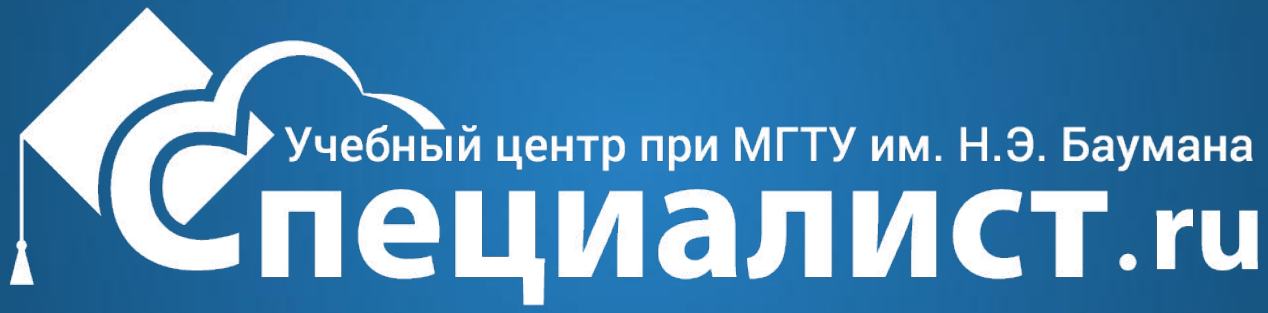


Java: Технология Enterprise Java Beans

Спецификация Java EE.

Java Platform, Enterprise Edition (Java EE)
https://javaee.github.io/javaee-spec/

JavaEE – спецификация, которая объединяет и интегрирует множество спецификаций.
Цель JavaEE - обеспечить разработку приложения и его развертывание на любом сервере приложения без изменения кода и конфигурационных файлов.

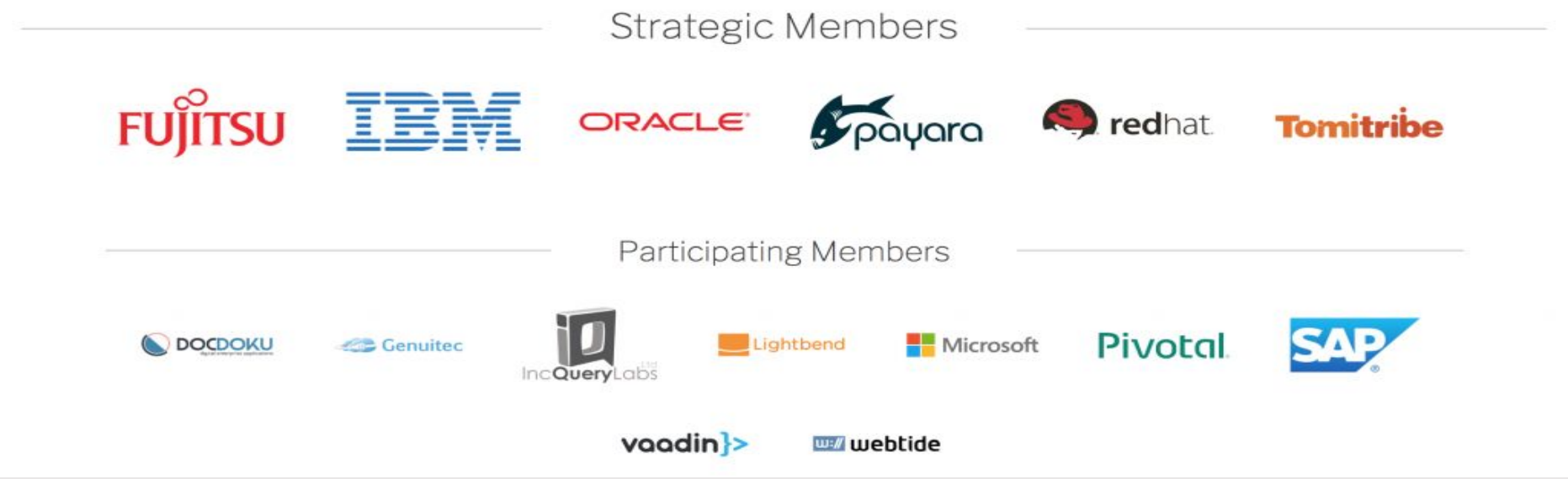
JavaEE – **сервер приложений** (application server) должен реализовывать спецификации JavaEE.
Приложения JavaEE сохраняют полную совместимость с предыдущими версиями, позволяя вашему приложению перемещаться до более новых версий сервера приложений без особых проблем.

JCP. Настоящие и будущее Java EE (JAKARTA EE).

Спецификации **Java EE 7** (релиз 2011 год).
Спецификации **Java EE 8** (релиз 2017 год).

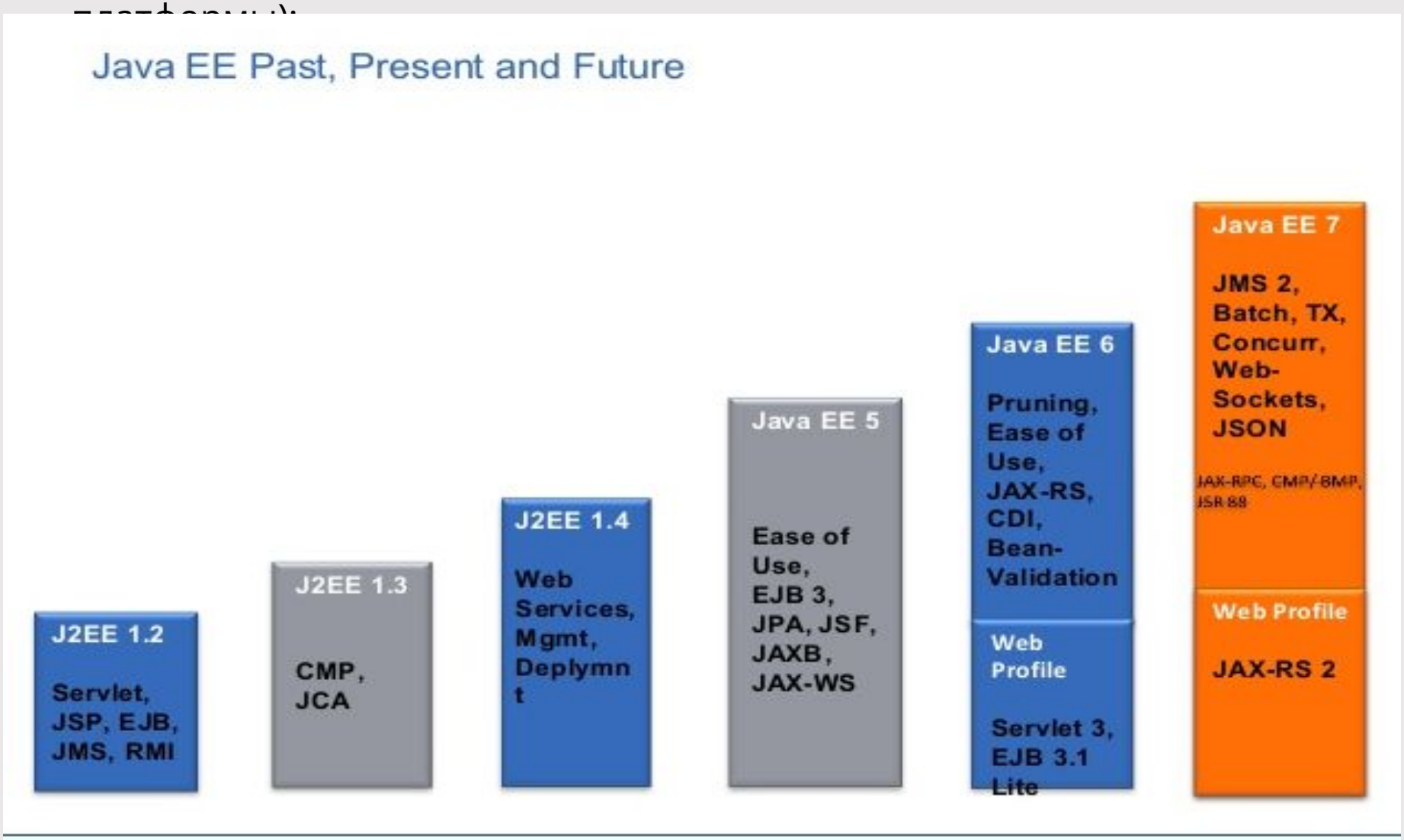
Java Community Process (JCP) - открытая организация созданная компанией **Sun Microsystems**.
Направление работы - определение будущих версий и функционала платформы Java.
JSR - запрос на спецификацию.
После одобрения спецификация выпускается для внедрения.

Jakarta EE Working Group - группа по развитию платформы Enterprise Java созданная в организации Eclipse Foundation.



Настоящие и будущее Java EE (JAKARTA EE).

Развитие технологии (платформы) Java Enterprise Edition (версии платформы):



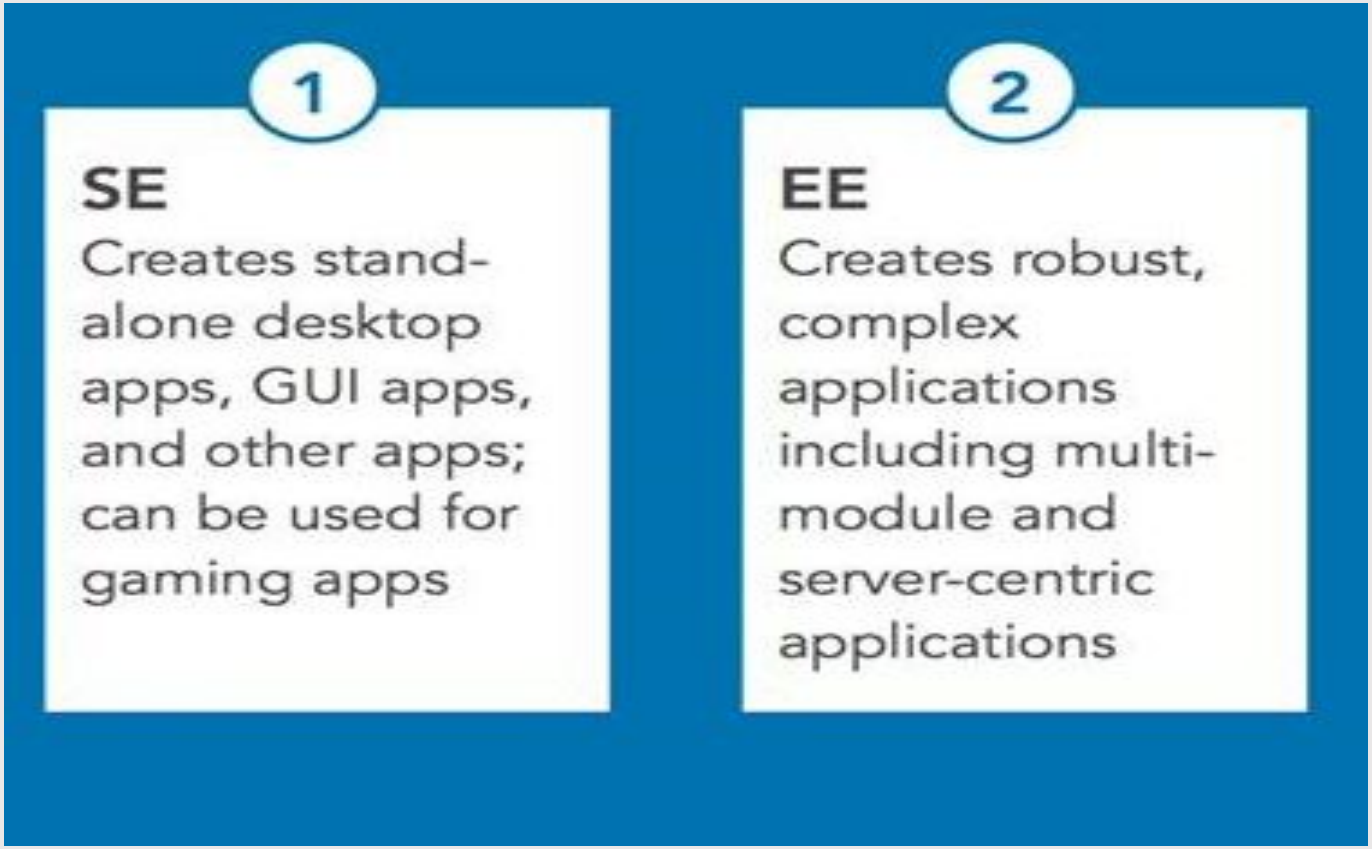
Настоящие и будущее Java EE (JAKARTA EE).

