



Delivering Excellence in Software Engineering



Контейнеры в Java Enterprise Edition

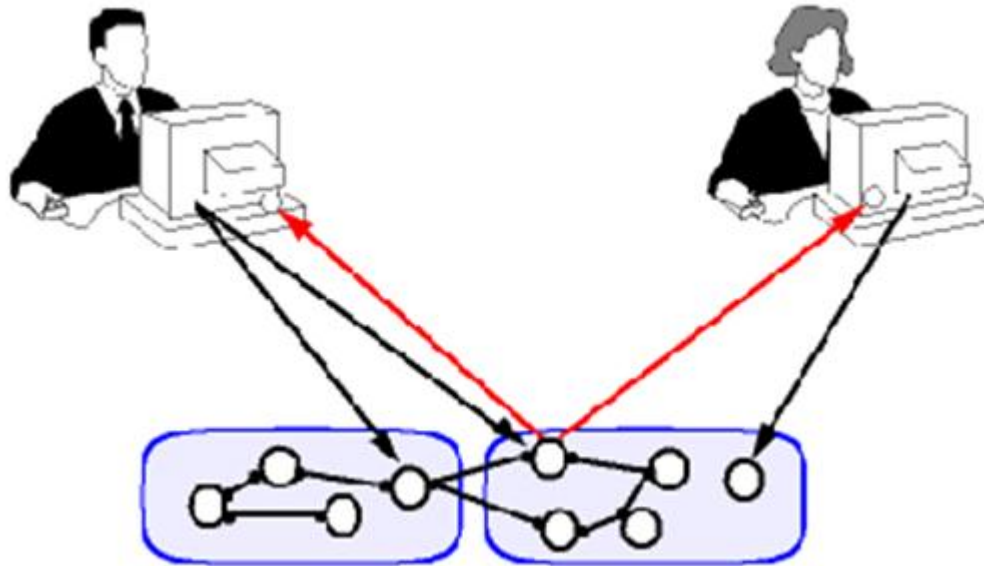
Елена Сирота

Руководитель Java лаборатории, RD Department.

Olena_Syrota@epam.com

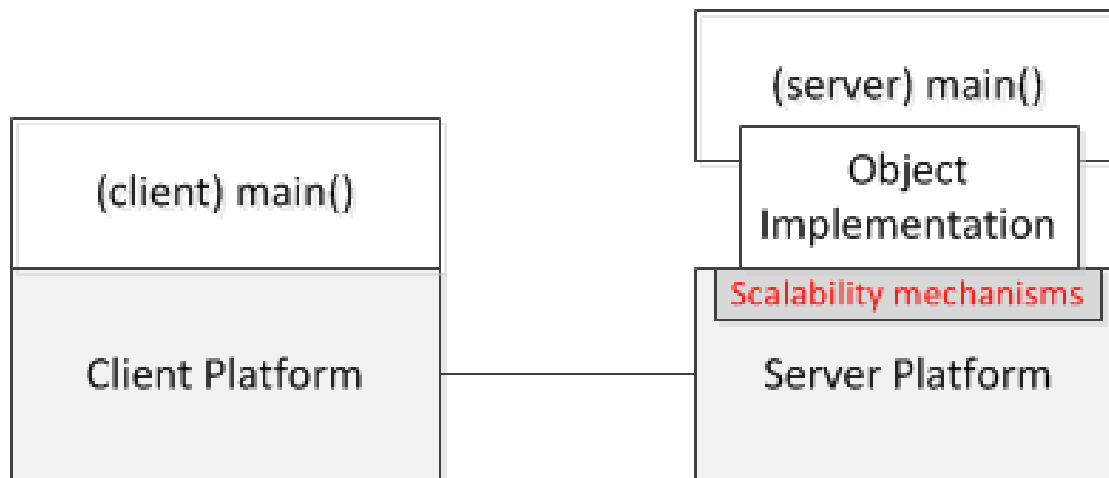
- Особенности корпоративных распределенных систем
- Контейнеры в Java EE
 - Типы
 - Назначение
- IoC контейнеры
 - Назначение

