**Введение**

Популярность в Интернете социальные сети начали завоёвывать в [1995 году](https://ru.wikipedia.org/wiki/1995_%D0%B3%D0%BE%D0%B4), с появлением американского портала [Classmates.com](https://ru.wikipedia.org/w/index.php?title=Classmates.com&action=edit&redlink=1). Проект оказался весьма успешным, что в следующие несколько лет спровоцировало появление не одного десятка аналогичных сервисов. Но официальным началом бума социальных сетей принято считать 2003—2004 года, когда были запущены [LinkedIn](https://ru.wikipedia.org/wiki/LinkedIn), [MySpace](https://ru.wikipedia.org/wiki/MySpace) и [Facebook](https://ru.wikipedia.org/wiki/Facebook).

**Цели и задачи**

Нашей целью было изучение принципов построения веб-приложения и применения их на практике для создания тематической социальной сети.

Задача нашей работы заключалась в реализации социальной сети на JavaEE с использованием фреймворка Struts 2.

**Основные сведения о социальных сетях**

**Социальная сеть**

Социальная сеть – это a platform to build social networks or social relations among people who share interests, activities, backgrounds or real-life connections. A social network service consists of a representation of each user (often a profile), his or her social links, and a variety of additional services. Social network sites are web-based services that allow individuals to create a public profile, to create a list of users with whom to share connections, and view and cross the connections within the system.[1] Most social network services are web-based and provide means for users to interact over the Internet, such as e-mail and instant messaging. Social network sites are varied and they incorporate new information and communication tools such as mobile connectivity, photo/video/sharing and blogging. Online community services are sometimes considered a social network service, though in a broader sense, social network service usually means an individual-centered service whereas online community services are group-centered. Social networking sites allow users to share ideas, pictures, posts, activities, events, interests with people in their network.

**История развития социальных сетей**

The potential for computer networking to facilitate newly improved forms of computer-mediated social interaction was suggested early on. Efforts to support social networks via computer-mediated communication were made in many early online services, including Usenet,ARPANET, LISTSERV, and bulletin board services (BBS). Many prototypical features of social networking sites were also present in online services such as America Online, Prodigy, CompuServe, ChatNet, and The WELL.

Early social networking on the World Wide Web began in the form of generalized online communities such as Theglobe.com (1995),Geocities (1994) and Tripod.com (1995). Many of these early communities focused on bringing people together to interact with each other through chat rooms, and encouraged users to share personal information and ideas via personal webpages by providing easy-to-use publishing tools and free or inexpensive webspace. Some communities - such as Classmates.com - took a different approach by simply having people link to each other via email addresses. PlanetAll started in 1996.

In the late 1990s, user profiles became a central feature of social networking sites, allowing users to compile lists of "friends" and search for other users with similar interests. New social networking methods were developed by the end of the 1990s, and many sites began to develop more advanced features for users to find and manage friends.This newer generation of social networking sites began to flourish with the emergence of SixDegrees.com in 1997, followed by Makeoutclub in 2000, Hub Culture and Friendster in 2002, and soon became part of the Internet mainstream. Friendster was followed by MySpace and LinkedIn a year later, and eventually Bebo. Friendster became very popular in the Pacific Island. Orkut became the first popular social networking service in Brazil (although most of its very first users were from the United States) and quickly grew in popularity in India (Madhavan, 2007).Attesting to the rapid increase in social networking sites' popularity, by 2005, it was reported that Myspace was getting more page views than Google. Facebook, launched in 2004, became the largest social networking site in the world[23] in early 2009. Facebook was first introduced (in 2004) as a Harvard social networking site, expanding to other universities and eventually, anyone.

***Причины популярности социальных сетей***

Причин, по которым владельцев аккаунтов на разных сайтах становится больше, масса:

* *Безграничность общения*. Общаться можно когда угодно и с кем угодно. Нет ни пространственных, ни временных ограничений. Это позволяет не только поддерживать связь с друзьями и родственниками, находящимися далеко, но и находить новые знакомства.
* *Возможность приобщаться к группам по интересам*. Различные объединения в социальных сетях позволяют искать единомышленников, интересную и нужную информацию, обмениваться разными навыками и опытом. Это очень удобно, когда в реальной жизни в той или иной области найти единомышленников сложно.
* *Для пользования социальными сетями не нужны какие-то особые навыки*. Если же вопросы все-таки возникают, то всегда поможет служба поддержки или специализированные сайты, вроде odinsecret.ru. Но обычно интерфейс порталов максимально понятен, поэтому разбираться в нем долго не приходится.
* *Возможность приобретать товары*. За несколько лет многие социальные сети превратились из площадок для общения в площадки, где можно найти все, что только может понадобиться. Особенно это касается распространенных товаров — одежды, обуви, книг, сувениров и многого другого. Это очень удобно, поскольку часто цены на предлагаемый товар дешевле, чем в интернет-магазинах.
* *Возможность получить оценку себя и своих занятий другими пользователями*. Большинство людей в своей жизни ориентируются на оценку окружающих, а в интернете ее получить достаточно легко: система «лайков» существует во всех социальных сетях.

Реализация проекта

*Основная идея*

Идея проекта заключается в обмене желаниями между пользователями. Смысл этой идеи состоит в том, чтобы предоставить возможность людям не только делиться своими мечтами, но и помогать другим в осуществлении их желаний. В этой социальной сети каждый сможет поделиться частичкой доброты и сделать кого-то в этом мире счастливым.

