МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный университет

имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра математических проблем управления и информатики

Допущен к защите

Зав. кафедрой В.С. Смородин

"\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

**Комплекс заданий на обучение разработке JSP-страниц в системе дистанционного обучения DL.GSU.BY**

Дипломная работа

Исполнитель

Студент группы ПМ-41 \_\_\_\_\_\_\_\_\_\_\_\_\_ Р.Д. Осадчий

Научный руководитель

к.т.н., доцент кафедры МПУиИ \_\_\_\_\_\_\_\_\_\_\_\_\_ М.С. Долинский

Рецензент

к.ф.-м.н., доцент кафедры ФиПМ \_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Якубович

Гомель 2018

**Содержание**

[Введение 3](#_Toc514726600)

[1 Инструментальные средства 5](#_Toc514726601)

[2 Теоретические основы разработки JSP 12](#_Toc514726602)

[2.1 Настройка окружения для разработки JSP 12](#_Toc514726603)

[2.1 Взаимодействие Intelij Idea c DL, DLVM 19](#_Toc514726604)

[2.2 Настройка и подключение Apache Tomcat к Intelij Idea 32](#_Toc514726605)

[2.3 Описание структуры приложения 36](#_Toc514726606)

[3 Технология создания JSP 38](#_Toc514726607)

[3.1 Сервлет для показа исходного кода JSP страницы 38](#_Toc514726608)

[3.2 Div контейнер для показа кода 39](#_Toc514726609)

[3.3 Ajax запрос к сервлету для показа кода 41](#_Toc514726610)

[3.4 Особенности разработки JSP для DL 42](#_Toc514726611)

[3.5 Технология внедрения примеров JSP 42](#_Toc514726612)

[5 Примеры JSP страниц 46](#_Toc514726613)

[6 Развитие интерактивных возможностей сайта DL 61](#_Toc514726614)

[6.1 Модификация HTML плеера для flash заданий 63](#_Toc514726615)

[6.2 Обновление конструктора для flash заданий 65](#_Toc514726616)

[6.3 Примеры интерактивных заданий на тему JSP 66](#_Toc514726617)

[Заключение 72](#_Toc514726618)

[Список используемых источников 73](#_Toc514726619)

[Приложение А 74](#_Toc514726620)

# Введение

В системе Distance Learning Belarus имеется курс WEB-технологии для начинающих. В данном курсе имеются задачи на разные темы , такие как :

* HTML
* CSS
* JavaScript

Однако для потенциальных разработчиков на DL этого недостаточно. Разработка основного функционала сайта приходится на плечи такой технологии как JSP и языка Java , так же ASP.NET.

Большой проблемой DL на протяжении всей истории существования является сильная текучка кадров. Причем новички обычно начинают что-то дописывать, имея мало знаний и опыта, а как только путем проб и ошибок их приобретают, уходят работать за деньги. Остающийся код зачастую не блещет ни качеством работы, ни документированностью. Часто такой код становится [мертвым](http://confluence.newit.gsu.by/display/dldev/Dead+Code), так как требует существенной поддержки, а никто из оставшихся не в силах в нем разобраться. В итоге, общее качество, гибкость и производительность DL оставляют желать лучшего.

Для облегчения обучения потенциальных разработчиков было принято решение выполнить курс полезных примеров по технологии JSP. Где будут собраны полезные приемы разработки, необходимые начальные навыки, а так же полезная литература.

Любая разработка не обходится без препятствий и проблем. В процессе разработке актуальными проблемами стали:

1) Возможность обновления проекта, при этом, не затрагивая, важные исходники DL. Так как при использовании боевых исходников DL и внедрении их на сайт приходилось перезагружать сервер, что являлось критичным.

2) Возможность поддержи системы контроля версий. Так как над проектом работал не один человек, это являлось критичным, обмен актуальными исходниками в последующем стал возможным благодаря системам контроля версий.

3) Создание адекватных и действительно полезных примеров. Код DL не блещет своей красотой, код писался очень давно, писали его, в том числе и студенты.

4) Создание адекватной теории и объяснения примеров.

5) Так же актуальной проблемой являлась до недавнего времени демонстрация исходного кода.

6) Несовпадение мнения сторон.

В процессе реализации данного проекта возникла необходимость поддержки проекта с примерами системами контроля версий. Такой системой было решено выбрать Git c хранилищем репозитория на github.com. Что позволило реализовать возможность разработки несколькими людьми, а так же настройка песочницы, где пользователи могли попробовать работать с примерами.

Стоит отметить и информационный портал DL называемый confluence, куда была перенесена вся имеющаяся информация по DL. Где молодые разработчики смогут найти различные руководства по настройке и советы от старшего поколения.

Так же у DL есть так называемая Jira, призванная улучшить разработку. Пользователи оставляют на форуме различные темы с багами и новыми разработками. Там можно обсудить проблему и посмотреть работы, проведенные над проблемой. Прежде чем что-либо внедрять необходимо, составить соответствующую заявку, описать суть проблемы, выставить приоритет и назначить разработчика, который будет решать проблему, зачастую разработчиком назначается человек, который и составляет запрос.

У DL есть своя система контроля версий с различными репозиториями для разных проектов.

Существует проблема с flash-плеером. Постепенно глобальные корпорации отказываются от flash плеера в пользу HTML5, таким образом, на данный момент прекращена поддержка flash-плеера в современных браузерах таких как: Chrome, Firefox и т.д. В задачи так же входило внедрение html-плеера и обновление конструктора для flash заданий и установка интерактивных задач на тему JSP. HTML плеер призван во много расширить возможности flash плеера, что касается графики и функционала. Во многом застой с разработкой плееров произошел из-за того что люди которые работали над первыми версия ушли , а заменить их некому.

Однако как прогресс не стоит на месте, так и система обучения не стоит на месте. Впечатляют и результаты, которые показывают обучающиеся в этой системе. Внедряется много интересных решений на сайт, появляется уйму новых идей и предложений требующих пристального внимания.

# 1 Инструментальные средства

**Java** — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее.

Язык Java зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов. Реализация проекта была начата на языке С++, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента — языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

Язык Java потребовался для создания интерактивных продуктов для сети Internet. Фактически, большинство архитектурных решений, принятых при создании Java, было продиктовано желанием предоставить синтаксис, сходный с С и С++. В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты С++.

**JSP** **(JavaServerPages)**— 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.

Код JSP-страницы транслируется в Java-код сервлета с помощью компилятора JSP-страниц [Jasper](https://ru.wikipedia.org/wiki/Apache_Tomcat#Jasper), и затем компилируется в [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) виртуальной машины [java](https://ru.wikipedia.org/wiki/Java)([JVM](https://ru.wikipedia.org/wiki/Java_Virtual_Machine)). [Контейнеры 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), способные исполнять JSP-страницы, написаны на платформонезависимом языке Java. JSP-страницы загружаются на сервере и управляются из структуры специального Java server packet, который называется [Java EE](https://ru.wikipedia.org/wiki/Java_EE) Web Application. Обычно страницы упакованы в файловые архивы [.war](https://ru.wikipedia.org/wiki/WAR_(%D1%82%D0%B8%D0%BF_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0)) и [.ear](https://ru.wikipedia.org/wiki/EAR).

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

Директивы обеспечивают глобальную информацию, касающихся конкретных запросов, направляемых в JSP, и предоставляют информацию, необходимую на стадии трансляции.

 Директивы всегда помещаются в начале JSP-страницы до всех остальных тегов, чтобы parser (анализатор) JSP при разборе текста в самом начале выделил глобальные инструкции. Таким, образом, JSP Engine (среда исполнения JSP), анализируя код, создает из JSP сервлет. Директивы представляют собой сообщения контейнеру JSP.

Синтаксис директив JSP выглядит следующим образом: <%@ директива имяАтрибута="значение" %>.

Директива taglib объявляет, что данная страница JSP использует библиотеку тегов, уникальным образом идентифицируя ее с помощью URI, и ставит в соответствие префикс тега, с помощью которого возможны действия в библиотеке. Если контейнер не может найти библиотеку тегов, возникает фатальная ошибка трансляции. Директива taglib имеет следующий синтаксис:

<%@ taglib uri="URI включаемой библиотеки тегов" prefix="имяПрефикса" %>.

Директива include позволяет вставлять текст или код в процессе трансляции страницы JSP в сервлет. Синтаксис директивы include имеет следующий вид: <%@ include file="Относительный URI включаемой страницы" %>.

Директива include имеет один атрибут - file. Она включает текст специфицированного ресурса в файл JSP. Эту директиву можно использовать для размещения стандартного заголовка об авторских правах на каждой странице JSP: <%@ include file="copyright.html" %>.

Контейнер JSP получает доступ к включаемому файлу. Если включаемый файл изменился, контейнер может перекомпилировать страницу JSP. Директива include рассматривает ресурс, например, страницу JSP, как статический объект.

Заданный URI обычно интерпретируется относительно JSP страницы, на которой расположена ссылка, но, как и при использовании любых других относительных URI, можно задать системе положение интересующего ресурса относительно домашнего каталога WEB-сервера добавлением в начало URI символа "/". Содержимое подключаемого файла обрабатывается как обычный текст JSP и поэтому может включать такие элементы, как статический HTML, элементы скриптов, директивы и действия.

Многие сайты используют небольшую панель навигации на каждой странице. В связи с проблемами использования фреймов HTML часто эта задача решается размещением небольшой таблицы сверху или в левой половине страницы, HTML код которой многократно повторяется для каждой страницы сайта. Директива include - это наиболее естественный способ решения данной задачи, избавляющий разработчика от кошмара рутины копирования HTML в каждый отдельный файл.

Поскольку директива include подключает файлы в ходе трансляции страницы, то после внесения изменений в панель навигации требуется повторная трансляция всех использующих ее JSP страниц. Если же подключенные файлы меняются довольно часто, можно использовать действие jsp:include, которое подключает файл в процессе обращения к JSP странице.

Выражение в странице JSP - это исполняемое выражение, написанное на языке скрипта, указанного в объявлении language (как правило Java). Результат выражения JSP, имеющий обязательный тип String, направляется в стандартный поток вывода out с помощью текущего объекта JspWriter. Если результат выражения не может быть приведен к типу String, возникает либо ошибка трансляции, если проблема была выявлена на этапе трансляции, либо возбуждается исключение ClassCastException, если несоответствие было выявлено в процессе выполнения запроса. Выражение имеет следующий синтаксис: <%= текст выражения %>.

Declarations (Declarations) предназначены для определения переменных и методов на языке скриптов, которые в дальнейшем используются на странице JSP. Синтаксис declarations имеет следующий вид : <%! код Java %>.

В страница JSP можно использовать два типа комментариев:

1. Выводимый комментарий (output comment).
2. Закомментированный блок (commented block).

Выводимый комментарий - это такой стиль комментирования, при котором комментарий выводится в выходной поток и отображается в браузере. Синтаксис комментариев данного типа следующий:<!-- Текст комментария -->.

Можно также использовать комментарии с динамическим содержимым. Для этого внутрь закомментированного блока должен быть встроен тег скриптлета, например: <!-- Начало комментария <%= expression %> продолжение комментария -->.

Закомментированный блок - это стиль комментирования, при котором тело закомментированного блока полностью игнорируется, не выводится в выходной поток и, следовательно, не отображается в браузере. Этот стиль обычно используется для поясняющих записей, касающихся фрагментов программного кода страницы JSP.Закомментированный блок имеет следующий синтаксис:<%-- Текст комментария --%>.

**Apache Tomcat** - это контейнер, который позволяет вам использовать интернет приложения такие, как Java сервлеты и JSP (серверные страницы Java). Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java. Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish. Разработка и поддержка Tomcat осуществляется фондом Apache Software Foundation и добровольцами. Пользователи имеют свободный доступ к исходным кодам и бинарным файлам Tomcat согласно лицензии Apache License 2.0. Номера версий Tomcat начинаются с 3.0.x .

Сервлет- является интерфейсом [Java](https://ru.wikipedia.org/wiki/Java), реализация которого расширяет функциональные возможности [сервера](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%B5)). Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.

Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения [веб-серверов](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). Для таких приложений технология [Java Servlet](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82) определяет HTTP-специфичные сервлет классы.

Пакеты javax.servlet и javax.servlet.http обеспечивают интерфейсы и классы для создания сервлетов.