Jakarta EE Working Group - группа по развитию платформы Enterprise Java созданная в организации Eclipse Foundation.

Спецификация JavaEE.

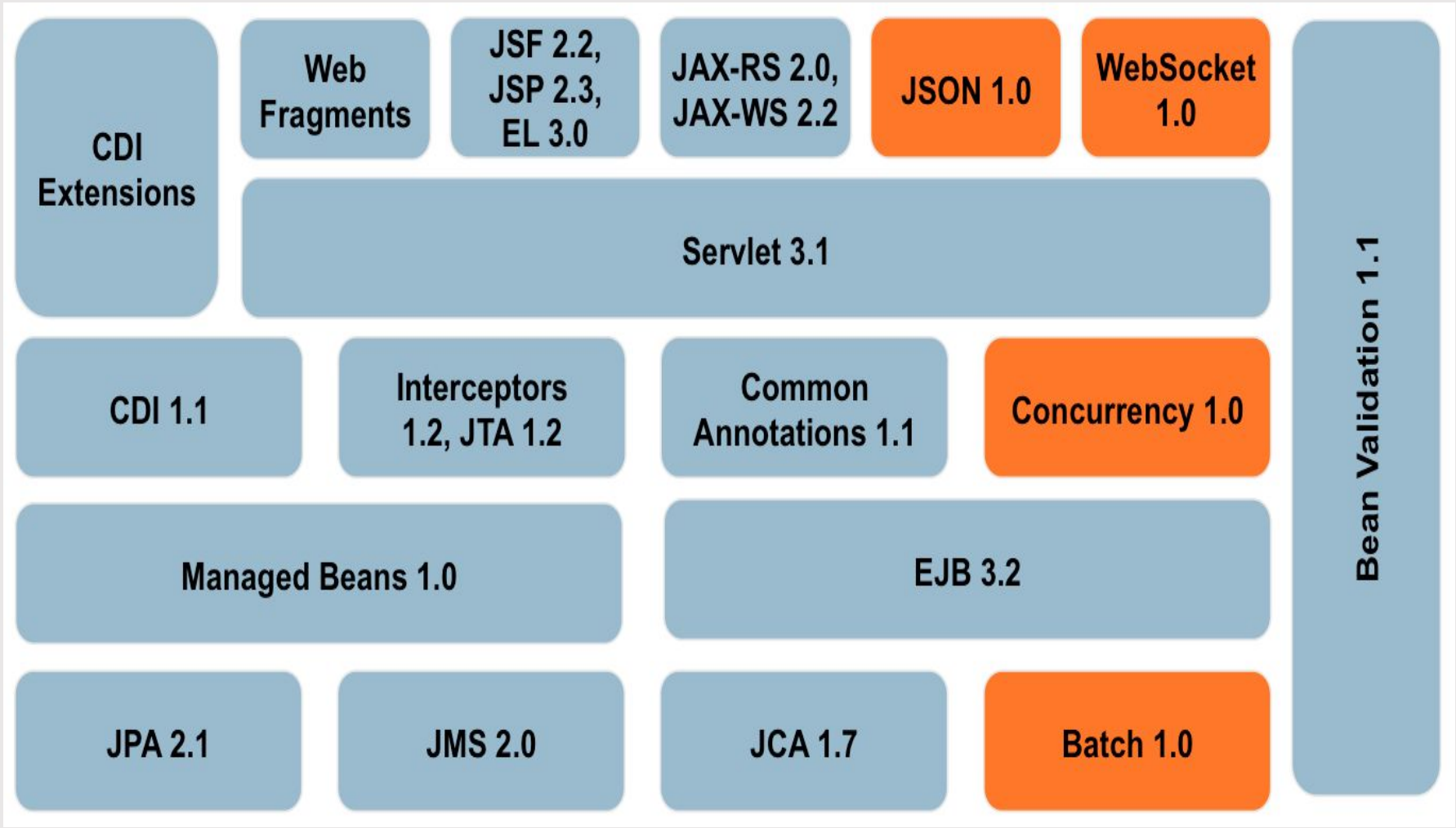
JavaEE – это расширенная версия **Java SE**.
В Java EE доступны все возможности языка Java, а также API Java SE.

Различия между Java SE и Java EE приложениями:



Спецификация JavaEE 7.

JavaEE 7 предлагает несколько новых спецификаций:
Batch, WebSocket, JSON, Concurrency, а также совершенствует существующие.

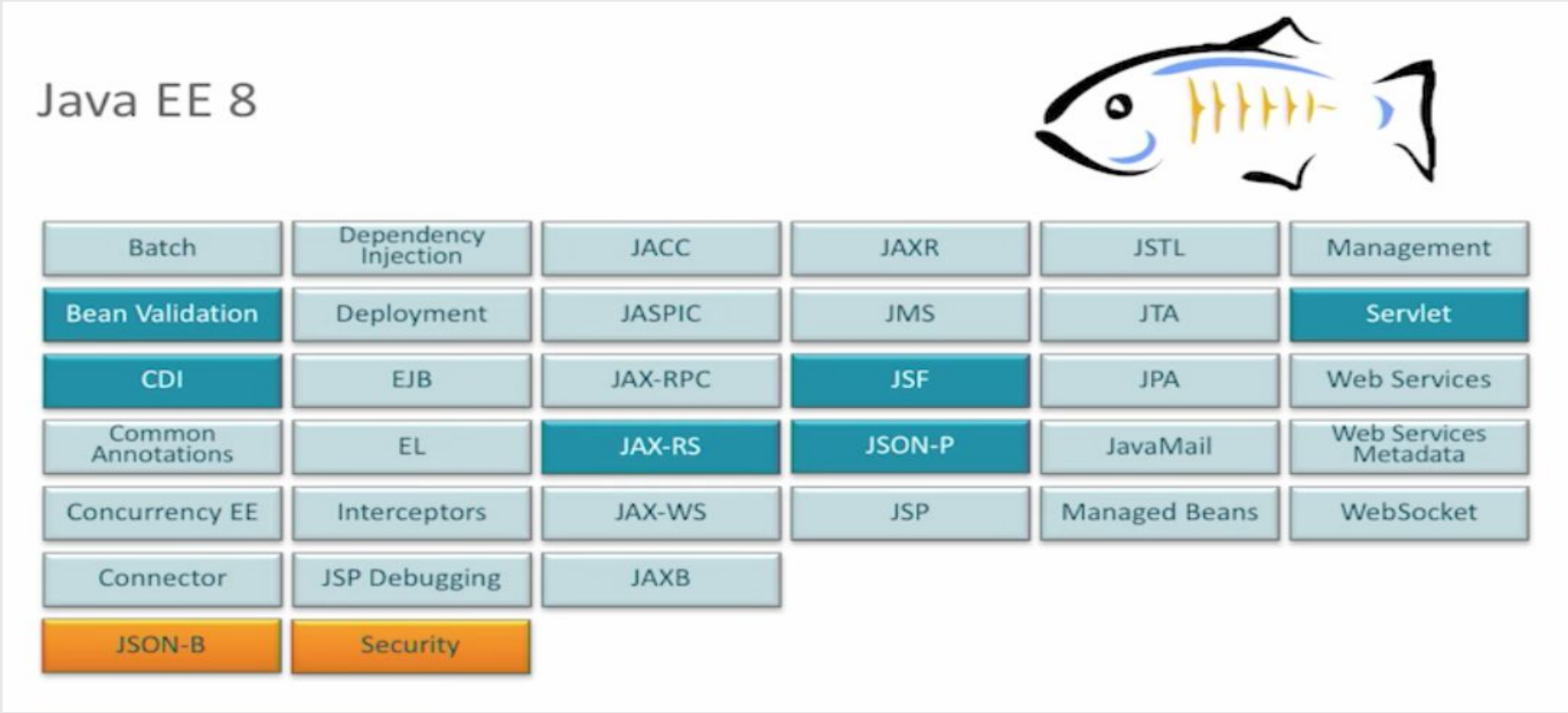


Спецификация JavaEE 8.

Java EE 8 - спецификация:
https://javaee.github.io/javaee-spec/download/JavaEE8_Platform_Spec_FinalRelease.pdf

Java EE 8 API:
https://javaee.github.io/javaee-spec/javadocs/

JavaEE 8 предлагает несколько новых спецификаций:



Спецификация JavaEE. Состав спецификации.

JavaEE - объединяющая спецификация:
http://www.oracle.com/technetwork/java/javaee/tech/index-jsp-142185.html

Спецификации (группы технологий):

- **Web Application Technologies** (Веб)
- **Enterprise Application Technologies** (Корпоративные)
- **Web Services Technologies** (Веб-службы)
- **Management and Security Technologies** (Управление, безопасность)
- **Java EE-related Specs in Java SE** (вспомогательные)

Состав спецификации JavaEE.

Web Application Technologies:
Java API for WebSocket
Java API for JSON Processing
Java Servlet
JavaServer Faces
Expression Language
JavaServer Pages
Standard Tag Library for JavaServer Pages (JSTL)

Enterprise Application Technologies:
Batch Applications for the Java Platform
Concurrency Utilities for Java EE
Contexts and Dependency Injection for Java
Dependency Injection for Java
Enterprise JavaBeans
Interceptors
Java EE Connector Architecture
Java Persistence
Common Annotations for the Java Platform
Java Message Service API
Java Transaction API (JTA)
JavaMail

Состав спецификации JavaEE.

