Тимур Машнин

bhv®

Google App Engine Java и Google Web Toolkit: разработка Web-приложений

РАЗРАБОТКА Web-ПРИЛОЖЕНИЙ ДЛЯ Google App Engine

ИСПОЛЬЗОВАНИЕ ПРОГРАММНОГО ИНТЕРФЕЙСА СЛУЖБ Google App Engine

СОЗДАНИЕ Ajax-ПРИЛОЖЕНИЙ НА ОСНОВЕ Google Web Toolkit

КОМПОНЕНТЫ GUI-ИНТЕРФЕЙСА Google Web Toolkit

РАБОТА GWT-ПРИЛОЖЕНИЯ НА СТОРОНЕ СЕРВЕРА

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Тимур Машнин

Google App Engine Java и Google Web Toolkit: разработка Web-приложений

Санкт-Петербург
«БХВ-Петербург»
2014

УДК 004.43:004.738.5
ББК 32.973.26-018.2
М38

Машнин Т. С.

М38 Google App Engine Java и Google Web Toolkit: разработка Web-приложений. — СПб.: БХВ-Петербург, 2014. — 352 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-0828-5

Книга посвящена разработке Web-приложений для платформы Google App Engine и на основе фреймворка Google Web Toolkit на языке программирования Java и с использованием среды разработки Eclipse. Рассмотрено создание проектов и запуск GWT-приложений и приложений для Google App Engine. Описано использование программного интерфейса служб платформы Google App Engine, создание GUI-интерфейса на основе фреймворка Google Web Toolkit, оптимизация и интернационализация GWT-приложения. Показано применение фреймворков UiBinder и Activities and Places для разработки клиентской части GWT-приложения, а также фреймворков GWT RPC и RequestFactory для разработки серверной части GWT-приложения.

Материал книги сопровождается большим количеством примеров с подробным анализом исходных кодов.

Для программистов

УДК 004.43:004.738.5
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 02.08.13.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 28,38.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-0828-5

Оглавление

Введение **7**
Платформа Google App Engine 7
Фреймворк Google Web Toolkit 9

ЧАСТЬ I. ПЛАТФОРМА GOOGLE APP ENGINE **11**

Глава 1. Начало работы с Google App Engine **13**
Установка инструментов разработки 13
Создание проекта приложения 18
Запуск приложения из среды Eclipse 23
Развертывание приложения на платформе App Engine 25
Регистрация приложения 25
Страница администрирования приложением 26
Загрузка приложения в App Engine 31
Создание пользовательских разделов консоли администрирования 32

Глава 2. Журналирование приложения **34**
Библиотека *java.util.logging* 34
Библиотека Log4j 37
LogService API 38

Глава 3. Определение местоположения пользователя **42**

Глава 4. Аутентификация пользователей **45**
Ограничения доступа к ресурсам в дескрипторе web.xml 45
Программный интерфейс Users API 48
Аутентификация с помощью Google Accounts 49
Аутентификация с помощью OpenID 50

Глава 5. Использование JSP/JSF-страниц в GAE-приложении **54**
Технология JSP 54
Использование JSTL 57
Технология JSF 60

Глава 6. Хранение данных приложения 65
App Engine Datastore 65
Datastore API 67
Служба Remote API 80
JDO и JPA 82
JDO 82
JPA 91
Objectify 100
Twig 108
Slim3 116
Google Cloud SQL 123
Google Cloud Storage и Blobstore 126
Google Cloud Storage 126
Blobstore 128
Сервис изображений 132
Служба Memcache 134

Глава 7. Поддержка сессий и HTTPS 137
Поддержка протокола HTTP/SSL 137
Использование сессий и cookie 138

Глава 8. Сервисы сообщений Mail, XMPP и Channel 140
Служба Mail 140
Отправка сообщений электронной почты 140
Получение сообщений электронной почты 141
Пример использования службы Mail 143
Служба XMPP 146
Отправка мгновенных сообщений 147
Получение мгновенных сообщений 147
Пример использования службы XMPP 150
Служба Channel 153

Глава 9. Фильтры и обработка ошибок 159
Фильтрация запросов и ответов 159
Обработка ошибок 161

Глава 10. Разработка Backend-приложений 164

Глава 11. Использование протокола OAuth 2.0 для получения доступа к Google-сервисам 168
Служба URL Fetch 172

Глава 12. Запланированные задачи и очередь задач 175
Служба Cron 175
Служба Task Queue 177
Очереди Push 177
Отложенные задачи *DeferredTask* 180
Очереди Pull 181

Глава 13. Службы поиска Search и Prospective Search 183
Служба Search 183
Служба Prospective Search 187

ЧАСТЬ II. ФРЕЙМВОРК GOOGLE WEB TOOLKIT 193

Глава 14. Начало работы с Google Web Toolkit 195
Установка плагинов фреймворка GWT 196
Создание проекта GWT-приложения 197
Структура проекта GWT-приложения 197
GWT-модули 198
Конфигурационный XML-файл определения GWT-модуля 200
Модель программирования фреймворка GWT 203
Запуск GWT-приложения в режиме разработки 208
Запуск GWT-приложения как Web-приложения 210

Глава 15. Компоненты графического интерфейса пользователя 214
Кнопка *Button* 215
Отличие метода *setText()* от метода *setHTML()* 215
Обработчики событий кнопки 216
Определение свойств кнопки 222
Кнопка *PushButton* 222
Переключатель *RadioButton* 224
Флажок *CheckBox* 225
Компонент выбора даты *DatePicker* 226
Кнопка *ToggleButton* 229
Текстовое поле *TextBox* 231
Поле ввода пароля *PasswordTextBox* 232
Текстовая область *TextArea* 233
Гиперссылка *Hyperlink* 234
Гиперссылка *Anchor* 236
Список выбора *ListBox* 237
Компоненты *Cell Widgets* 238
Столбец *CellList* 239
Таблица *CellTable* 249
Таблица *DataGrid* 255
Дерево *CellTree* 257
Дерево *CellBrowser* 259
Панель меню *MenuBar* 260
Дерево *Tree* 263
Поле подсказки *SuggestBox* 265
Редактор текста *RichTextArea* 267
Таблица *FlexTable* 269
Таблица *Grid* 271
Всплывающие окна *PopupPanel*, *DecoratedPopupPanel*, *LoggingPopup* и *DialogBox* 273
Уведомление *NotificationMole* 276
Панели с закладками *TabPanel* и *TabLayoutPanel* 276
Загрузчик файлов *FileUpload* и форма *FormPanel* 279
Скрытое поле *Hidden* 280

Фрейм *Frame* 281
Изображение *Image* 281
Метка *Label* 283
Метка *HTML* 284
Метка *InlineHTML* 284
Метка *InlineLabel* 284
Медиакомпоненты *Audio* и *Video* 284
Компонент *Video* 284
Компонент *Audio* 285
Компонент *Canvas* 286
Панели компоновки 287
Панели *AbsolutePanel*, *RootPanel*, *RootLayoutPanel* и *LayoutPanel* 287
Панели *StackPanel*, *DecoratedStackPanel* и *StackLayoutPanel* 290
Панели *DockPanel*, *DockLayoutPanel* и *SplitLayoutPanel* 292
Панели *HorizontalPanel* и *VerticalPanel* 294
Панель *FlowPanel* 295
Панель *HTMLPanel* 295
Панель *HeaderPanel* 296
Панели *SimplePanel*, *DecoratorPanel*, *FocusPanel* и *SimpleLayoutPanel* 296
Панель *ScrollPanel* 298
Панель *DisclosurePanel* 299
Панель *CaptionPanel* 299
Изменение внешнего вида GWT-компонентов 300
Фреймворк UiBinder 304

Глава 16. Интернационализация GWT-приложения 307
Статическая интернационализация 308
Динамическая интернационализация 312
Интернационализация UiBinder 312

Глава 17. Программный интерфейс JavaScript Native Interface 314

Глава 18. Оптимизация GWT-приложения 316
Разделение кода (Code Splitting) 316
Отложенное связывание (Deferred Binding) 318

Глава 19. Поддержка истории Web-браузера 321

Глава 20. Фреймворк Activities and Places 324

Глава 21. Взаимодействие GWT-приложения с сервером 330
Фреймворк GWT RPC 330
Библиотека GWT HTTP Client 335
Фреймворк RequestFactory 341

Литература 349

Предметный указатель 350

Введение

Корпорация Google среди своих многочисленных Web-продуктов, включающих в себя поисковые и рекламные сервисы, сервисы Gmail и Google Docs, различные инструменты коммуникации, публикации, разработки, анализа и картографии, предлагает платформу Google App Engine и фреймворк Google Web Toolkit.