Возможности системы

**Используемые технологии**

* **JDK 1.8.0, JavaEE (v 7)**
* **Struts 2**
* **Tomcat 8.0.5**
* **MySql**

**Дополнительно**

* **JSP**
* **JSTL?**
* **Log4j**
* **JavaScript**
* **Ajax**
* **CSS**
* **HTML**
* **XML**
* **Intellij IDEA 14.1.1 (Java EE IDE)**

**Java**

Java является объектно-ориентированным языком программирования.

ООП — методология программирования, основанная на представлении программного продукта в виде совокупности объектов, каждый из которых является экземпляром конкретного класса. ООП использует в качестве базовых элементов взаимодействие объектов.

Основные принципы ООП:

Инкапсуляция (encapsulation) — принцип, объединяющий данные и код, манипулирующий этими данными, а также защищающий данные от прямого внешнего доступа и неправильного использования. Другими словами, доступ к данным класса возможен только посредством методов этого же класса. Наследование (inheritance) — процесс, посредством которого один класс может наследовать свойства другого класса и добавлять к ним свойства и мето- ды, характерные только для него.

Полиморфизм (polymorphism) — механизм, использующий одно и то же имя метода для решения похожих, но несколько отличающихся задач в различных объектах при наследовании из одного суперкласса. Целью полиморфизма является использование одного имени при выполнении общих для суперкласса и подклассов действий.

История

Объектно-ориентированный язык Java, разработанный в компании Sun Microsystems в 1995 году для оживления графики на стороне клиента с помощью апплетов, в настоящее время используется для создания переносимых на различные платформы и операционные системы программ. Язык Java нашел широкое применение в Интернет-приложениях, добавив на статические и клиентские веб-страницы динамическую графику, улучшив интерфейсы и реализовав вычислительные возможности. Но объектно-ориентированная парадигма и кроссплатформенность привели к тому, что уже буквально через несколько лет после создания язык практически покинул клиентские страницы и перебрался на серверы. На стороне клиента его место заняли языки JavaScript, Adobe Flash и проч.

Системная библиотека классов языка Java содержит классы и пакеты, реализующие и расширяющие базовые возможности языка, а также сетевые средства, взаимодействие с базами данных, графические интерфейсы и многое другое.

Основные особенности языка

Программы на Java [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), выполняемый [виртуальной машиной Java](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как [интерпретатор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80).

Достоинством подобного способа выполнения программ является полная независимость байт-кода от [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [оборудования](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на Java:

* применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы ([JIT](https://ru.wikipedia.org/wiki/JIT" \o "JIT)-технология) с возможностью сохранения версий класса в машинном коде,
* широкое использование [платформенно-ориентированного кода](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (native-код) в стандартных библиотеках,
* аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология [Jazelle](https://ru.wikipedia.org/wiki/Jazelle), поддерживаемая некоторыми процессорами фирмы [ARM](https://ru.wikipedia.org/wiki/ARM_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F))).

По данным сайта shootout.alioth.debian.org, для семи разных задач время выполнения на Java составляет в среднем в полтора-два раза больше, чем для C/C++, в некоторых случаях Java быстрее, а в отдельных случаях в 7 раз медленнее[[12]](https://ru.wikipedia.org/wiki/Java" \l "cite_note-slow_java-12). С другой стороны, для большинства из них потребление памяти Java-машиной было в 10—30 раз больше, чем программой на C/C++. Также примечательно исследование, проведённое компанией [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)), согласно которому отмечается существенно более низкая производительность и бо́льшее потребление памяти в тестовых примерах на Java в сравнении с аналогичными программами на [C++](https://ru.wikipedia.org/wiki/C%2B%2B)[[13]](https://ru.wikipedia.org/wiki/Java#cite_note-13)[[14]](https://ru.wikipedia.org/wiki/Java#cite_note-14)[[15]](https://ru.wikipedia.org/wiki/Java#cite_note-15).

Идеи, заложенные в концепцию и различные реализации среды виртуальной машины Java, вдохновили множество энтузиастов на расширение перечня языков, которые могли бы быть использованы для создания программ, исполняемых на виртуальной машине[[16]](https://ru.wikipedia.org/wiki/Java" \l "cite_note-16). Эти идеи нашли также выражение в спецификации общеязыковой инфраструктуры [CLI](https://ru.wikipedia.org/wiki/Common_Language_Infrastructure), заложенной в основу платформы [.NET](https://ru.wikipedia.org/wiki/.NET_Framework) компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft).

**Java EE** (Enterprise Edition) представляет собой широко используемую платформу, содержащую набор взаимосвязанных технологий, которые существенно сокращают стоимость и сложность разработки, развертывания многоуровневых серверных приложений, а также управления ими. Платформа Java EE основана на платформе Java SE и предоставляет набор интерфейсов API (интерфейсов разработки приложений) для разработки и запуска портируемых, надежных, масштабируемых и безопасных серверных приложений.

Актуальная версия Java EE имеет номер 7.0.

