Глава 1
Краткий обзор Java EE 7

Сегодняшние предприятия существуют в условиях мировой конкуренции. Им нужны приложения, отвечающие их производственным нуждам, которые, в свою очередь, с каждым днем усложняются. В эпоху глобализации компании могут действовать по всему миру, имея представительства на разных континентах, работать в различных странах круглосуточно, без выходных, иметь по несколько центров обработки данных и международные системы, работающие с разными валютами и временными зонами. При этом им необходимо сокращать расходы, увеличивать быстродействие своих сервисов, хранить бизнес-данные в надежных и безопасных хранилищах, а также иметь несколько мобильных и веб-интерфейсов для клиентов, сотрудников и поставщиков.

Большинству компаний необходимо совмещать эти противоречивые требования с существующими корпоративными информационными системами (EIS), одновременно разрабатывая приложения типа «бизнес — бизнес» для работы с партнерами или системы типа «бизнес — потребитель» с использованием мобильных приложений, в том числе с возможностью геолокации. Довольно часто компании требуется координировать корпоративные данные, которые хранятся в разных местах, обрабатываются несколькими языками программирования и передаются с помощью разных протоколов. И конечно же, во избежание серьезных убытков необходимо предотвращать системные сбои, сохраняя при этом высокую доступность, масштабируемость и безопасность. Изменяясь и усложняясь, корпоративные приложения должны оставаться надежными. Именно для этого была создана платформа Java Enterprise Edition (Java EE).

Первая версия Java EE (изначально известная как J2EE) предназначалась для решения задач, с которыми сталкивались компании в 1999 году, а именно для работы с распределенными компонентами. С тех пор программным приложениям пришлось адаптироваться к новым техническим решениям, таким как веб-службы SOAP и RESTful. На сегодняшний день платформа Java EE отвечает этим техническим требованиям, регламентируя различные способы работы в стандартных спецификациях. Спустя годы Java EE изменилась, стала насыщеннее и проще в использовании, а также более мобильной и интегрированной.

В этой главе будут представлены общие сведения о Java EE. После описания внутренней архитектуры, компонентов и сервисов я расскажу о том, что нового в Java EE 7.

Понимание Java EE

Если вам нужно произвести какие-то операции с наборами объектов, вы не разрабатываете для этого свою собственную хеш-таблицу; вы используете API кол-

лекций (интерфейс программирования приложений). Точно так же, если вам требуется простое веб-приложение или безопасная, интероперабельная распределенная прикладная система, оперирующая транзакциями, вы не захотите разрабатывать низкоуровневые API-интерфейсы, а будете использовать платформу Java EE. Так же как платформа Java Standard Edition (Java SE) предоставляет API для работы с коллекциями, Java EE предоставляет стандартный способ работы с транзакциями через Java API для транзакций (JTA), с сообщениями через службу сообщений Java (JMS) и с сохраняемостью через интерфейс JPA. Java EE располагает рядом спецификаций, предназначенных для корпоративных приложений. Она может рассматриваться как продолжение платформы Java SE для более удобной разработки распределенных, надежных, мощных и высокодоступных приложений.

Версия Java EE 7 стала важной вехой в развитии платформы. Она не только продолжает традиции Java EE 6, предлагая более простую модель разработки, но и добавляет свежие спецификации, обогащая наработанный функционал новыми возможностями. Кроме того, контекст и внедрение зависимостей (CDI) становится точкой интеграции между всеми новыми спецификациями. Релиз Java EE 7 почти совпадает с 13-й годовщиной выпуска корпоративной платформы. Она объединяет преимущества языка Java и опыт последних 13 лет. Java EE выигрывает как за счет динамизма сообществ свободных разработчиков, так и за счет строгой стандартизации группы Java Community Process (JCP). На сегодняшний день Java EE — это хорошо документированная платформа с опытными разработчиками, большим сообществом пользователей и множеством развертываемых приложений, работающих на серверах компаний. Java EE объединяет несколько интерфейсов API, которые могут использоваться для построения стандартных компонентно-ориентированных многозвенных приложений. Эти компоненты развертываются в различных контейнерах, предлагая серию служб.

