УДК 00.004.4

Ю.Э. Парфенов, В.Н. Федорченко

Харьковский национальный экономический университет, Харьков

ИСПОЛЬЗОВАНИЕ ПЛАТФОРМЫ JAVA EE 7 ДЛЯ РАЗРАБОТКИ КОРПОРАТИВНЫХ ПРИЛОЖЕНИЙ

В работе рассмотрены вопросы организации корпоративных вычислений и использования серверных платформ для корпоративных приложений. Изложены основные принципы организации спецификации Java EE. Анализируются новые возможности платформы Java EE 7 и ее применение для разработки корпоративных приложений.

Ключевые слова: корпоративные вычисления, серверная платформа, контейнер, сервис распределенных транзакций, сервис безопасности, сервис событий, веб-служба, сервлет, EJB, JSF, JPA.

Введение

Критически важным для успешного ведения бизнеса является снижение издержек предприятий и времени отклика их сервисов, надежное и безопасное хранение бизнес-данных, наличие мобильных и веб-интерфейсов для клиентов, сотрудников и поставщиков [1].

Таким образом, приложения, предназначенные для удовлетворения потребностей бизнеса, неизбежно становятся все более и более сложными.

«Корпоративные вычисления» – это универсальная категория для обозначения программного и аппаратного обеспечения, которое предназначено для удовлетворения потребностей крупных организаций.

Эти потребности обычно включают [4]:

- Обеспечение высокого уровня надежности, в том числе на базе внутренней избыточности, для того, чтобы можно было вести бизнес даже при выходе из строя одной из систем.

- Обеспечение высокого уровня безопасности, в том числе – надежной системы безопасности баз данных и возможности создания специфических профилей доступа для разных категорий пользователей.

- Наличие централизованной системы хранения данных, которая собирает и организует данные всей организации, а также управляет доступом к данным в соответствие с протоколами безопасности.

- Возможность добавления и настройки приложений по мере необходимости, а также наличие упрощенного способа предоставления приложениям необходимого доступа к данным.

- Минимизация времени отклика на действия пользователей.

Цель статьи и постановка заданий

Целью настоящей работы является анализ особенностей корпоративных приложений и обоснование рекомендаций по выбору корпоративной вычислительной платформы для разработки и использования таких приложений.

Особое внимание в работе уделено архитектуре платформы Java EE и анализу новых возможностей версии Java EE 7.

Основная часть

У истоков архитектур серверных платформ для корпоративных приложений находится Common Object Request Broker Architecture and Specification (CORBA), разработанная группой OMG. Однако, многочисленные недостатки технологии CORBA постепенно свели на нет ее применение в большинстве корпоративных систем.

В настоящее время, «корпоративные вычисления» предусматривают построение аппаратной инфраструктуры и программной платформы, к которой могут быть «подключены» все приложения предприятия.

Корпоративные приложения должны проектироваться, разрабатываться и внедряться с меньшими финансовыми затратами, большей скоростью и с меньшим количеством ресурсов. Именно поэтому была создана платформа Java Enterprise Edition (Java EE) – корпоративная вычислительная платформа Oracle на базе Java [1].

Java EE является общей спецификацией, которая связывает и объединяет другие спецификации. Провайдеры технологий должны обеспечивать выполнение определенных требований, чтобы объявить свою продукцию совместимой с Java EE.

Архитектура Java EE в значительной степени базируется на модульных компонентах, работающих на сервере приложений. Программное обеспечение для Java EE в первую очередь разрабатывается на языке программирования Java.

Всего было выпущено несколько версий Java EE [1]: J2EE 1.2, J2EE 1.3, J2EE 1.4, Java EE 1.5, Java EE 1.6, Java EE 1.7

Java EE 7, выпущенная 12 июня 2013 г., добавляет много новых спецификаций, а также улучшает существующие. Наиболее важной целью

платформы Java EE 7 является упрощение разработки путем обеспечения общей среды для компонентов различных видов [2].

Рассмотрим основные элементы архитектуры платформы Java EE.

Модель приложения Java EE базируется на языке программирования Java и виртуальной машине Java. Она определяет архитектуру реализации сервисов в виде многоуровневых приложений, которые обеспечивают масштабируемость, доступность и управляемость, необходимую для приложений уровня предприятия. Эта модель делит работу, необходимую для реализации многоуровневого сервиса на следующие части:

- бизнес-логику и логику представления, реализуемые разработчиком;
- стандартные системные сервисы, предоставляемые Java EE.