- Java Enterprise Edition – класс корпоративных распределенных систем



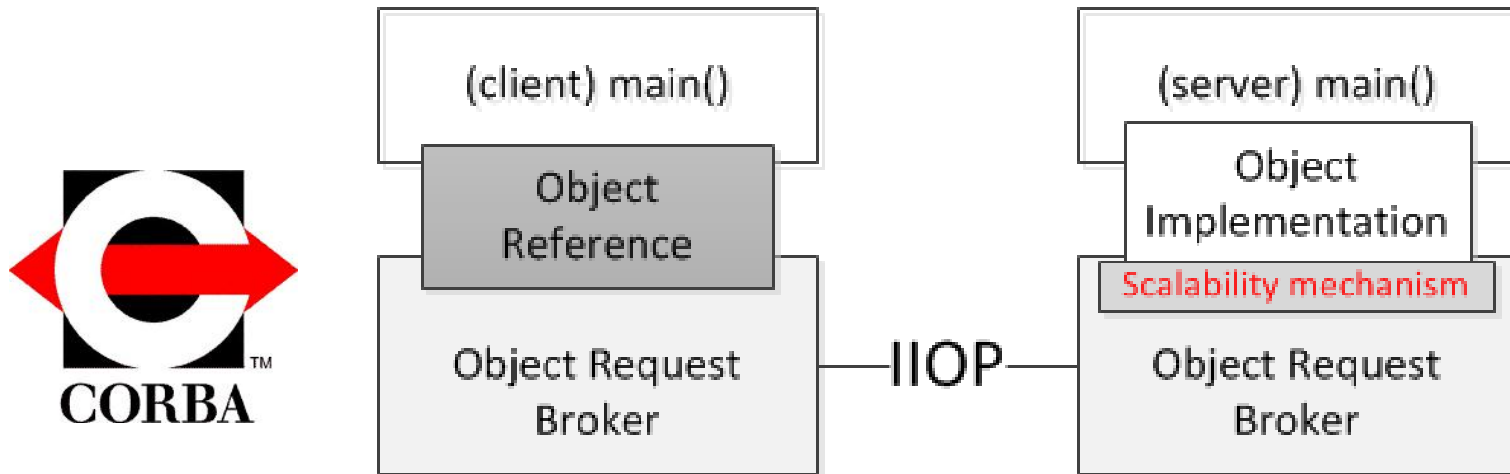
- Что в таких системах распределено?
- Чем характеризуются корпоративные распределенные системы?

- Высокая нагрузка на серверную часть
- Характеристики архитектуры:
 - Серверная часть – сложная архитектура, содержит механизмы по обеспечению работоспособности в условиях высокой нагрузки
 - Клиентская часть – простая архитектура