Платформа Google App Engine

Google App Engine (GAE) является PaaS-платформой (Platform as a Service, платформа как сервис) и представляет собой платформу облачного хостинга Web-приложений в инфраструктуре Google, содержащей центры данных и серверы. Платформа GAE избавляет от необходимости самостоятельной поддержки инфраструктуры развернутого сайта и заботы о его масштабировании. При этом платформа GAE сама регулирует использование серверов и центров данных инфраструктуры Google для обеспечения работы сайта. Развернутое Web-приложение может использовать бесплатно 500 Мбайт дискового пространства и бесплатно обслуживать около 5 млн запросов в месяц (квоты использования ресурсов можно посмотреть на странице по адресу **https://developers.google.com/appengine/docs/quotas**). Необходимость сайта в увеличении потребляемых ресурсов удовлетворяется автоматически за отдельную плату, при этом оплата производится только за фактически используемые дополнительные ресурсы. Таким образом, при росте проекта отпадает необходимость в аренде/покупке серверов и заказе VDS-серверов.

Для работы Web-приложений платформа GAE обеспечивает среды выполнения Java, Python и Go.

Платформа GAE создает для выполнения Web-приложений "песочницы", которые накладывают ограничения на работу приложений.

- GAE-приложение может обмениваться данными с другими приложениями только с помощью HTTP/HTTPS-запросов и сообщений электронной почты. Поддержку HTTP/HTTPS-запросов обмена данными с другими серверами в Интернете обеспечивает служба GAE-платформы URL Fetch, а обмен сообщениями электронной почты обеспечивает GAE-служба Mail.

- GAE-приложение может взаимодействовать с дисковым пространством хостинга только путем чтения из тех ресурсных файлов, которые были загружены вместе с приложением. Для хранения и работы с данными GAE-приложение может использовать GAE-службы Blobstore, Images, Datastore и Memcache. Служба Blobstore обеспечивает работу приложения с большими двоичными объектами, размером до 2 Гбайт. Служба Images обеспечивает преобразования изображений, размером до 1 Мбайт, включая изменение размера, поворот, отражение, кадрирование и коррекцию контрастности и цветопередачи. Служба Datastore обеспечивает хранение данных приложения в двух типах хранилищ — High Replication Datastore (HRD) (по умолчанию) и Master/Slave Datastore (в настоящее время не рекомендуется). Выбор хранилища производится при регистрации приложения в GAE. Хранилище HRD — это хранилище с высоким уровнем репликации данных по нескольким центрам данных, а хранилище Master/Slave обеспечивает репликацию данных по системе "главное/подчиненное хранилище" и является менее затратным по сравнению с хранилищем HRD, которое, в свою очередь, обеспечивает большую доступность к данным. Служба Memcache обеспечивает хранение данных приложения объемом до 1 Мбайт в кэше памяти.
- При обработке клиентского запроса GAE-приложение не может создавать фоновые процессы и должно обрабатывать запрос в течение определенного времени — в настоящее время это 30 секунд. Если запрос не обрабатывается в течение этого времени, то возникает исключение. Размер запроса и ответа не должен превышать 10 Мбайт.

Кроме того, накладываются ограничения на размер отдельного файла приложения — до 10 Мбайт, на общий размер всех файлов приложения — до 150 Мбайт, и на общее количество файлов приложения — до 3000.

Помимо служб URL Fetch, Mail, Blobstore, Images, Datastore и Memcache платформа GAE предоставляет в пользование приложению службы Namespaces, OAuth, Task Queue, Cron, Users и XMPP.

Служба Namespaces позволяет разделить клиентов GAE-приложения по пространствам имен. Служба OAuth дает возможность клиенту предоставить доступ одного приложения к другому от своего имени. Служба Task Queue обеспечивает для приложения создание фоновых задач, не обрабатывающих клиентский запрос, и постановку их в очередь выполнения. Служба Cron позволяет создавать приложению запланированные задачи, выполняемые в конкретное время или регулярно через определенные периоды. Служба Users помогает приложению организовать аутентификацию/авторизацию клиента. Служба XMPP позволяет приложению организовать сервис мгновенных сообщений.

Кроме GAE-службы Datastore, обеспечивающей хранение данных приложения в нереляционной базе данных, организовать хранение данных GAE-приложения позволяют такие Google-сервисы, как Google Cloud SQL — хранение данных в реляционной базе данных, основанной на MySQL, и Google Cloud Storage — хранение больших объектов, размером до терабайта.

Среда выполнения Java платформы GAE включает в себя виртуальную машину JVM и Servlet-контейнер. Применение виртуальной машины JVM обеспечивает

создание GAE-приложений с использованием JVM-языков JavaScript, Ruby, Scala и др.

Использование классов платформы Java для GAE-приложений ограничено белым списком классов среды выполнения Java (**https://developers.google.com/appengine/docs/java/jrewhitelist?hl=ru-RU**).

Следуя ограничениям "песочницы" платформы GAE, Java-приложение не может создавать экземпляры классов `java.lang.ThreadGroup`, `java.lang.Thread`, `java.util.concurrent.ThreadPoolExecutor`, `java.util.Timer`, использовать классы записи библиотеки `java.io`. Методы `exit()`, `gc()`, `runFinalization()`, `runFinalizersOnExit()`, `inheritedChannel()`, `console()`, `load()`, `loadLibrary()`, `setSecurityManager()` класса `java.lang.System` не работают.

Среда выполнения Python платформы GAE содержит стандартную библиотеку Python, функции которой могут использоваться с учетом ограничений "песочницы", а также библиотеки Django, WebOb и PyYAML. В приложение могут добавляться и другие библиотеки, созданные на чистом языке Python.

Регистрация приложения в App Engine осуществляется с помощью Web-страницы консоли администратора по адресу **https://appengine.google.com/**, при этом на один аккаунт разработчика можно зарегистрировать до 10 приложений. В процессе регистрации для приложения создается поддомен домена **appspot.com** и формируется URL-адрес доступа к приложению в виде **http://[*идентификатор приложения*].appspot.com/**. После регистрации приложение можно привязать к домену, зарегистрированному в бизнес-приложении Google Apps (**http://www.google.com/apps/intl/ru/business/**). Само добавление приложения в App Engine после его регистрации осуществляется с помощью инструментов разработки GAE-приложений.

Для разработки приложений GAE-платформа предоставляет наборы Google App Engine SDK для Java, Python и Go, обеспечивающие эмуляцию соответствующих сред выполнения, работу GAE-служб и инструменты загрузки приложений в GAE-платформу. Для разработки Java-приложений предоставляется также Google-модуль для среды разработки Eclipse.

Фреймворк Google Web Toolkit

Google Web Toolkit (GWT) обеспечивает создание RIA (Rich Internet Application) Ajax-приложений на основе Java-кода.

GWT-приложение разрабатывается на языке Java, и его код на стадии разработки содержит определение GUI-интерфейса, обработку событий интерфейса и работу с данными. Затем Java-код GWT-приложения, содержащий определение GUI-интерфейса и обработку его событий, компилируется в JavaScript-код Web-страницы клиента, а на стороне сервера остается лишь Java-код Web-сервисов, отвечающих за работу с данными. При этом JavaScript-код Web-страницы содержит Ajax-клиентов Web-сервисов.

GWT-компилятор Java-to-JavaScript создает на выходе набор версий приложения, обеспечивающий совместимость с различными Web-браузерами и интернационали-

зацию приложения. В результате запускаемый на стороне клиента код является компактным и быстро загружается, т. к. содержит только те ресурсы, которые требуются конкретным Web-браузером. Кроме того, для ускорения запуска приложения можно создавать точки разделения JavaScript-кода, в которых приложение будет производить Ajax-загрузку дополнительного кода.

Программный интерфейс JavaScript Native Interface (JSNI) GWT-платформы позволяет также при разработке встраивать напрямую JavaScript-код в Java-код приложения, что обеспечивает использование сторонних JavaScript-библиотек.

GWT-фреймворк обеспечивает поддержку HTML5-тегов `<canvas>`, `<audio>` и `<video>`.