**VMware Workstation** — программное обеспечение [виртуализации](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), предназначенное для компьютеров [x86-64](https://ru.wikipedia.org/wiki/X86-64) операционных систем [Microsoft Windows](https://ru.wikipedia.org/wiki/Windows) и [Linux](https://ru.wikipedia.org/wiki/Linux). Позволяет пользователю установить одну или более [виртуальных машин](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) на один физический компьютер и запускать их параллельно с ним. Каждая виртуальная машина может выполнять свою операционную систему, включая [Microsoft Windows](https://ru.wikipedia.org/wiki/Windows), [Linux](https://ru.wikipedia.org/wiki/Linux), [BSD](https://ru.wikipedia.org/wiki/BSD), и [MS-DOS](https://ru.wikipedia.org/wiki/MS-DOS). VMware Workstation разработана и продается компанией [VMware](https://ru.wikipedia.org/wiki/VMware), подразделением [EMC Corporation](https://ru.wikipedia.org/wiki/EMC).

VMware Workstation поддерживает [мосты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BC%D0%BE%D1%81%D1%82) с сетевым адаптером реального компьютера, а также создание общих папок с виртуальной машиной. Программа может [монтировать](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) реальные CD или DVD диски или [ISO образы](https://ru.wikipedia.org/wiki/ISO-%D0%BE%D0%B1%D1%80%D0%B0%D0%B7) в виртуальные оптические приводы, при этом виртуальная машина будет считать, что приводы настоящие. Виртуальные жесткие диски хранятся в файлах [.vmdk](https://ru.wikipedia.org/w/index.php?title=.vmdk&action=edit&redlink=1).

VMware Workstation в любой момент может сохранить текущее состояние виртуальной машины (снимок). Данные снимки позже могут быть восстановлены, что возвращает виртуальную машину в сохраненное состояние.

VMware Workstation включает в себя возможность объединять несколько виртуальных машин в группу, которую можно включать, выключать, приостанавливать или возобновлять как единый объект, что является полезным для тестирования технологий клиент-сервер.

Также доступен [VMware Player](https://ru.wikipedia.org/wiki/VMware_Player) - похожий пакет виртуализации, однако урезанный. Данный пакет бесплатен для некоммерческого использования и свободен в распространении.

**GIT** – это полностью бесплатное программное обеспечение, поддерживающее множество ОС, таких как Mac, Linux, Windows и Solaris. Скачать соответствующие дистрибутивы можно с [официального сайта GIT.](https://git-scm.com/downloads) Вот несколько функций GIT достойных упоминания:

Распределенная система управления версиями, GIT следует принципу одноранговой сети – **peer to peer** (равный к равному) в отличии от других систем вроде Subversion (SVN), которая основана на модели **client-server** (клиент-сервер).

1) GIT позволяет разработчикам иметь множество совершенно независимых веток кода. Создание, удаление и объединение этих веток происходит без каких-либо проблем и больших затрат времени.

2) В GIT все операции атомарны; это означает, что любое действие может быть полностью удачным или провалиться (без каких-либо изменений). Это действительно важно, так как в некоторых системах контроля версий (вроде CVS), где действия не атомарны, некоторые повисшие операции по всему хранилищу, могут оставить его в нестабильном состоянии.

3) В отличии от других VCS, таких как SVN или CVS где метаданные хранятся в скрытых папках (.cvs, .svn, и т.д.), в GIT все данные расположены в каталогах .git.

4) Он использует модель данных, которая помогает обеспечить криптографическую целостность всего, что присутствует в репозитории. Каждый раз когда файлы добавляются или коммитятся, генерируются их контрольные суммы; аналогичный процесс происходит при их извлечении.

5) Еще одна превосходная функция, присутствующая в GIT – это его **индекс**. В пределах индекса, разработчики могут форматировать коммиты и просматривать их до фактического применения.

Git хранит свои данные в виде данные набора слепков небольшой файловой системы. Каждый раз, когда вы фиксируете текущую версию проекта, Git, по сути, сохраняет слепок того, как выглядят все файлы проекта на текущий момент. Ради эффективности, если файл не менялся, Git не сохраняет файл снова, а делает ссылку на ранее сохранённый файл.

**IntelliJ IDEA** — [интегрированная среда разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для многих языков программирования, в частности [Java](https://ru.wikipedia.org/wiki/Java), [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), [Python](https://ru.wikipedia.org/wiki/Python), разработанная компанией [JetBrains](https://ru.wikipedia.org/wiki/JetBrains).

Первая версия появилась в январе [2001 года](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3), которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными [свободными](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%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) инструментами разработчиков, такими как [CVS](https://ru.wikipedia.org/wiki/Concurrent_Versions_System), [Subversion](https://ru.wikipedia.org/wiki/Subversion), [Apache Ant](https://ru.wikipedia.org/wiki/Apache_Ant), [Maven](https://ru.wikipedia.org/wiki/Maven) и [JUnit](https://ru.wikipedia.org/wiki/JUnit). В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке [Ruby](https://ru.wikipedia.org/wiki/Ruby).

Начиная с версии 9.0, среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью [свободной](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%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) версией, доступной под лицензией [Apache 2.0](https://ru.wikipedia.org/wiki/Apache_License), в ней реализована полная поддержка [Java SE](https://ru.wikipedia.org/wiki/Java_SE), [Groovy](https://ru.wikipedia.org/wiki/Groovy), [Scala](https://ru.wikipedia.org/wiki/Scala_(%D1%8F%D0%B7%D1%8B%D0%BA_%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%D1%8F)), а также интеграция с наиболее популярными [системами управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8). В редакции Ultimate Edition, доступной под коммерческой лицензией, реализована поддержка [Java EE](https://ru.wikipedia.org/wiki/Java_EE), [UML](https://ru.wikipedia.org/wiki/UML)-диаграмм, подсчёт [покрытия кода](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0), а также поддержка других систем управления версиями, языков и [фреймворков](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA).

**TortoiseSVN** — это бесплатный клиент для системы контроля версий Subversion (свободная централизованная система управления версиями, официально выпущенная в 2004 году компанией CollabNet), выполненный как расширение оболочки Windows и распространяемый под лицензией GPL.

Будучи клиентом Subversion, TortoiseSVN позволяет управлять файлами и папками во времени. Файлы хранятся в центральном хранилище, в котором запоминается каждое изменение, сделанное в хранимых файлах и папках. Это даёт возможность восстанавливать старые версии файлов и изучать историю их изменения. Поэтому Subversion и другие системы контроля версий часто называют «машинами времени» для файловой системы.

**DLVM** — виртуальная машина DL, локальный сервер DL где собрана вся функциональность основного сайта DL. DLVM поддерживает java 8, так же на виртуальной машине находится база данных DL, а так же сокращенное число задач.

Главными преимуществами DLVM является :

1) Упрощение разработки нового или обновление старого функционала

2) Упрощение обучения новых разработчиков

3) Возможность разработки без вмешательства в работу основного сайта DL

4) Поддержка системы контроля версий.

5) Поддержка Tomcat8.

# 2 Теоретические основы разработки JSP

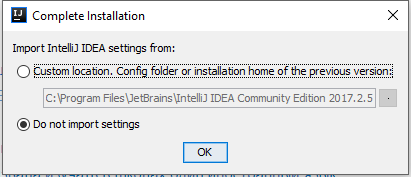
## 2.1 Настройка окружения для разработки JSP

IntelliJ IDEA – одна из самых мощных универсальных сред разработки. Она поддерживает такие языки, как PHP, JavaScript, Sass, Ruby, Python и конечно же Java.

Существует две версии IntelliJ IDEA – Community и Unlimited. Первая – бесплатная, но ограниченная. В ней не поддерживаются фреймворки Spring, Vaadin, GWT, языки JavaScript и TypeScript, SQL и многое другое. Вторая включает в себя всевозможные способы для коллективной разработки, но бесплатна будет только 30 дней пробного периода.

Для установки в ОС Windows нам потребуется скачать с[официального сайта](https://www.jetbrains.com/idea/download/index.html) исходник с расширением exe.  Далее запускаем его и следуем инструкциям установщика.

При первом запуске IntelliJ IDEA перед вами выскочит диалоговое окно с требованием указать путь до файла с настройками. Так как это наш первый опыт знакомства с IDE, то выбираем пункт «Do not import settings». Если данное окно появилось после обновления или переустановки – выберите исходную директорию.



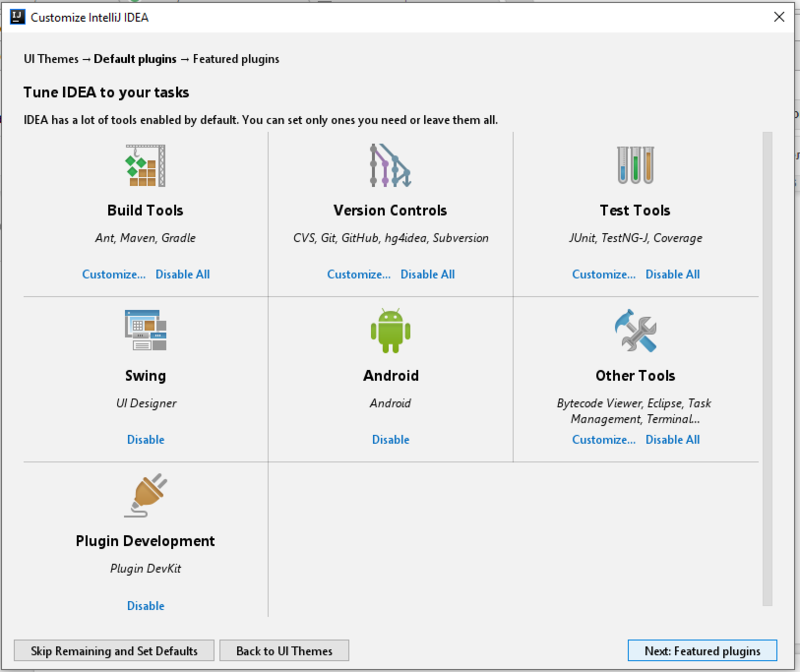
**Рисунок 1 – Диалоговое окно с настройками**

Следующим шагом настройки intellij idea будет выбор темы. ПО умолчанию их две – стандартная (светлая) и Darcula (тёмная).



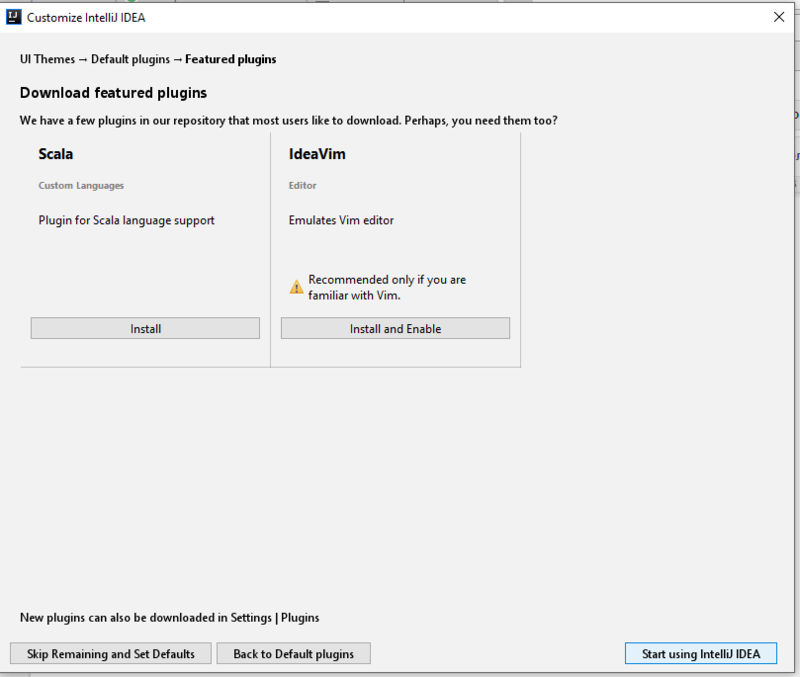
**Рисунок 2 – Диалоговое окно с выбором темы**

Как уже было написано выше, IDEA поддерживает огромное количество вспомогательных инструментов. Поэтому следующим шагом будет их отключение для увеличения производительности. Для этого в очередном диалоговом окне выбираем «Disable All» , кроме Versions. Их в дальнейшем можно подключить позднее в меню настроек.