Логика приложения делится по функциональному признаку на компоненты. Компоненты Java EE являются автономными программными модулями, которые собираются в приложение Java EE вместе со связанными с ними классами и файлами и взаимодействуют с другими компонентами. Они устанавливаются на разных машинах, в зависимости от яруса, к которому принадлежит компонент в многоуровневой среде Java EE.

Составные части приложения Java EE представлены следующими компонентами (рис.1):

- компоненты клиентского яруса, функционирующие на клиентской машине;
- компоненты веб-яруса, которые выполняются на сервере Java EE;
- компоненты бизнес-уровня, также выполняющиеся на сервере Java EE;
- программное обеспечение информационно-аналитического яруса работает на соответствующем сервере.

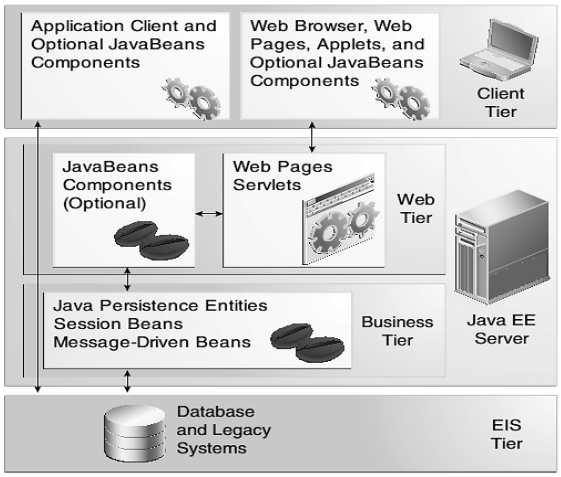

Рис. 1. Многоярусное приложение Java EE

Хотя приложения Java EE могут включать все уровни, как правило, они считается трехъярусными приложениями, так как они распределены между клиентскими компьютерами, компьютером сервера Java EE и компьютером сервера баз данных.

Спецификация Java EE определяет следующие компоненты Java EE:

- клиентские приложения и апплеты, которые выполняются на клиентской стороне;
- компоненты Java Servlet, JSF и JSP – это веб-компоненты, которые выполняются на сервере;
- компоненты EJB – это бизнес-компоненты, которые выполняются на сервере.

Компоненты Java EE пишутся на языке программирования Java и компилируются таким же образом, как и любая программа на этом языке. Различия между компонентами Java EE и «стандартными» Java-классами состоят в том, что компоненты Java EE собираются в приложение Java EE, проверяются на соответствие спецификации Java EE и разворачиваются на сервере Java EE, который управляет их жизненным циклом.

Кроссплатформенная архитектура платформы Java EE, основанная на использовании компонентов, делает приложения Java EE несложными в разработке, так как бизнес-логика сосредоточена в повторно используемых компонентах.

Также, за счет того, что сервер Java EE обеспечивает набор низкоуровневых сервисов в виде контейнера для каждого типа компонентов, разработчик может направить свою деятельность на решение бизнес-задач.

Контейнеры являются интерфейсом между компонентом и низкоуровневой платформозависимой функциональностью, которая предназначена для поддержки компонента.

Прежде, чем любой компонент может быть выполнен, он должен быть добавлен в модуль Java EE и развернут в его контейнере. Этот процесс включает указание настроек контейнера для каждого компонента и, собственно, приложения Java EE, что позволяет настроить базовую поддержку со стороны сервера Java EE: безопасность, управление транзакциями, обнаружение компонентов, удаленное соединение.

Контейнер также управляет жизненным циклом компонентов EJB и сервлетов, пулом соединений с базой данных, «постоянными» данными и доступом к API платформы Java EE.

