Министерство Науки и Образования Российской Федерации

Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ»

Кафедра МО ЭВМ

Реферат

по дисциплине «человеко-машинное взаимодействие»

Современные технологии java для разработки приложений:

новые тенденции на примере Java Server Faces

**Выполнил: Эмман П.А.**

**гр. 3351**

**Проверил: Первицкий А.Ю.**

**Санкт-Петербург**

**2007**

Оглавление

[Что такое технология Java? 3](#_Toc186490767)

[Конкурентные преимущества java 4](#_Toc186490768)

[Обзор технологий java 4](#_Toc186490769)

[Появление Web-разработки 9](#_Toc186490770)

[Для чего нужна технология JSF? 10](#_Toc186490771)

[Удобство разработки 11](#_Toc186490772)

[Задачи при разработке Web-приложений с использованием JSF 11](#_Toc186490773)

[Вывод 12](#_Toc186490774)

[Список использованной литературы 13](#_Toc186490775)

# Что такое технология Java?

Технология Java (версии 2) является одновременно объектно-ориентированным языком программирования и платформой, разработанной компанией Sun Microsystems. Технология Java основывается на концепции виртуальной Java-машины (Java virtual machine - JVM) -- транслятора между языком и нижележащей операционной системой и аппартаной платформой. Все реализации языка программирования должны содержать JVM, за счет чего программы, написанные на Java, работают на любой операциооной системе, где существует соотвествующая версия JVM.

*Язык программирования Java* является необычным, поскольку с одной стороны Java программы *компилируются* (в промежуточный язык, называемый Java байт-код), а с другой стороны они *интерпретируются* (байт-код разбирается и исполняется в рамках JVM). Компиляция делается единожды, в то время, как интерпретация происходит каждый раз при запуске программы. Компилированный байт-код является формой оптимизированного машинного кода для JVM; интерпретатор и есть реализация JVM.

*Платформа Java*, которая существует в трех различных версиях, состоит из JVM и прикладного программного интерфейса Java (Java API - Java Application Programming Interface) -- большого набора готовых программных компонентов упрощающих разработку и развертывание апплетов (applets) и приложений, включая надежные, безопасные и интероперабельнные приложения масштаба предприятия. Java API сгруппирован в библиотеки классов и интерфейсов; библиотеки частно называют *пакетами (packages).*

**Различные версии платформы Java**

Когда технология Java набрала популярность, разработчики стали требовать что-то более простое. В ответ на поступавшие просьбы Sun Microsystems создала три версии (редакции) платформы Java - Standard Edition, Enterprise Edition и Micro Edition.

Подробнее о каждой их них :

**J2SE** (Java 2 Standard Edition) - это стандартный Java 2 SDK (Software Develpment Kit), инструментарий, среду исполнения, и программные интерфейсы (APIs) для разработчиков, которые пишут, развертывают и запускают апплеты и приложения на языке программирования Java.

**J2EE** (Java 2 Enterprise Edition). Enterprise версия упрощает создание и развертывание приложений масштаба предприятия, базируя их на стандартизированных модульных компонентых, предоставляя полный набор сервисов для этих компонент, автоматически решая при этом многие вопросы поведения приложений без сложного программирования.

**J2ME** (Java 2 Micro Edition). Эта версия является высокооптимизированной версией среды исполнения Java, которая ориентирована на потребительский сегмент, в частности, ряд малых (физически и по объемам памяти) устройств, начиная от смартфонов или пейджеров до компьютерных приставок.

# Конкурентные преимущества java

Основным достоинством языка Java является его переносимость между платофрмами и операционными системами, а также возможность доставки приложений по относительно недорогим каналам с невысокой пропускной способностью, таким как коммутируемые (dial-up) соединения. Язык также поразительно масштабируем -- существующие приложения могут быть достаточно легко адаптированы для устройств с ограниченными ресурсами памяти. Помимо этого, поскольку Java проектировалась для безопасной работы в сетях, она позволяет это делать при работе через интернет. По сути, технология Java расширяет вычислительные возможности пользователей от настольных приложений до ресурсов развернутых в сети.