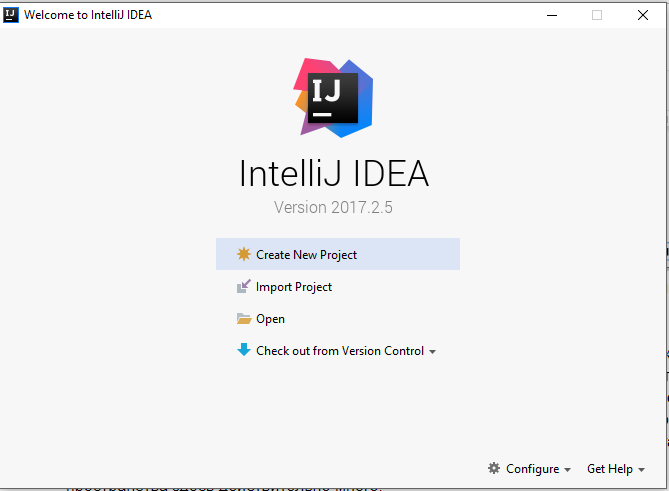
**Рисунок 3 – Диалоговое окно с вспомогательными инструментами**

Далее idea предложит установить плагины, пропускаем данное диалоговое окно.



**Рисунок 4 – Диалоговое окно с плагинами**

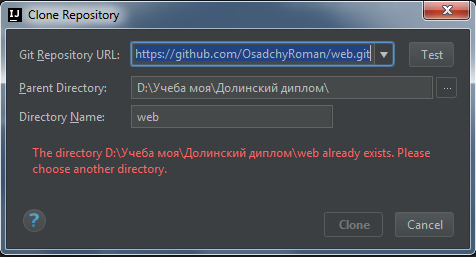
Теперь, когда установка и настройка IntelliJ IDEA закончена, можно приступить к работе. Начинается она, разумеется, с создания IntelliJ IDEA проекта.



**Рисунок 5 – Диалоговое окно для создания проекта**

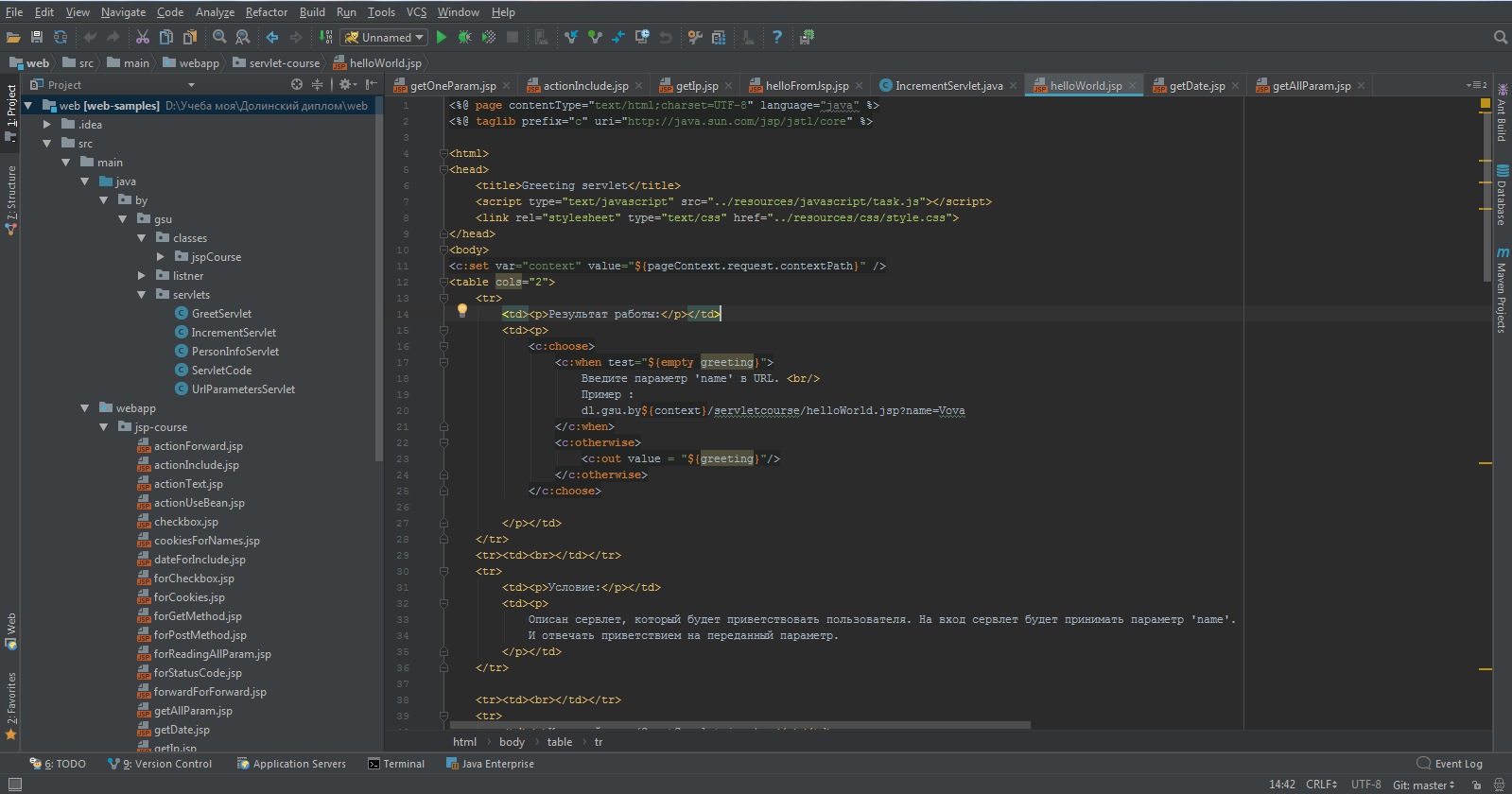
Следующее диалоговое окно предлагает лишь 4 варианта: создать новый проект (Create New Project), открыть существующий (Open), импортировать проект (Import Project) или выйти из системы контроля версий (Check out from Version Control). Выбираем первый пункт. Подробнее о том, как создать проект IntelliJ IDEA, что делать уже внутри среды, лучше почитать на [официальном сайте](https://www.jetbrains.com/idea/documentation/), потому что возможностей для кастомизации рабочего пространства здесь действительно много.

Для проекта с примерами по JSP в диалоговом окне для создания проекта необходимо нажать Check out from Version, для того чтобы среда разработки ,подхватила проект с github.com , где лежит текущий репозиторий проекта с примерами. В выпадающем списке необходимо выбрать GitHub. После чего появится диалоговое окно Clone Repository



**Рисунок 6 – Диалоговое окно Clone Repository**

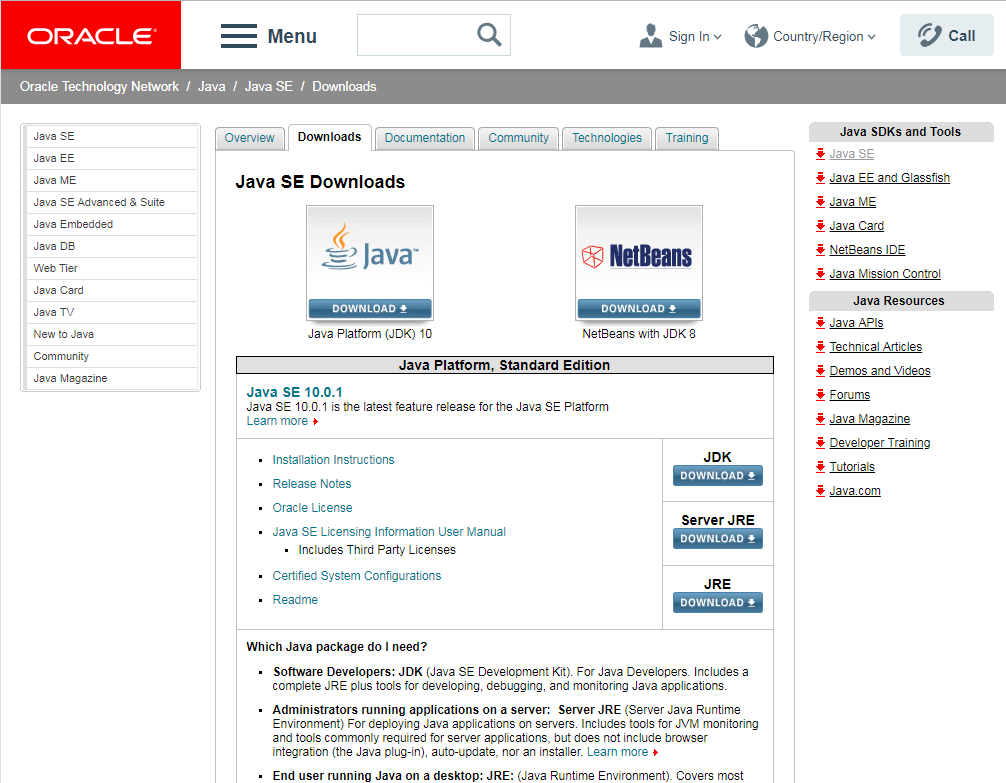
В диалоговом окне показанном на рис.6 требуется в поле Git Repository URL. В поле необходимо вставить ссылку: https://github.com/OsadchyRoman/web . В поле Parent Directory необходимо выбрать каталог куда вы хотите поместить копию проекта из GitHub. В поле Directory Name необходимо ввести название каталога. Далее необходимо нажать кнопку Clone.



**Рисунок 7 – Открытый клонированный проект с github**

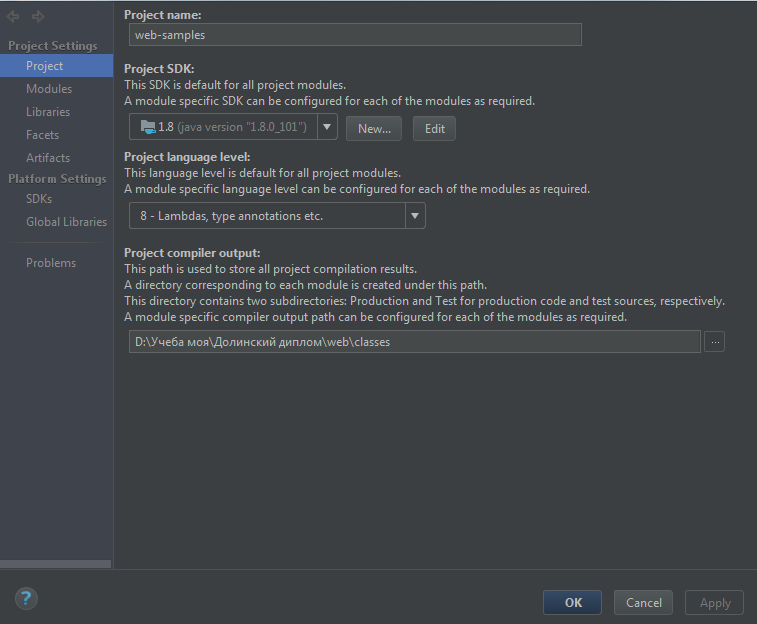
После проделанных действий необходимо скачать и настроить JDK.Java Development Kit (сокращенно JDK) — бесплатно распространяемый компанией Oracle Corporation комплект разработчика приложений на языке Java, включающий в себя компилятор Java(javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java(JRE). В состав JDK не входит интегрированная среда разработки на Java, поэтому разработчик, использующий только JDK, вынужден использовать внешний текстовый редактор и компилировать свои программы, используя утилиты командной строки.

Скачать JDK необходимо скачать по ссылке : <http://www.oracle.com/technetwork/java/javase/downloads/index.html>



**Рисунок 8 – Сайт oracle.com**

Вам необходимо выбрать и скачать необходимую версию JDK , на данный момент нужен JDK 8 , в последующем с обновлением Tomcat , лучше уточнять актуальную версию. В открытом проекте нужно нажать на вкладку file->Project Structrue.



**Рисунок 9 – Диалоговое окно Project Structure**

Далее нажмите на кнопку New и выберите пункт JDK, и укажите путь до скачанного вами JDK. Примените изменения нажав кнопку Apply.

## 2.1 Взаимодействие Intelij Idea c DL, DLVM

После установки у себя на компьютере DLVM начинающие разработчики задаются рядом вопросов, а именно:

1) Как сделать так чтобы DLVM подхватила ваши изменения в java коде(jsp, servlet и т.д)?