Web Services Technologies:
Java API for RESTful Web Services (JAX-RS)
Implementing Enterprise Web Services
Java API for XML-Based Web Services (JAX-WS)
Web Services Metadata for the Java Platform
Java API for XML-Based RPC (JAX-RPC) (Optional)
Java APIs for XML Messaging
Java API for XML Registries (JAXR)

Management and Security Technologies:
Java Authentication Service Provider Interface for Containers
Java Authorization Contract for Containers
Java EE Application Deployment (Optional)
J2EE Management
Debugging Support for Other Languages

Java EE-related Specs in Java SE:
Java Architecture for XML Binding (JAXB) 2.2
Java API for XML Processing (JAXP) 1.3
Java Database Connectivity 4.0
Java Management Extensions (JMX) 2.0
JavaBeans Activation Framework (JAF) 1.1
Streaming API for XML (StAX) 1.0

JavaEE - сервер приложений.

JavaEE-сервер приложений - комплекс программ, реализующих концепцию (спецификацию) JavaEE и позволяющих запускать JavaEE-приложения.

Application Servers (поддерживают спецификацию Java EE):

- Apache TomEE
- GlassFish / Payara
- Wildfly / JBoss EAP (Red Hat)
- Weblogic (Oracle)
- IBM WebSphere Application Server (IBM)
- SAP NetWeaver Application Server (SAP)

Java based HTTP (Web) server and Java Servlet container
(частично поддерживают спецификацию JavaEE - light Java EE):

- Apache Tomcat
- Jetty

JavaEE - сервер приложений.

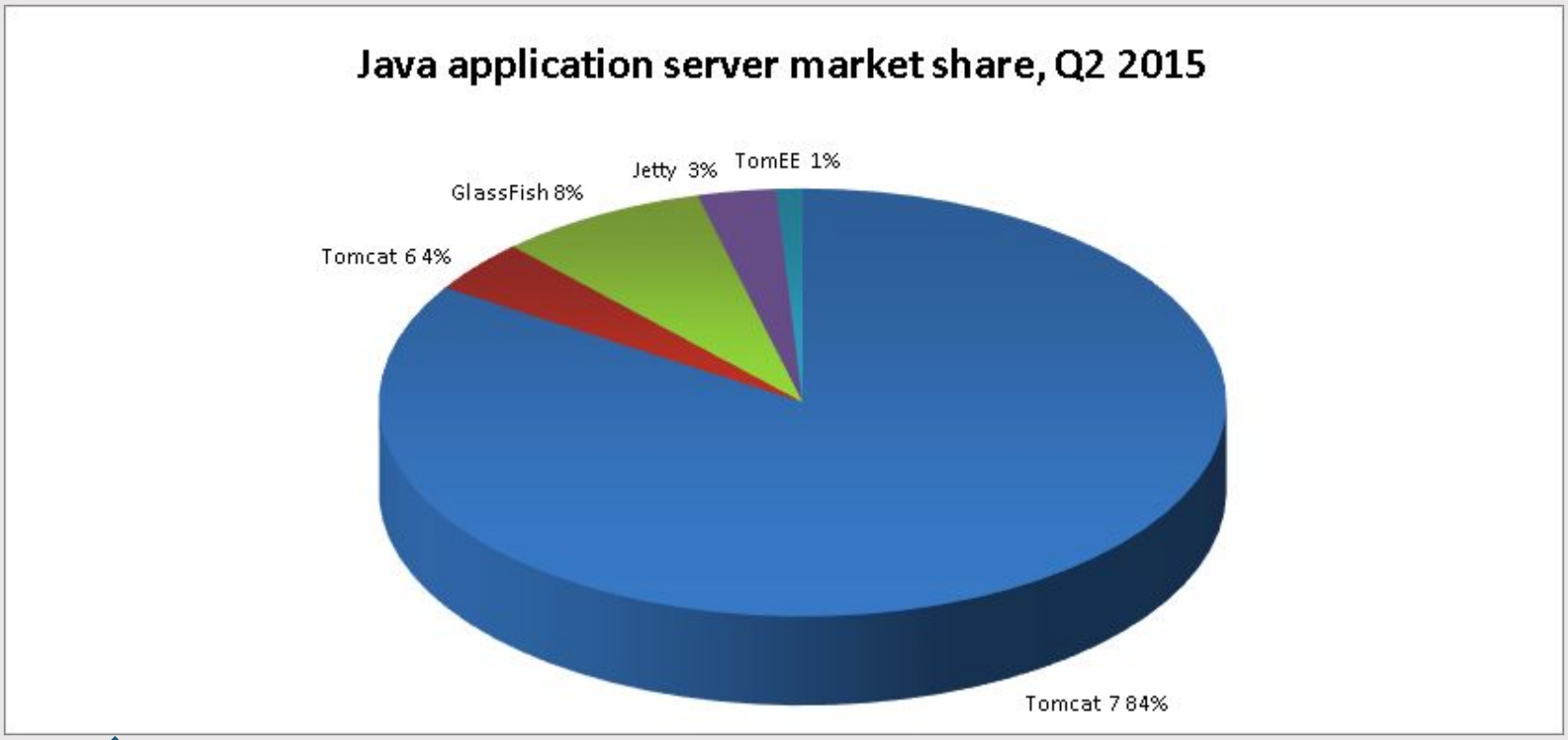
Application Servers (поддерживают спецификацию Java EE) – “большая пятерка”:

- Apache TomEE
- GlassFish / Payara
- Wildfly / JBoss EAP (Red Hat)
- Weblogic (Oracle)
- IBM WebSphere Application Server (IBM)

JavaEE - сервер приложений.

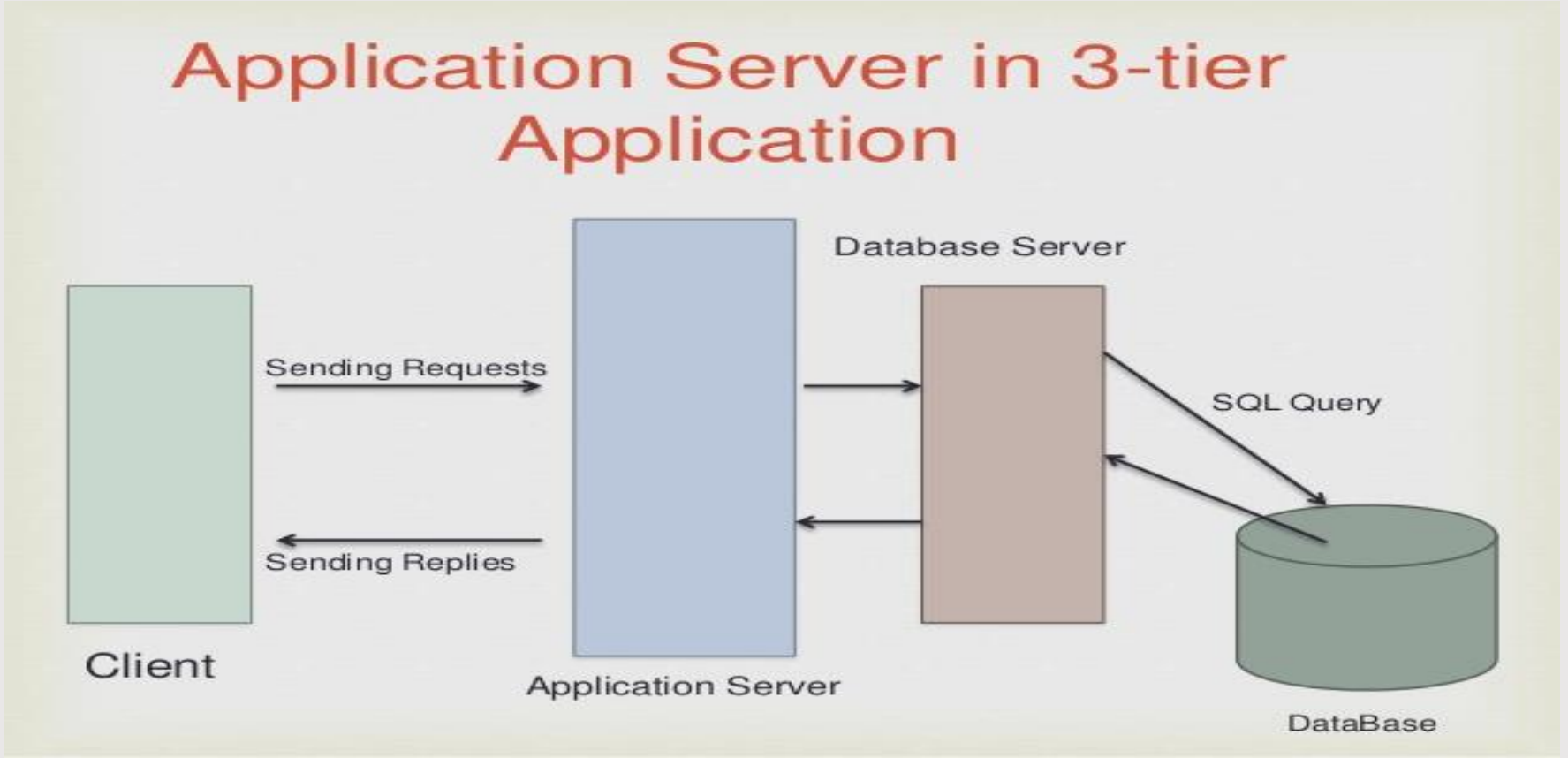
Application Servers (поддерживают спецификацию Java EE) – “большая пятерка”:

- Apache TomEE
- GlassFish / Payara
- Wildfly / JBoss EAP (Red Hat)
- Weblogic (Oracle)
- IBM WebSphere Application Server (IBM)



JavaEE - сервер приложений. 3-х уровневая архитектура.

Трёхуровневая архитектура (**3-Tier**) — архитектурная модель, предполагающая наличие трёх компонентов:
клиента,
сервера приложений (к которому подключено клиентское приложение) и
сервера баз данных (с которым работает сервер приложений).



Разработка приложения на платформе JavaEE.

Java SE - необходимые знания и API:

1) ООП
2) Аннотации (annotation)
3) Обработка исключений (exception handling)
4) JNDI API, JDBC API

Java SE 8 API (документация)**:**
https://docs.oracle.com/javase/8/docs/api/

Java EE спецификации:

- Web Application Technologies (Веб)
- Enterprise Application Technologies (Корпоративные)
- Web Services Technologies (Веб-службы)
- Management and Security Technologies (Управление, безопасность)
- Java EE-related Specs in Java SE (вспомогательные)

Java EE API (документация)**:**
https://javaee.github.io/javaee-spec/javadocs/

Разработка приложения на платформе JavaEE.