Технология Java является важной составной частью инициативы IBM -- бизнеса по требованию (On Demand Business).Java (так же как и Linux) была одной из первых технологий, поддерживающих открытые стандарты на корпоративном рынке, тем самым открывая дверь для использования XML и Web служб для совместного использования информации и приложений различными направлениями бизнеса. Помимо этого, технология Java является основой многих продуктов IBM и услуг технологического консалтинга.

# Обзор технологий java

Возможно, умение обращаться с "солянкой" из аббревиатур технологий Java, является такой же большой работой, как и управление развитием языка в рамках Java Community Process. Ниже приведен частичный список компонент, опицональных пакетов и расширений для каждой версии языка. Другими словами, это те части, которые и формируют платформу в целомю Они дополнены коротким описанием и ссылкой на ресурсы, которые показывают их место в мире платформы Java. *Обратите внимание, что многие из перечисленных ниже ресурсов доступны в различных редакциях платформы Java*.

**Технологии в J2SE:**

* **Java Access Bridge** для for Microsoft Windows выполняет роль моста, который позволяет вспомогательной технологии, базирующейся на Windows, взаимодействовать с Java Accessibility API.
* **JavaBeans Component Architecture** (***компонентная архитектура JavaBeans***) является платформенно-нейтральной спецификацией, которая определяет набор стандартных программных интерфейсов (APIs) для компонент платформы Java.
* **Javadoc** является утилитой для генерации из документационных комментариев в исходных кодах документации по API в формате HTML
* **Java Foundation Classes (Swing)** (JFC) -- это набор библиотек слассов Java, предназначенных для поддержки создания графических приложений и работы с графикой на клиентских местах, построенных на Java.
* **Java Platform Debugger Architecture** (JPDA) является инфраструктурой, обеспечивающей поддержку отладки в Java 2. JPDA включает три многоуровневых API:
  + JDI (Java Debug Interface) -- высокоуровневый программно-языковой интерфейс, включающих поддержку удаленной отладки.
  + JDWP (Java Debug Wire Protocol) определяет формат информации и запросов, передаваемых между отлаживаемым процессом и инструментарием отладчика.
  + JVMDI (Java Virtual Machine Debug Interface) является низкоуровневым "родным" интерфейсом, которым определяет набор сервисов, которые JVM должна предоставить для отладки.