При переходе на версию 5.0 изменилось и название спецификации с J2EE [[ˌdʒeɪˈtuː ˌiːˈiː](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82" \o "Международный фонетический алфавит)] (**Java 2 Platform, Enterprise Edition**), на **Java Platform, Enterprise Edition**, сокращённо **Java EE**.

|  |  |  |
| --- | --- | --- |
| **Версия** | **Полное имя** | **Дата публикации** |
| 1.0 | Java 2 Platform Enterprise Edition, v 1.0 | декабрь 1999 |
| 1.2 | Java 2 Platform Enterprise Edition, v 1.2 | 2000 |
| 1.2.1 | Java 2 Platform Enterprise Edition, v 1.2.1 | 23 мая 2000 |
| 1.3 | Java 2 Platform Enterprise Edition, v 1.3 | 24 сентября 2001 |
| 1.4 | Java 2 Platform Enterprise Edition, v 1.4 | 24 ноября 2003 |
| 5.0 | Java Platform, Enterprise Edition, v 5 | 11 мая 2006 |
| 6.0 | Java Platform, Enterprise Edition, v 6 | 6 декабря 2009 |
| 7.0 | Java Platform, Enterprise Edition, v 7 | 12 июня 2013 |
| 8.0 | Java Platform, Enterprise Edition, v 8 | ~ III квартал 2016[[1]](https://ru.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition" \l "cite_note-1) |

Java EE включает в себя стандарты следующих технологий:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Описание** | **Использованная версия** | | | |
| **J2EE 1.4** | **JEE 5** | **JEE 6** | **JEE 7** |
| **2003 г.** | **2006 г.** | **2009 г.** | **2013 г.** |
| [EJB](https://ru.wikipedia.org/wiki/Enterprise_JavaBeans) | Enterprise JavaBeans — спецификация технологии серверных компонентов, содержащих [бизнес-логику](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) | 2.1 | 3.0 | 3.1 | 3.2 |
| [JPA](https://ru.wikipedia.org/wiki/Java_Persistence_API) | Java Persistence API | Нет | 1.0 | 2.0 | 2.1 |
| [Сервлет](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82) | Обслуживание запросов веб-клиентов | 2.4 | 2.5 | 3.0 | 3.1 |
| [JSP](https://ru.wikipedia.org/wiki/JSP) | JavaServer Pages — динамическая генерация веб-страниц на стороне сервера | 2.0 | 2.1 | 2.2 | 2.3 |
| [JSTL](https://ru.wikipedia.org/wiki/JSTL) | JavaServer Pages Standard Tag Library | Нет | 1.2 | 1.2 | 1.2 |
| [JSF](https://ru.wikipedia.org/wiki/JavaServer_Faces) | JavaServer Faces — компонентный серверный фреймворк для разработки веб-приложений на технологии Java | Нет | 1.2 | 2.0 | 2.2 |
| [JAX-WS](https://ru.wikipedia.org/wiki/JAX-WS) | Java API for XML Web Services — создание веб-сервисов | 1.0 | 1.2 | 1.2 | 2.2 |
| JAX-RS | Java API for RESTful Web Services — создание RESTful веб-сервисов | Нет | Нет | 1.1 | 2.0 |
| [JNDI](https://ru.wikipedia.org/wiki/JNDI) | Java Naming and Directory Interface — служба каталогов | 1.2 | 1.2 | 1.2 |  |
| [JMS](https://ru.wikipedia.org/wiki/Java_Message_Service) | Java Message Service — обмен сообщениями | 1.1 | 1.1 | 1.1 | 2.0 |
| [JTA](https://ru.wikipedia.org/wiki/Java_Transaction_API) | Java Transaction API | 1.0.1B | 1.1 | 1.1 | 1.2 |
| [JAAS](https://ru.wikipedia.org/wiki/Java_Authentication_and_Authorization_Service) | Java Authentication and Authorization Service — Java-реализация [PAM](https://ru.wikipedia.org/wiki/Pluggable_Authentication_Modules) | 1.0 | 1.0 | 1.0 |  |
| [JavaMail](https://ru.wikipedia.org/wiki/JavaMail) | Получение и отправка электронной почты | 1.2 | 1.4 | 1.4 | 1.5 |
| [JACC](https://ru.wikipedia.org/w/index.php?title=Java_Authorization_Contract_for_Containers&action=edit&redlink=1) | Java Authorization Contract for Containers | 1.0 | 1.1 | 1.1 | 1.4 |
| [JCA](https://ru.wikipedia.org/w/index.php?title=J2EE_Connector_Architecture&action=edit&redlink=1) | J2EE Connector Architecture | 1.5 | 1.5 | 1.6 | 1.6 |
| JAF | JavaBeans Activation Framework | 1.0 | 1.1 | 1.1 | 1.1 |
| StAX | Streaming API for XML | Нет | 1.0 | 1.0 | 1.0 |
| CDI | Context and Dependency Injection | Нет | Нет | 1.0 | 1.1 |

API в Java EE

Java EE API, включает в себя несколько технологий, которые расширяют функциональность базовых Java SE API-интерфейсов.

**javax.servlet. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=4) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=4)]

Спецификация [сервлетов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82_(Java)) определяет набор программных интерфейсов для обслуживания [HTTP](https://ru.wikipedia.org/wiki/HTTP)-запросов. Она включает в себя спецификации [JavaServer Pages](https://ru.wikipedia.org/wiki/JSP).

**javax.websocket. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=5) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=5)]

Спецификация [Java API](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_Java_API) для [WebSocket](https://ru.wikipedia.org/wiki/WebSocket) определяет набор программных интерфейсов для обслуживания WebSocket-соединений.

**javax.faces. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=6) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=6)]

Этот пакет является корнем [JavaServer Faces API](https://ru.wikipedia.org/wiki/JavaServer_Faces). JSF — это технология для построения пользовательских интерфейсов из компонентов.

**javax.faces.component. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=7) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=7)]

Этот пакет — составная часть [JavaServer Faces (JSF) API](https://ru.wikipedia.org/wiki/JavaServer_Faces).

**javax.el. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=8) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=8)]

Этот пакет определяет классы и интерфейсы для языка выражений в Java EE. Expression Language (EL) — это простой язык, изначально предназначенный для удовлетворения конкретных потребностей разработчиков веб-приложений. Он используется, в частности, в JSF для связывания компонентов (backing) beans и в CDI name beans, но может быть использован для других платформ.