Виды контейнеров [2] (рис. 2):

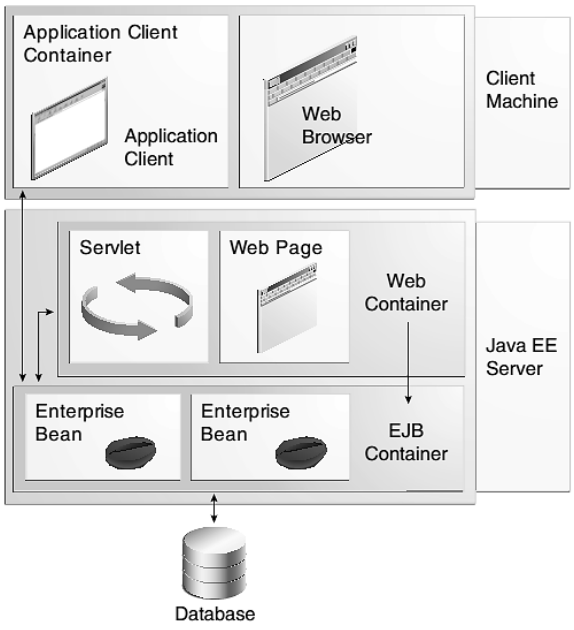

Рис. 2 Java EE-сервер и контейнеры

Сервер Java EE. Это среда выполнения для программных продуктов Java EE. Серверы приложений Java EE обеспечивают системные сервисы в соответствии с четко определенными открытыми промышленными стандартами.

Благодаря этому разработчики приложений могут разрабатывать программы в соответствии со спецификацией Java EE, не зависящие от конкретного сервера приложений. Таким образом, приложения Java EE могут быть развернуты на любом сервере приложений Java EE.

«Эталонной» реализацией платформы Java EE 7 является сервер приложений Oracle GlassFish. Кроме того, есть много свободно распространяемых серверов приложений, которые поддерживают Java EE: JBoss Application Server, Apache Geronimo, Apache TomEE, Caucho Resin и т.д.

Контейнер EJB – управляет выполнением компонентов EJB для приложений Java EE. Компоненты EJB и их контейнер выполняются на сервере Java EE.

Веб-контейнер – управляет выполнением веб-страниц, сервлетов и некоторых компонентов EJB, функционирующих на сервере Java EE.

Контейнер клиентских приложений – управляет выполнением компонентов клиентских приложений.

Контейнер апплетов – управляет выполнением апплетов. Состоит из веб-браузера и Java Plug-in, работающих на клиентской стороне.

Java EE 7 предлагает новые возможности, которые увеличивают производительность труда разработчиков, и обеспечивает дальнейшее совершенствование способов удовлетворения возросших требований предприятий к корпоративным платформам.

Таким образом, разработчикам придется писать меньше рутинного кода, они будут иметь лучшую поддержку новейших веб-приложений и фреймворков, а также получат доступ к расширенной масштабируемости, более совершенной и простой функциональности. Предприятия получат новые возможности за счет портативной пакетной обработки и улучшенной масштабируемости.

В Java EE 7 добавлено ряд новых спецификаций, которые обеспечивают совершенствование функциональности платформы. Также была выполнена ревизия нескольких существующих спецификаций компонентов для упрощения их использования.

Основные особенности новых спецификаций Java EE 7 описаны ниже [3]:

1) Java API for WebSocket:

- позволяет определять клиентские и серверные «конечные точки» веб-сокета декларативно с помощью аннотаций для POJO или программно с помощью реализации необходимых интерфейсов;

- обеспечивает конфигурацию, специфическую для сервера (отображение, которое идентифицирует «конечную точку» веб-сокета в URI-пространстве контейнера; протоколы, поддерживаемые «конечной точкой»; расширения, необходимые для приложений);

- предлагает специфичные для клиента конфигурации, например, предоставление пользовательских алгоритмов конфигурации;

- обеспечивает интеграцию с существующими технологиями Java EE.

2) Java API для JSON Processing:

- потоковый API предоставляет возможность обработки и генерации JSON;

- API объектной модели создает древовидную структуру с произвольным доступом для представления данных JSON в памяти.

3) Пакетные приложения на платформе Java:

- обеспечивает описание пакетного задания с использованием Job Specification Language;

- описывает пакетную модель программирования, используя интерфейсы, абстрактные классы и аннотации;

- предлагает блочный и пакетный стили обработки работ.

4) Утилиты обеспечения параллелизма для Java EE:

- обеспечивают возможности параллелизма для компонентов приложения Java EE без ущерба для целостности контейнера;

- определяют управляемые объекты: ContextService, ManagedScheduledExecutorService, ManagedThreadFactory и ManagedExecutorService.