* **Java 2D API** -- это набор классов для продвинутого использования 2D графики работы с изображениями, который предоставляет всестороннюю поддержку формирования изображений и альфа-каналов изображений; набор классов для точного определения цвета и переходов; набор операторов, предназначенных для работы с изображениями на дисплеях.
* **Java Web Start** позволяет вам упростить развертывание прилжоений, предоставляя возможность загрузки и запуска полнофункциональных приложений (например, электронные таблицы) посредством единственного клика, без прохождения процесса инсталляции.
* **Java Database Connectivity** (JDBC) -- это прикладной программный интерфейс (API) который позволяет вам обращаться к большинству табличных источников данных из Java, предоставляя возможность единообразного доступа к различным SQL серварм баз данных, а также к другим табличным источникам данных, таким как электронные таблицы или плоские файлы
* **Remote Method Invocation** (RMI) предоставляет самозагружающуюся службу именования (naming service), не урезает типы данных и использует сериализацию объектов, что позволяет разработчикам создавать распределенные Java приложения, когда метод удаленного Java объекта может быть вызван из другой виртуальной Java машины, потенциально на разных хостах.
* **Java Advanced Imaging** (JAI) -- это прикладной программный интерфейс (API), который предоставляет набор объектно-ориентированных интерфейсов, которые поддерживают простую высокоуровневую модель, позволяющую легко манипулировать изображениями
* **Java Authentication and Authorization Service** (JAAS) -- это пакет, который позволяет сервисиам посредством реализации Java версии стандартного каркаса встраиваемого модуля аутенфикации (PAM -- Pluggable Authentication Module -- framework), а также посредством авторизациипользователей.
* **Java Cryptography Extension** (JCE) -- это набор пакетов, которые предоставляют основу (framework) и реализацию алгоритмов криптографии, генерации и согласования ключей, а также алгоритмы Message Authentication Code (MAC). Они также предоставляют поддержку симметричных, несимметричных, блоковых и потоковых шифров, и поддерживают безопасные потоки и sealed objects.
* **Java Data Objects** (JDO) -- это абстрактная Java модель персистентности, описанная в терминах интерфейсов Java, которая позволяет разработчикам приложений напрямую сохранять их доменную модель Java в долговременном хранилище (базе данных). Потенциально данная модель может заменить такие методы работы с данными как прямой ввод/вывод, сериализация, JDBC и [EJB](http://www-128.ibm.com/developerworks/ru/java/newto/#EJB) Bean Managed Persistence (BMP) или Container Managed Persistence (CMP) Entity Beans.
* **Java Management Extensions** (JMX) (Расширения управления Java) предоставляют средства для построения распределенных, модульных и динамических Web-приложений, для управления и мониторинга устройств, приложений, и сетей, основанных на предоставлении сервисов (service-driven networks. **JMX Remote** расширяет спецификацию JMX, предоставляя возможность стандартизирования средств доступа к своим агентам удаленно.
* **Java Media Framework** (JMF) позволяет добвалять в Java прилжоения аудио, видео и другую временную медиа-информацию.
* **Java Naming and Directory Interface** (JNDI) вляется стандартным расширением, которое предоставляет Java приложениям унифицированный интерфейс к множеству служб имен и каталогов на предприятии, давая возможность использовать бесшовные механизмы взаимодействия с гетерогенными служюами имен и каталогов на предприятии
* **Java Secure Socket Extensions** (JSSE) -- это набор пакетов, которые позволяют использовать безопасные Inernet соединения, реализуя Java версию протоколов SSL (Secure Sockets Layer) и TLS (Transport Layer Security), включая функциональность криптования данных, аутенфикации сервера, целостность сообщений и опциональную аутенфикацию клиентов.
* Спецификация **Java Speech API** (JSAPI), которая также включет спецификации Grammar Format (JSGF) и Markup Language (JSML) specifications, позволяет приложениям Java встраивать голосовые технологии в интерфейс пользователя. JSAPI определяет кросс-платформенный API, обеспечивающий поддержку команд и управления устройст распознавания речи, систем речевого ввода речи и синтезаторов голоса.
* **Java 3D** -- это прикладной программный интерфейс, который разработчики могут использовать для встравивания в Java приложения масштабируемой, платформенно-независимой 3D графики, предоставляя набор объектно-ориентированных интерфейсов, которые поддерживают простую высокоуровневую программную модель.