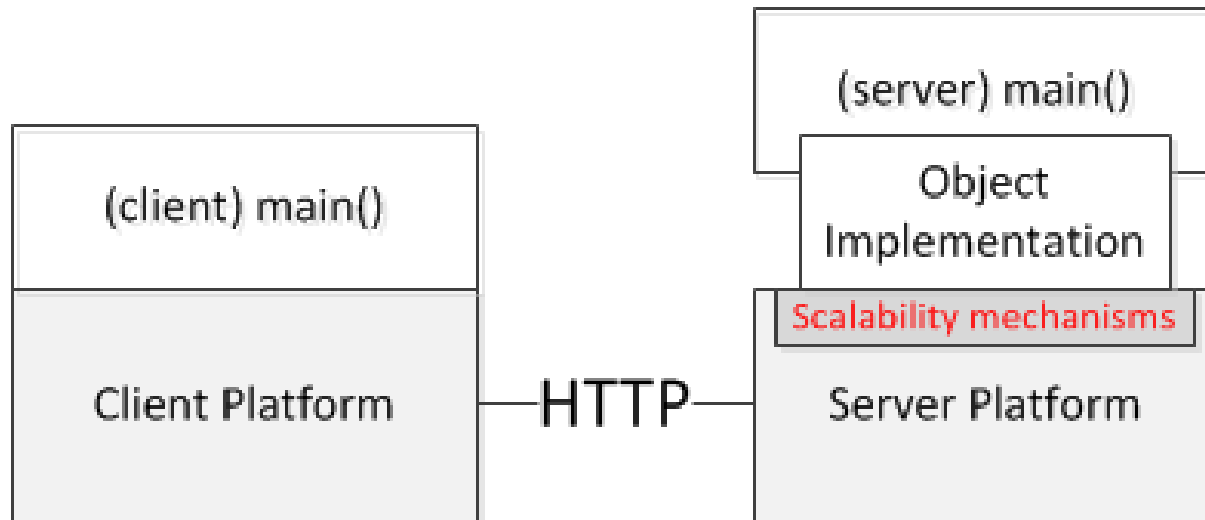
Источники архитектуры

- Предложенная архитектура происходит из Corba
 - Реализация объекта с бизнес-логикой находится на сервере
 - Клиент получает и работает с удаленной ссылкой на этот объект



Корпоративные системы с веб-интерфейсом

- Для корпоративных систем с веб-интерфейсом характерно следующее:



- Что общего на всех приведенных рисунках?
- Для корпоративных систем характерным является использование **“server platform”** и **серверных объектов**.
- Что такое “server platform”?
- Какие типы серверных объектов Вы знаете?

“Server Platform”

- Java “Server Platform” = Java Enterprise Edition
 - это Java + набор enterprise-сервисов
 - HTTP/HTTPS
 - Java Transaction Service (JTS)
 - JDBC
 - Java Persistence API (JPA)
 - Java Messaging Service (JMS)
 - Java Naming and Directory Interface (JNDI)
 - Java Mail
 - Java EE Connector Architecture
 - Security Services
 - Web Services
 - Management
 - Deployment
- Ваше мнение - зачем все эти сервисы?

СЕРВЕРНЫЕ ОБЪЕКТЫ, КОНТЕЙНЕРЫ

Жизненный цикл серверных объектов

- Жизненным циклом серверных объектов управляет **контейнер**



- Объекты, чьим жизненным циклом управляет контейнер, еще называют "managed beans"

Как создать серверный объект?

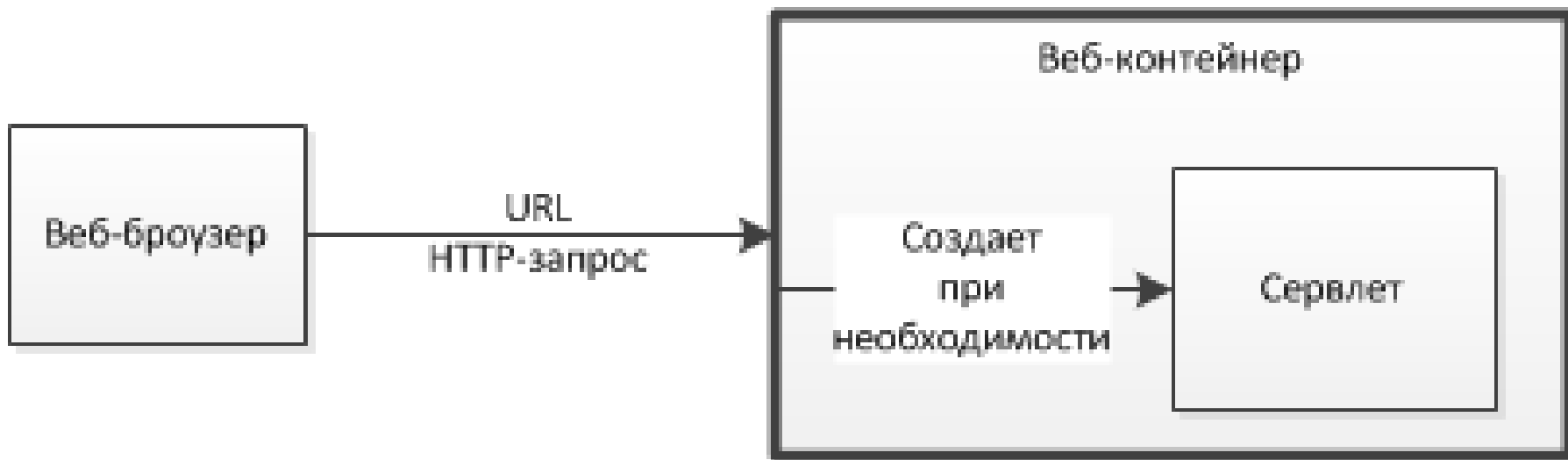
Вспомним типы серверных объектов. Как их создать?