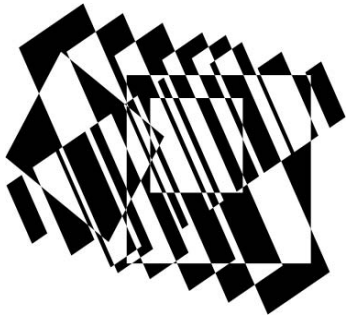
Фреймворк Google Web Toolkit (GWT) содержит:

- GWT SDK — включает в себя Java-библиотеки программного интерфейса GWT-платформы, GWT-компилятор Java-кода в JavaScript-код, локальный сервер разработки, позволяющий запускать и отлаживать Java-код приложения без его компиляции в JavaScript-код;
- Speed Tracer — расширение Web-браузера Chrome, позволяющее анализировать производительность GWT-приложения;
- Google Plugin for Eclipse (GPE) — плагин, обеспечивающий разработку GWT-приложений в среде Eclipse;
- GWT Designer — плагин, обеспечивающий визуальное редактирование GUI-интерфейса GWT-приложения в среде Eclipse.

Для создания GUI-интерфейса приложения GWT-фреймворк предлагает набор готовых панелей и виджетов, из которых можно компоновать клиентскую часть приложения.

Серверную часть приложения, содержащую Web-сервисы данных для клиентской части, можно создавать с помощью библиотек GWT-RPC и RequestFactory GWT-фреймворка.

В данной книге рассматривается создание Java-приложений для развертывания на платформе Google App Engine и создание приложений на основе фреймворка Google Web Toolkit.



ЧАСТЬ I

Платформа Google App Engine

Глава 1. Начало работы с Google App Engine

Глава 2. Журналирование приложения

Глава 3. Определение местоположения пользователя

Глава 4. Аутентификация пользователей

Глава 5. Использование JSP/JSF-страниц в GAE-приложении

Глава 6. Хранение данных приложения

Глава 7. Поддержка сессий и HTTPS

Глава 8. Сервисы сообщений Mail, XMPP и Channel

Глава 9. Фильтры и обработка ошибок

Глава 10. Разработка Backend-приложений

Глава 11. Использование протокола OAuth 2.0 для получения доступа к Google-сервисам

Глава 12. Запланированные задачи и очередь задач

Глава 13. Службы поиска Search и Prospective Search

ГЛАВА 1

Начало работы с Google App Engine

Установка инструментов разработки

Разработка на основе платформы Google App Engine (GAE) подразумевает создание Web-приложений. Для разработки Web-приложений на языке Java платформа GAE предлагает использовать набор Eclipse-плагинов, облегчающих организацию проекта, тестирование и развертывание приложения. Поэтому в качестве первого шага скачаем по адресу **http://www.eclipse.org/downloads/** последнюю версию среды Eclipse IDE for Java EE Developers, обеспечивающую все необходимые инструменты разработки Java Web-приложений. При этом предполагается, что набор JDK платформы Java уже установлен на компьютере (**http://www.oracle.com/technetwork/java/javase/downloads/index.html**).

> ***Примечание***
> При написании книги использовался релиз Indigo SR2 среды Eclipse.

После распаковки скачанного дистрибутива запустим среду Eclipse, закроем страницу приветствия **Welcome** кнопкой **Workbench** и в меню **Help** выберем команду **Install New Software** — в результате откроется окно мастера инсталляции Eclipse-плагинов (рис. 1.1).

Нажмем кнопку **Add** и в поле **Name** введем **Google**, а в поле **Location** — адрес установки Google-модуля, для Eclipse Indigo **http://dl.google.com/eclipse/plugin/3.7** (рис. 1.2).

Нажмем кнопку **OK**. После поиска в Интернете в окне мастера отобразится набор плагинов Google-модуля (рис. 1.3).

Плагин Google App Engine Tools for Android обеспечивает создание клиент-серверных приложений для платформы Android, в которых клиентская часть размещается в мобильном Android-устройстве, а серверная часть — в облаке GAE-платформы. Данный плагин требует предварительной установки плагина Android Development Tools (ADT) для разработки Android-приложений в среде Eclipse.

Плагин Google Plugin for Eclipse обеспечивает поддержку проектов GAE- и GWT-приложений в среде Eclipse.