2) Как подключить проект DL к Intellij Idea?

3) Как создать патч для Code Review?

4) Как коммитить изменения после Code Review?

По порядку, для начала расскажу где и как именно взять актуальный java код.

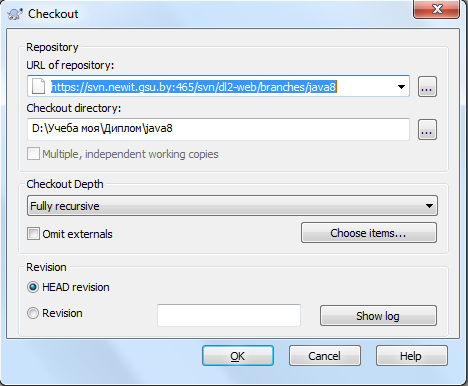
Актуальные исходники java можно скачать из репозитория DL. Для этого вам потребуется система контроля версий Tortoise SVN. Tortoise SVN необходимо скачать с официального сайта, по ссылке : https://tortoisesvn.net/downloads.html



**Рисунок 10 – Официальный сайт Tortoise SVN**

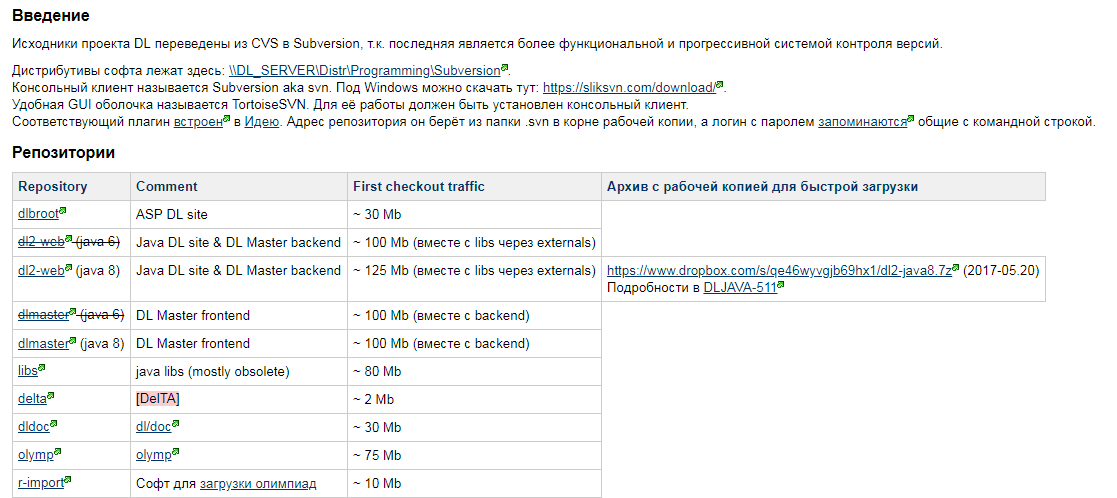
После того как вы скачаете и установите SVN, проверьте чтобы в файлах был svn.exe, он нужен для подключения SVN к Intellij Idea.

Для того чтобы скачать актуальные исходники java, нужно сперва создать папку для исходников. Далее необходимо зайти в папку и нажать правой кнопкой и выбрать пункт SVN Checkout… . После чего вы должны увидеть диалоговое окно Checkout, показанное на рис.2 .



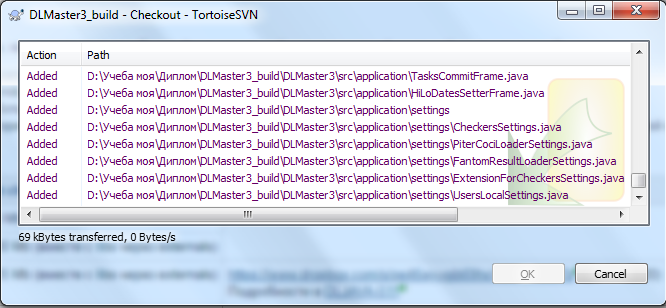
**Рисунок 11 – Диалоговое окно Checkout**

Обратите внимание что на рис.2 показана ссылка с репозиторием на исходники java8, перед работой уточните какая версия Tomcat стоит на самой новой DLVM. В поле URL of repository введите URL репозитория где лежать java исходники. URL можно найти по ссылке : http://confluence.newit.gsu.by/display/dldev/Subversion .



**Рисунок 12 – Актуальные репозитории DL**

После того как найдете нужный репозиторий, для java репозиторий с пометкой dl-web(java 8), вставьте нужный URL в поле URL of repository. После чего нажмите ок. Должна начаться загрузка из репозитория

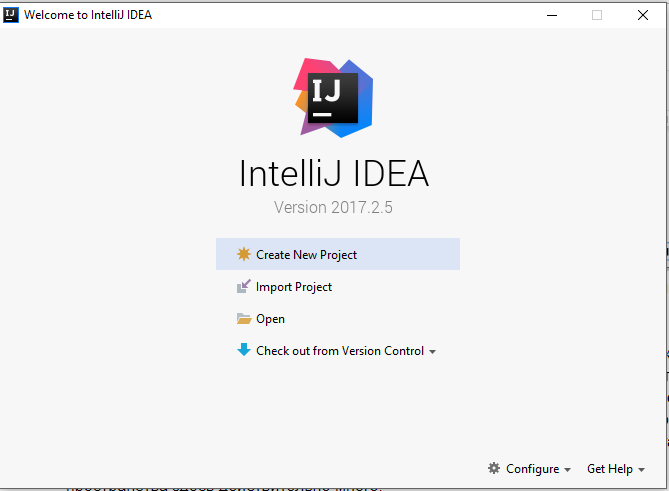


**Рисунок 14 – Загрузка исходников**

На этом загрузка java исходников закончена. Подключите проект к Intelij Idea.

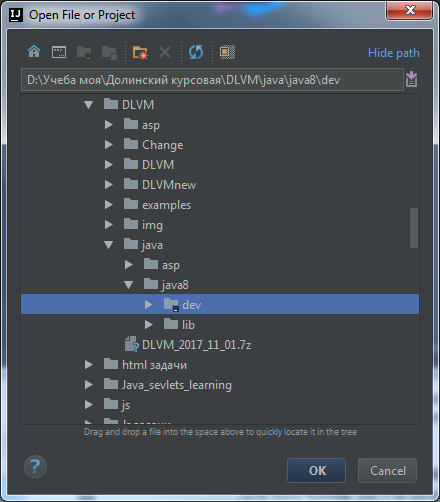
**Как подключить проект с исходниками DL к Intellij Idea?**

Для того чтобы подключить проект с исходниками необходимо в диалоговом окне показанном на рис.5 нажать Open.



**Рисунок 13 – Диалоговое окно Welcom to Intellij Idea**

После того как нажмете кнопку Open, в окне Open File or Project укажите путь до каталога dev скаченных вами исходников и нажмите ок. Пример приведен на рис.6.



**Рисунок 15 – Диалоговое окно Welcom to Intellij Idea**

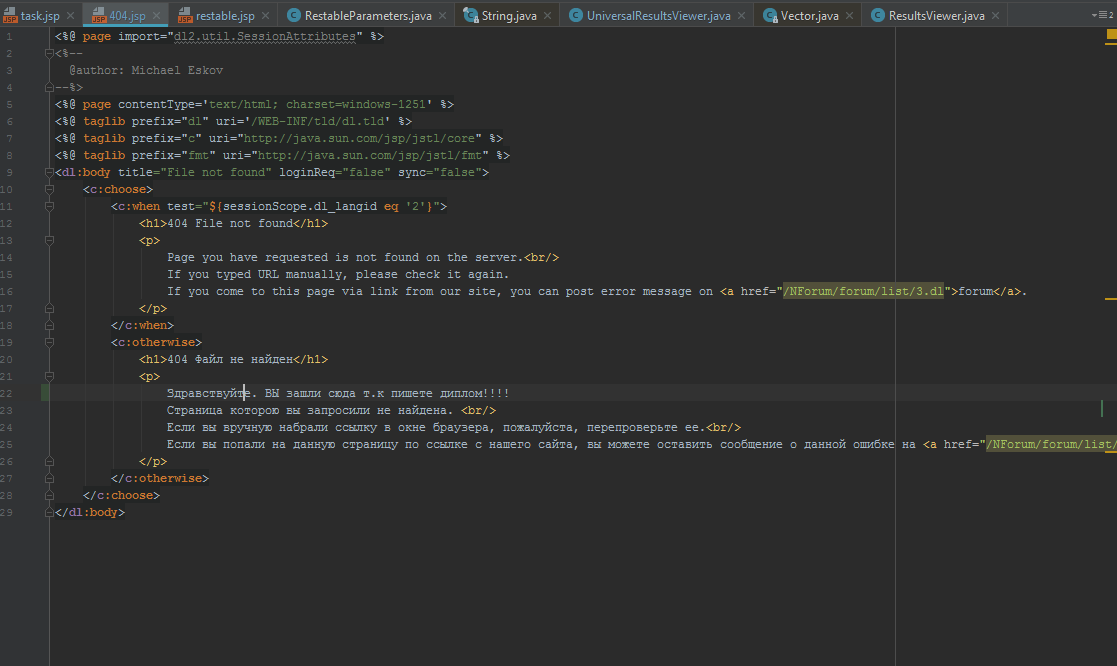
**Как сделать так чтобы DLVM подхватила ваши изменения?**

Для того чтобы DLVM подхватила ваши изменения необходимо :

1) Внести изменения в нужный файл

2) Нажать кнопку выполнения.

Ниже приведен пример в котором , мы добавим приветственную фразу в jsp – файл. Изменим файл 404.jsp , добавив фразу ”Здравствуйте. ВЫ зашли сюда т.к пишете диплом”.



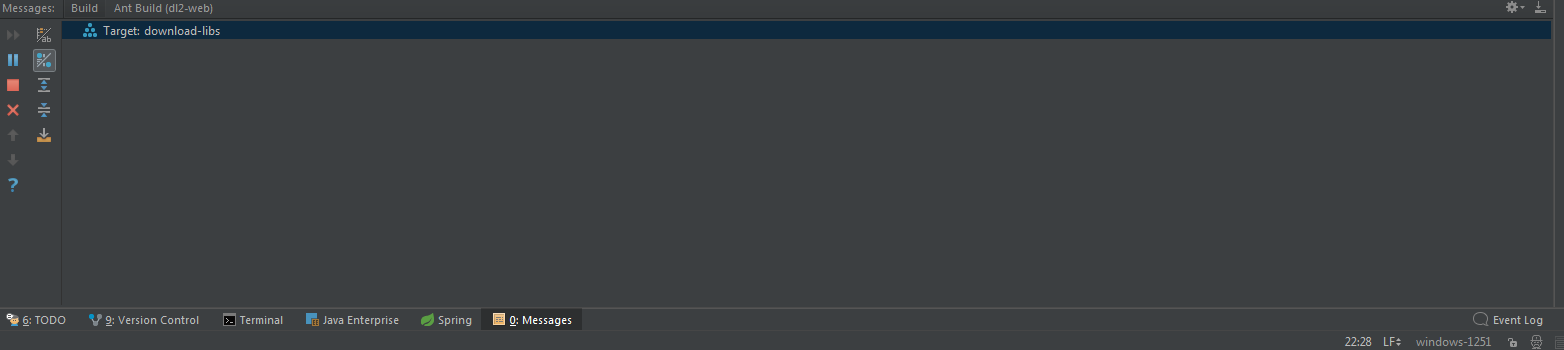
**Рисунок 16 – Изменение файла 404.jsp**

После того как Вы внесли изменения нажмите зеленую кнопку Run.

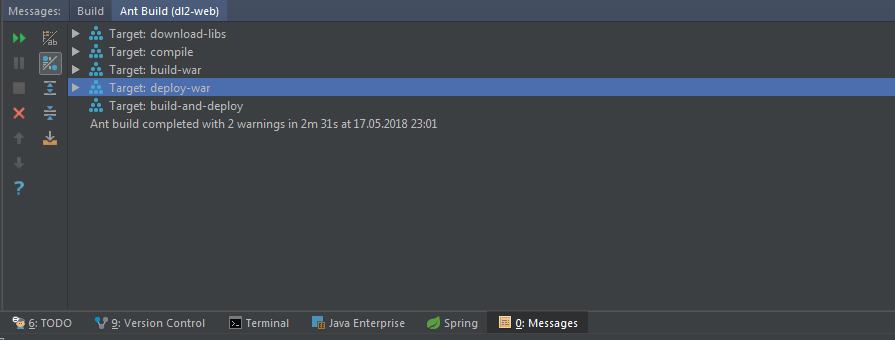


**Рисунок 17 – Зеленая кнопка Run**

После чего среда разработки начнет загрузку библиотек , копиляцию проекта, создавать war и отправит war на DLVM. В war файл упаковывается структура проекта. По сути war файл – это архив с структурой вашего приложения.