~~new Servlet()~~

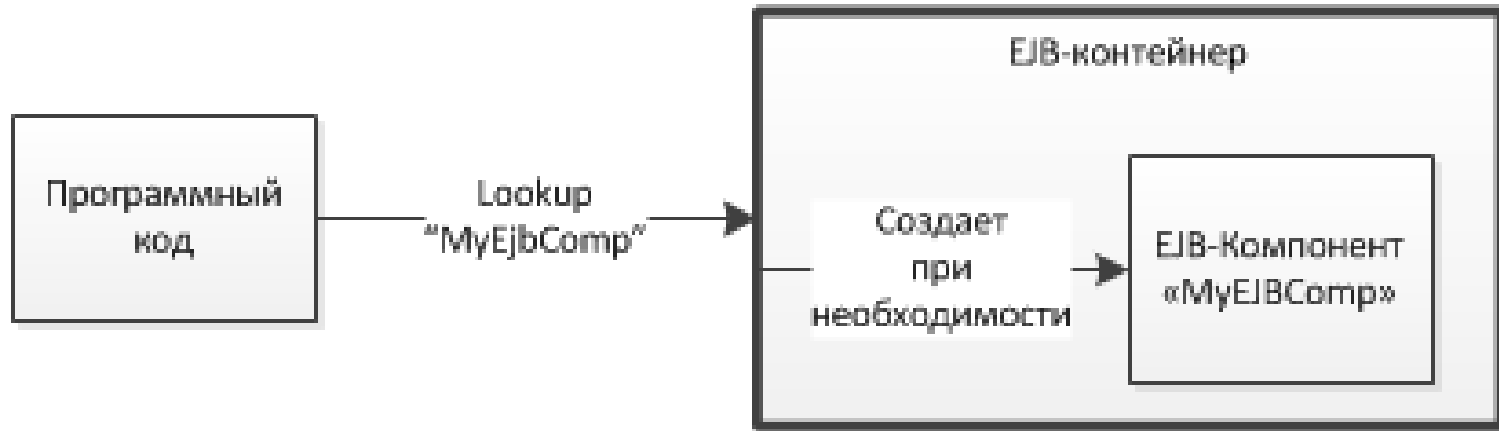
~~new EjbComponent()~~

~~new IocContainerManagedObject()~~

Как вызвать сервлет?



Как создать Ejb-компонент?



`@Stateless MyEjbComponent implements MyEjbCompLocal {...}`

Способ 1:

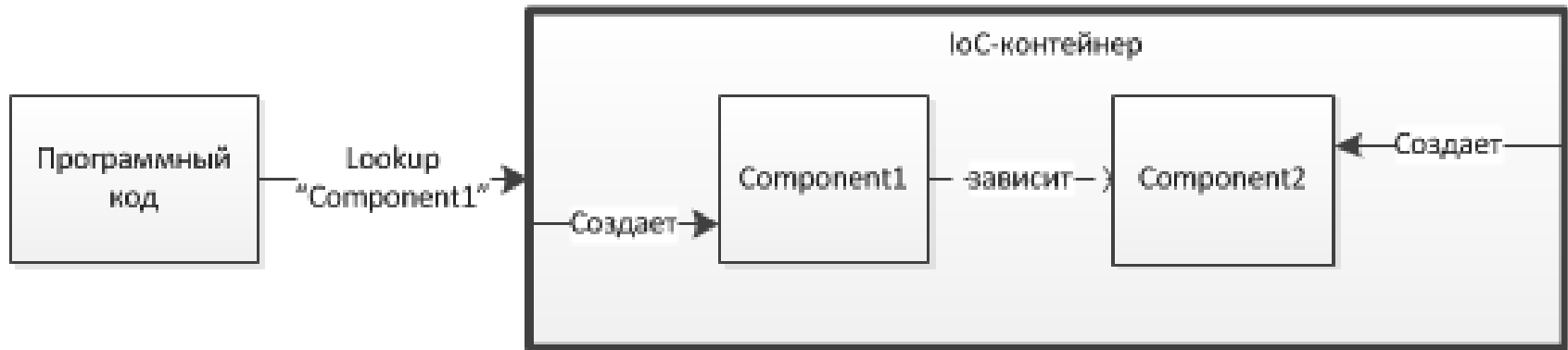
```
public Test {  
    public foo () throws NamingException {  
        MyEjbCompLocal my = new InitialContext().lookup("NameOfComponent");  
        ... }  
}
```