Архитектура

Java EE состоит из набора спецификаций, реализуемых различными контейнерами. Контейнерами называются средства среды времени выполнения Java EE, предоставляющие размещенным на них компонентам определенные службы, например управление жизненным циклом разработки, внедрение зависимости, параллельный доступ и т. д. Такие компоненты используют точно определенные контракты для сообщения с инфраструктурой Java EE и с другими компонентами. Перед развертыванием они должны упаковываться стандартным способом (повторяя структуру определенного каталога, который может быть сжат в архивный файл). Java EE представляет собой расширенный набор функций платформы Java SE, что означает, что API-интерфейсы Java SE могут использоваться любыми компонентами Java EE.

Рисунок 1.1 иллюстрирует логические взаимосвязи между контейнерами. Стрелками представлены протоколы, используемые одним контейнером для доступа к другому. Например, веб-контейнер размещает сервлеты, которые могут обращаться к компонентам EJB по протоколу RMI-IIOP.

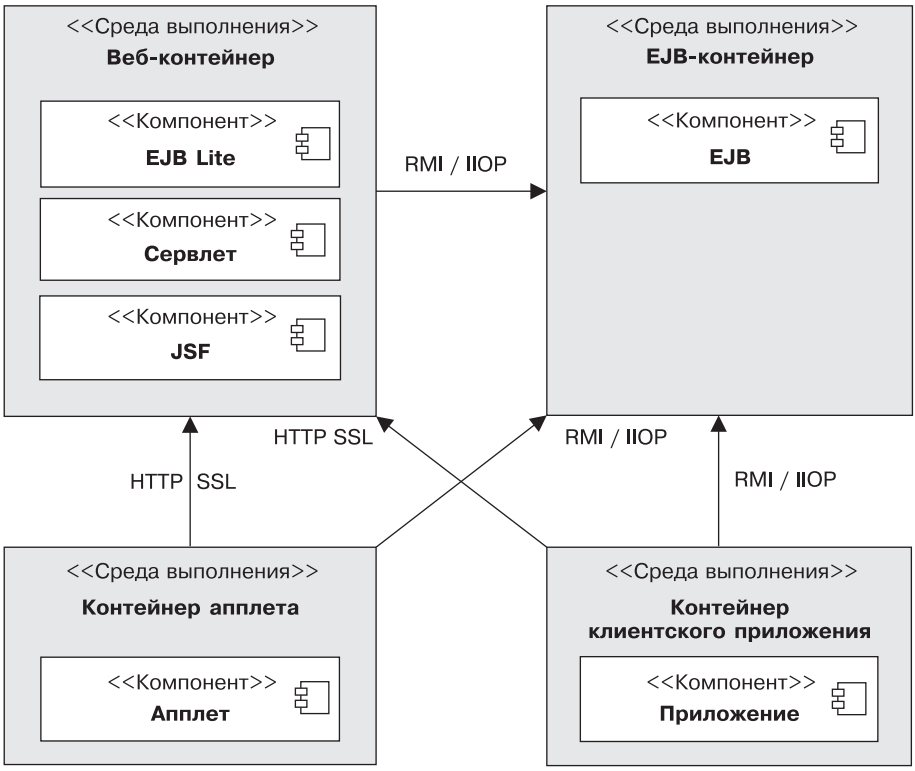

Рис. 1.1. Стандартные контейнеры Java EE

Компоненты

В среде времени выполнения Java EE выделяют четыре типа компонентов, которые должна поддерживать реализация.