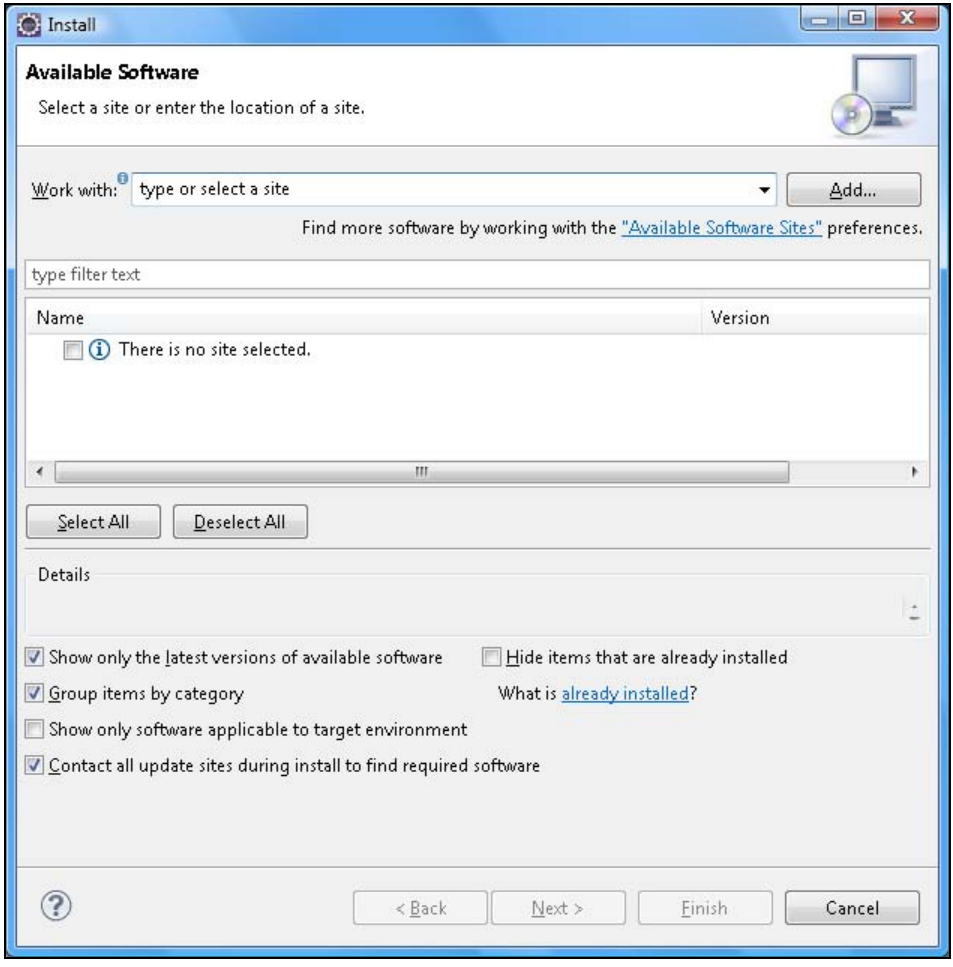


Рис. 1.1. Мастер установки Eclipse-плагинов

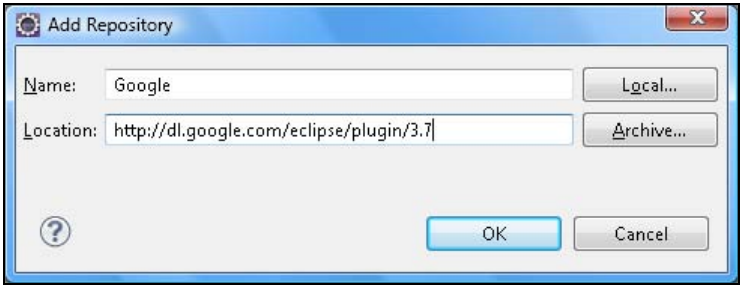


Рис. 1.2. Определение адреса установки Google-модуля

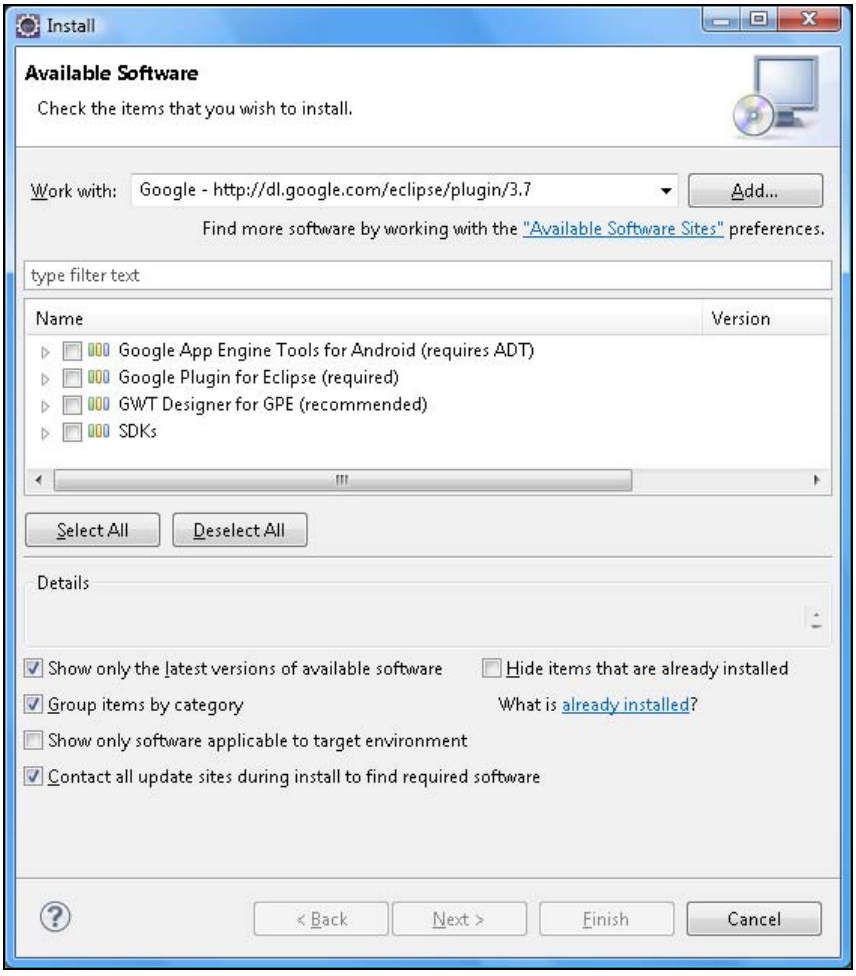


Рис. 1.3. Набор плагинов Google-модуля для среды Eclipse

Плагин GWT Designer for GPE предоставляет средства визуального редактирования GUI-интерфейса GWT-приложений.

Плагин SDKs обеспечивает установку наборов Google App Engine Java SDK и GWT SDK, необходимых для разработки GAE- и GWT-приложений в среде Eclipse.

Для начала отметим флажки **Google Plugin for Eclipse** и **Google App Engine Java SDK**, нажмем кнопку **Next** мастера и, следуя инструкциям, установим выбранные плагины с перезапуском среды Eclipse.

В результате установки плагина Google Plugin for Eclipse (GPE) в Workbench-окне среды Eclipse появятся кнопки **Google Services and Development Tools**  и 

, а в диалоговом окне команды **New | Other** меню **File** появятся разделы **Google** и **Google Web Toolkit** (рис. 1.4).

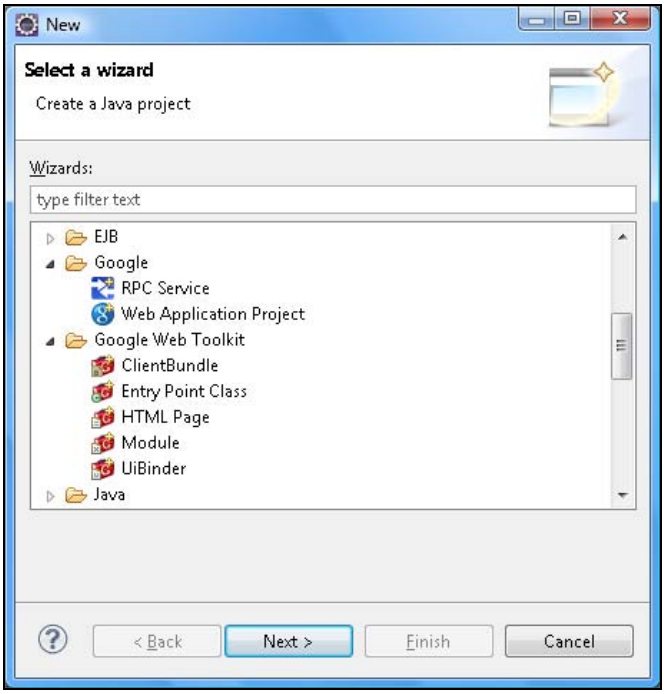


Рис. 1.4. Разделы **Google** и **Google Web Toolkit** окна команды **New | Other** меню **File**

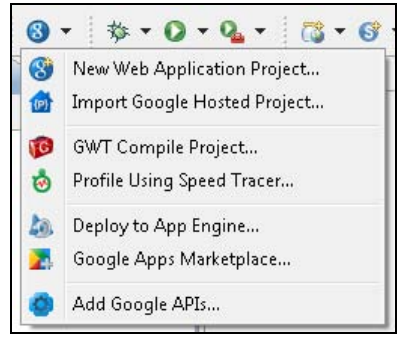


Рис. 1.5. Команды кнопки **Google Services and Development Tools**

Кнопка **Sign in to Google** позволяет пройти аутентификацию/авторизацию для доступа к таким Google-сервисам из среды Eclipse, как Project Hosting, Google SQL Service, Google Apps Marketplace и Google App Engine.

Сервис Project Hosting (**http://code.google.com/hosting/**) предоставляет возможность размещения открытых проектов в удаленном VCS-репозитории (Version Control System) для организации командной работы над ними. Данный сервис поддерживает такие системы контроля версий, как Subversion, Mercurial и Git.

Сервис Google SQL Service представляет собой Web-сервис, обеспечивающий создание, администрирование и использование MySQL базы данных для GAE-приложения.

Сервис Google Apps Marketplace (**http://www.google.com/enterprise/marketplace/**) является магазином Web-приложений, которые интегрируются с бизнес-приложением Google Apps (**http://www.google.com/enterprise/apps/business/**).

Раскрытая кнопка **Google Services and Development Tools** показывает набор команд (рис. 1.5):

- **New Web Application Project** — запускает мастер создания проекта приложения платформ GAE и GWT;
- **Import Google Hosted Project** — позволяет импортировать проект, размещенный на хостинге сервиса Project Hosting, в Workspace-пространство среды Eclipse;

- **GWT Compile Project** — запускает мастер компиляции Java-кода проекта GWT-приложения в JavaScript-код;
- **Profile Using Speed Tracer** — обеспечивает анализ производительности приложения с помощью инструмента Speed Tracer — расширения Web-браузера Google Chrome;
- **Deploy to App Engine** — запускает мастер развертывания приложения на GAE-платформе;
- **Google Apps Marketplace** — обеспечивает загрузку приложения в магазин Google Apps Marketplace;
- **Add Google APIs** — позволяет добавить в путь приложения программный интерфейс API различных Google-сервисов.

Раздел **Google** команды **New | Other** меню **File** среды Eclipse содержит мастера:

- **RPC Service** — на основе Entity-компонентов создает для GWT-приложения RequestFactory-сервис данных;
- **Web Application Project** — создает проекты GAE- и GWT-приложений.

Раздел **Google Web Toolkit** команды **New | Other** меню **File** среды Eclipse содержит мастера:

- **ClientBundle** — создает для GWT-приложения ClientBundle-интерфейс, группирующий ресурсы приложения, такие как изображения, CSS-стили, текстовые файлы и др., в единый модуль, что ускоряет загрузку и запуск клиентской части приложения;
- **Entry Point Class** — создает для GWT-приложения EntryPoint-класс — точку входа в GWT-модуль;
- **HTML Page** — создает для GWT-приложения HTML-страницу, обеспечивающую загрузку клиентской части приложения;
- **Module** — создает для GWT-приложения новый GWT-модуль;
- **UiBinder** — создает для GWT-приложения UiBinder-шаблон, декларативно определяющий GUI-интерфейс приложения.