Способ 2:

```
public Test {  
    @EJB MyEjbCompLocal comp;  
}
```

Способ 2 работает, если жизненным циклом объекта Test управляет контейнер

Как создать Spring-компонент?



```
public class Component 1 {
    private Component 2;
    public void setComponent2(Component2 c) {...}
}
```

Создание экземпляра Component1

```
ApplicationContext appContext = new
    ClassPathXmlApplicationContext(
        "/application-context.xml");
Component1 cmp1 = (Component1)
appContext.getBean("cmp1");
```

Конфигурация

(application-context.xml):

```
<bean id="cmp1" class="Component1">
    <property name="component2">
        <ref bean="cmp2"/>
    </property>
</bean>
<bean id="cmp2" class="Component2"/>
```

Типы контейнеров

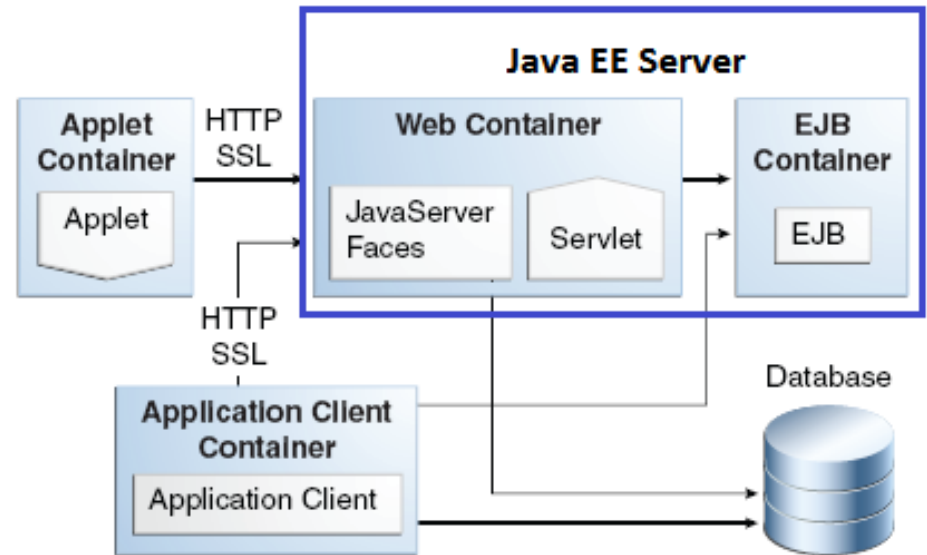
- Java EE контейнеры:

- Веб-контейнер
- EJB-контейнер
- Application Client Container
- Applet Container