Основные черты обновленных спецификаций описываются далее:

1) Java API for RESTful Web Services:

- предлагает новый клиентский API, который может использоваться для доступа к веб-ресурсам и обеспечения интеграции с поставщиками JAX-RS;

- поддерживает асинхронную обработку как в API клиента, так и в API сервера;

- определяет фильтры сообщений, и сущности-перехватчики для настройки обработки запросов и ответов на клиенте и сервере;

- поддерживает согласование контента на стороне сервера, используя qsfactor;

- обеспечивает декларативную валидацию полей, свойств и параметров, внедренных с помощью аннотаций, а также аннотации ограничений для классов ресурсов.

2) Java Message Service

- были сделаны некоторые изменения для упрощения использования JMS (Connection, Session и другие объекты, в которых есть метод close, сейчас реализуют интерфейс java.lang.Autocloseable, что позволяет им использовать оператор Java SE 7 «try-with-resources»; были добавлены новые методы для создания сессии без необходимости использования чрезмерного количества аргументов; был добавлен новый метод GetBody, который позволяет приложению извлечь тело сообщения непосредственно из объекта Message без необходимости приведения типов);

- «производитель» сообщений может теперь указать, что сообщение не должны быть доставлены до истечения определенного промежутка времени;

- были добавлены новые методы для асинхронной отправки сообщений;

- JMS-провайдеры должны теперь установить свойство сообщения JMSXDeliveryCount.

3) Expression Language (EL):

- EL может быть сконфигурирован и использован вне контейнера Java EE с помощью ELProcessor;

- в EL теперь включен синтаксис лямбда-выражений, поддерживается использование лямбда-выражений при операциях с коллекциями;

- чтобы сделать EL более выразительным, были добавлены новые операторы.

4) Enterprise JavaBeans:

- поддержка EJB 2.1, EJB QL и “конечных точек” веб-служб на базе JAX-RPC сделана опциональной;

- расширенный контракт компонентов Message-Driven Beans (MDB) на базе интерфейса слушателя сообщений позволяет использовать все public-методы компонента MDB в качестве методов слушателя сообщений (это позволит разрабатывать пользовательские адаптеры ресурсов для компонентов MDB);

- были определены группы EJB API с четкими правилами для контейнера EJB Lite относительно поддержки других групп API (это поможет определить порядок добавления полной функциональности EJB в программный продукт, который не поддерживает полный профиль Java EE);

- в EJB Lite добавлены асинхронные вызовы сессионных EJB и служба EJB Timer Service;

- для сессионных EJB с поддержкой состояния добавлена опция для отключения возможности их перехода в пассивное состояние.

5) Сервлеты:

- определяют стандартный механизм для обновления существующего HTTP–подключения для использования с разными протоколами с помощью HttpUpgradeHandler;

- предлагают неблокирующую обработку запросов и ответов для асинхронных сервлетов;

- определяют правила распространения ограничений безопасности на HTTP-методы.

6) Java Server Faces:

- поток JSF обеспечивает инкапсуляцию связанных представлений с точками входа и выхода, определенными в приложении;

- контракты библиотек ресурсов позволяют разработчикам применять шаблоны фейслетов ко всему приложению;

- разметка, совместимая с HTML5, обеспечивает почти полный контроль над любым элементом пользовательского интерфейса на веб-странице;

- введение представлений без поддержки состояния позволяет приложениям с компонентами JavaScript управлять состоянием, вместо JSF.

7) Java Persistence:

- схема базы данных и таблицы могут генерироваться с помощью свойств из пакета javax.persistence.schema-generation;

- добавлены несинхронизированные Persistence Contexts, поэтому они не обязательно должны быть перечислены в транзакции;

- в Criteria API теперь поддерживается пакетное обновление/удаление;

- предустановленные и пользовательские функции можно вызвать с помощью FUNCTION;

- хранимые процедуры можно вызвать при помощи StoredProcedureQuery и аннотации @NamedStoredProcedureQuery.

8) Перехватчики:

- привязка перехватчиков с использованием InterceptorBinding теперь является частью спецификации вместо CDI;

- с помощью аннотации @AroundConstruct можно задать метод перехватчика, который получает обратный вызов при вызове конструктора целевого класса;

- используя связывание перехватчиков, могут быть созданы приоритетные диапазоны для упорядочивания перехватчиков.

