЕТОД clear() очищает содержимое элемента.

2. Работа с input-полями:

- МЕТОД createInput();
- МЕТОД insertInput();
- МЕТОД insertInputAfter().

3. Особые процессы:

- МЕТОД removeHandler();
- МЕТОД includeScript() подключает внешний js-файл;
- МЕТОД script() добавляет прописанные вручную js-обработки;
- МЕТОД addEvent() создает новое событие;
- МЕТОД call() вызывает заданную js-функцию с указанными параметрами;
- МЕТОД alert() создает сообщение;
- МЕТОД redirect() осуществляет перенаправление на другую страницу, возможно, через некоторое время.

Как работать с сайтом? При нажатии на ссылку **Работа с элементами** видим страницу, изображенную на рис. 1.3.

Пособие по изучению AJAX к книге

Методы `jQueryResponse` для работы с элементами

- o Метод `assign()`
`jQueryResponse->assign($sTarget, $sAttribute, $sData);`
 - заменяет значение `$sAttribute` элемента `$sTarget` на `$sData`.
- o Метод `append()`
`jQueryResponse->append($sTarget, $sAttribute, $sData);`
 - добавляет `$sData` в конец значения атрибута `$sAttribute` элемента `$sTarget`
- o Метод `prepend()`
`jQueryResponse->prepend($sTarget, $sAttribute, $sData);`
 - добавляет `$sData` в начало значения атрибута `$sAttribute` элемента `$sTarget`.
- o Метод `replace()`
`jQueryResponse->replace($sTarget, $sAttribute, $sSearch, $sData);`
 - заменяет найденное по маске `$sSearch` значение атрибута `$sAttribute` элемента `$sTarget` на `$sData`.
- o Метод `remove()`
`jQueryResponse->remove($sTarget);`
 - удаляет элемент `$sTarget`
- o Метод `create()`
`jQueryResponse->create($sParent, $sTag, $sId, $sType = "");`
 - создает элемент `$sTag` с `id - $sId`, как потомка от `$sParent` и типом `$sType`.
 Метод удобно использовать для создания элементов форм
- o Метод `insert()`
`jQueryResponse->insert($sBefore, $sTag, $sId);`
 - вставляет элемент `$sTag` с `id - $sId` до элемента `$sBefore`.
- o Метод `insertAfter()`
`jQueryResponse->insertAfter($sAfter, $sTag, $sId);`
 - вставляет элемент `$sTag` с `id - $sId` после элемента `$sAfter`.
- o Метод `clear()`
`jQueryResponse->clear($sTarget, $sAttribute);`
 - очищает значение атрибута `$sAttribute` элемента `$sTarget`.

Рис. 1.3. Выбор методов для работы с элементами

Примеры `jQueryResponse->assign`

Выберите параметры команды и посмотрите результат выполнения
в окне Результат и код - в окне Код

`$sTarget`
`$sAttribute`
`$sData`

Результат

div1
span11
span13
div11
div2
span12
div3
div4
Готово

Рис. 1.4. Выбор параметров функции `assign()`

Далее выбираем какой-нибудь метод, например `assign()`, и попадаем на страницу, изображенную на рис. 1.4. Выбираем в форме значения атрибутов и нажимаем кнопку **Выполнить**.

Результат работы функции видим на рис. 1.5. Изменился фон элемента `span13`. В блоке **Код** видим код отправленной команды.

Теперь вернемся по ссылке **Назад** и выберем метод `remove()` (рис. 1.6). В раскрывающемся списке выберем элемент, например `span13`.

Рис. 1.5. Вид страницы после выполнения функции `assign()`

Рис. 1.6. Выбор параметров функции `remove()`

Нажимаем на кнопку **Выполнить** и видим результат выполнения функции (рис. 1.7). Элемент `span13` удален.