Для использования в GAE-проекте страниц Java Server Pages (JSP) требуется Java-компилятор javac.exe, содержащийся в наборе Java SE Development Kit (JDK). По умолчанию среда Eclipse использует не набор JDK, а среду выполнения JRE, поэтому после установки плагинов GPE и GAE Java SDK, перед созданием GAE-проекта, в меню **Windows** среды Eclipse выберем команду **Preferences**, в диалоговом окне которой откроем раздел **Java | Installed JREs** и кнопкой **Add** добавим предварительно инсталлированный набор JDK (**http://www.oracle.com/technetwork/java/javase/downloads/index.html**) (рис. 1.6).

В настоящее время GAE-платформа поддерживает версии Java 5 и 6, поэтому необходимо проверить, чтобы уровень компиляции, установленный в разделе **Java | Compiler** диалогового окна команды **Preferences**, а также версия Java раздела **Project Facets** диалогового окна свойств GAE-проекта соответствовали версии Java раздела **Java | Installed JREs**.

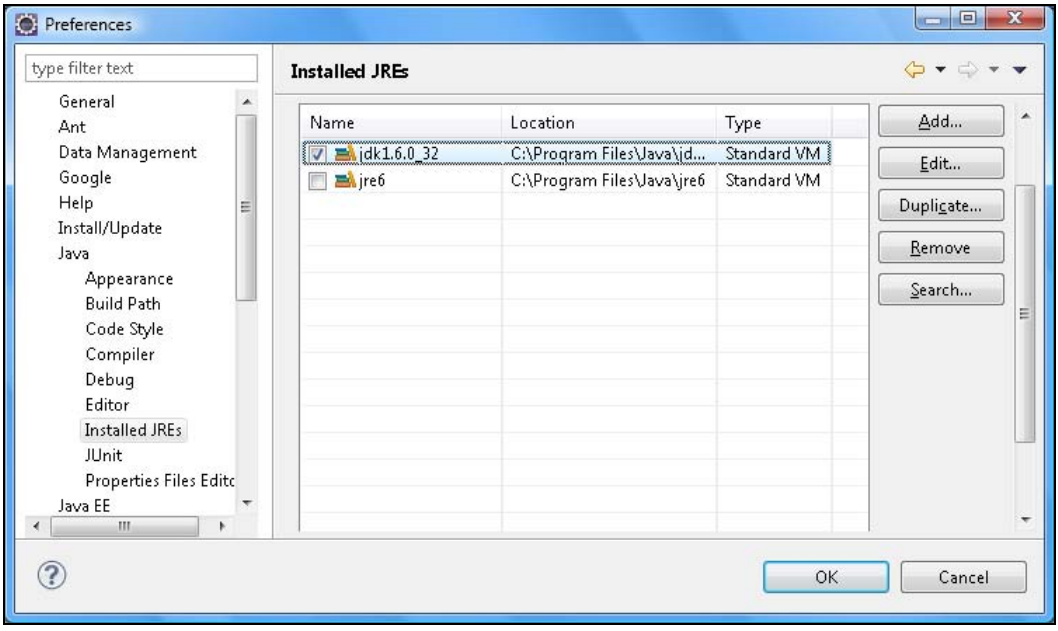


Рис. 1.6. Установка использования JDK по умолчанию

Создание проекта приложения

Теперь все готово для создания проекта GAE-приложения. В меню **File** среды Eclipse выберем команду **New | Other | Google | Web Application Project** и нажмем кнопку **Next**. В окне мастера создания проекта, в поле **Project name** введем имя проекта, в поле **Package** — имя пакета проекта, отметим флажок **Use Google App Engine** и нажмем кнопку **Finish** (рис. 1.7).

В результате в Workspace-пространстве среды Eclipse будет сгенерирована основа проекта GAE-приложения.

В путь GAE-проекта будут добавлены библиотеки среды JRE и набора GAE Java SDK для локального запуска GAE-приложения, предоставляющего Servlet-контейнер Web-сервера Jetty и обеспечивающего эмуляцию работы GAE-служб.

Папка src сгенерированного GAE-проекта содержит пакет с классом сервлета, который расширяет класс `javax.servlet.http.HttpServlet` и отвечает за обработку HTTP-запросов к приложению (листинг 1.1). Сгенерированный Servlet-класс переопределяет метод `doGet()` класса `HttpServlet`, обрабатывающий HTTP GET-запрос, в котором в ответ на запрос записывается строка "Hello, world". Для вызова данного сервлета его класс указан в дескрипторе web.xml развертывания приложения с помощью тега `<servlet-class>`, при этом тег `<url-pattern>` указывает относительный URL-адрес вызова сервлета. Дескриптор web.xml располагается в папке war\WEB-INF GAE-проекта. Тег `<welcome-file>` дескриптора web.xml определяет страницу приветствия приложения.

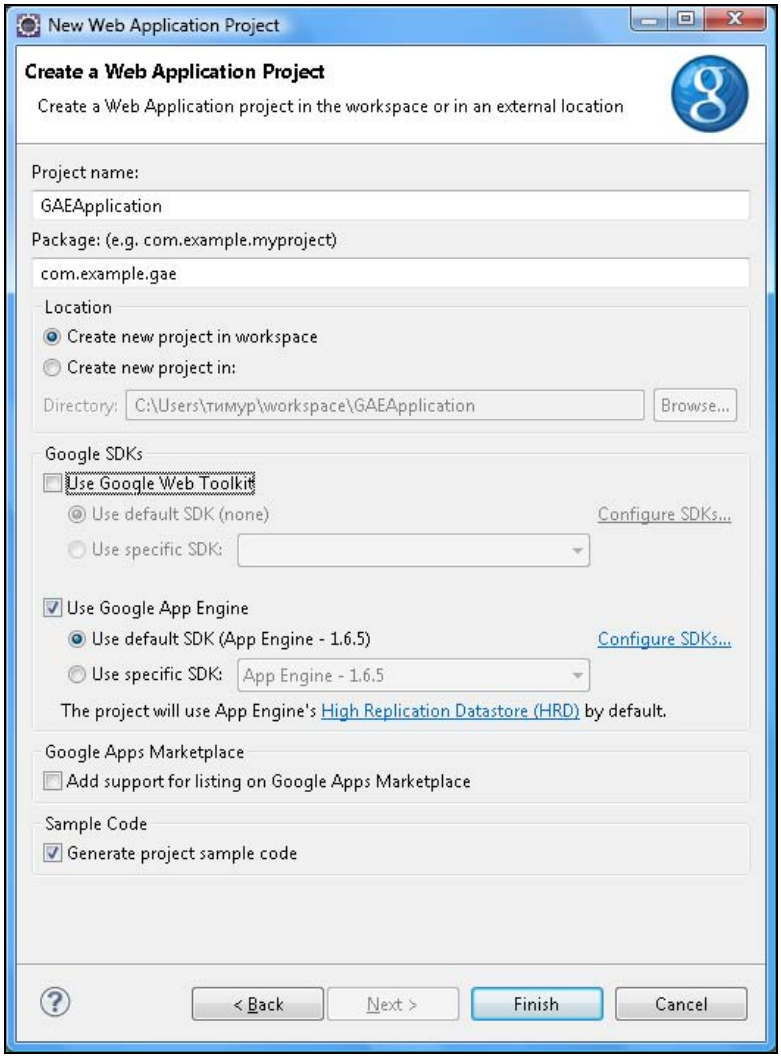


Рис. 1.7. GPE-мастер создания проекта

Листинг 1.1. Код Servlet-класса GAE-приложения

```
package com.example.gaeapplication;
import java.io.IOException;
import javax.servlet.http.*;
@SuppressWarnings("serial")
public class GAEApplicationServlet extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse resp)
      throws IOException {
    resp.setContentType("text/plain");
    resp.getWriter().println("Hello, world");
}}
```

Сгенерированная страница index.html приветствия приложения каталога war GAE-проекта содержит ссылку на относительный URL-адрес вызова сервлета, указанный в теге <url-pattern> дескриптора web.xml (листинг 1.2).