**Технологии в J2EE:**

* **Java API for XML Processing** (JAXP) поддерживает обработку XML документов с использованием DOM, SAX или XSLT, позволяя приложениям разбирать и трансформировать XML документы, независящие от конкретной реализации работы с XML, предоставляя разработчикам гибкость при замене XML процессоров, без внесения изменений в программный код самих приложений.
* **Java API for XML Registries** (JAXR) предоставляет единообразный и стандартный API для доступа к XML реестрам различных типов (инфраструктуру для построяния, развертывания и поиска Web служб).
* **Java API for XML-based RPC** (JAX-RPC) позволяет разработчикам создавать и нтероперабельные Web службы на базе протоколоа SOAP.
* **SOAP with Attachments API for Java** (SAAJ) позволяет разработчиками порождать и принимать сообщения, соотвествующие спецификации SOAP 1.1 и SOAP c вложениями
* **Common Object Request Broker Architecture** (CORBA) -- это открытый стандарт для расперделенных вычислений, который дополняет Java, предоставляя инфраструктуру для распределенных объектов, службы, поддерживающие эту инфраструктуру и интероперабельность с другими языками.
* **ECperf** -- это эталонный тест, созданный для оценки производительности и масштабируемости Web-ситем. Он включает EJB компоненты, которые собственно и составляют приложение; Web клиента, использующего [JSP](http://www-128.ibm.com/developerworks/ru/java/newto/#JSP) (Java Server Pages), для тестирования пользователя и интерактивной работы, сценарии схемы базы данных и программы загрузки; make-файлы и дескрипторы развертывания (deployment descriptors); и управляющую программу, для реализации правил исполнения теста и симуляции клиентской нагрузки.
* **Enterprise JavaBeans** (EJB) -- это технология, которая использует компонентную модель для упрощения разработки приложений промежуточного слоя, предоставляя автоматическую поддержку для таких служб, как транзакции, безопасность, взаимодействие с базой данных и многие другие.
* **Java Authorization Contract for Containers** (Java ACC) -- это спецификация, которая определяет установку и конфигурацию провайдеров авторизации для их использования в контейнерах. Java ACC определяет интерфейсы, которые провайдеры должны предоставлять, чтобы инструментарий развертывания конейнера мог создавать управлять коллекциями доступа, соотвествующих определенным ролям.
* **JavaMail** -- это API, который предоставляет набор абстрактных классов, которые моделируют почтовую систему.
* **Java Message Service** (JMS) -- это API, который добавляет инфраструктуру провайдеров, позволяющую разрабатывать переносимые Java приложения, работающих по приницпу обмена сообщениями. JMS определяет общий набор концепций обмена сообщениями и программные стратегии, которые будут поддерживаться всеми JMS совместимыми системами обмена сообщениями.
* **JavaServer Faces** (JSF) предоставляют программную модель, которая помогает разработчикам верстать Web приложения создавая на странице повторно используемые компоненты пользовательского инттерфейса, связывая эти компоненты с источниками данных и увязывая программные события на стороне клиента с их обработчиками на стороне сервера.
* Технология **JavaServer Pages** (JSP) позволяет Web разработчикам быстро создавать и поддерживать платформенно-независимые Web страницы, отдельно от пользовательского интерфейса и содержимого страниц, так, чтобы дизайнеры могли измнить макет страницы без изменения динамического содержимого. Технология использует XML-подобные теги, которые включают логику, отвечающую за генерацию содержимого на странице.
* Технология **Java Servlets** (***сервлеты***) -- это подход Java к расширению функционала Web серверов. Они это делают за счет предоставления компонентного, платформенно-независимого метода создания Web приложений без тех ограничений производительности, которые накладывают CGI программы.
* **J2EE Connector Architecture** (JCA) определяет стандартную архитектуру для свзяывания платформ J2EE с копоративными информационными системами (EISs -- Enterprise Information Systems), определя набор масштабируемых, безопасных и транзакционных механизмов, позволяющих поставщику копортаивной системы предоставить стандартный адаптер ресурсов, который встраивается в сервер приложений.
* **Спецификация J2EE Management** (JMX) определяет информационную модель управления для платформы J2EE -- J2EE Management Model. Модель разработана таким образом, чтобы обеспечить интероперабельность со множеством систем управления и протоколами, и включает стандартные отображения модели на Общую модель управления Common Information Model (CIM), стандартной базой данных управляющей информации (SNMP Management Information Base - MIB) и на объектную модель Java через резидентный компонент EJB - J2EE Management EJB Component (MEJB).
* **Java Transaction API** (JTA) это высокоуровневый API, независящий от реализации и используемых протоколов, который позволяет приложениям и серверам приложений осуществлять доступ к транзакциям. **Java Transaction Service** (***Сервис транзакций Java***) (JTS) специфицирует реализацию менеджера транзакций, поддерживающего JTA и реализующего на уровне ниже API отображение на Java спецификации Службы объектных транзацкий 1.1 (Object Transaction Service -- OTS) OMG. JTS распространяет транзакции используя Internet Inter-ORB Protocol (IIOP).