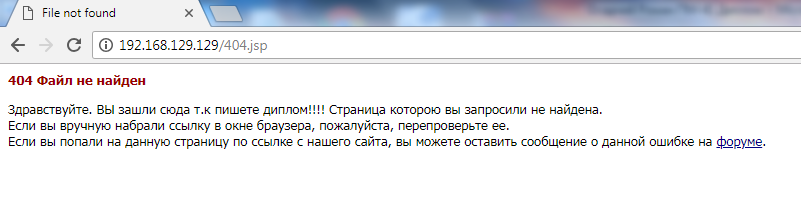


**Рисунок 18 – Начало сборки war и загрузка библиотек**



**Рисунок 19 – Завершение сборки war, отправка файлов на DLVM**

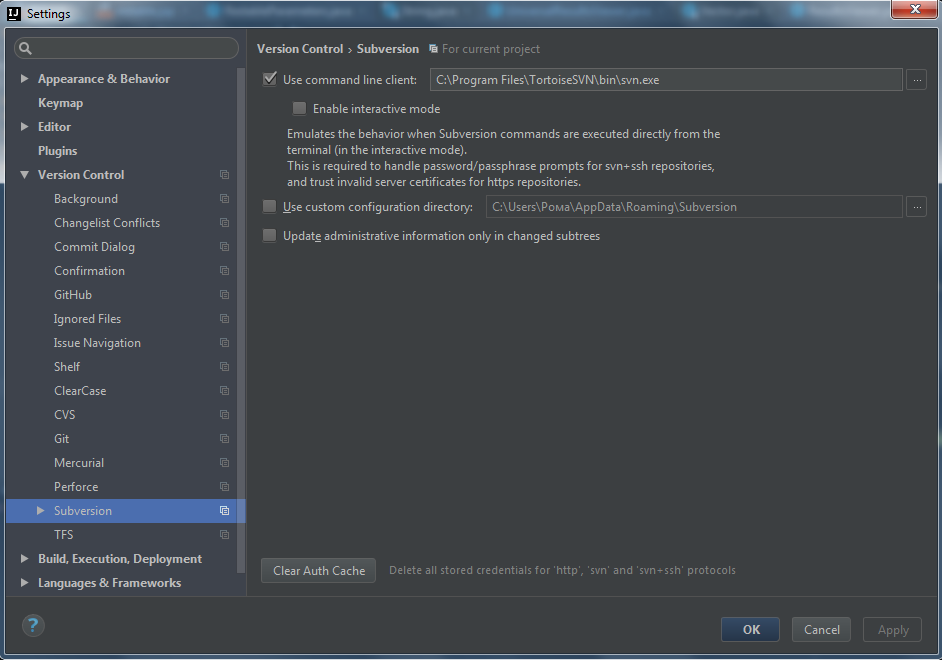
На рис.8 изображен результат проведенных ранее действий.



**Рисунок 20 – Результат изменения**

**Как создать патч для Code Review?**

Необходимо для начала подключить SVN к Intelij Idea. Для того чтобы это сделать необходимо зайти в среду разработки нажать на вкладку File->Settings->Version Control->Subversion.



**Рисунок 21 – Вкладка Settings -> Subversion**

Поставьте галочку Use command line client. И установите путь до файла svn.exe. Нажмите Apply. Теперь ваша среда разработки готова к созданию патча.

Для того чтобы создать патч необходимо найти в среде разработки вкладку VCS.

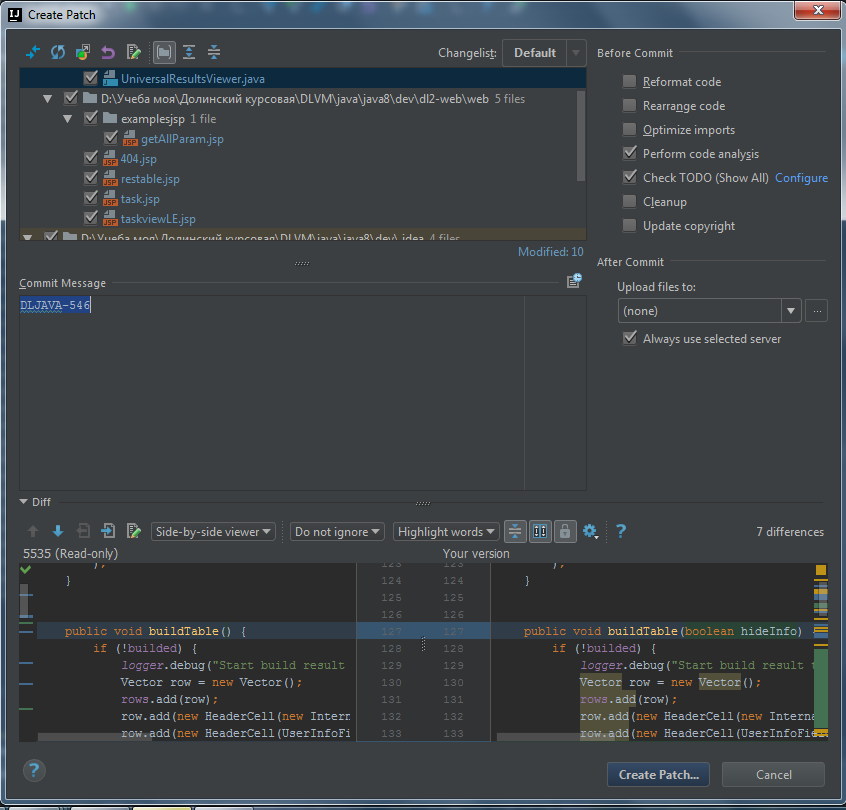


**Рисунок 22 – Пользовательская панель**



**Рисунок 23 – Вкладка VCS**

Нажмите на вкладку VCS. После чего выберете пункт Create Patch.



**Рисунок 24 – Окно Create Patch**

Выберите нужные файлы и нажмите Create Patch и найдите patch у себя на компьютере.

**Как коммитить изменения после Code Review?**

Для того чтобы коммитить изменений необходимо нажать на вкладку VCS которая показана на рис.24. И найти пункт Commit, далее появится окно где необходимо выбрать нужные файлы и указать заявку из Jira.

Большой проблемой DL на протяжении всей истории существования является сильная текучка кадров. Причем новички обычно начинают что-то дописывать, имея мало знаний и опыта, а как только путем проб и ошибок их приобретают, уходят работать за деньги. Остающийся код зачастую не блещет ни качеством работы, ни документированностью. Часто такой код становится [мертвым](http://confluence.newit.gsu.by/display/dldev/Dead+Code), так как требует существенной поддержки, а никто из оставшихся не в силах в нем разобраться. В итоге, общее качество, гибкость и производительность DL оставляют желать лучшего.

Коренными причинами указанной хронически неблагоприятной ситуации являются потеря знаний прошлых поколений разработчиков, отсутствие куратора по используемым технологиям и архитектурным решениям и недостаточная организация процесса разработки. Внедрение нового ПО, подкрепленное организационными мерами, должно помочь в решении указанных проблем.  
 Главная задача - это правильная организация процесса накопления и сохранения информации о проекте и используемых технологиях и передачи её новым разработчикам. Должно быть "никто не забыт, ничто не забыто".

Идиомы Jira:

1) Обсуждение по почте всех вопросов по разработке и поддержке DL, имеющих более чем сиюминутное значение, не оставляет следов для потомков. Поэтому все важные моменты необходимо сохранять в JIRA или Confluence, чтобы осталось для потомков, чтобы можно было сделать поиск, чтобы уведомления автоматически разослались всем заинтересованным лицам, чтобы везде можно было использовать краткую ссылку, всегда отражающую актуальное состояние проблемы.

2) Каждое задание на разработку должно быть четко описано в запросе в JIRA с кратким, но конкретным указанием требуемой функциональности или имеющейся проблемы "назначено" соответствующему разработчику. Все обсуждение задачи должно вестись там же.

3) Суть проблемы в заявке и план её решения необходимо описывать ДО начала работы с кодом, чтобы потом не пришлось много переделывать на нашедшийся более правильный и эффективный способ.

4) В случае добавления новой функциональности разработчик должен кратко описать принятые технические и архитектурные решения.

5) Каждая модификация исходников (коммит) должна быть [правильно прокомментирована](http://confluence.newit.gsu.by/display/dldev/Subversion#Subversion-commits) с указанием номера заявки и кратким текстовым описанием сути изменений. JIRA автоматически связывает правильные коммиты с соответствующими запросами. SVN настроен так, что недокументированные коммиты не принимаются.

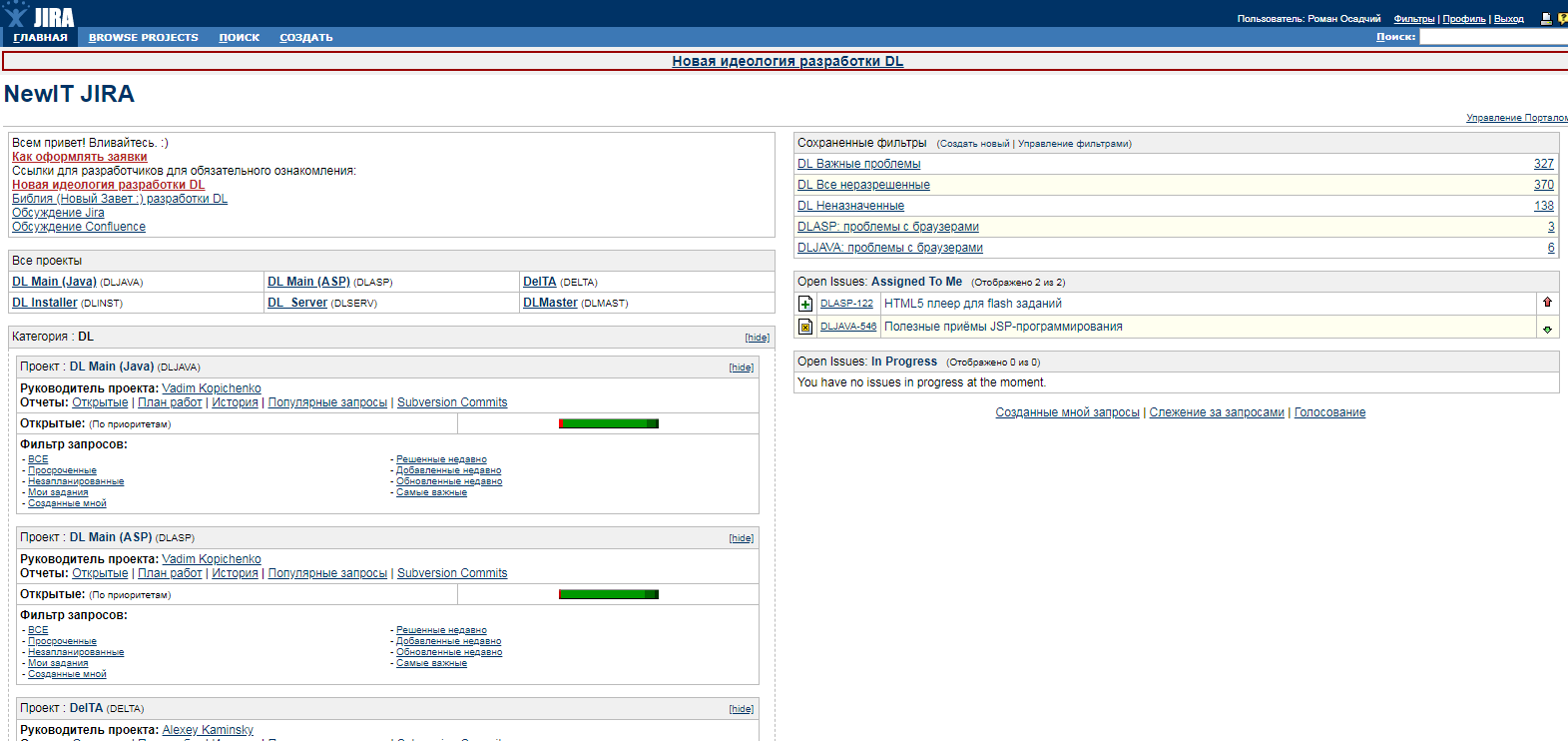
6) JIRA автоматически рассылает почтовые уведомления об изменениях всем разработчикам и заинтересованным лицам. Confluence [тоже так может](http://confluence.newit.gsu.by/display/dldev/Confluence#Confluence-mail), но каждый пользователь управляет своей подпиской сам. Поэтому необходимость в отдельной почтовой переписке сводится к минимуму. Даже в этом минимуме следует слать не текст своих соображений, а ссылку на страницу, где они описаны. Иначе, с большой вероятностью, они будут вскоре всеми забыты.