Листинг 1.2. Код страницы index.html приветствия GAE-приложения

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Hello App Engine</title>
</head>
<body>
  <h1>Hello App Engine!</h1>
  <table>
    <tr>
    <td colspan="2" style="font-weight:bold;">Available Servlets:</td>
    </tr>
    <tr>
      <td><a href="gaeapplication">GAEApplication</a></td>
    </tr>
  </table>
</body>
</html>
```

Папка src\META-INF содержит конфигурационный файл jdoconfig.xml, определяющий для приложения конфигурацию механизма Java Data Objects (JDO) сохранения объектов в базе данных.

Каталог war GAE-проекта содержит все файлы, необходимые для развертывания приложения в Servlet-контейнере, поэтому в папку war\WEB-INF\lib включены библиотеки программного интерфейса API GAE-служб.

Конфигурационный файл appengine-web.xml папки war\WEB-INF GAE-проекта содержит настройки развертывания приложения на GAE-платформе. Файл appengine-web.xml указывает зарегистрированный GAE-идентификатор приложения, версию приложения, а также его список статических и ресурсных файлов и другие настройки работы приложения.

Изначально сгенерированный файл appengine-web.xml в теге <application> содержит пустой GAE-идентификатор, и для загрузки приложения в облако платформы GAE необходимо заполнить тег <application> GAE-идентификатором, полученным в результате регистрации приложения с помощью GAE-консоли **https://appengine.google.com/**.

Версия приложения указывается в теге <version> файла appengine-web.xml и необходима для создания новой версии приложения в облаке GAE-платформы или для замены приложения с тем же номером версии.

Для повышения эффективности GAE-платформа обслуживает отдельно статические файлы приложения, такие как изображения, CSS-стили, JavaScript-файлы

и др., которые могут обслуживаться простым HTTP-сервером, а не сервером приложений, вызывающим сервлеты и обрабатывающим JSP-страницы.

По умолчанию все файлы каталога war приложения, за исключением JSP-файлов и файлов папки WEB-INF, рассматриваются GAE-платформой как статические файлы. Отдельно указать статические файлы можно в файле appengine-web.xml с помощью тега:

```
<static-files>
  <include path="" expiration="" />
  <exclude path="" />
</static-files>
```

Здесь тег `<include>` указывает путь для включения файлов в список статических файлов, а тег `<exclude>` — путь для исключения файлов из списка статических файлов приложения. Атрибут `expiration` указывает время кэширования браузером статических файлов (по умолчанию 10 мин) в формате `d` (дни) `h` (часы) `m` (минуты) `s` (секунды).

Тег `<public-root>` файла appengine-web.xml позволяет переопределить корневой каталог для статических файлов приложения.

Все остальные файлы WAR-каталога приложения считаются ресурсными файлами и обслуживаются сервером приложений. Уточнить список ресурсных файлов приложения можно в файле appengine-web.xml с помощью тега:

```
<resource-files>
  <include path="" />
  <exclude path="" />
</resource-files>
```

Здесь тег `<include>` указывает путь для включения файлов в список ресурсных файлов, а тег `<exclude>` — путь для исключения файлов из списка ресурсных файлов приложения.

Указать кодировку приложения по умолчанию можно в файле appengine-web.xml с помощью тега:

```
<env-variables>
  <env-var name="DEFAULT_ENCODING" value="" />
</env-variables>
```

Файл appengine-web.xml может содержать также следующие теги.

- `<system-properties>` — посредством дочерних тегов `<property name="" value=""/>` определяет системные свойства, которые могут быть доступны из Java-кода с помощью объекта `java.util.Properties`, получаемого методом `System.getProperties()`. Сгенерированный файл appengine-web.xml содержит свойство `java.util.logging.config.file` со значением WEB-INF/logging.properties, указывающим расположение конфигурационного файла для модуля ведения журнала java.utils.logging. В конфигурационном файле logging.properties установлен уровень ведения журнала по умолчанию — `.level = WARNING`. Просматривать журнал приложения можно в консоли администрирования **https://**

appengine.google.com/ в разделе **Main | Logs**. Выводить свои сообщения в журнал можно с помощью Java-кода:

```
private static final Logger log = Logger.getLogger([имя класса].class.getName());
log.info("");
log.warning("");
log.severe("");
```

Кроме того, GAE-платформа сама устанавливает ряд системных свойств при запуске приложения и JVM: `com.google.appengine.runtime.environment`, `com.google.appengine.runtime.version`, `com.google.appengine.application.id`, `file.separator`, `path.separator`, `line.separator`, `java.version`, `java.vendor`, `java.vendor.url`, `java.class.version`, `java.specification.version`, `java.specification.vendor`, `java.specification.name`, `java.vm.vendor`, `java.vm.name`, `java.vm.specification.version`, `java.vm.specification.vendor`, `java.vm.specification.name`, `user.dir`.

- `<ssl-enabled>` — с помощью значения `false` отключает использование для приложения протокола HTTPS.
- `<sessions-enabled>` — с помощью значения `true` включает использование сессий. При этом с помощью тега `<async-session-persistence enabled="true"/>` можно определить асинхронную запись данных сессии в хранилище.
- `<inbound-services>` — посредством дочерних тегов `<service>[служба]</service>` включает использование GAE-служб:
  - `channel_presence` — приложение получает уведомления о присоединении или отсоединении клиента от Channel-канала;
  - `mail` — приложение может получать электронные письма;
  - `xmpp_message` — приложение может получать мгновенные сообщения. При этом включение службы xmpp_presence позволяет приложению получать уведомления об изменении статуса клиента в службе XMPP, включение службы xmpp_subscribe — получать уведомления об изменении статуса подписки на сообщения между клиентом и приложением, включение службы xmpp_error позволяет приложению получать сообщения об ошибках XMPP-службы;
  - `warmup` — служба посылает запрос для предварительной инициализации нового экземпляра приложения до получения им клиентского запроса, что сокращает время ожидания клиентом ответа от нового экземпляра приложения. GAE-платформа создает новый экземпляр приложения, например, при увеличении количества клиентских запросов к существующему экземпляру приложения.
- `<warmup-requests-enabled>` — с помощью значения `false` отключает warmup-запросы.
- `<precompilation-enabled>` — с помощью значения `false` отключает оптимизацию загрузки классов. Служба warmup работает с приложением, имеющим постоян-

ный трафик с клиентом. Если же клиент посылает запрос к наработавшему приложению, этот запрос, называемый loading-запросом, инициализирует экземпляр приложения, и клиент будет ожидать некоторое время ответ на свой первый запрос. Precompilation-оптимизация ускоряет работу loading-запроса как минимум на 30%.

- `<admin-console>` — с помощью дочерних тегов `<page name="" url="" />` добавляет страницы администрирования приложением в консоль администрирования **https://appengine.google.com/**.
- `<static-error-handlers>` — с помощью дочерних тегов `<handler file="" error-code=""/>` переопределяет страницы отображения ошибки клиенту по умолчанию, где код ошибки `error-code` может быть `over_quota`, `dos_api_denial` и `timeout`.
- `<threadsafe>` — с помощью значения `true` включает доставку параллельных запросов к серверу.

Перед запуском GAE-приложения из среды Eclipse удостоверимся, что версия Java раздела **Project Facets** диалогового окна свойств GAE-проекта соответствует версии среды JRE проекта. Для этого в окне **Project Explorer** щелкнем правой кнопкой мыши на узле проекта, в контекстном меню выберем команду **Properties** и раздел **Project Facets** ее диалогового окна (рис. 1.8).

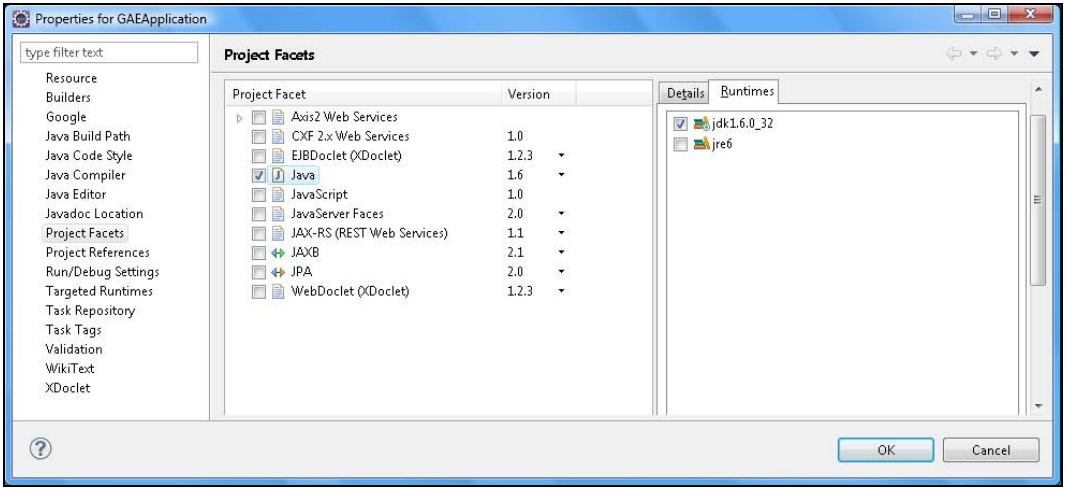


Рис. 1.8. Установка версии Java проекта GAE-приложения

Запуск приложения из среды Eclipse

Для запуска GAE-приложения из среды Eclipse в окне **Project Explorer** щелкнем правой кнопкой мыши на узле проекта и в контекстном меню выберем команду **Run As | Web Application**. В результате в окно **Console** среды Eclipse будет выведено сообщение:

```
... AM com.google.apphosting.utils.jetty.JettyLogger info
INFO: Logging to JettyLogger(null) via com.google.apphosting.utils.jetty.JettyLogger
... AM com.google.apphosting.utils.config.AppEngineWebXmlReader readAppEngineWebXml
INFO: Successfully processed ...\workspace\GAEApplication\war\WEB-INF/appengine-web.xml
... AM com.google.apphosting.utils.config.AbstractConfigXmlReader readConfigXml
INFO: Successfully processed ...\workspace\GAEApplication\war\WEB-INF/web.xml
... PM com.google.appengine.tools.development.DevAppServerImpl start
INFO: The server is running at http://localhost:8888/
... com.google.appengine.tools.development.DevAppServerImpl start
INFO: The admin console is running at http://localhost:8888/_ah/admin
```

В выведенном сообщении указан адрес локального вызова приложения **http://localhost:8888/** и адрес локальной консоли администрирования **http://localhost:8888/_ah/admin**.

Набрав в адресной строке Web-браузера адрес **http://localhost:8888/**, можно будет увидеть страницу приветствия приложения со ссылкой вызова сервлета приложения (рис. 1.9).

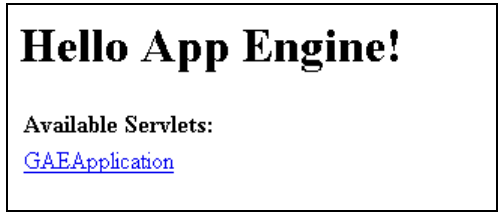


Рис. 1.9. Страница приветствия GAE-приложения

Набрав в адресной строке Web-браузера адрес **http://localhost:8888/_ah/admin**, можно будет увидеть страницу консоли администрирования (рис. 1.10).

Рис. 1.10. Страница локальной консоли администрирования GAE-приложения

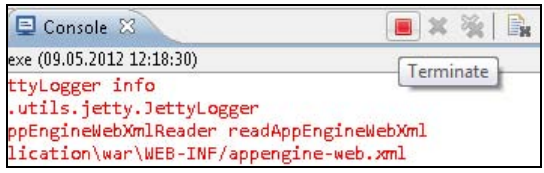


Рис. 1.11. Остановка GAE-приложения кнопкой **Terminate**

Остановить работающий сервер с GAE-приложением можно, нажав кнопку **Terminate** окна **Console** среды Eclipse (рис. 1.11).

Развертывание приложения на платформе App Engine

Регистрация приложения

Для того чтобы развернуть приложение на платформе GAE, его необходимо предварительно зарегистрировать. Для регистрации GAE-приложения откроем Web-браузер и попробуем войти в консоль администрирования по адресу **https://appengine.google.com/**. Однако сервис консоли администрирования является защищенным сервисом и его аутентификацию/авторизацию проходят лишь пользователи, имеющие свой Google-аккаунт. Поэтому предварительно откроется страница, приглашающая ввести логин/пароль Google-аккаунта или зарегистрировать его (рис. 1.12).

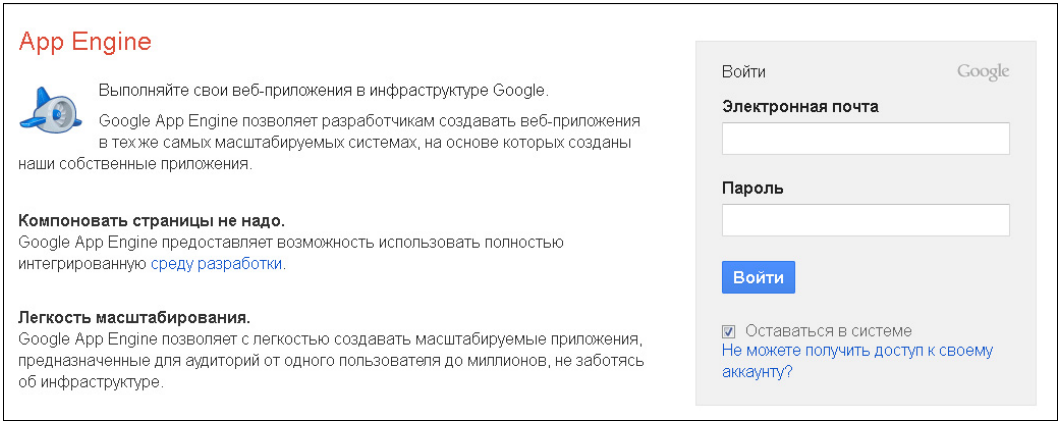


Рис. 1.12. Страница аутентификации консоли администрирования

После ввода логина/пароля и нажатия кнопки **Войти** откроется страница консоли администрирования.

Кнопка **Create Application** консоли администрирования позволяет зарегистрировать новое приложение. Страница, открывающаяся при нажатии кнопки **Create Application**, отображает уведомление об оставшемся количестве приложений, ко-

торые можно еще зарегистрировать на данный Google-аккаунт: максимальное число — 10. Поле **Application Identifier** предлагает ввести идентификатор приложения, при этом кнопка **Check Availability** позволяет проверить его уникальность. Поле **Application Title** предлагает ввести отображаемый при аутентификации заголовок приложения.

Раздел **Authentication Options (Advanced)** страницы регистрации GAE-приложения с помощью переключателей позволяет выбрать систему аутентификации пользователей, которая может быть использована GAE-приложением:

- **Open to all Google Accounts users (default)** — аутентификация с помощью Google-аккаунта;
- **Restricted to the following Google Apps domain** — аутентификация с помощью аккаунта домена бизнес-приложения Google Apps;
- **(Experimental) Open to all users with an OpenID Provider** — аутентификация с помощью идентификатора OpenID.

Раздел **Storage Options (Advanced)** страницы регистрации GAE-приложения с помощью переключателей **High Replication (default)** и **Master/Slave** позволяет выбрать тип хранилища данных приложения — распределенное хранилище или главное/подчиненное хранилище (в настоящее время не рекомендуется). Данный раздел после отмены хранилища Master/Slave был исключен.

Нажатие кнопки **Create Application** страницы регистрации GAE-приложения завершает регистрацию приложения.

Теперь при входе в консоль администрирования **https://appengine.google.com/** ее страница будет содержать ссылку на зарегистрированное приложение, при щелчке по которой откроется страница администрирования приложением.

Страница администрирования приложением

Левая панель страницы администрирования приложением содержит разделы **Main**, **Data**, **Administration**, **Billing** и **Resources**.

Раздел **Main** содержит подразделы **Dashboard**, **Instances**, **Logs**, **Versions**, **Backends**, **Cron Jobs**, **Task Queues**, **Quota Details**.

Раздел **Dashboard** показывает общую информацию о работе приложения в App Engine, включающую в себя набор графиков потребления ресурсов приложением, информацию о работающих экземплярах приложения, статус оплаты хостинга приложения, таблицу потребления ресурсов приложением в пределах установленной квоты (**https://developers.google.com/appengine/docs/quotas**), информацию о URI-запросах к приложению и об их ошибках. Набор графиков потребления ресурсов приложением состоит из следующих графиков:

- Requests/Second — количество запросов в секунду в зависимости от времени, включая динамические, статические и кэшированные запросы;
- Requests by Type/Second — количество запросов в секунду в зависимости от времени с дифференциацией по типам запросов: динамическим, статическим (к статическим ресурсам приложения) и кэшированным;

- Milliseconds/Request — время ожидания ответа от приложения для динамического запроса в зависимости от времени;
- Errors/Second — количество ошибок приложения в секунду в зависимости от времени;
- Bytes Received/Second — количество байтов запроса в секунду в зависимости от времени;
- Bytes Sent/Second — количество байтов ответа в секунду в зависимости от времени;
- CPU Seconds Used/Second — загруженность процессора в зависимости от времени с дифференциацией по общей загруженности и загруженности, связанной с использованием API GAE-платформы;
- Milliseconds Used/Second — обработка запроса приложением в зависимости от времени;
- Number of Quota DenialsSecond — количество отклонений запросов к приложению из-за превышения установленной квоты (**https://developers.google.com/appengine/docs/quotas**);
- Instances — работа экземпляров приложения в зависимости от времени с дифференциацией по активным экземплярам и экземплярам учитываемых квот;
- Memory Usage (MB) — использование памяти в зависимости от времени.

Раздел **Instances** показывает информацию о работе экземпляров приложения — количество созданных экземпляров приложения, среднее количество запросов в секунду, среднюю задержку ответа и среднее количество памяти, занимаемой экземпляром. Для каждого экземпляра приложения отображается среднее количество запросов в секунду за последнюю минуту, средняя задержка ответа за последнюю минуту, количество успешно обработанных запросов, количество ошибок, время жизни экземпляра, количество памяти, ссылка на журнал, тип экземпляра — динамический или резидентный и кнопка **Shutdown** уничтожения экземпляра.