**Технологии в J2ME:**

* **Connected Limited Device Configuration** (CLDC) -- это одна из двух конфигураций, которые конфигурруют среду исполнения Java для мобильных стройств с ограниченными ресурсами. CDLC определяет основной набор библиотек и возможностей виртуальной Java машины, которые должны присутствовать в каждой реализации среды J2ME, которые включают виртуальную машину K (KVM)
* **Mobile Information Device Profile** (MIDP) -- это одна из двух конфигураций, которые конфигурруют среду исполнения Java для мобильных стройств с ограниченными ресурсами. MIDP предоставляет основную функциональность приложений, включая пользовательский интерфейс, возможности сетевого соединения, локальное хранилище данных и управление жизненным циклом приложений.
* **Connected Device Configuration** (CDC) -- это основанная на стандартах инфратсрутктура (framework) для создания и доставки приложений, которые могут совместно использоваться рядом потребителей и встроенных устройств, подключенных к сети.
* Раздел [developerWorks Wireless technology](http://www.ibm.com/developerworks/wireless/) (на английском) содержит несколько статей и учебных пособий по технологиям, связанным с J2ME.

**Сетевые технологии Java:**

* **JAIN** -- это набор прикладных программных интерфейсов на базе Java, позволяющих осуществлять быструю разработку телекоммуникационных приложений и служб следующего поколения. **JSLEE** - это технология событийно управляемого компонентного контейнера, разработанного для высокопроизводительных, асинхронных, отказоустойчивых серверов приложений.
* **Java Metadata Interface** (JMI) -- это спецификаци, которая реализует платформменно-нейтральную инфраструктуру, позволяющую создавать, хранить, очуществлять доступ, искать и обмениваться метаданными. Она базируется на спецификации Meta Object Facility (MOF), разработанной Object Management Group (OMG) и состоит из набора базовых вретфактов мобелирования, описанных с использованием языка UML (Unified Modeling Language).
* **JavaGroups** -- это программный инструментарий (библиотека прикладных программных интерфейсов), предназначенный для проектирования, реализации и экспериментирования с решениями для распределенных систем.
* **Jini** -- это открытая архитектура для создания высокоадаптивных сетецентричных программных и аппартаных служб. Системная архитектура Jini состоит из трех категорий: программной модели, инфраструктуры и служб.
* **JXTA** -- это набор открытых протоколов, которые позволяют любому стройству, подсоединенному к сети, взаимодействовать и работать по приницпу равноправнх узлов (P2P). JXTA узлы создают виртуальную сеть, где любой узел может взаимодействовать с другими узлами и ресурсами напрямую, даже если некоторые узлы или ресурсы находятся за брэндмауэром или NAT, либо используют другие сетевые траспорты.

### 

### Появление Web-разработки

Взрывообразное распространение Web-разработки произошло в середине 1990-х годов. С появлением языка программирования Java стало можно создавать распределенные Web-приложения и одновременно решать наиболее серьезные клиент-серверные проблемы при помощи новых возможностей:

* **Ограниченные взаимодействия.** Web-модель запрос/ответ обладает всеми характеристиками терминальной разработки. Пользователь вводит информацию в форму, выполняет запрос и получает ответ. "Болтливость" при взаимодействиях исчезла, а производительность повысилась.
* **Архитектуры, не использующие общие ресурсы.** Основанная на сервлетах модель программирования могла не сохранять состояния. Это означало, что один сервлет мог обслуживать любого клиента, а фиксированный пул сервлетов мог обслуживать намного большее число клиентов. Вы не должны были резервировать сервлет для каждого пользователя. Производительность улучшилась еще больше.
* **Общие стандарты для клиентов.** Неожиданно, развернув общий браузер у всех клиентов, вы могли создать один интерфейс и взаимодействовать практически со всеми клиентами. Поддержка многих браузеров была проблематичной, но это и близко не было так трудно, как поддержка родных библиотек пользовательских интерфейсов. Многие проблемы переносимости просто исчезли.
* **Более совершенная модель развертывания.** Используя браузер в качестве стандартного клиентского приложения, стало намного проще распространять разработки. Компания могла развернуть приложение на паре Интернет-серверов и предоставить доступ к ним для всего предприятия. Сетевая архитектура часто могла распределять запросы по нескольким серверам, поэтому увеличить мощность стало так же просто, как установить еще один сервер. Развертывание на стороне клиента было настолько простым, что сводилось к проверке установки у клиента нужного браузера. Обслуживание значительно упростилось.

Производительность, масштабируемость и переносимость, все стало намного лучше, и Интернет-революция включила последнюю передачу. Но мы должны были пойти на некоторые значительные компромиссы.

### Для чего нужна технология JSF?

Вообще говоря, JSF – это технология построения Web-приложений.

Зачем же нужен еще один фреймворк для построения веб-приложений, ведь уже есть сервлеты, JSP, Struts, Tapestry и еще много других библиотек? Дело в том, что такие технологии как сервлеты, JSP (и некоторые другие) требуют от программиста много рутинной работы, которая напрямую не относится к кодированию бизнес логики приложения. Большинство этих библиотек не скрывают от программиста, что он работает с простыми HTTP запросами и ответами. И программисту приходится самому разбирать запросы, выбирать из ник параметры, обрабатывать данные, проверять их на непротиворечивость (к примеру введенные пользователем в форму), самому формировать ответы, и многое другое. Кроме того для остальных фреймворков зачастую нет IDE, облегчающих процесс создания приложений. Еще одной проблемой является то, что даже популярные фреймворки развиваются только работе энтузиастов, а не являются промышленными стандартами.

JSF проектировалась для решения обозначенных выше проблем. Разработанная быть гибкой, технология JavaServer Faces усиливает существующие, стандартные концепции пользовательского интерфейса (UI) и концепции Web-уровня без привязки разработчика к конкретному языку разметки, протоколу или клиентскому устройству. Классы компонент пользовательского интерфейса, поставляемые вместе с технологией JavaServer Faces, содержат функциональность компонент, а не специфичное для клиента отображение, открывая тем самым возможность рендеринга JSF-компонент на различных клиентских устройствах. Совмещая функциональность компонент интерфейса пользователя со специальными рендерерами, разработчики могут конструировать специальные теги для заданного клиентского устройства. В качестве удобства технология JSF предоставляет специфичный рендерер и специальную библиотеку JSP-тегов для рендеринга на HTML-клиенте, позволяя разработчикам приложений на J2EE платформе использовать технологию JSF в своих приложениях.

1. Во-первых, JSF – это спецификация, а спецификация, в свою очередь, - это стандарт.
2. Во-вторых, раз все реализации соответствуют стандарту, то производители программного обеспечения могут создавать IDE и другие инструменты для работы с JSF. Например: IBM WebShpere Application Developer или Sun One Studio Creator
3. В-третьих, JSF значительно упрощает построение пользовательского интерфейса для Web-приложений. Создатели JSF постарались сделать так, чтобы написание JSF приложений больше походило на написание простых приложений для обычных настольных систем. Для достижения этого эффекта, работа с HTTP запросами и ответами спрятана глубоко в недрах реализации, программист же получает в свое распоряжение компонентную архитектуру, набор стандартных UI компонентов (кнопки, чекбоксы, радиокнопки, текстовые поля, таблицы), как и в обычных оконных приложениях. И разработка интерфейса просто сводится к построению нужного UI из этих компонентов. JSF также использует модель event/listener из JavaBeans (как и библиотека Swing). UI компоненты генерируют события (такие как щелчок по кнопке, выбор определенного пункта из списка), а слушатели могут регистрироваться для обработки данных событий.