7)Confluence должен стать основным источником информации о проекте. Ключевым преимуществом по сравнению со старым вариантом хранения DL/Doc в CVS является.

На примере ниже показано как создавать заявку в Jira. Для этого нужно

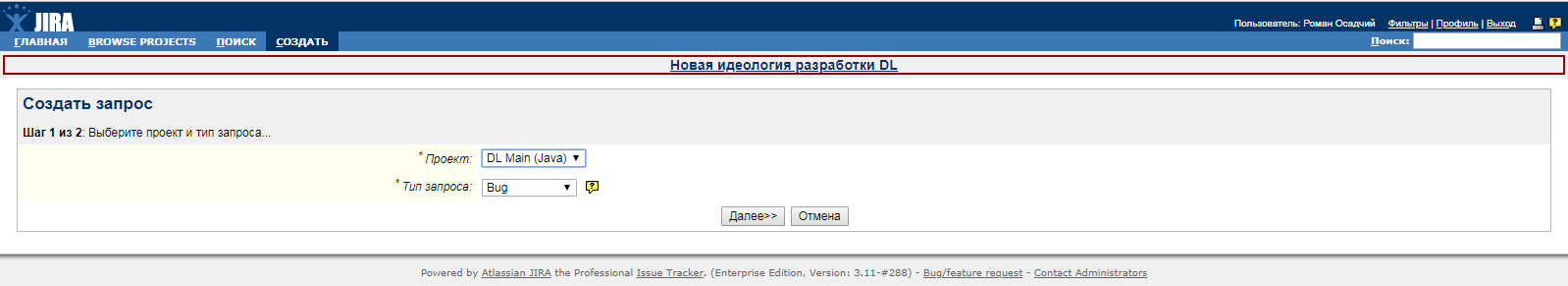
Зарегистрироваться по ссылке: <http://jira.newit.gsu.by/secure/Dashboard.jspa> .

После того как вы зарегистрировались, и вошли вы должны увидеть форум.



**Рисунок 25 – Форум Jira**

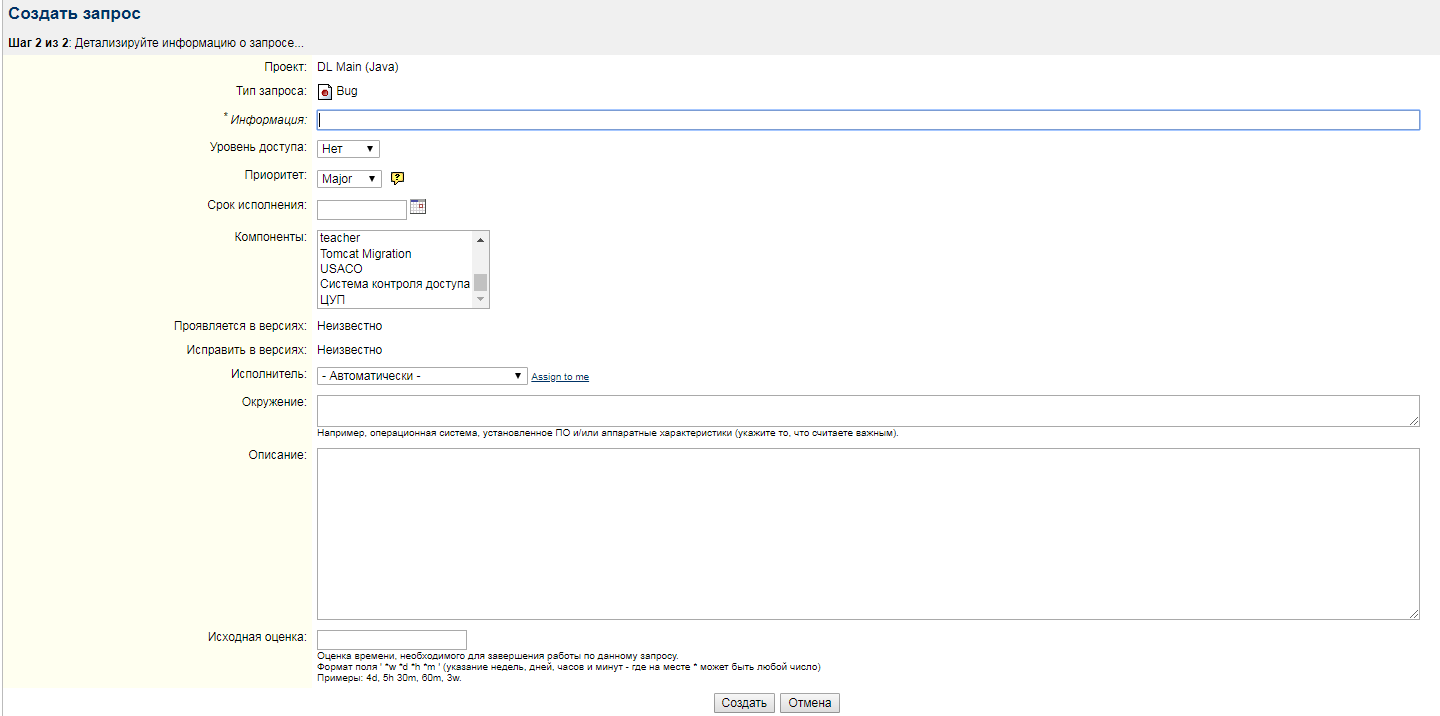
Нажмите вкладку ”Создать”.



**Рисунок 26 – Вкладка ”Создать”**

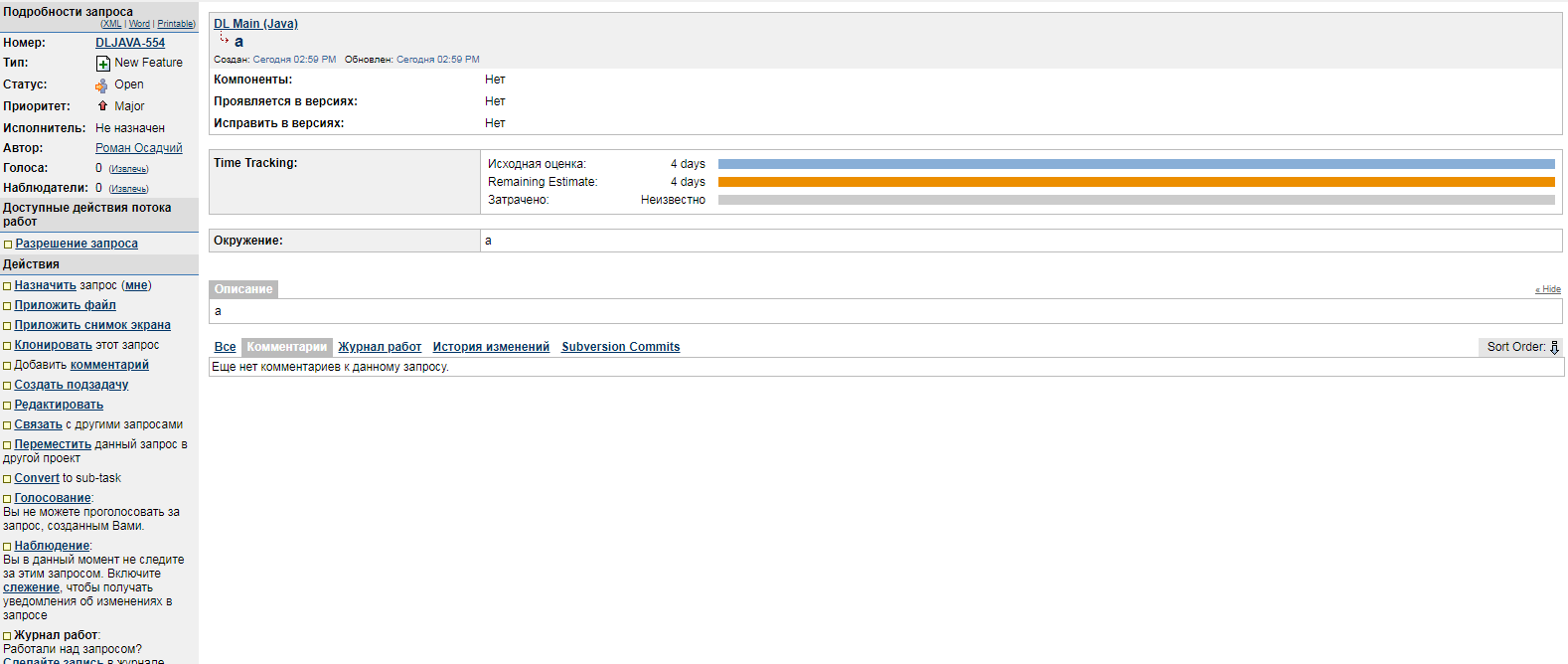
В поле проект выберите проект, в котором проводились изменения.

В поле тип запроса выберите необходимый тип запроса и нажмите кнопку далее.



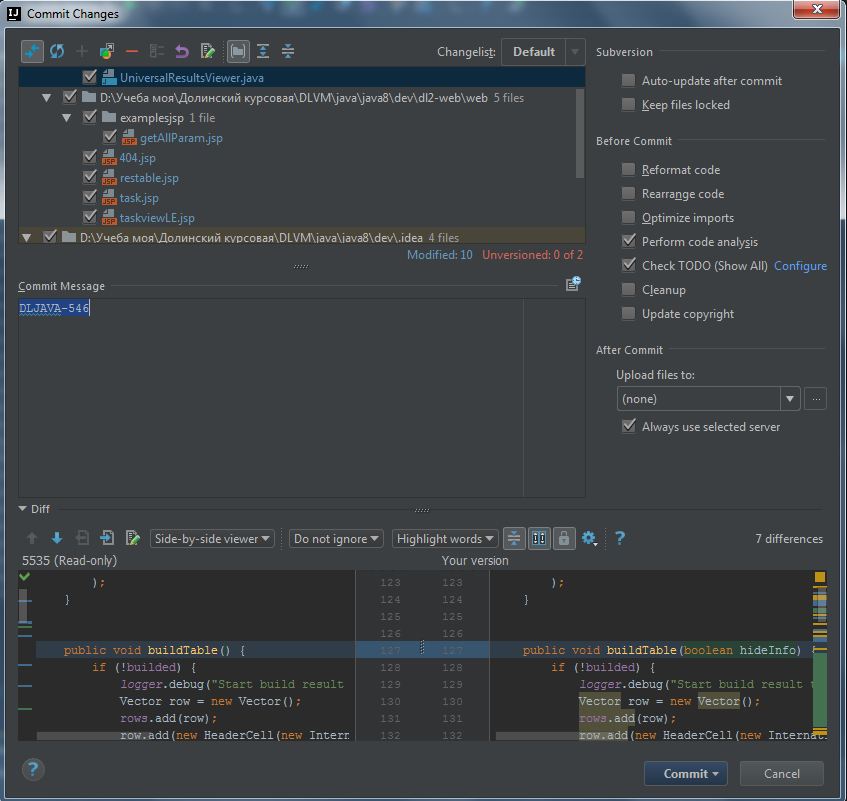
**Рисунок 27 – Вкладка создания запроса**

Во вкладке создания запроса необходимо описать проблему, которую вы нашли, либо которую решили. Если это что-то новое, описать цели создания. Также необходимо выбрать разработчика, который будет решать возникшую проблему. Зачастую если вы создаете тему, то Вы и решаете проблему. Подробно опишите проблему и нажмите кнопку создать.