**javax.enterprise.inject. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=9) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=9)]

Эти пакеты определяют аннотации вложения для [(CDI) API](http://jcp.org/en/jsr/detail?id=299).

**javax.enterprise.context. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=10) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=10)]

Эти пакеты определяют контекст аннотации и интерфейсы для [(CDI) API](http://jcp.org/en/jsr/detail?id=299).

**javax.ejb. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=11) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=11)]

[Enterprise JavaBean (EJB)](https://ru.wikipedia.org/wiki/Enterprise_JavaBeans) определяет набор API-интерфейсов удалённого вызова процедур (с использованием[RMI](https://ru.wikipedia.org/wiki/RMI" \o "RMI) или RMI-[IIOP](https://ru.wikipedia.org/wiki/GIOP" \o "GIOP)), управление параллелизмом, инъекции зависимости и контроля доступа для бизнес-объектов. Этот пакет содержит классы и интерфейсы, определяющие связь между корпоративным компонентом и его клиентом, а также между корпоративным компонентом и EJB-контейнером.

**javax.validation. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=12) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=12)]

Этот пакет содержит аннотации и интерфейсы для поддержки декларативной проверки [Bean Validation](https://ru.wikipedia.org/w/index.php?title=Bean_Validation&action=edit&redlink=1) ([*англ.*](https://en.wikipedia.org/wiki/Bean_Validation)) API. Bean Validation обеспечивает единый способ обеспечения ограничения на bean (например, классов [JPA](https://ru.wikipedia.org/wiki/Java_Persistence_API)модели), что может быть реализовано с помощью cross-layer.

**javax.persistence. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=13) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=13)]

Этот пакет содержит классы и интерфейсы, которые определяют связь между поставщиком, управляемым классом и клиентом [Java Persistence API (JPA)](https://ru.wikipedia.org/wiki/Java_Persistence_API).

**javax.transaction. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=14) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=14)]

Этот пакет предоставляет [Java Transaction API (JTA)](https://ru.wikipedia.org/wiki/Java_Transaction_API), который содержит интерфейсы и аннотации для взаимодействия с поддержкой транзакций, предлагаемой Java EE.

**javax.security.auth.message. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=15) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=15)]

Этот пакет предоставляет Java SPI (JASPIC), который содержит интерфейсы и классы для создания модулей аутентификации для безопасных "Java EE"-приложений.

**javax.enterprise.concurrent. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=16) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=16)]

Этот пакет предоставляет интерфейсы для взаимодействия непосредственно с платформой Java EE, по умолчанию управляет запуском потоков.

**javax.jms. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=17) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=17)]

Этот пакет определяет [Java Message Server (JMS) API](https://ru.wikipedia.org/wiki/Java_Message_Service) (Application Programming Interface). JMS API предоставляет для программ Java распространенный способ создавать, отправлять, получать и читать сообщения системы корпоративного обмена сообщениями.

**javax.batch.api. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=18) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=18)]

Этот пакет определяет вступление AP для Java EE [batch applications](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5). Batch applications API предоставляет средства для запуска длительных фоновых задач, которые, возможно, связаны с большим объёмом данных.

**javax.resource. \***[[править](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&veaction=edit&vesection=19) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&action=edit&section=19)]

Этот пакет определяет [Java EE Connector Architecture](https://ru.wikipedia.org/w/index.php?title=Java_EE_Connector_Architecture&action=edit&redlink=1) ([*англ.*](https://en.wikipedia.org/wiki/Java_EE_Connector_Architecture)). Java EE Connector Architecture (JCA) является Java-технологией для решения проблем подключения серверов приложений к их корпоративным информационным системам (КИС) в рамках интеграции корпоративных приложений (EAI).

**Struts 2**

|  |  |
| --- | --- |
| **Apache Struts 2** | |
| [Apache Struts Logo](http://en.wikipedia.org/wiki/File:Struts_logo.gif) | |
| [**Developer(s)**](http://en.wikipedia.org/wiki/Software_developer) | [Apache Software Foundation](http://en.wikipedia.org/wiki/Apache_Software_Foundation) |
| **Initial release** | October 10, 2006 |
| [**Stable release**](http://en.wikipedia.org/wiki/Software_release_life_cycle) | 2.3.20 / December 7, 2014 |
| **Written in** | [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) |
| [**Operating system**](http://en.wikipedia.org/wiki/Operating_system) | [Cross-platform](http://en.wikipedia.org/wiki/Cross-platform) |
| [**Platform**](http://en.wikipedia.org/wiki/Computing_platform) | [Cross-platform](http://en.wikipedia.org/wiki/Cross-platform) ([JVM](http://en.wikipedia.org/wiki/Java_Virtual_Machine)) |
| [**License**](http://en.wikipedia.org/wiki/Software_license) | [Apache License](http://en.wikipedia.org/wiki/Apache_License) 2.0 |
| **Website** | [struts.apache.org](http://struts.apache.org/) |

**Apache Struts 2** — фреймворк с открытым исходным кодом для создания **Java EE** веб-приложений. Основывается на Java Servlet API и расширяет его, в архитектурном плане реализует шаблон проектирования **MVC**.

Model-view-controller (MVC, «Модель-представление-поведение», «Модель-представление-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные.

Основная цель применения этой концепции состоит в разделении бизнес-логики (модели) от её визуализации (представления, вида). За счет такого разделения повышается возможность повторного использования.