### Удобство разработки

Технология JavaServer Faces включает:

Набор API для представления компонент пользовательского интерфейса (UI) и управления их состоянием, обработкой событий и валидацией вводимой информации, определения навигации, а также поддержку интернационализации (i18n) и доступности (accessibility).

Специальная библиотека JSP тегов для выражения интерфейса JSF на JSP странице.

Следует отметить, что компоненты - это всего лишь JavaBeans, которые имеют определенную функциональность, но не отвечают за то, каким образом они выглядят для пользователя (ответственным за отображение компонента являются специальные JavaBeans – Renderers).

Существует в JSF такое понятие, как ***конвертор***. Как видно из названия, они конвертируют данные введенные в определенный компонент в объект, и из объекта в строку (для отображения данных в UI компоненте). Это несомненно удобно для разработчика.

К компоненту может быть присоединен ***валидатор*** - объект, который может проверять данные, введенные пользователем. Помимо имеющихся стандартных валидаторв, можно написать свой.

Кроме того программист получает в свое распоряжение мощную систему навигации (правила навигации между страницами описываются в специальном файле конфигурации), что избавляет от рутинной работы по проверке ссылок можеду JSP-страницами и позволяет сосредоточиться на более высоком уровне представления взаимодействия пользователя с системой.

# Задачи при разработке Web-приложений с использованием JSF

Для создания типичного JSF-приложения необходимо последовательно решить следующие задачи:

* Создать JSP-страницу, содержащую html разметку, пользовательские теги, теги серверных элементов управления. Современные IDE позволяют обычно просто нарисовать UI.
* Определить правила навигации между страницами в конфигурационном файле.
* Разработать т.н. backing bean. Обычно этот класса получают наследованием класса FacesBean, в котором реализовано много полезных методов, в том числе, связанных с жизненным циклом jsf-страницы. При необходимости можно эти методы переопределить.
* Описать созданный backing bean в конфигурационном файле приложения(/WEB-INF/managed-beans.xml)

# Вывод

Можно рассматривать JSF как технологию, поднимающую разработку Web-приложений и, в частности, UI, на более высокий уровень. Можно перечислить ряд серьезных преимуществ, которые обеспечивает фреймворк.

Во-первых, при использовании IDE, поддерживающей JSF, разработчику нет необходимости отвлекаться на ручную реализацию UI средствами HTML , обратботку данных в запросе, их валидацию. Как бы залогом правильной работы UI приложения выступает правильность понимания взаимодейсвтия пользователя с программным продуктом (в принципе, прецедентов), что очень важно.

Во-вторых, копонентная модель MVC позволяет отделить внутреннюю логику приложения от интерфейса, что также очень удобно.

В-третьих, быстрая разработка интерфейса позволяет оперативно составлять горизонтальные прототипы приложения для демонстрации их пользователю и согласования с ним функциональности Web-приложения.

Также можно смело добавить, что при дальнейшей разработке инструментальных средств возможно дальнейшее существенное упрощение разработки Web-приложения. Например задание навигации посредством построения орграфа, где узлами являются страницы, а ребрами – переходы.

# Список использованной литературы

1. Core JavaServer Faces DAVID GEARY, CAY HORSTMANN Addison Wesley, Июнь 15, 2004 ISBN: 0-13-146305-5
2. Kito D. Mann JavaServer Faces in Action (Forwarded by Ed Burns) Manning, 2004
3. <http://www-128.ibm.com/developerworks/ru/edu/os-ag-jsf1/resources.html>
4. <http://www-128.ibm.com/developerworks/ru/java/newto/>
5. <http://www.exadel.com/tutorial/jsf/jsftutorial-kickstart.html>
6. <http://java.sun.com/javaee/javaserverfaces/1.1/docs/tlddocs/>
7. <http://www.ibm.com/developerworks/ru/library/j-cb03216/index.html>