ПО для создания Java Enterprise-приложений:

- Java SE Development Kit 8 **(JDK 8u201+)**
- Apache Ant, Apache Maven, Gradle - фреймворки для автоматизации сборки проектов
- Интегрированная среда разработки (IDE): Netbeans, Eclipse
- JavaEE - сервер приложений
- СУБД (Oracle, PostgreSQL, MySQL/MariaDB, Microsoft SQL Server)

Сборка Web/Enterprise-приложений:

Сборка Java Web/Enterprise приложения осуществляется с помощью инструмента Apache Maven.
Информация для сборки проекта содержится в XML-файле **pom.xml**.
С помощью pom.xml конфигурируются зависимости от других проектов, индивидуальные фазы процесса построения проекта (build process).
Результат сборки WAR-файл – файл с расширением **.war** (файл в формате JAR).

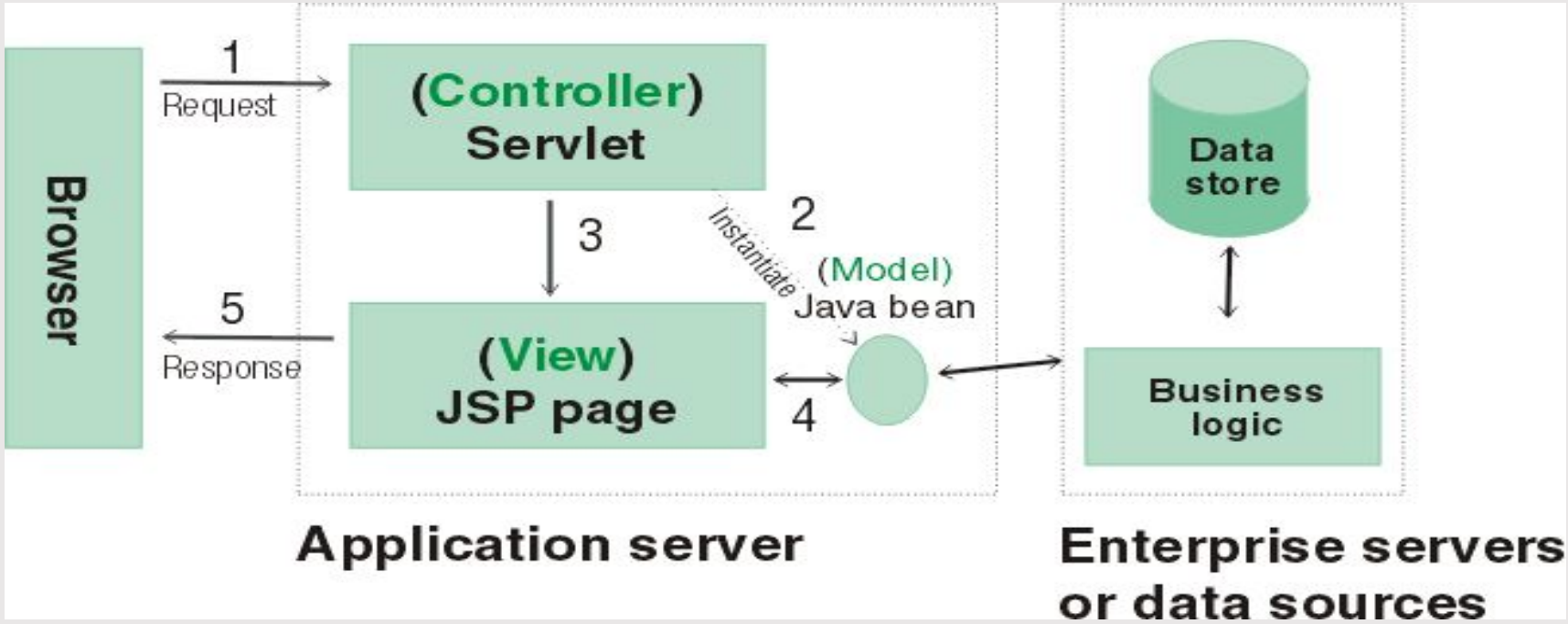
JavaEE-сервера приложений поддерживают развертывание приложения на сервере из WAR-файлов!

Разработка Web-приложения.

Веб-приложение — клиент-серверное приложение, в котором клиентом выступает **браузер**, а сервером — **веб-сервер**.

Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети.

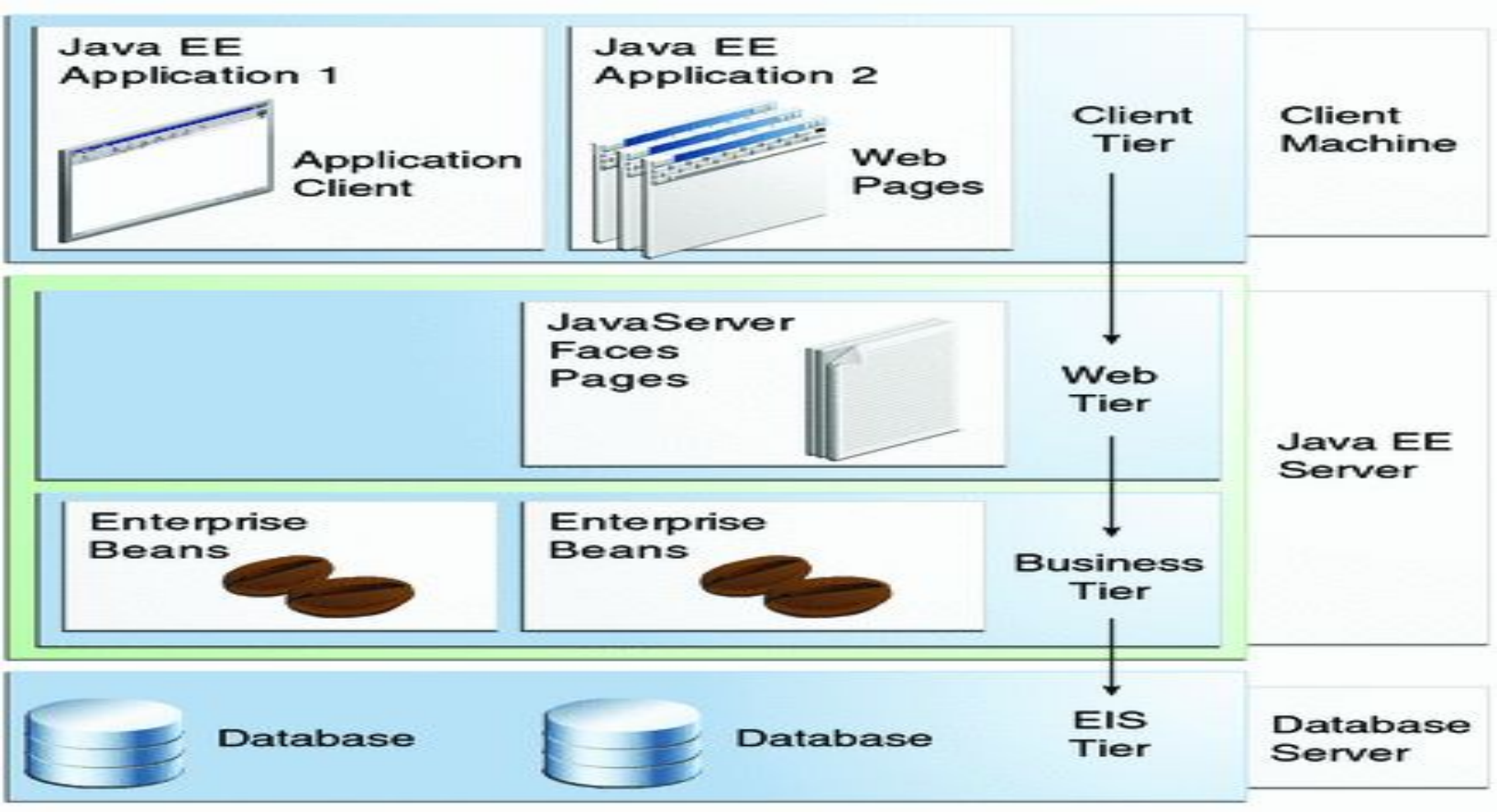
Клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами.



Технология Enterprise JavaBeans (EJB).

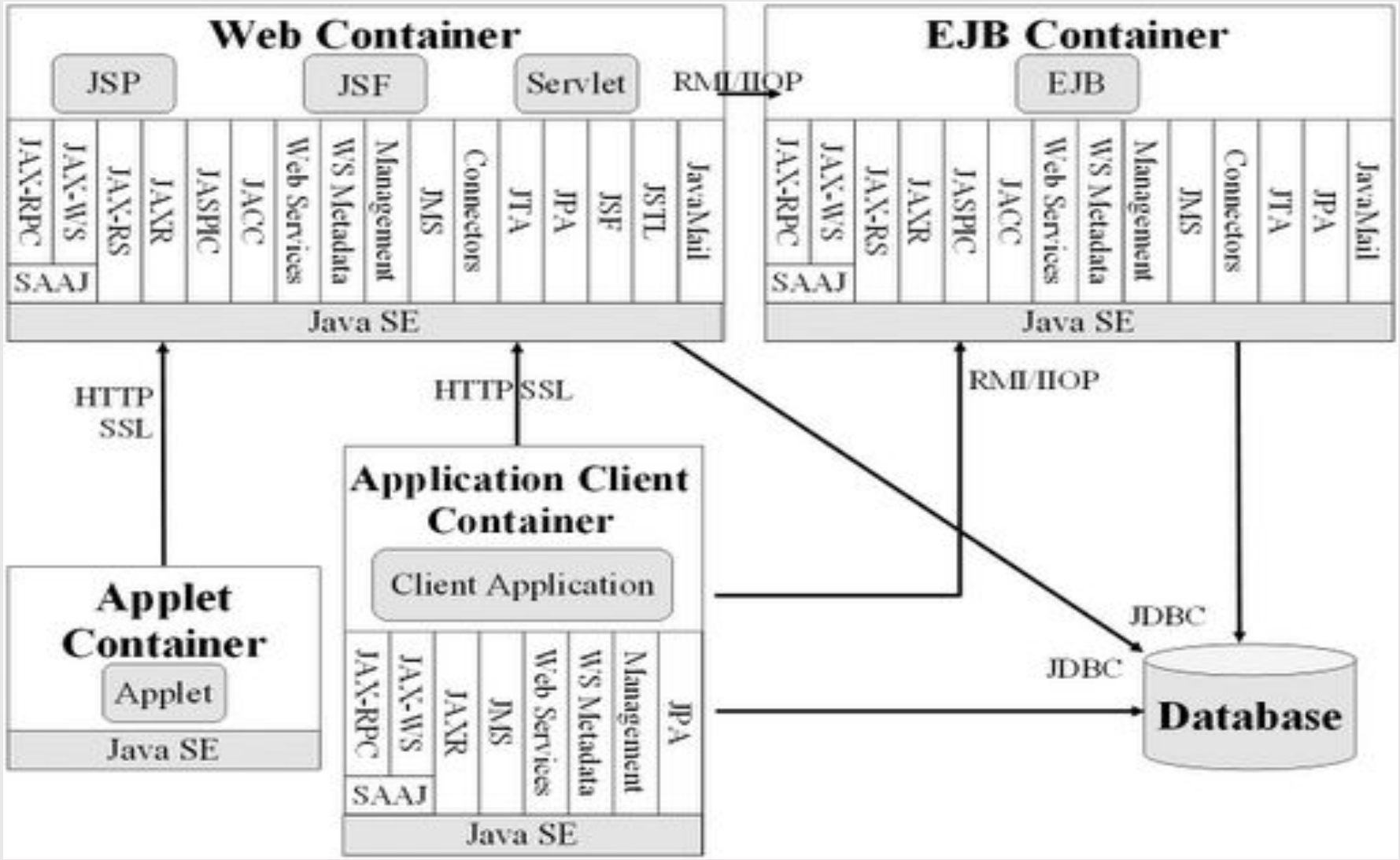
Enterprise JavaBeans (**EJB**) - спецификация технологии написания и поддержки серверных компонентов, содержащих бизнес-логику. Является частью **JavaEE**.