9) Contexts and Dependency Injection:

- позволяет автоматическое включение CDI для компонентов с помощью аннотаций области видимости и компонентов EJB;

- поддерживает глобальное упорядочивание и активацию перехватчиков, декораторов и альтернатив с помощью аннотации @Priority;

- добавляет аннотацию @Vetoed для простого программного отключения классов.

10) Bean Validation:

- ограничения валидации могут быть применены к параметрам и возвращаемым значениям произвольных методов и конструкторов;

- были переработаны и усилены механизмы интеграции с CDI;

- целевая группа может быть изменена при выполнении каскадной валидации.

11) Java Transaction:

- аннотация @Transactional обеспечивает для приложений возможность декларативного управления границами транзакций для компонентов, управляемых CDI, а также для управляемых компонентов Java EE на уровне классов и методов;

- аннотация @TransactionScoped предназначена для указания экземпляров компонентов, жизненный цикл которых ограничен текущей активной JTA-транзакцией.

12) JavaMail:

- аннотации @MailSessionDefinition и @MailSessionDefintions определяют сессию MailSession, которая необходимо зарегистрировать в JNDI.

13) Java EE Connector Architecture:

обеспечивает аннотации для определения объектов, администрируемых коннектором, и фабрик для регистрации в JNDI:

@AdministeredObjectDefinition,

@AdministeredObjectDefintions,

@ConnectorFactoryDefinition,

@ConnectorFactoryDefinitions

Выводы

Таким образом, платформа Java EE 7 позволяет разработчикам корпоративных приложений воспользоваться преимуществами шаблонов, фреймворков и технологий корпоративного сектора для того, чтобы поставлять современные программные продукты, обладающие высокой эффективностью, гибкостью и простотой освоения.

Список литературы

1. Antonio Goncalves. Beginning Java EE 7, apress, 2013, 567 p.

2. The Java EE 7 Tutorial, Release 7 for Java EE Platform, Oracle, 2013. [Электронный ресурс]. – Режим доступа: http://docs.oracle.com/javaee/7/tutorial/doc/home.htm

3. Arun Gupta. Java EE 7 Essentials, 2013, O'Reilly, 362 p.

4. Enterprise Computing: What's All the Buzz? [Электронный ресурс]. – Режим доступа: http://www.techopedia.com/2/28102/enterprise/platforms/enterprise-computing-whats-all-the-buzz

5. Новые перспективы Java Enterprise с Polyglot JVM. [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/company/epam_systems/blog/169321/

Рецензент: д-р техн. наук, проф. И.В. Рубан, Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков.

Авторы:

ПАРФЕНОВ Юрий Эдуардович
Харьковский национальный экономический университет, Харьков, кандидат технических наук, старший научный сотрудник, доцент кафедры информационных систем.
Раб. тел. – 702-18-31, E-mail – Yurii.Parfonov@m.hneu.edu.ua

ФЕДОРЧЕНКО Владимир Николаевич
Харьковский национальный экономический университет, Харьков, кандидат технических наук, доцент, доцент кафедры информационных систем.
Раб. тел. – 702-18-31, E-mail – is@ksue.edu.ua

ВИКОРИСТАННЯ ПЛАТФОРМИ JAVA EE 7 ДЛЯ РОЗРОБКИ КОРПОРАТИВНИХ ДОДАТКІВ

Ю.Е. Парфьонов, В.М. Федорченко

У роботі розглянуто питання організації корпоративних обчислень та використання серверних платформ для корпоративних застосунків. Викладені основні принципи організації специфікації Java EE. Аналізуються нові можливості платформи Java EE 7 та її використання для розроблення корпоративних застосунків.

Ключові слова: корпоративні обчислення, серверна платформа, контейнер, сервіс розподілених транзакцій, сервіс безпеки, сервіс подій, веб-служба, сервлет, EJB, JSF, JPA.

USING OF JAVA EE 7 PLATFORM TO DEVELOP ENTERPRISE APPLICATIONS

Y. Parfyonov, V. Fedorchenko

In the article considered issues of organization of enterprise computing and using of server platforms to develop enterprise applications. The main principles of the organization of Java EE specification are highlighted. New features of Java EE platform 7 and its application for developing enterprise application are analyzed.

Keywords: *enterprise computing, server platform, container, distributed transaction service, security service, event service, web service, servlet, EJB, JSF,JPA.*