В обычном **Java EE** веб-приложении клиент, как правило, отправляет информацию на сервер из формы. Введённые данные передаются сервлету, который обрабатывает её, при необходимости взаимодействуя с базой данных, и формирует HTML ответ. Для больших проектов (например, для портала с сотнями страниц), такой подход является неадекватным, так как объединяет бизнес логику и логику отображения, что делает поддержку чрезвычайно сложной. **Struts** был создан для того, чтобы чётко разделить модель (бизнес-логику), представление (HTML-страницы) и контроллер (отвечающий за передачу данных от модели к представлению и обратно).

Конфигурация задается в файле struts.xml

It uses and extends the[Java Servlet](http://en.wikipedia.org/wiki/Java_Servlet" \o "Java Servlet) [API](http://en.wikipedia.org/wiki/Application_programming_interface) to encourage developers to adopt a [model–view–controller](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) architecture. The [WebWork](http://en.wikipedia.org/wiki/WebWork) framework spun off from [Apache Struts](http://en.wikipedia.org/wiki/Apache_Struts) aiming to offer enhancements and refinements while retaining the same general architecture of the original Struts framework. In December 2005, it was announced that WebWork 2.2 was adopted as Apache Struts 2, which reached its first full release in February 2007.

**JSP**

**JSP** (**JavaServer Pages**) — технология, позволяющая веб-разработчикам создавать содержимое, которое имеет как статические, так и динамические компоненты. Страница JSP содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов [HTML](https://ru.wikipedia.org/wiki/HTML),[SVG](https://ru.wikipedia.org/wiki/SVG), [WML](https://ru.wikipedia.org/wiki/WML), или [XML](https://ru.wikipedia.org/wiki/XML), и JSP- элементы, которые конструируют динамическое содержимое. Кроме этого могут использоваться библиотеки JSP-тегов, а также [EL](https://ru.wikipedia.org/wiki/Expression_Language) ([Expression Language](https://ru.wikipedia.org/wiki/Expression_Language)), для внедрения Java-кода в статичное содержимое JSP-страниц.

JSP является платформонезависимой, переносимой и легко расширяемой технологией для разработки веб-приложений.

Технология Java Server Pages (JSP) была разработана компанией Sun Microsystems (в настоящее время поглощена компанией Oracle), чтобы облегчить создание страниц с динамическим содержанием. В то время как сервлеты наилучшим образом подходят для выполнения контролирующей функции приложения в виде обработки запросов и определе- ния содержания и вида ответа, страницы JSP выполняют функцию отображения результатов работы приложения в виде текстовых документов типа HTML, XML, WML и некоторых других. JSP поддерживают как JavaScript, так и HTML-теги. JavaScript обычно используется, чтобы добавить функциональные возможно- сти на уровне HTML-страницы.

Принято разделять динамическое и статическое содержимое JSP.

* Динамические ресурсы. Результаты их деятельности изменяются во время выполнения приложения. Обычно представлены в виде выражений Expression Language, библиотек тегов и тегов разработчика.
* Статические ресурсы. Не изменяются сами в процессе работы (HTML, JavaScript, изображения и т. д.). Смысл разделения динамического и статического содержания в том, что статические ресурсы могут находиться под управлением HTTP-сервера в то время, как динамические нуждаются в движке (JSP Engine) и в большинст- ве случаев в доступе к уровню данных. Рекомендуется разделить и разрабатывать параллельно две части приложения: часть, состоящяя только из динамических ресурсов, и часть, состоящяя только из статических ресурсов.

Некоторые преимущества использования JSP-технологии над другими ме- тодами создания динамического содержания страниц:

— разделение динамического и статического содержания. Возможность раз- делить логику приложения и дизайн веб-страницы снижает сложность раз- работки веб-приложений и упрощает их поддержку;

— независимость от платформы. Технологии Java не зависят от платформы, следовательно, JSP могут выполняться практически на любом веб-сервере. Разрабатывать JSP можно на любой платформе;

— многократное использование компонентов. Использование JavaBeans и Enterprise JavaBeans (EJB) позволяет многократно использовать компоненты, что ускоря- ет создание веб-приложений;

— теги.

Спецификация JSP содержит библиотеку стандартных тегов JSTL, по- зволяет разработчику создавать собственные теги, кроме того, теги обеспе- чивают возможность использования JavaBean и обращение к классам бизнес-логики. Содержимое Java Server Pages (теги HTML, теги JSP и скрипты) переводится в сервлет код-сервером. Этот процесс ответствен за трансляцию как динамических, так и статических элементов, объявленных внутри файла JSP. Об архи- тектуре сайтов, использующих JSP/Servlet-технологии, часто говорят, как о thin-client (использование ресурсов клиента незначительно), потому что большая часть логики выполняется на сервере.

Жизненный цикл

Процессы, выполняемые с файлом JSP при первом вызове:

— браузер делает запрос к странице JSP;

— JSP-engine анализирует содержание файла JSP и создает сервлет с кодом, основанным на исходном тексте файла JSP, при этом engine транслирует статическое содержимое в методы вывода и помещает его в метод \_jspService(). Полученный сервлет будет ответственен за генерацию статических элемен- тов, определенных во время разработки. Динамические элементы трансли- руются в java-код;

— код сервлета компилируется в файл \*.class. и загружается в контейнер. В итоге сервлет на основе JSP установлен и готов к работе;

— выполняется метод init() сервлета;

— вызываeтся метод \_jspService(), и сервлет логически исполняется, форми- руя экземпляр response;

— комбинация статического HTML и графики вместе с результатами исполне- ния динамических элементов, определенных в оригинале JSP, пересылают- ся браузеру через выходной поток объекта ответа HttpServletResponse.