Уровни бизнес-логики (**Business Tier**) и представления (**Web Tier**) в JavaEE- приложении на сервере приложений (**JavaEE Application Server**):



Контейнеры JavaEE.

Web и EJB контейнеры:



Контейнеры JavaEE и сетевые протоколы.

Компоненты в контейнерах могут вызываться с помощью различных протоколов.
Протоколы поддерживаемы в JavaEE:

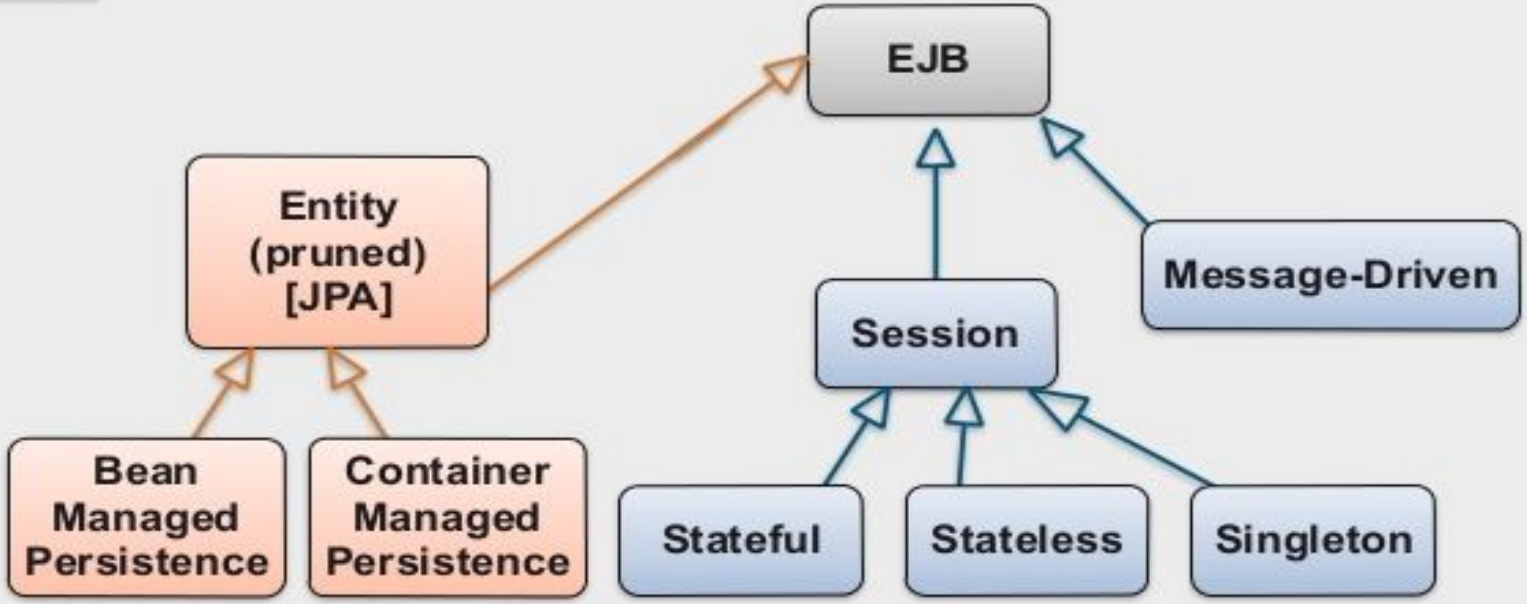
- HTTP - веб протокол
  Серверный API для работы с HTTP определяется сервлетами, JSP страницами, JSF, веб-службами SOAP и RESTful

- HTTPS - расширение веб протокола HTTP для поддержки шифрования в целях повышения безопасности : HTTP + SSL

- RMI-IIOP – удаленный вызов методов (RMI)
  Представляет собой расширение технологии RMI, которое используется для интеграции с архитектурой CORBA.

Технология Enterprise JavaBeans (EJB).Типы компонент.

- **Session** (сессионные)
- **Message-Driven** (управляемые сообщениями)



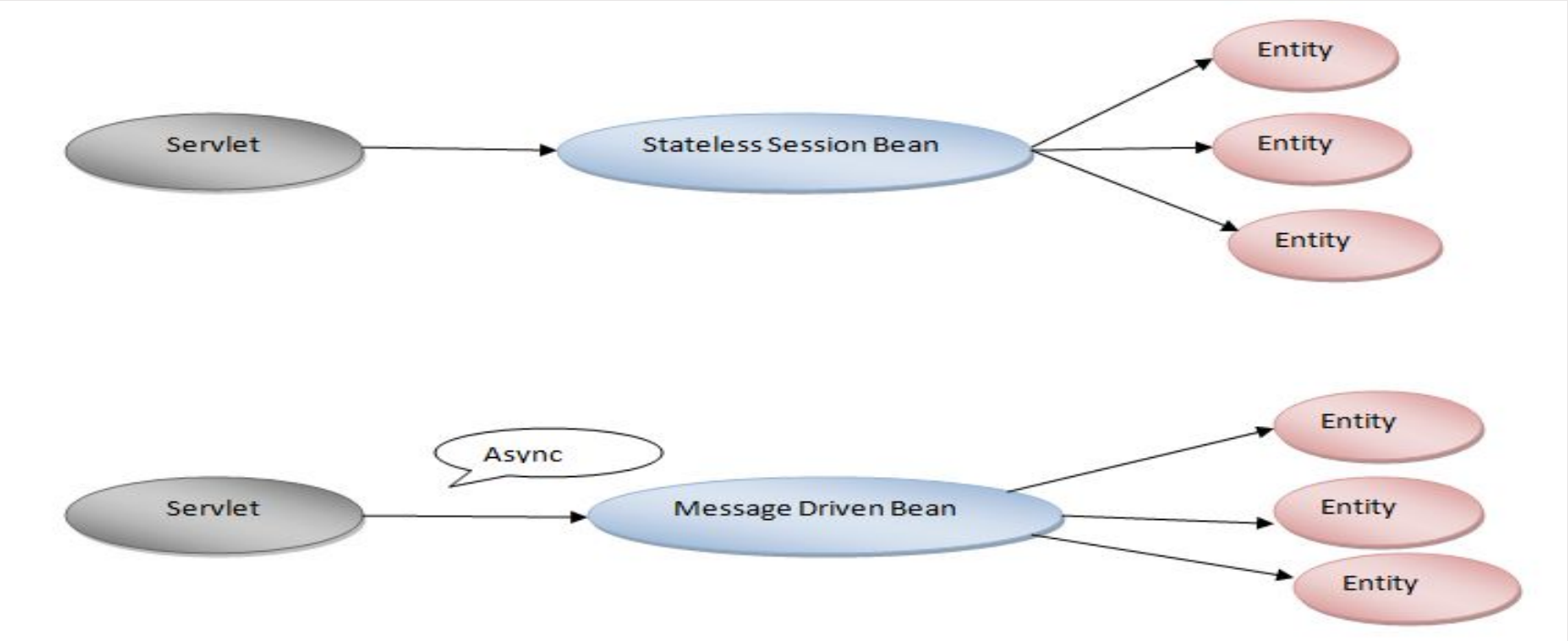
Bean Type	*Annotation*
Session Bean	*Performs a task for a client; optionally, may implement a web service*
Message-driven Bean	*Acts as a listener for a particular messaging type, such as the Java Message Service API*

Технология Enterprise JavaBeans (EJB).

Когда использовать Enterprise JavaBeans (EJB) ?

- Масштабируемость (scalability)
- Поддержка транзакций (transaction management)
- Подключение различных клиентов (diversity of clients)

Какие типы компонент использовать при создании JavaEE-приложения (Stateless, Stateful, Message-Driven)?



Технология Enterprise JavaBeans (EJB). Session Beans.

Сессионные EJB компоненты (Session Beans):

- **Stateless** - компоненты без сохранения состояния
- **Stateful** - компоненты с сохранением состояния
- **Singleton** - существует только один экземпляр компонента

Технология Enterprise JavaBeans (EJB). Session Beans.

- **Stateless** - компоненты без сохранения состояния
- **Singleton** - существует только один экземпляр компонента

EJB компоненты не сохраняющие состояние (**Stateless**) и синглтоны (**Singleton**) похожи друг на друга, тем что они не поддерживают диалога с выделенным клиентом (**conversational state**).
Оба типа компонента позволяют получать доступ любому клиенту!

Синглентон (**Singleton**) предоставляет доступ к одному экземпляру компонента.

Компоненты обоих типов (**Stateless и Singleton**) имеют одинаковый жизненный цикл (**lifecycle**).

Технология Enterprise JavaBeans (EJB). Session Beans.