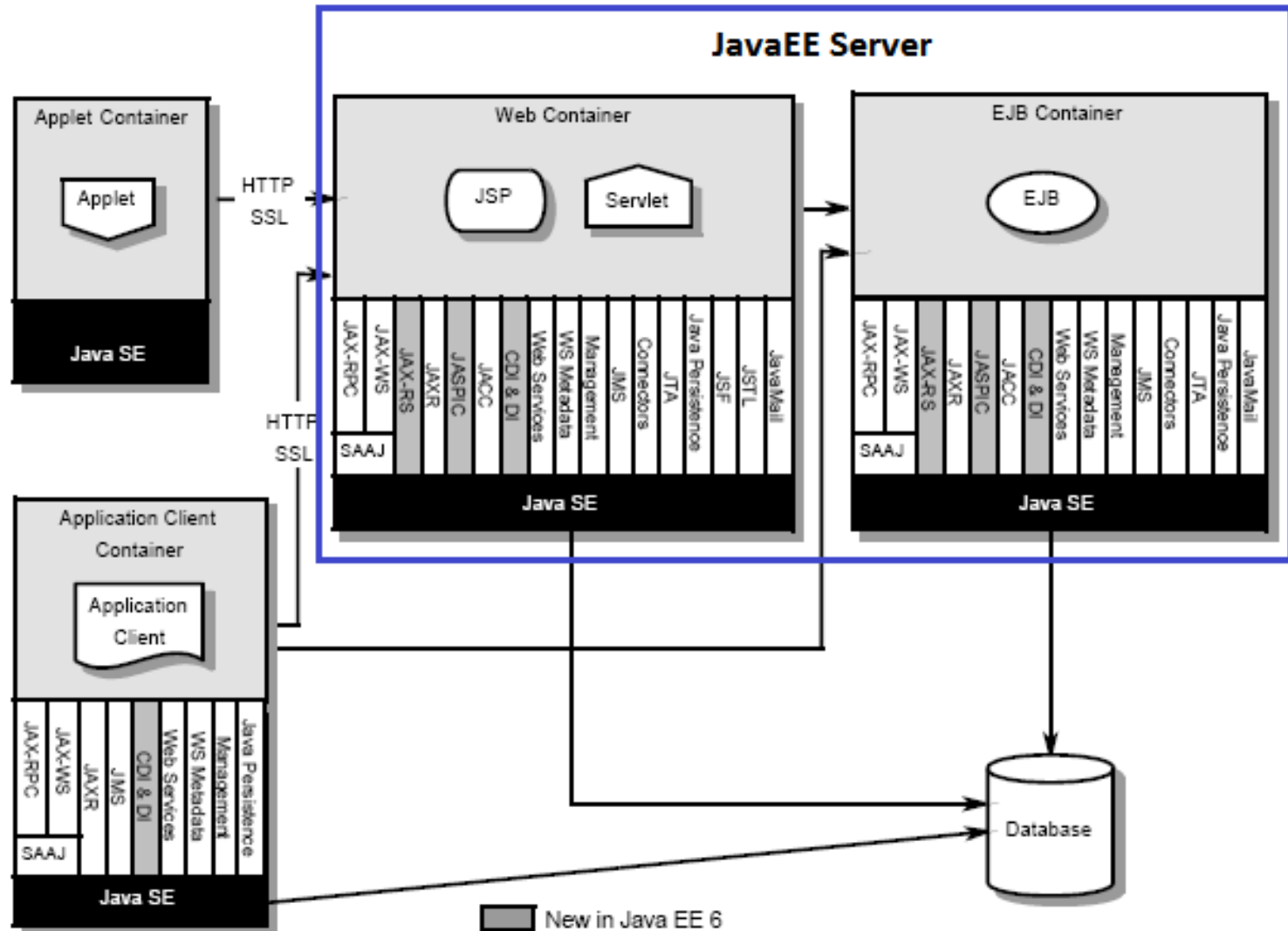
- IoC-контейнеры:

- Spring IoC Container
- PicoContainer
- Google Guice
- и другие

- Java EE 6 имеет IoC-контейнер "CDI" (Contexts and Dependency Injection)