Следующие обращения к файлу JSP просто вызовут метод \_jspService() сер- влета. Сервлет используется до тех пор, пока сервер не будет остановлен или сер- влет не будет выгружен из контейнера. Результат работы JSP можно легко предста- вить, зная правила трансляции JSP в сервлет, в частности, в его метод \_jspService().

**Tomcat**

**Tomcat** (в старых версиях — **Catalina**) — [контейнер сервлетов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80_%D1%81%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82%D0%BE%D0%B2) с открытым исходным кодом, разрабатываемый [Apache Software Foundation](https://ru.wikipedia.org/wiki/Apache_Software_Foundation). Реализует спецификацию [сервлетов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82) и спецификацию [JavaServer Pages](https://ru.wikipedia.org/wiki/JavaServer_Pages) ([JSP](https://ru.wikipedia.org/wiki/JSP)) и [JavaServer Faces](https://ru.wikipedia.org/wiki/JavaServer_Faces) ([JSF](https://ru.wikipedia.org/wiki/JSF)). Написан на языке [Java](https://ru.wikipedia.org/wiki/Java).

**Tomcat** позволяет запускать [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), содержит ряд программ для самоконфигурирования.

Tomcat используется в качестве самостоятельного [веб-сервера](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), в качестве сервера контента в сочетании с [веб-сервером](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80)[Apache HTTP Server](https://ru.wikipedia.org/wiki/Apache_HTTP_Server), а также в качестве [контейнера сервлетов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80_%D1%81%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82%D0%BE%D0%B2) в [серверах приложений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) [JBoss](https://ru.wikipedia.org/wiki/JBoss) и [GlassFish](https://ru.wikipedia.org/wiki/GlassFish).

**Контейнер сервлетов**  — программа, представляющая собой [сервер](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)), который занимается системной поддержкой [сервлетов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82) и обеспечивает их жизненный цикл в соответствии с правилами, определёнными в спецификациях. Может работать как полноценный самостоятельный веб-сервер, быть поставщиком страниц для другого веб-сервера, например [Apache](https://ru.wikipedia.org/wiki/Apache), или интегрироваться в [Java EE](https://ru.wikipedia.org/wiki/Java_EE) [сервер приложений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9). Обеспечивает обмен данными между сервлетом и клиентами, берёт на себя выполнение таких функций, как создание программной среды для функционирующего сервлета, идентификацию и авторизацию клиентов, организацию сессии для каждого из них.

MySql

**MySQL** (МФА: [[maɪ ˌɛskjuːˈɛl]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82))[[3]](https://ru.wikipedia.org/wiki/MySQL#cite_note-3) — [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [реляционная система управления базами данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94). Разработку и поддержку MySQL осуществляет корпорация [Oracle](https://ru.wikipedia.org/wiki/Oracle), получившая права на торговую марку вместе с поглощённой [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems), которая ранее приобрела шведскую компанию [MySQL AB](https://ru.wikipedia.org/wiki/MySQL_AB). Продукт распространяется как под [GNU General Public License](https://ru.wikipedia.org/wiki/GNU_General_Public_License), так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм[репликации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)" \o "Репликация (вычислительная техника)).

MySQL является решением для малых и средних приложений. Входит в состав серверов [WAMP](https://ru.wikipedia.org/wiki/WAMP), [AppServ](https://ru.wikipedia.org/w/index.php?title=AppServ&action=edit&redlink=1), [LAMP](https://ru.wikipedia.org/wiki/LAMP) и в портативные сборки серверов [Денвер](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)), [XAMPP](https://ru.wikipedia.org/wiki/XAMPP), [VertrigoServ](https://ru.wikipedia.org/w/index.php?title=VertrigoServ&action=edit&redlink=1). Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа [MyISAM](https://ru.wikipedia.org/wiki/MyISAM), поддерживающие полнотекстовый поиск, так и таблицы [InnoDB](https://ru.wikipedia.org/wiki/InnoDB), поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

26 февраля 2008 года [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) приобрела [MySQL AB](https://ru.wikipedia.org/wiki/MySQL_AB) за [$](https://ru.wikipedia.org/wiki/$)1 млрд[[4]](https://ru.wikipedia.org/wiki/MySQL" \l "cite_note-4), 27 января 2010 года [Oracle](https://ru.wikipedia.org/wiki/Oracle" \o "Oracle) приобрела [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) за [$](https://ru.wikipedia.org/wiki/$)7,4 млрд[[5]](https://ru.wikipedia.org/wiki/MySQL" \l "cite_note-5) и включила MySQL в свою линейку [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94" \o "СУБД).[[6]](https://ru.wikipedia.org/wiki/MySQL#cite_note-6)

Сообществом разработчиков MySQL созданы различные ответвления кода, такие, как [Drizzle](https://ru.wikipedia.org/w/index.php?title=Drizzle&action=edit&redlink=1" \o "Drizzle (страница отсутствует)) ([*англ.*](https://en.wikipedia.org/wiki/Drizzle_(database_server))), [OurDelta](https://ru.wikipedia.org/w/index.php?title=OurDelta&action=edit&redlink=1" \o "OurDelta (страница отсутствует)), [Percona Server](https://ru.wikipedia.org/w/index.php?title=Percona_Server&action=edit&redlink=1)и [MariaDB](https://ru.wikipedia.org/wiki/MariaDB). Все эти ответвления уже существовали на момент [поглощения](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D1%8F_%D0%B8_%D0%BF%D0%BE%D0%B3%D0%BB%D0%BE%D1%89%D0%B5%D0%BD%D0%B8%D1%8F) компании Sun корпорацией Oracle.