- **Stateless** - компоненты без сохранения состояния

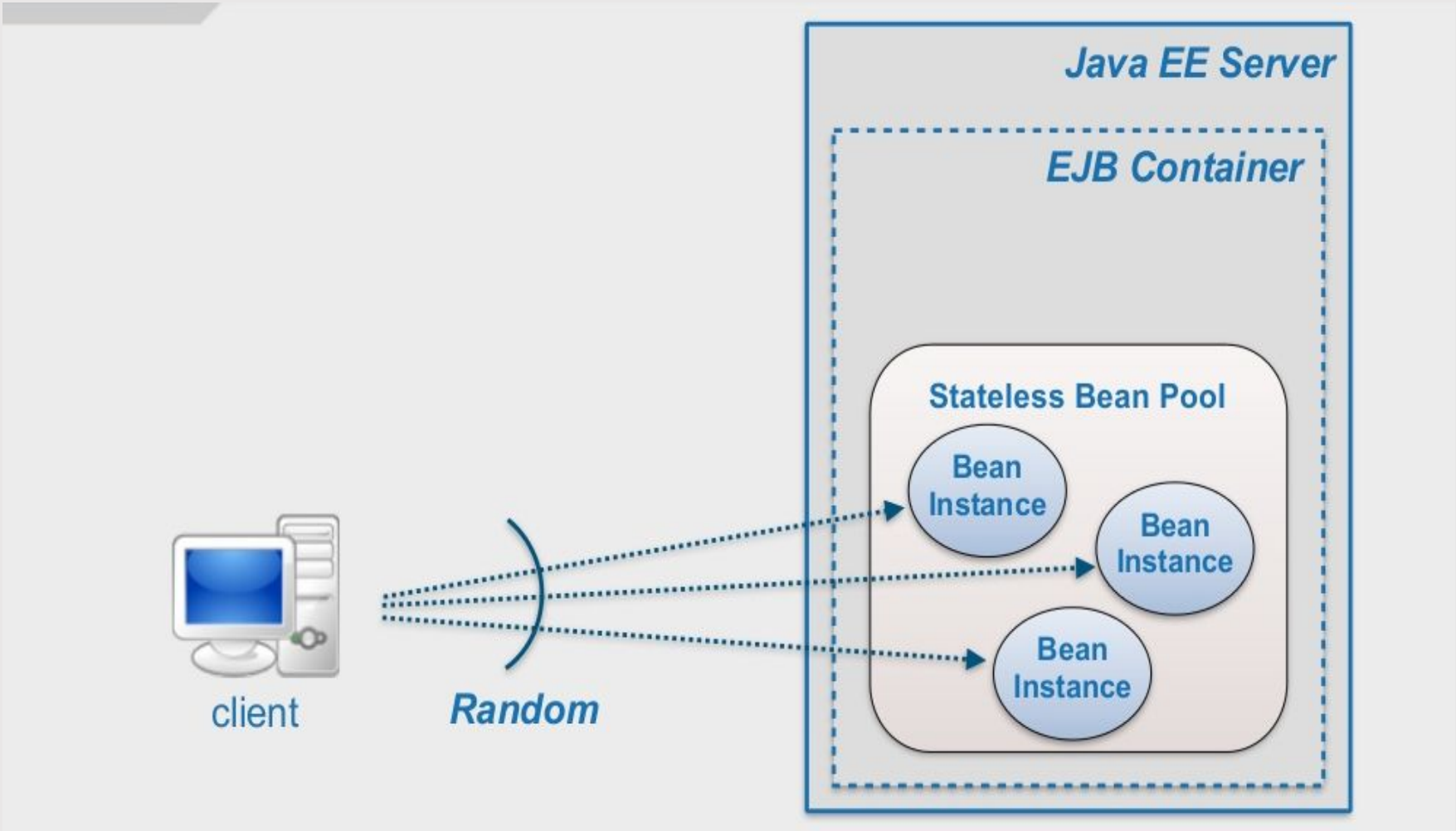
Используется **@Stateless** – аннотация для класса компонента

Аннотации методов класса компоненты без сохранения состояния:

- @PostConstruct
- @PreDestroy

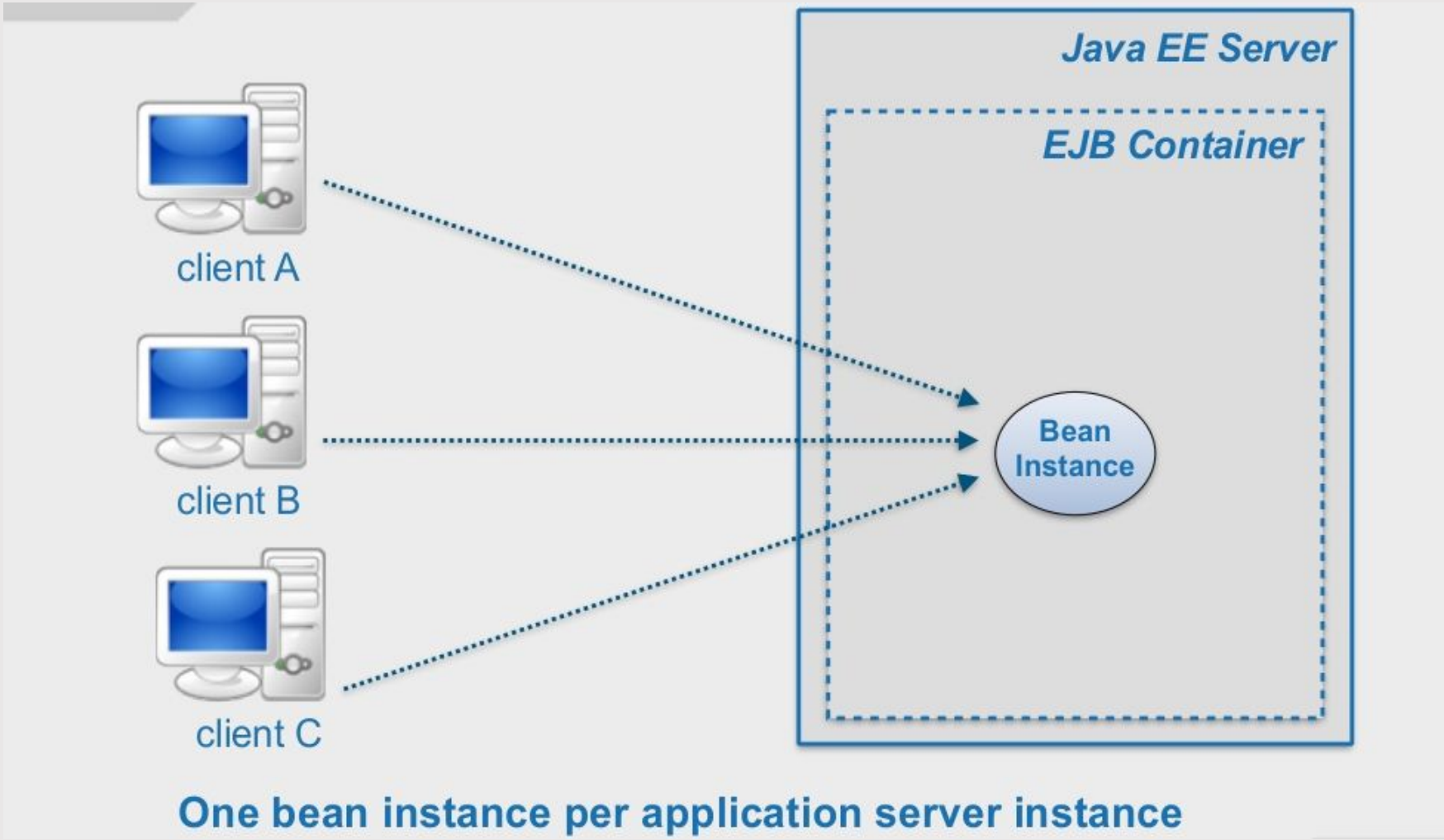
EJB. Stateless Session Beans.

Stateless компоненты (Stateless Session Beans):



EJB. Singleton Session Beans.

Singleton компоненты (Singleton Session Beans):



EJB. Stateful Session Beans.

Stateful - компоненты с сохранением состояния

Компоненты хранящие состояние (**Stateful**) программно незначительно отличаются от компонентов не хранящих состояния (**Stateless**) и синглтонов (**Singleton**).

Отличие заключается в том, что эти компоненты поддерживают диалог с клиентом (**conversational state**) и следовательно имеют иной (более сложный) жизненный цикл (**lifecycle**).

EJB-контейнер создает экземпляр компонента и назначает его только **одному клиенту**.
Каждый запрос клиента передается только данному экземпляру компонента!

EJB. Stateful Session Beans.

Stateful - компоненты с сохранением состояния

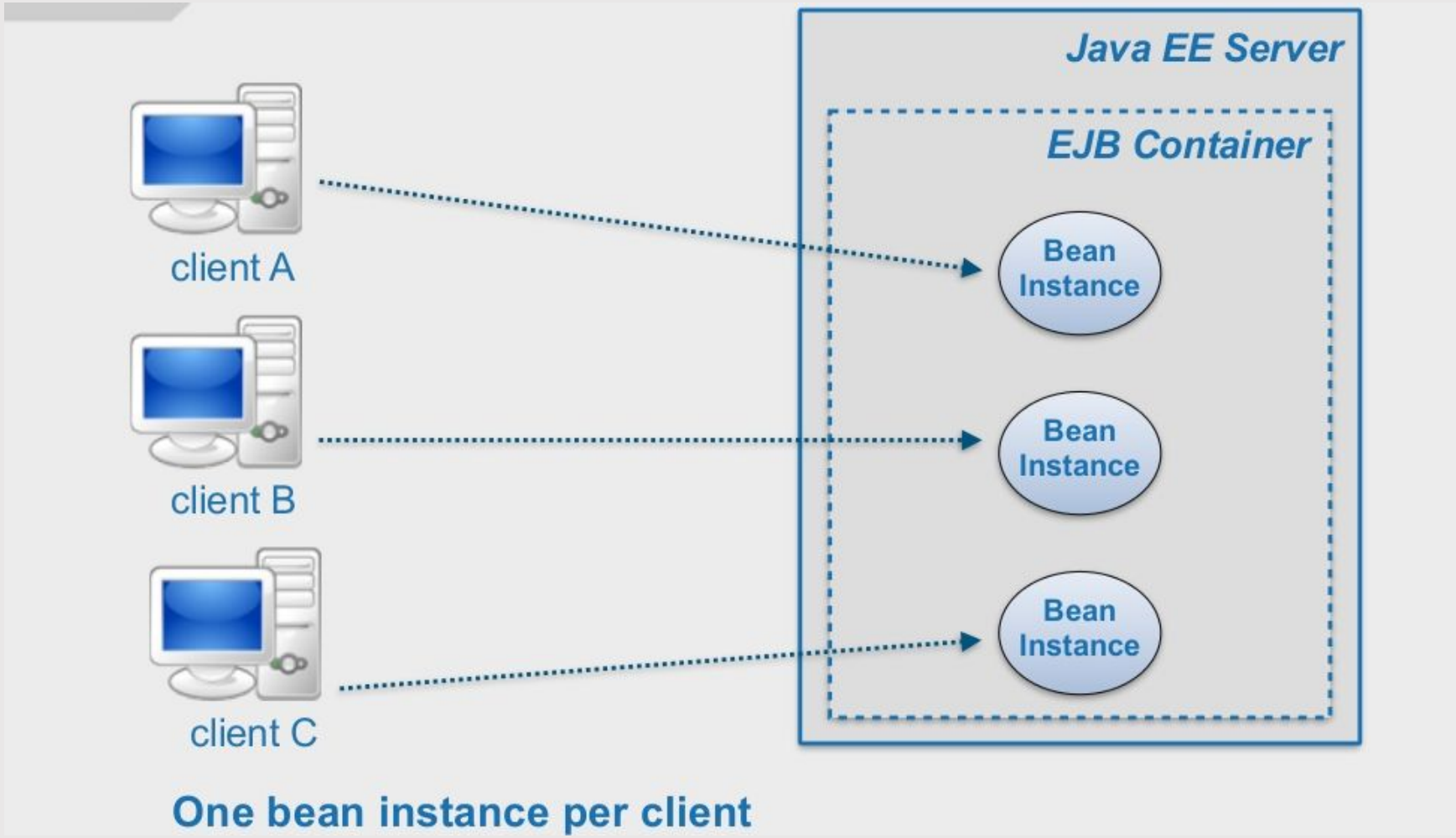
Используется **@Stateful** – аннотация для класса компонента.