**Рисунок 28 – Созданный запрос**

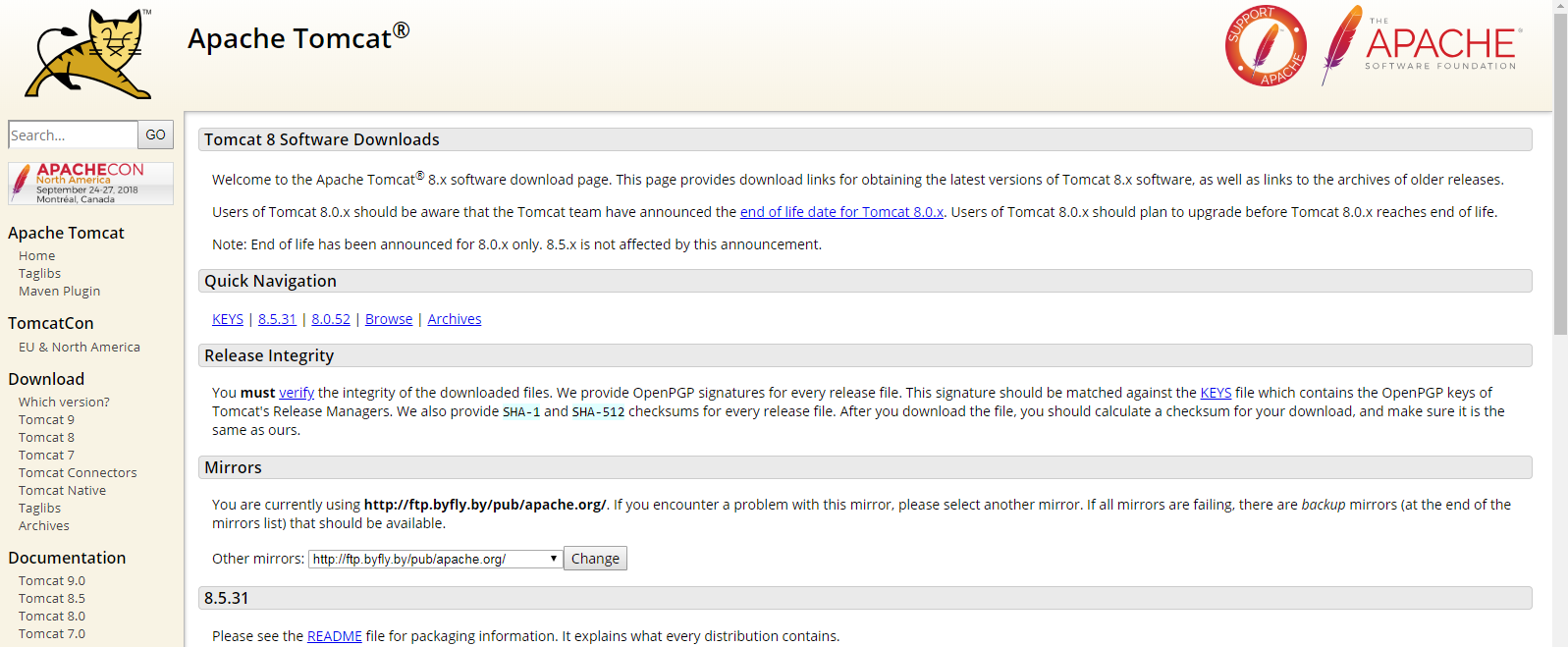
Обратите внимание на номер запроса в данном примере номер запроса: [DLJAVA-546](http://jira.newit.gsu.by/browse/DLJAVA-554). Номер запроса в Jira необходим для того чтобы связывать запрос и коммиты. Теперь можем приступать к коммиту.



**Рисунок 29 – Commit Changes**

## 2.2 Настройка и подключение Apache Tomcat к Intelij Idea

Первым шагом необходимо скачать Apache Tomcat с официального сайта. По ссылке : http://tomcat.apache.org/download-80.cgi .



**Рисунок 30 – Официальный сайт Apache Tomcat**

Выберите версию для скачивания. Я использую версию 7.0.82. После того как нашли нужную вам версию скачайте .zip архив и распакуйте его в нужное вам место.

Для того чтобы подключить Apache Tomcat к Intelij Idea найдите на панели кнопку Select Run/Debug Configuration.

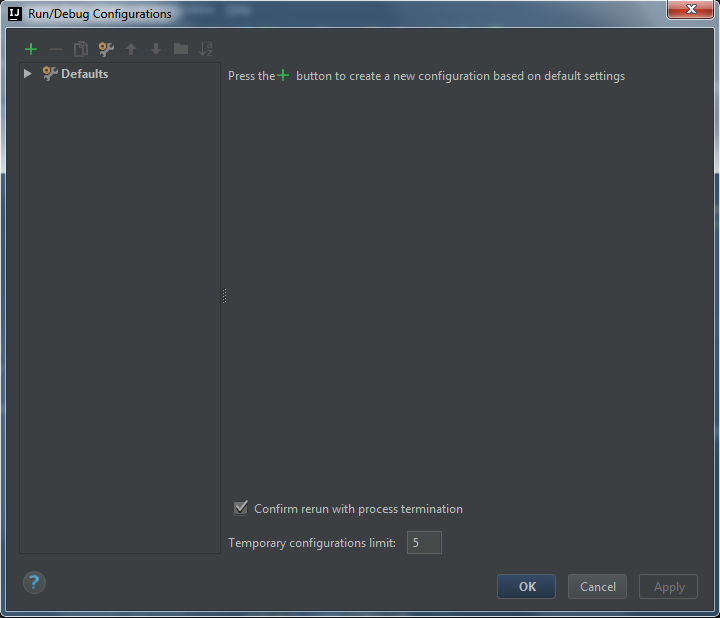


**Рисунок 31 – Пользовательская панель Intelij Idea**



**Рисунок 32 – Кнопка Select Run/Debug Configuration**

После того как вы нажмете Select Run/Debug Configuration перед вами появится окно Run/Debug Configuration.



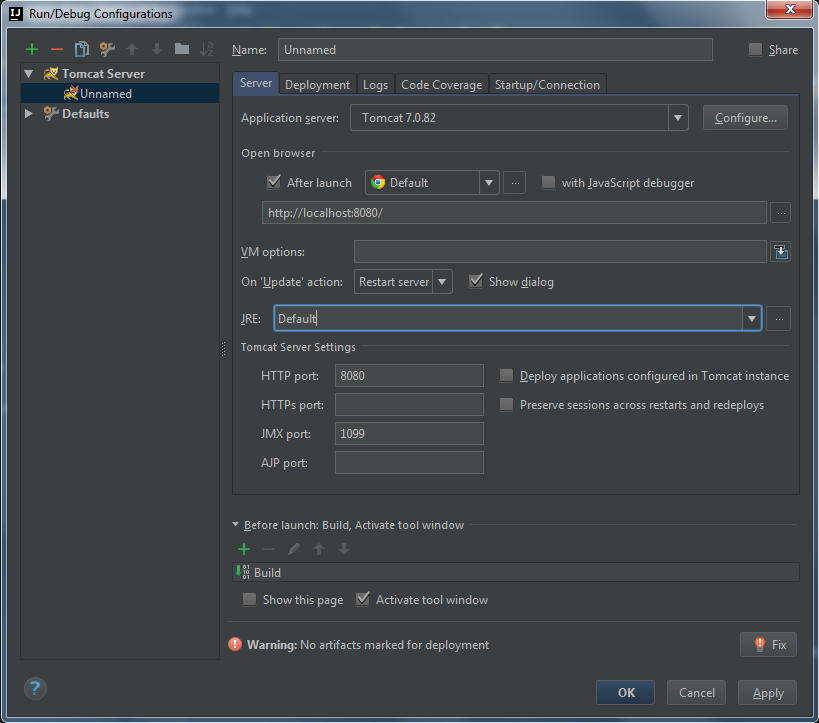
**Рисунок 33 – Окно Run/Debug Configuration**

В окне Run/Debug Configuration необходимо нажать на кнопку Add New Configuration, для того чтобы создать новую конфигурацию основанную на стандартных настройках.



**Рисунок 34 – Кнопка Add New Configuration**

В выпадающем списке необходимо найти вкладку Tomcat Server и в выпадающем списке выберите Local. Перед вами появятся настройки Tomcat.



**Рисунок 35 – Окно c настройками Tomcat**

В настройках Tomcat введите поле Name. После того как ввели поле Name нажмите кнопку Configure… . И выберите каталог куда распаковали архив с Tomcat. Раздел настроек Open browser отвечает за то, в каком браузере будет открываться ваше приложение и по какому адресу будет доступно ваше приложение в браузере. Поле JRE отвечайте за версию JDK установите его Default или выберите нужную Вам, если у вас несколько JDK. HTTP port оставьте стандартным. Если у вас есть сообщение No artifacts marked for deployment. Нажмите на кнопку Fix. И выберите пункт: web-samples:war.

****

**Рисунок 36 – Сообщение об ошибке и кнопка Fix**

Затем нажмите Apply и попробуйте запустить приложение. Если что-то не работает проверьте сделали ли вы все правильно

## 2.3 Описание структуры приложения

У каждого приложения имеется своя структура. В приложении с полезными примерами она тоже имеется. Так как приложение состоит из нескольких курсов : JSP, Servlet, необходимо было четкое разделение этих курсов, что отразилось так же на структуре проекта.

JSP файлы с примерами по JSP лежать по следующему пути: src\webapp\jsp-course.

JSP файлы с примерами по курсу Servlet находятся по пути: src\webapp\servlet-course.

Класс файлы относящиеся к сервлетам лежат по пути src\main\java\by\gsu\servlets.

Если для создания JSP примеров вам необходимо создать класс, его следует положить по пути src\main\java\by\gsu\classes\jspCourse .

По пути src\main\java\by\gsu\classes\listner. Находится файл-фильтр он нужен для того чтобы можно было тестировать JavaScript на стороннем сервисе. Так как без фильтра браузеры не допускают использование другого функционала, если только приложением это не разрешено. Этот подход называется правило одного источника.

Файл .gitignore необходим для того чтобы система контроля версий Git отличала файлы которые нужны и файлы которые являются мусором, к примеру таковым является каталог .idea. Другим разработчикам совершенно не важна информация о вашей idea.

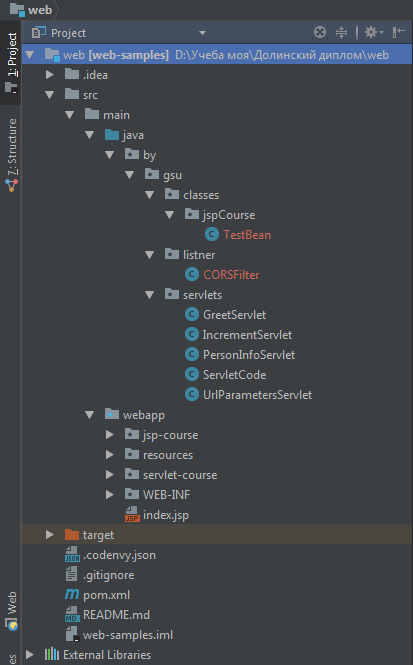
Файл pom.xml(Project object model) - это общая модель проектов. В ней описываются такие общие характеристики как имя, версия, авторы и их контактная информация, VCS проекта и вообще связанные с ним сетевые ресурсы, тип проекта (например библиотека или web-модуль, в оригинальной терминологии называется packaging, хотя влияет не только на тип получаемого артефакта), связи с другими проектами, используемые при сборке плагины и описания способа их задействования.

Файл дескриптор web.xml развертывания определяет какие URL будут передаваться определенному сервлету, какие URL требуют аутентификации и др. Находится в WAR приложения в  WEB-INF/ директории. Web.xml — часть стандарта сервлета для веб-приложений.

Дескриптор развертывания веб-приложений описывает классы, ресурсы и конфигурацию приложения, а так же как сервер будет использовать их для выполнения веб-запросов. Когда сервер получает запрос для приложения, он использует дескриптор развертывания для отображения URL запроса к коду, который должен этот запрос обработать. Файл web.xml определяет соответствие между путями URL и сервлетами, которые эти URL будут обрабатывать. Веб-сервер использует эту конфигурацию, чтоб определить сервлет для обработки данного запроса и вызвать метод класса, который соответствует методу запроса (например метод doGet() для HTTP.GET запросов).