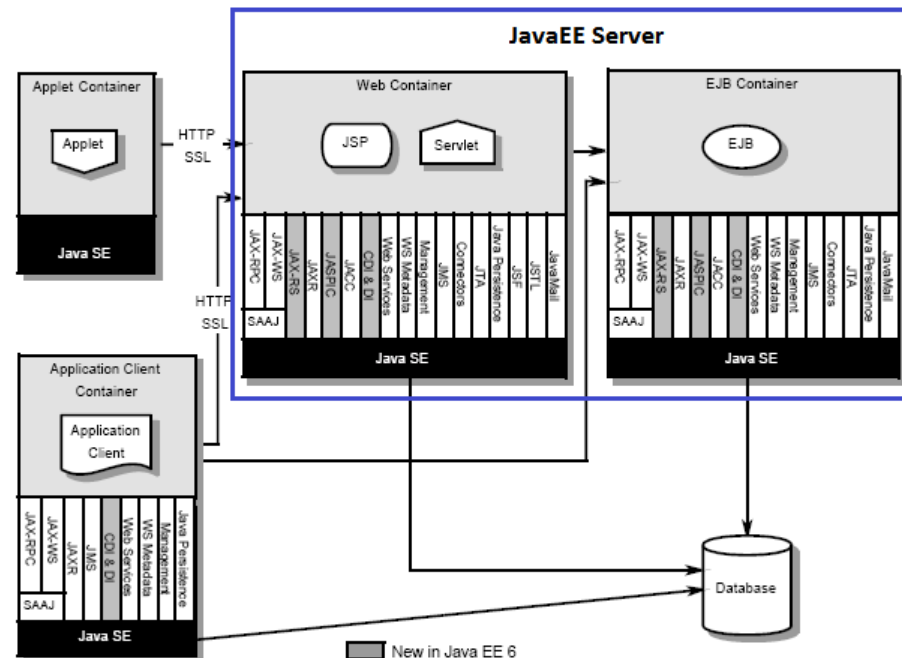


Контейнеры Java EE



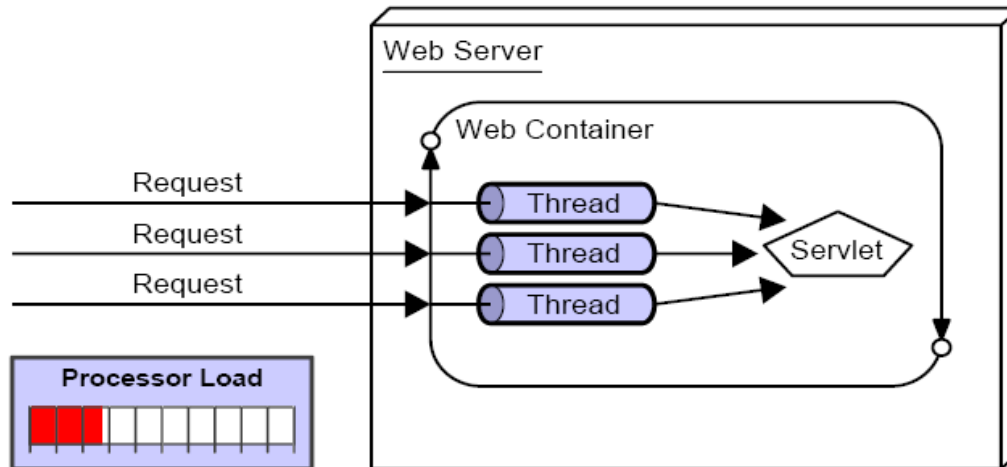
Задачи Java EE контейнеров . Задача 2 (из 3)

- Задача 2.
 - Обеспечить работоспособность системы под высокой нагрузкой
- Это задача для веб-контейнера, EJB-контейнера



- Ваши предположения по способу реализации

- Как веб-контейнер «выдерживает» высокую нагрузку:
 - Thread-pool для HTTP-запросов
 - Создается только один экземпляр любого сервлета. По сути сервлет – это синглтон



- Как EJB-контейнер обеспечивает работоспособность EJB-компонент под высокой нагрузкой?
- Stateless Session Beans
 - Поддерживается пул EJB-компонентов
- Statefull Session Beans
 - Поддерживается механизм «пассивации/активации»

- Задача 3.
- Задача для EJB-контейнера - обеспечить enterprise-возможности EJB-компонентов:
 - Декларативное управление транзакциями на уровне методов
 - Декларативное управление безопасностью на уровне методов
 - Вызов методов
 - удаленный
 - как веб-сервиса
 - по таймеру
 - асинхронный (Java EE 6)
 - Конкурентный доступ к EJB-компонентам (EJB-компоненты – thread-safe)
 - JMX-мониторинг EJB-компонентов

- Application Server
 - Веб-контейнер
 - EJB-контейнер
- Web Server
 - Веб-контейнер

(Java web server = servlet container)