Аннотации методов класса компоненты с сохранением состояния:

- @PostConstruct
- @PrePassivate
- @PostActivate
- @PreDestroy
- @Remove

EJB. Stateful Session Beans.

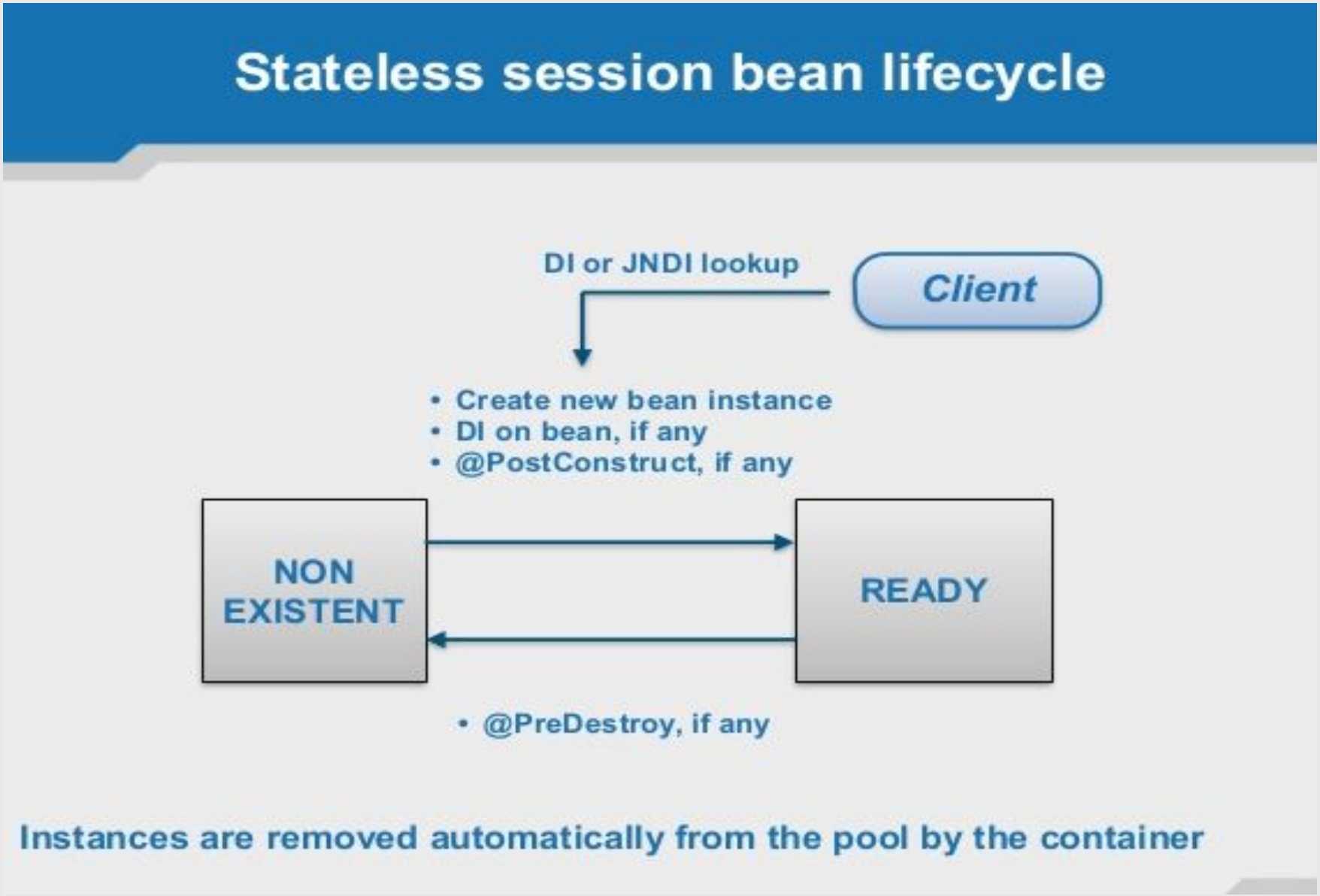
Stateful компоненты (Stateful Session Beans):



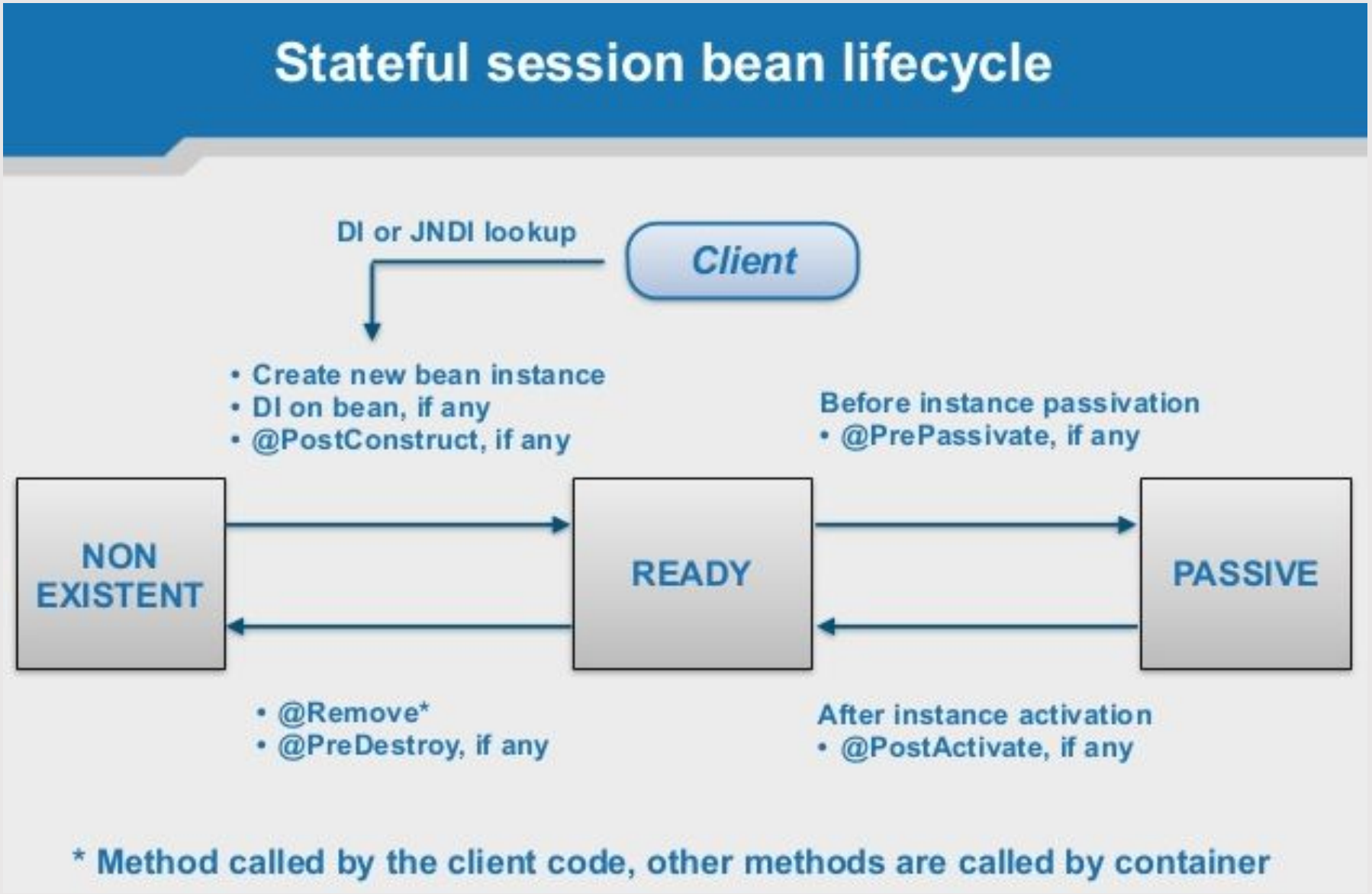
Сессионные EJB компоненты (Session Beans):

Stateless	Stateful	Singleton
Has no client association	Each client has its own bean instance	Instantiated once per application
Has no state between calls	Stores state between client calls	Each client obtains single state
Pooled in memory	May be passivated to disk, cached	-
Client couldn't manage lifecycle	Removable by client	Client couldn't manage lifecycle
Does not support concurrency	Does not support concurrency	Supports concurrency
Implements WS	Couldn't implement WS	Implements WS

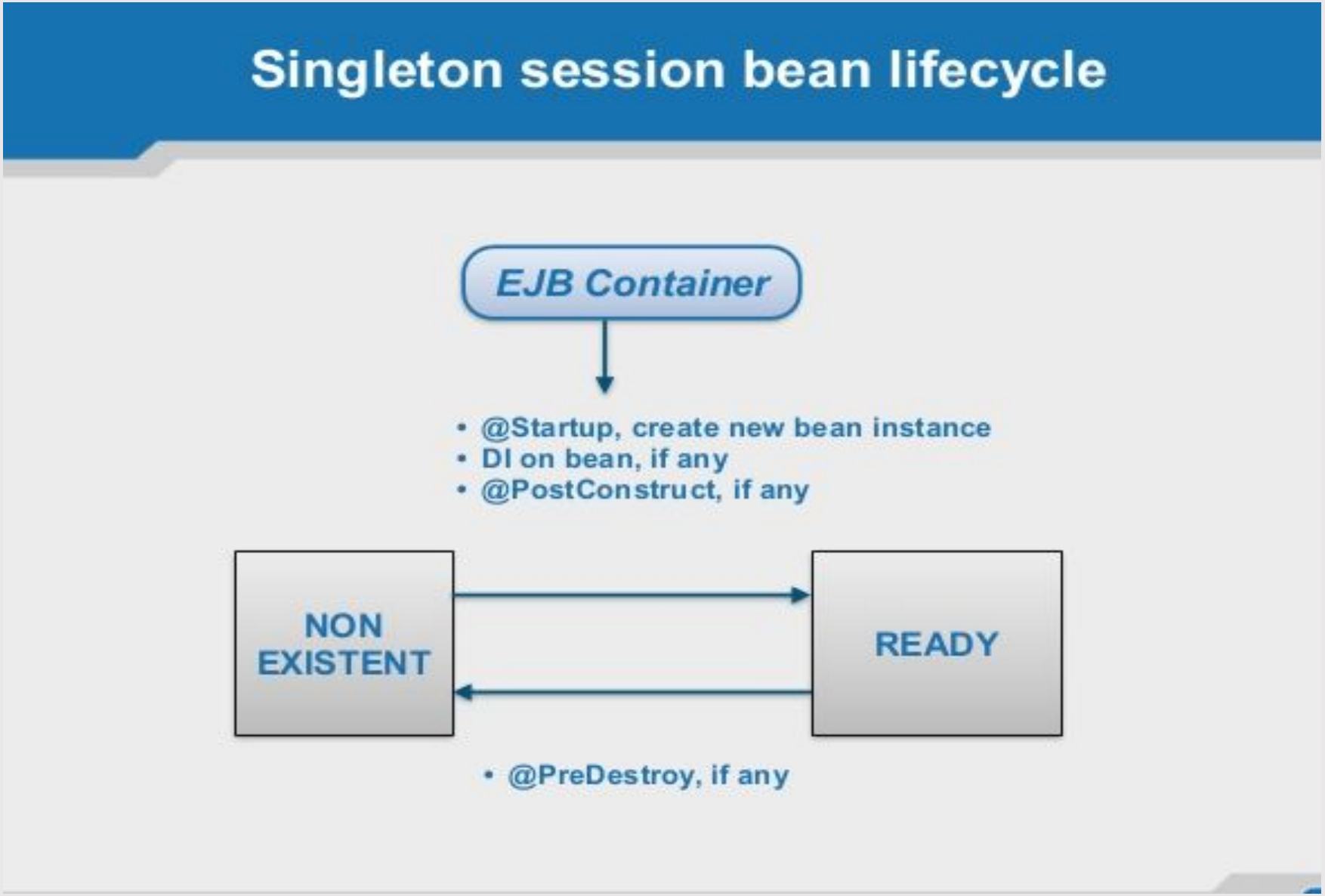
EJB. Stateless Session Bean – жизненный цикл.