**Шаблоны проектрирования**

**Шаблон проектирования** или **паттерн** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *design pattern*) в [разработке программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) — повторимая [архитектурная конструкция](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), представляющая собой решение проблемы [проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) в рамках некоторого часто возникающего [контекста](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82).

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в [код](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0); это лишь пример решения задачи, который можно использовать в различных ситуациях. [Объектно-ориентированные](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Объектно-ориентированное программирование) шаблоны показывают [отношения](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_(%D1%84%D0%B8%D0%BB%D0%BE%D1%81%D0%BE%D1%84%D0%B8%D1%8F)&action=edit&redlink=1) и [взаимодействия](https://ru.wikipedia.org/wiki/%D0%92%D0%B7%D0%B0%D0%B8%D0%BC%D0%BE%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D0%B5) между [классами](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) или [объектами](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются [идиомами](https://ru.wikipedia.org/wiki/%D0%98%D0%B4%D0%B8%D0%BE%D0%BC%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют **архитектурные шаблоны**, они охватывают собой архитектуру всей [программной системы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0).

MVC

**Model-view-controller** (**MVC**, «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких [шаблонов проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), с помощью которых модель приложения, [пользовательский интерфейс](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения [архитектурного каркаса](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BA%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), когда переходят от теории к реализации в конкретной предметной области.

Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

* *Модель* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Model*). Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.
* *Представление*, *вид* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *View*). Отвечает за отображение информации (визуализацию). Часто в качестве представления выступает [форма (окно)](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%BD%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)" \o "Окно (программирование)) с графическими элементами.
* *Контроллер* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Controller*). Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

DAO

В [программном обеспечении](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) **data access object** (DAO) — это объект, который предоставляет абстрактный [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) к какому-либо типу [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) или механизму хранения. Определённые возможности предоставляются независимо от того, какой механизм хранения используется и без необходимости специальным образом соответствовать этому механизму хранения. Этот [шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) применим ко множеству языков программирования, большинству программного обеспечения, нуждающемуся в хранении информации и к большей части баз данных, но традиционно этот шаблон связывают с приложениями на платформе [Java Enterprise Edition](https://ru.wikipedia.org/wiki/Java_EE), взаимодействующими с реляционными базами данных через интерфейс [JDBC](https://ru.wikipedia.org/wiki/JDBC), потому что он появился в рекомендациях от фирмы [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems)

Data Access Object

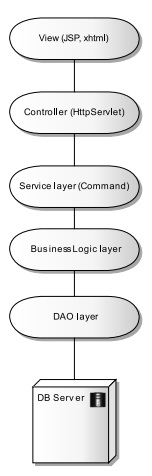
При создании информационной системы выявляются некоторые слои, кото- рые отвечают за взаимодействие различных частей приложения. Связь с базой данных является важной частью любой системы, поэтому всегда выделяется часть кода, ответственная за передачу запросов в БД и обработку полученных от нее ответов. Общее определение шаблона Data Access Object трактует его как прослойку между приложением и СУБД. DAO абстрагирует бизнес- сущности системы и отражает их на записи в БД. DAO определяет общие способы использования соединения с БД, моменты его открытия и закрытия или извлечения и возвращения в пул. В общем случае DAO можно опреде- лять таким образом, чтобы была возможность подмены одной модели базы JDBC 359 данных другой. Например: реляционную заменить на объектную или, что про- ще, MySQL на Oracle. В практическом программировании такие глобальные задачи ставятся крайне редко, поэтому будет приведено несколько способов организации взаимодействия с БД, отличающихся уровнем использования кон- некта к БД и организацией работы с бизнес-сущностями. Вершина иерархии DAO представляет собой класс или интерфейс с описа- нием общих методов, которые будут использоваться при взаимодействии с та- блицей или группой таблиц. Как правило, это методы выбора, поиска сущно- сти по признаку, добавление, удаление и замена информации.

DAO. Уровень логики

На практике чаще всего возникает необходимость при выполнении запроса пользователя обращаться сразу к нескольким ветвям DAO и использовать при этом единственное соединение с БД. В этом случае соединение с БД создается или извлекается из пула до создания экземпляров DAO, а закрывается, соответ- ственно, после выполнения всех обращений к БД.

архитектурное взаимодействие

В большинстве приложений используются не сер- влеты или JSP по отдельности, а их архитектурное взаимодействие. Страница JSP представляет вид для результатов выполнения запроса клиента, а сервлет отвечает за вызов классов бизнес-логики и передачу результатов выполнения бизнес-логики в соответству- ющую JSP и ее вызов. Т.е. сервлеты не генерируют от- вета сами, а только выступают в роли контроллера за- просов. Такая архитектура построения приложений носит название MVC (Model/View/Controller). Model — классы бизнес-логики и длительного хранения, View — страницы JSP, Controller — сервлет. Применение шаблона на практике приводит к тому, что трехуровневая базовая модель распадается на мно- гоуровневую. Число и назначение слоев может существенно отличаться в зависимости от разработанного архитектурного решения. В предлагаемом ниже подходе добавляются: уро- вень служб, обрабатывающих объект запроса, уровень логики, имплементирующий конкретные бизнес-процессы приложения и уровень организации взаимодей- ствия с базой данных.