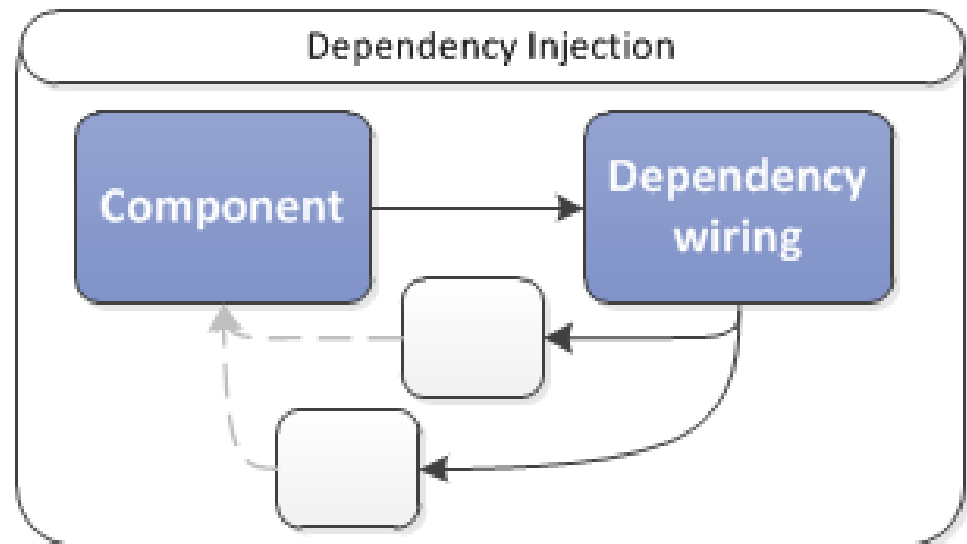
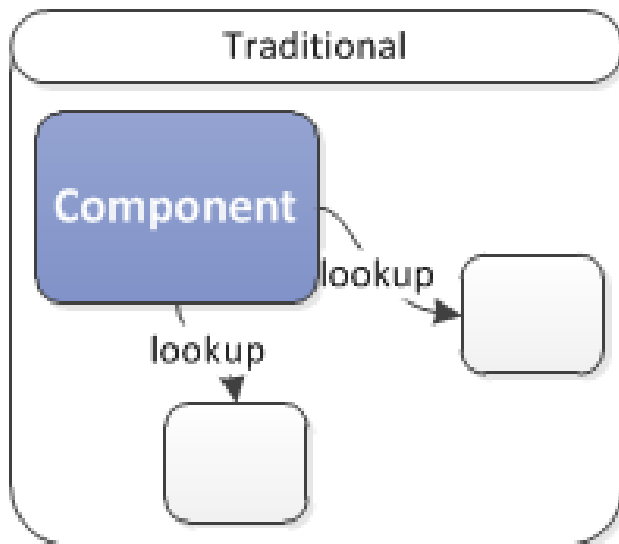
IOS-КОНТЕЙНЕРЫ

В чем состоит «Inversion of control»?

- Речь идет о объектах, чьим жизненным циклом управляет контейнер (managed bean)
- Объект может быть жестко «привязан» к контейнеру:
 - Реализация специфического интерфейса
 - Программный lookup зависимых “managed” объектов с помощью специального API контейнера
- IoC помогает «отвязать» объект от контейнера
 - Не нужно реализовать специфический интерфейс
 - Не нужно делать lookup зависимых “managed” объектов на основе API
- **Инверсия контроля состоит в том, что объект не «привязан» к контейнеру и зависимостями управляет сам контейнер с помощью **dependency injection**.**

Dependency Injection

- Когда контейнер инстанцирует компонент, он «собирает» все его зависимости
- «Сборка» отделена от компонента
 - в коде компонента нет обращений к контейнеру с просьбой lookup зависимых компонентов



- IoC-контейнеры:
 - Spring IoC Container
 - PicoContainer
 - Google Guice
 - другие
- Java EE 6 имеет IoC-контейнер “CDI” (Contexts and Dependency Injection)

- Ваше мнение?

- Убирают hard-code dependencies
- Test Driven Development
 - Возможность протестировать программный код без погружения в сервер приложений
 - Резко увеличивается скорость тестирования (тестирование можно оперативно проводить при сборке приложения)

- ORB Basics.
http://www.omg.org/gettingstarted/orb_basics.htm
- **Ke Jin.** Why and what of Inversion of Control.
<http://www.pocomatic.com/docs/whitepapers/ioc/>
- **Martin Fowler.** Inversion of Control Containers and the Dependency Injection pattern. -
<http://martinfowler.com/articles/injection.html#InversionOfControl>
- The Spring Framework - Reference Documentation. Chapter 3. IoC Container.
<http://static.springsource.org/spring/docs/2.0.x/reference/beans.html>