EJB. Stateful Session Bean – жизненный цикл.



EJB. Singleton Session Bean – жизненный цикл.



Создание компонент EJB. Модель программирования.

Большинство спецификаций **JavaEE 8** используют одну и ту же модель программирования.

В JavaEE 8 **компоненты EJB,** сервлеты, веб-службы SOAP и REST являются **аннотируемыми классами** с опциональными дескрипторами развертывания XML.

Java-*аннотация* — в языке Java специальная форма синтаксических метаданных, которая может быть добавлена в исходный код.

Аннотации используются для анализа кода, компиляции или выполнения.

Аннотируемы пакеты, ***классы***, **методы**, **переменные** и **параметры**.

Правила наименования компонент EJB (Naming Convention).

Item	Syntax	Example
EJB name	<*name*>Bean	AdderBean
EJB class	<*name*>Bean	AdderBean
Business interface	<*name*>	Adder (AdderRemote AdderLocal)

Создание компонент EJB. Удаленный и локальный интерфейс.

1) Определение типа клиента для работы с **EJB-компонентом**:
локальный **(local)**, удаленный **(remote)**, веб-служба **(webservice)**.

В зависимости от типа клиента классу EJB компонента потребуется реализовывать удаленный, локальный или не реализовывать интерфейс.

Аннотация **@Remote** – обозначает удаленный бизнес-интерфейс.
Аннотация **@Local** – обозначает локальный бизнес-интерфейс.

2) Класс EJB-компонента – любой Java класс, который реализовывает бизнес-логику:

- Класс должен использовать аннотации **@Stateless**, **@Stateful**, **@Singleton** или быть описан в дескрипторе развертывания.
- У класса должен быть **public** конструктор без аргументов – используется контейнером для создания экземпляра класса.
- Класс должен быть **public**, не должен быть абстрактным (**abstract**).
- Аргумент и возвращаемое значение удаленного метода должны относиться к допустимым типам RMI.

Создание компонент EJB. Пример класса компонента.

// Пример:
// Класс EJB-компонента **без сохранения состояния**
// (не реализующего интерфейс)

```
@Stateless
public class CalculatorBean {

    // публичный конструктор класса
    public CalculatorBean(){
        System.out.println("CalculatorBean constructor!");
    }

    // публичный метод
    public double calculate(double a, double b, String operation) {

            // ...
    }
}
```

Создание компонент EJB. Пример класса компонента.

// Пример:
// Класс EJB-компонента **синглтона** (не реализующего интерфейс)

```
@Singleton
@Startup
public class ProductCatalogBean {

    // публичный конструктор класса
    public ProductCatalogBean (){
        System.out.println("ProductCatalogBean constructor!");
    }

    // публичный метод getItems
    public List<Item> getItems() {

            // ...
    }
}
```

Создание компонент EJB. Пример класса компонента.

// Пример:
// Класс EJB-компонента **синглтона** (не реализующего интерфейс)

@Startup – *аннотация для* **синглтона** *позволяет инициализировать EJB компонент во время запуска приложения на сервере приложений.*

```
@Singleton
@Startup
public class ProductCatalogBean {

// ...
```

Singleton Session Bean – цепочка вызовов.

В некоторых случаях важно задать порядок инициализации для **singleton**-компонент.

@DependsOn – *аннотация для* **синглтона** *содержит одну или несколько строк, каждая из которых определяет имя* **синглтон-***компонента.*

```
// Пример:
@Singleton
public class ConfigBean {

// ...

@DependsOn("ConfigBean")
@Singleton
public class ProductCatalogBean{

// ...
```

Методы обратного вызова компонент EJB.

Каждый тип сессионных компонентов имеет свой собственный управляемый контейнером жизненный цикл.

Переход из одного состояния в другое может быть перехвачен контейнером
для вызова методов, аннотированных одним из способов:

- **@PostConstruct** –
  Метод должен быть вызван сразу после создания экземпляра компонента.
  В подобных методах часто выполняется инициализация.

- **@PreDestroy** –
  Метод должен быть вызван сразу после того как контейнер уничтожит экземпляра компонента.
  Используется для освобождения раннее инициализированных ресурсов.

- **@PrePassivate** – только для **Stateful** компонентов.
  Должен быть вызван перед тем, как контейнер “пассивизирует “ экземпляр.
  Может быть использован для освобождения ресурсов (закрытие соединений).

- **@PostActivate** – только для **Stateful** компонентов.
  Должен быть вызван перед тем, как контейнер “активизирует“ экземпляр.
  Может быть использован для инициализации ресурсов (получение соединений).

Вызов компонент EJB.

Для вызова метода в EJB-компоненте, клиент не создает экземпляр компонента непосредственным образом
(с использованием оператора **new**).
Клиенту нужна ссылка на EJB-компонент.

Ссылку можно получить:

1) С помощью JNDI-поиска.
В этом случае используется JNDI-API.

2) С использованием внедрения зависимостей –
dependency injection (DI).
В этом случае используются аннотации **@EJB** и **@Inject.**
Внедрение зависимостей позволяет автоматически внедрять ссылку на EJB-компонент.

Вызов компонент EJB. Переносимое JNDI-имя.

С помощью JNDI-поиска может быть получена ссылка на компонент с указанием **переносимого** JNDI-имени компонента.

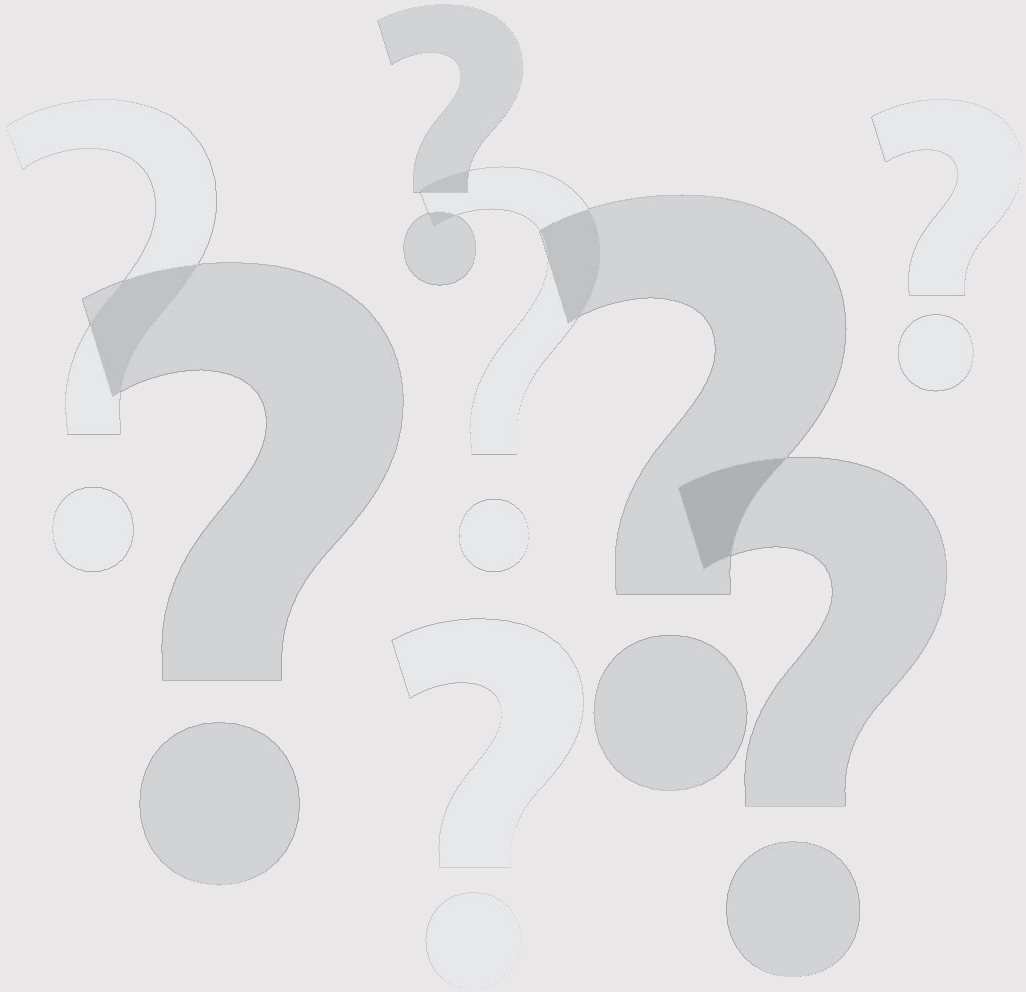
Начиная с **EJB 3.1** JNDI-имена были определены, благодаря чему код является переносимым,
т. е. нет зависимости от определенного сервера приложений.

Определение переносимого JNDI-имени в спецификации JavaEE:

java:<область видимости>[/<имя приложения>]/<имя модуля>/имя EJB компонента[!имя интерфейса]

// Пример - JNDI-имя :
java:global/**TestWebApp1**/**CardSessionBean**

Вопросы?



Выбирайте Центр «Специалист» – ведущего поставщика образовательных услуг в России!

info@specialist.
ru
+7 (495)
232-32-16