Экземпляр приложения состоит из работающего экземпляра среды выполнения, подключенных библиотек, кода приложения и памяти, выделенной для работы. Наличие нескольких экземпляров приложения обеспечивает параллельную обработку входящих запросов, при этом за создание новых экземпляров приложения при увеличении количества входящих запросов отвечает планировщик GAE-платформы. Повлиять на работу планировщика GAE-платформы можно с помощью установок раздела **Administration | Application Settings | Performance** консоли администрирования.

Экземпляр приложения называется *динамическим*, если его жизненный цикл регулируется планировщиком GAE-платформы, или *резидентным*, если работает все время, при этом за его создание отвечает разработчик — администратор приложения. Экземпляр приложения может быть, помимо динамического или резидентного, *внешним* (Frontend) или *фоновым* (Backend). Frontend-экземпляры по умолчанию обрабатывают входящие запросы, и их работа ограничена установленной квотой (**https://developers.google.com/appengine/docs/quotas**). Backend-экземпляры, по

сравнению с Frontend-экземплярами, не ограничены 30-секундным периодом обработки запроса и могут использовать большую память (до 1 Гбайт) и больше загружать процессор (до 4,8 ГГц). Соответственно работа Frontend- и Backend-экземпляров приложения отдельно учитывается в использовании установленной квоты. Создать резидентные Frontend-экземпляры можно, только если приложение имеет платный хостинг и с помощью установки **min idle instances** раздела **Administration | Application Settings | Performance** консоли администрирования. Backend-экземпляры приложения связаны со специальным Backend-сервером, создаваемым для приложения GAE-платформой. Приложение может иметь несколько Backend-серверов, с каждым из которых может быть связано несколько Backend-экземпляров. Свойства Backend-серверов приложения определяются в его конфигурационном backends-файле. В отличие от динамических Frontend-экземпляров, Backend-экземпляры не создаются автоматически планировщиком GAE-платформы, он может лишь запускать и останавливать динамический Backend-сервер, с которым связано определенное число Backend-экземпляров. Backend-серверы назначаются резидентными (по умолчанию) или динамическими с помощью конфигурационного backends-файла приложения.

Раздел **Logs** отображает записи в журнале приложения с возможностью фильтрации записей по уровню (списки **Debug**, **Info**, **Warning**, **Error**, **Critical**) и датам (раздел **Options**).

Раздел **Versions** показывает развернутые версии приложения. Версия приложения указывается в теге `<version>` конфигурационного файла appengine-web.xml приложения. При загрузке приложения с уже существующим номером версии код развернутого приложения заменяется, при загрузке приложения с другим номером версии создается новая версия приложения. Сделать запрос к конкретной версии приложения можно с помощью URL-адреса вида **http://[*номер_версии*].[*идентификатор_приложения*].appspot.com/**. При наличии нескольких версий приложения в верхней части консоли администрирования появится раскрывающийся список с номерами версий, позволяющий отображать информацию для конкретной версии приложения.

Кнопка **Make Default** раздела **Versions** позволяет назначить версию приложения версией по умолчанию, принимающей входящие запросы к приложению, кнопка **Delete** удаляет выбранную версию приложения.

С помощью кнопки **Add Traffic Split** раскрывающегося подраздела **Traffic Splitting** раздела **Versions** можно разделить входящий трафик по версиям приложения для тестирования пользовательских предпочтений.

Раздел **Backends** отображает информацию о работе Backend-серверов приложения: тип сервера — динамический или резидентный, время и Google-аккаунт развертывания сервера, класс классификации сервера, количество его Backend-экземпляров, опции, кнопки запуска, остановки и уничтожения сервера. Квота потребления ресурсов Backend-сервером определяется его классом — B1 (128 Мбайт, 600 МГц), B2 (по умолчанию, 256 Мбайт, 1,2 ГГц), B4 (512 Мбайт, 2,4 ГГц), B8 (1024 Мбайт, 4,8 ГГц). Свойства Backend-сервера определяются в конфигурационном backends-

файле приложения. Использование Backend-серверов приложением ограничено их общим количеством 5, с каждым из которых может быть связано не более 20 Backend-экземпляров, и общей используемой памятью не более 10 Гбайт.

Раздел **Cron Jobs** показывает запланированные задачи приложения, представляющие собой HTTP GET-вызовы определенных URL-адресов в заданное время. Вызываемые URL-адреса и их расписания вызова определяются в конфигурационном cron-файле приложения. Количество запланированных задач приложения ограничено 20.

Раздел **Task Queues** позволяет управлять очередями задач приложения, определенными в конфигурационном queue-файле приложения. Приложение в своем коде может добавлять задачи, представляющие собой фоновые HTTP-запросы, в очереди, объявленные в конфигурационном queue-файле приложения, а разработчик — администратор приложения в консоли администрирования может приостановить очередь, немедленно выполнить отдельную задачу, удалить задачу или всю очередь.

Раздел **Quota Details** показывает квоту использования ресурсов GAE-платформы, установленную для приложения. При превышении установленной квоты отдельного ресурса он станет недоступным для приложения до тех пор, пока квота не пополнится. Так, при превышении дневной квоты, она обновится в полночь по тихоокеанскому времени. При превышении квоты, связанной с ресурсами инициализации запроса к приложению, запрос возвратит код HTTP 403, при превышении квоты, связанной с другими ресурсами, будет сгенерировано исключение.

Раздел **Data** консоли администрирования содержит подразделы **Datastore Indexes**, **Datastore Viewer**, **Datastore Statistics**, **Blob Viewer**, **Prospective Search**, **Text Search**, **Datastore Admin**, **Memcache Viewer**.

Раздел **Datastore Indexes** показывает информацию о indexes-таблицах приложения. Индексы — это таблицы, содержащие результаты запросов приложения к Datastore-хранилищу GAE-платформы. GAE-платформа использует indexes-таблицы для быстрой обработки запросов приложения к Datastore-хранилищу данных. О требуемых для запросов приложения индексах GAE-платформа узнает при развертывании приложения из его конфигурационного datastore-indexes-файла, который создается автоматически при разработке приложения в процессе его тестирования.

Раздел **Datastore Viewer** позволяет просматривать данные приложения, хранящиеся в Datastore-хранилище GAE-платформы.

Раздел **Datastore Statistics** показывает информацию об использовании приложением ресурсов Datastore-хранилища GAE-платформы.

Раздел **Blob Viewer** позволяет просматривать данные приложения, хранящиеся в Blobstore-хранилище GAE-платформы.

Раздел **Prospective Search** показывает информацию о зарегистрированных приложением запросах на данные, которые соответствуют определенным критериям. Программный интерфейс Datastore API позволяет осуществлять запросы к статическим данным хранилища. Служба Prospective Search дает возможность зарегистри-