**Сервлеты?**

Приложение Java, запускаемое и выполняемое в контейнере сервера прило- жений. Клиент общается с таким приложением посредством веб-браузера. Никаких дополнительных приложений на стороне клиента устанавливать не тре- буется. Сервлеты поддерживаются виртуальной машиной JVM, что предотвра- щает утечки памяти и обеспечивает функционирование garbage collection. Каждому клиенту сервлет выделяет независимый поток выполнения. Клиент посылает приложению HTTP-запрос, сервлет генерирует ответ и возвращает его клиенту в виде html-документа. Сервлет: — компонент приложений Java Enterprise Edition; — загружается веб-сервером в контейнер; — выполняется на стороне сервера; — обрабатывает клиентские запросы; — динамически генерирует ответы на запросы; — находится в состоянии ожидания, если запросы отсутствуют; — принимает запросы от других сервлетов (Servlet chaining); — поддерживает соединения с ресурсами. Наибольшее распространение получили сервлеты, обрабатывающие кли- ентские запросы по протоколу HTTP. Технология сервлетов является оболоч- кой протокола HTTP и поддерживает его как транспорт передачи данных от клиента серверу и обратно. Контейнер сервлетов поддерживает также про- токол HTTPS (HTTP и SSL) для защищаемых запросов. Сервлеты в промышленном программировании используются для: — приема входящих данных от клиента; — взаимодействия с бизнес-логикой системы; — динамической генерации ответа клиенту.

**Log4j**

В процессе функционирования сложных приложений необходимо вести журнал сообщений и ошибок, чтобы была возможность отследить время входа и выхода пользователя из системы, возникновение исключительных ситуаций, сбоев и т. д. Существуют различные API регистрации сообщений и ошибок, среди которых на данный момент можно выделить следующие. 1. Log4J.

Apache Log4J был первым регистратором. Изначально обладает ка- чественной архитектурой, в следствие чего быстро занял доминирующие позиции и применяется в большинстве промышленных приложений. Разработан в рамках проекта Jakarta Apache.

В итоге logger регистрирует и направляет вывод события в пункт назначе- ния, определяемый направляющим элементом, в формате, заданном формати- рующим элементом.

Log4j

В современном практическом программировании представляет основной инструмент журналирования событий. Формирует журнал сообщений (отла- дочных, информационных, системных, security, сообщений об ошибках).

Logger

Основным элементом API регистрации событий и ошибок является регистратор org.apache.log4j.Logger, который управляет регистрацией сооб- щений. Вывод регистратора может быть направлен на консоль, в файл, базу данных, GUI-компонент или сокет. Это компонент приложения, принимающий и выполняющий запросы на запись в регистрационный журнал.

Apache Log4J поддерживает несколько способов конфигурации. Позволяет управлять своим поведением во время исполнения. Каждый класс приложения может иметь свой собственный logger или быть прикреплен к общему для всего приложения. Регистраторы образуют иерар- хию, как и пакеты Java. Каждый логгер имеет имя, описывающее иерархию, к которой он принадлежит. Разделителем является точка. Принцип полностью аналогичен формированию имени пакета в Java. Регистратор может быть создан или получен с помощью статического мето- да getLogger(String name) или getLogger(Class name), где name — имя пакета или класса. На вершине иерархии находится корневой регистратор. Он всегда существует и у него нет имени. Ссылку на корневой регистратор можно полу- чить статическим методом getRootLogger(). У каждого регистратора есть уровень сообщения по возрастанию (TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF), который управляет выводом сообщений. Для вывода сообщений конкретного уровня используются методы trace(), debug(), info(), warn(), error(), fatal(). Чтобы вывести информацию о возникшем исключении в качестве второго параметра, в перечисленные методы нужно передать объект класса, производного от Throwable. Для вывода сообще- ния необходимо, чтобы уровень выводимого сообщения был не ниже, чем уровень регистратора (TRACE < DEBUG < INFO < WARN < ERROR < FATAL < OFF), т. е. если уровень регистратора INFO, то вызов logger.debug("message") не даст никакого эффекта, т. к. DEBUG < INFO. Уровень регистратора можно указать с помощью метода setLevel(Level level), который принимает объект класса Level, содержащий одноименные константы для каждого уровня. Если уровень регистратора не указывается, то применяется уровень, унаследованный от его родителя. Уровень корневого регистратора DEBUG. Таким образом, сообще- ния, выводимые с уровнем ниже установленного, в лог не попадут.

И в этом заключается основное преимущество — можно вставлять в программный код вывод информации на различных уровнях (об ошибках — на уровне ERROR, о нормальном ходе выполнения — на уровне INFO, отладочную — на уровне DEBUG), а потом гибко регулировать, что именно будет выводиться. Некоторые общие методы для вывода сообщений: ПРИЛОЖЕНИЕ 2 768 log(Priority priority, Object message, Throwable t) — выводит сообщения указанного уровня с информацией об исключительной ситуации t; log(Priority priority, Object message)  — выводит сообщения указанного уровня.

**Описание проекта**

**Будущее развитие**

В будущем планируется расширение и совершенствование системы:

* Добавление пользователя с правами администратора
* Поиск по возможностям
* Рекомендации пользователей (подборка пользователей с желаниями, соответствующим возможностям данного пользователя).
* Возможность восстановления пароля
* Возможность подписки на рассылки
* Подтверждение адреса электронной почты
* Возможность удаления сообщений, чатов
* Увеличение фото
* При регистрации и поиске пользователей – выбор стран из предложенного списка
* Поздравление именинников (изменение дизайна страницы – добавление дополнительных элементов)
* Ввод ключевых слов при описании своих возможностей.
* Страницы ошибок
* responsive design
* поддержка браузеров

**Заключение**

**Список литературы**

Блинов, И.Н., Романчик, В. С.

Java. Методы программирования : уч.-мет. Пособие — Минск : издательство «Четыре четверти», 2013. — 896 с. ISBN 978-985-7058-30-3

**Приложение1(Схема бд)**

**Приложение2(use cases)**