- *Апплеты* представляют собой приложения из графического пользовательского интерфейса (GUI), выполняемые в браузере. Они задействуют насыщенный интерфейс Swing API для производства мощных пользовательских интерфейсов.
- *Приложениями* называются программы, выполняемые на клиентской стороне. Как правило, они относятся к графическому пользовательскому интерфейсу (GUI) и применяются для пакетной обработки. Приложения имеют доступ ко всем средствам среднего звена.
- *Веб-приложения* (состоят из сервлетов и их фильтров, слушателей веб-событий, страниц JSP и JSF) выполняются в веб-контейнере и отвечают на запросы HTTP от веб-клиентов. Сервлеты также поддерживают конечные точки веб-служб SOAP и RESTful. Веб-приложения также могут содержать компоненты EJBLite (подробнее об этом читайте в гл. 7).
- *Корпоративные приложения* (созданные с помощью технологии Enterprise Java Beans, службы сообщений Java Message Service, интерфейса Java API для транзакций, асинхронных вызовов, службы времени, протоколов RMI-IIOP) вы-

полняются в контейнере EJB. Управляемые контейнером компоненты EJB служат для обработки транзакционной бизнес-логики. Доступ к ним может быть как локальным, так и удаленным по протоколу RMI (или HTTP для веб-служб SOAP и RESTful).

Контейнеры

Инфраструктура Java EE подразделяется на логические домены, называемые контейнерами (см. рис. 1.1). Каждый из контейнеров играет свою специфическую роль, поддерживает набор интерфейсов API и предлагает компонентам сервисы (безопасность, доступ к базе данных, обработку транзакций, присваивание имен каталогам, внедрение ресурсов). Контейнеры скрывают техническую сложность и повышают мобильность. При разработке приложений каждого типа необходимо учитывать возможности и ограничения каждого контейнера, чтобы знать, использовать один или несколько. Например, для разработки веб-приложения необходимо сначала разработать уровень фреймворка JSF и уровень EJB Lite, а затем развернуть их в веб-контейнер. Но если вы хотите, чтобы веб-приложение удаленно вызывало бизнес-уровень, а также использовало передачу сообщений и асинхронные вызовы, вам потребуется как веб-, так и EJB-контейнер.

Java EE использует четыре различных контейнера.

- *Контейнеры апплетов* выполняются большинством браузеров. При разработке апплетов можно сконцентрироваться на визуальной стороне приложения, в то время как контейнер обеспечивает безопасную среду. Контейнер апплета использует модель безопасности изолированной программной среды («песочницы»), где коду, выполняемому в «песочнице», не разрешается «играть» за ее пределами. Это означает, что контейнер препятствует любому коду, загруженному на ваш локальный компьютер, получать доступ к локальным ресурсам системы (процессам либо файлам).
- *Контейнер клиентского приложения (ACC)* включает набор Java-классов, библиотек и других файлов, необходимых для реализации в приложениях Java SE таких возможностей, как внедрение, управление безопасностью и служба именования (в частности, Swing, пакетная обработка либо просто класс с методом `main()`). Контейнер ACC обращается к EJB-контейнеру, используя протокол RMI-IIOP, а к веб-контейнеру — по протоколу HTTP (например, для веб-служб).
- *Веб-контейнер* предоставляет базовые службы для управления и исполнения веб-компонентов (сервлетов, компонентов EJB Lite, страниц JSP, фильтров, слушателей, страниц JSF и веб-служб). Он отвечает за создание экземпляров, инициализацию и вызов сервлетов, а также поддержку протоколов HTTP и HTTPS. Этот контейнер используется для подачи веб-страниц к клиент-браузерам.
- *EJB-контейнер* отвечает за управление и исполнение компонентов модели EJB (компонент-сеансы EJB и компоненты, управляемые сообщениями), содержащих уровень бизнес-логики вашего приложения на Java EE. Он создает новые сущности компонентов EJB, управляет их жизненным циклом и обеспечивает реализацию таких сервисов, как транзакция, безопасность, параллельный доступ, распределение, служба именования либо возможность асинхронного вызова.