Для привязки («маппинг») URL к сервлету, необходимо определить элементы  <servlet> и <servlet-mapping>.

Элемент <servlet-mapping> определяет шаблон URL и имя сервлета, который будет обрабатывать запросы, соответствующие шаблону. Шаблон должен содержать полный путь URL, включая (/), следующий за именем домена.

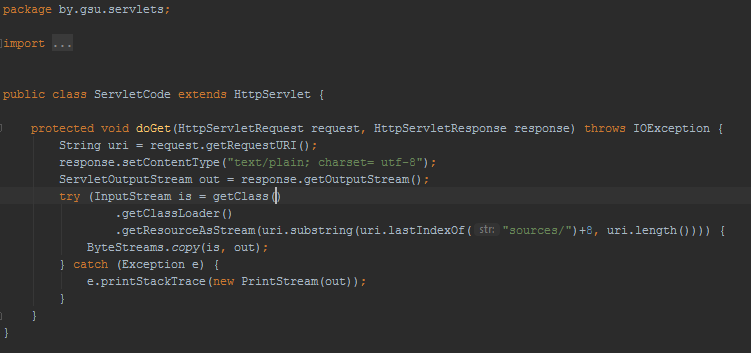


**Рисунок 37 – Структура приложения**

# 3 Технология создания JSP

## 3.1 Сервлет для показа исходного кода JSP страницы

При создании полезных примеров возникал ряд проблем, в последующем с которым боролись разработчики. Одной из проблем было экранирование исходного кода примеров на страницу. Это было очень неудобно, примеры получались громоздкие и неудобно править их. Решением этой проблемы стал сервлет для показа кода. Он находится в структуре проекта: src\main\java\by\gsu\servlets\ServletCode.java



**Рисунок 38 – Сервлет для показа исходного кода**

В данном сервлете параметр uri – это расположение запрошенного файла в структуре приложения.

Строкой ”response.setContentType(”text/plain; charset = utf-8”);” устанавливаем кодировку. Далее в входной поток помещаем нужный файл. И отправляем его в поток вывода сервлета.



**Рисунок 39 – Результат работы сервлета для показа кода**

Что в итоге облегчило создание полезных примеров и способствовало развитию.

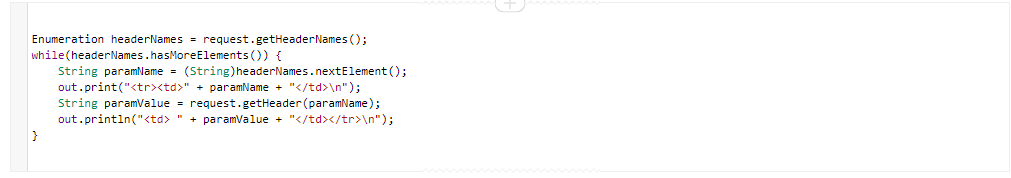
## 3.2 Div контейнер для показа кода

В каждом web приложении есть back-end и front-end. Фронтенд — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса. Бекенд — программно-аппаратная часть сервиса. Фронтенд и бекенд — это вариант архитектуры программного обеспечения.

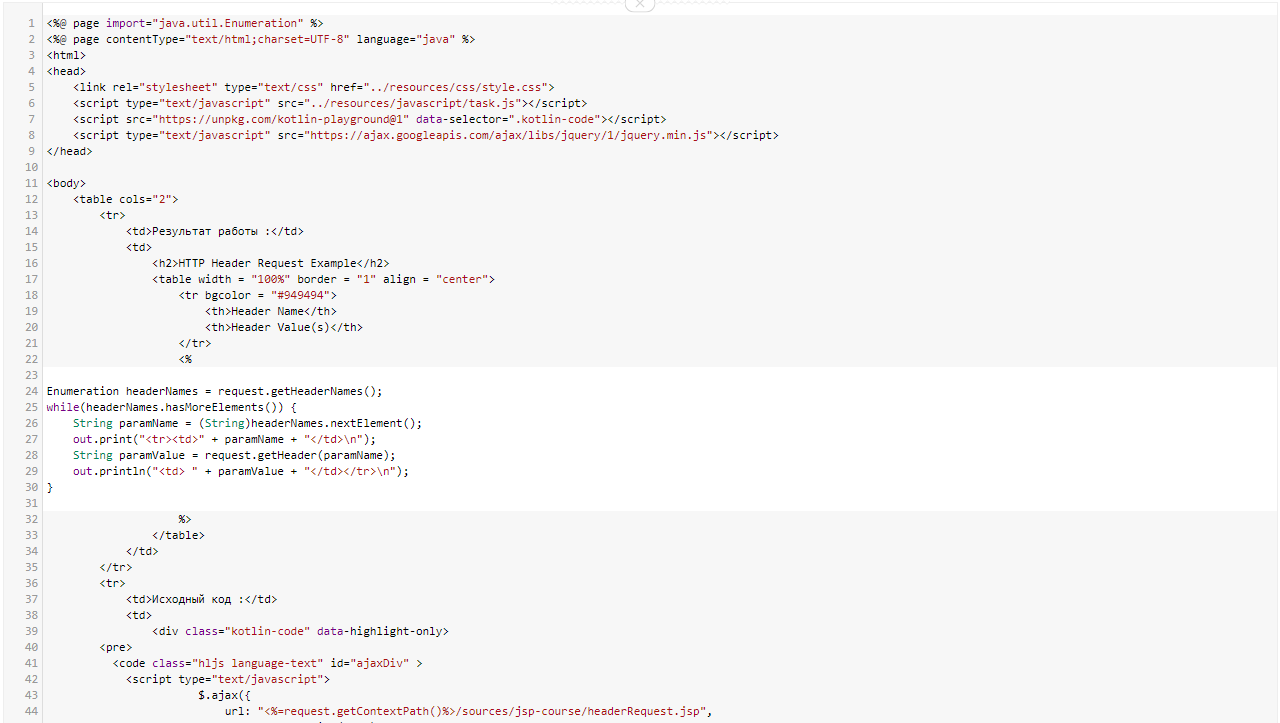
Так как у нас уже есть сервлет для показа кода, необходим и механизм который будет отображать этот исходный код на страницах примеров.

Первым вариантом для этого стал тег <iframe> в который и загружался исходный код. Однако пришлось быстро от этого отказаться так как возможность выделения наиболее важных частей кода затруднялась и приходилось выполнять ”разрезание” кода на стороне сервера, что являлось некорректным. В последующем пришлось заменить этот способ наиболее практичный.

Был внедрен контейнер который позволил бы отображать наиболее важные части кода или скрывать не нужные части, а так же нумеровать строки.



**Рисунок 40 – Отображенная, нужная часть кода**



**Рисунок 41 – Отображение всего кода**

Для того чтобы разместить на странице контейнер для показа кода необходимо разместить следующий код:

<div class="kotlin-code" data-highlight-only folded-button="true">  
 <pre>  
 <code class="hljs language-text" id="jspCode">  
 </code>  
 </pre>  
 </div>

Однако это лишь контейнер без кода. Как вставить код написано в пункте 3.3.

Для того чтобы выделить нужную часть кода необходимо на самой странице примера разместить комментарии-маркеры //sampleStart и //sampleEnd. Лучше их размещать таким образом: <%//sampleStart%> необходимый код <%//sampleEnd%>. Хоть в скрипте и есть код который игнорирует подобные комментарии-маркеры, однако лучше всего размещать их именно так , чтобы не возникло каких-либо проблем и отображения ненужных элементов на странице.

## 3.3 Ajax запрос к сервлету для показа кода

AJAX (аббревиатура от «**A**synchronous avascript **A**nd **X**ml») – технология обращения к серверу без перезагрузки страницы.

За счет этого уменьшается время отклика и веб-приложение по интерактивности больше напоминает десктоп.

Несмотря на то, что в названии технологии присутствует буква X (от слова XML), использовать XML вовсе не обязательно. Под AJAX подразумевают любое общение с сервером без перезагрузки страницы, организованное при помощи JavaScript.

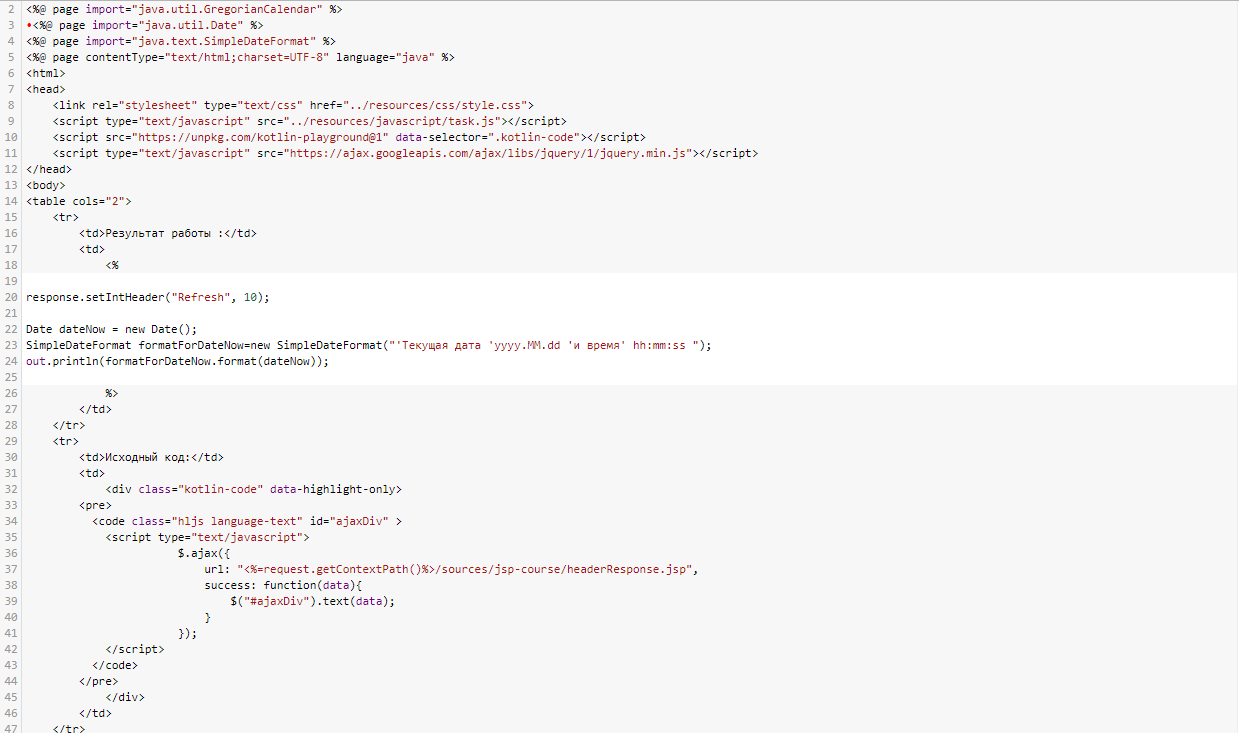
В первую очередь AJAX полезен для форм и кнопок, связанных с элементарными действиями: добавить в корзину, подписаться, и т.п.

Сейчас – в порядке вещей, что такие действия на сайтах осуществляются без перезагрузки страницы.

Мы использовали Ajax запрос для обращения к сервлету показа кода. Код Ajax запроса выглядит следующим образом:

var codeLoad = function(url, element) {  
 $.ajax({ url: url, cache : false})  
 .done(function(response) {  
 $(element).text(response);  
 });  
 };

В функцию приходит URL и id элемента куда нужно вставить код, ответ приходит от сервлета как текст.



**Рисунок 42 – Ответ от сервлета и отображение кода**

## 3.4 Особенности разработки JSP для DL

При разработке JSP для DL руководствуйтесь главными правилами:

1) Помните что JSP – это UI что означает User Interface, не размещайте на странице функциональность, которая отвечает за серверную часть. Вообще старайтесь избегать использование скриптлетов.

2) Старайтесь не усложнять, воспользуйтесь поиском или полезными ресурсами, сейчас литературы более чем достаточно.

3) Не бойтесь пробовать, экспериментировать.

Созданные нами полезные примеры призваны помочь новым разработчикам, направить их.

Основные требования к примерам:

1) После того как вы создали пример, необходимо описать всю функциональность и использованные средства по ссылке: <http://confluence.newit.gsu.by/pages/viewpage.action?pageId=35586049>

2) Создайте ссылку на странице примера на теорию.

## 3.5 Технология внедрения примеров JSP

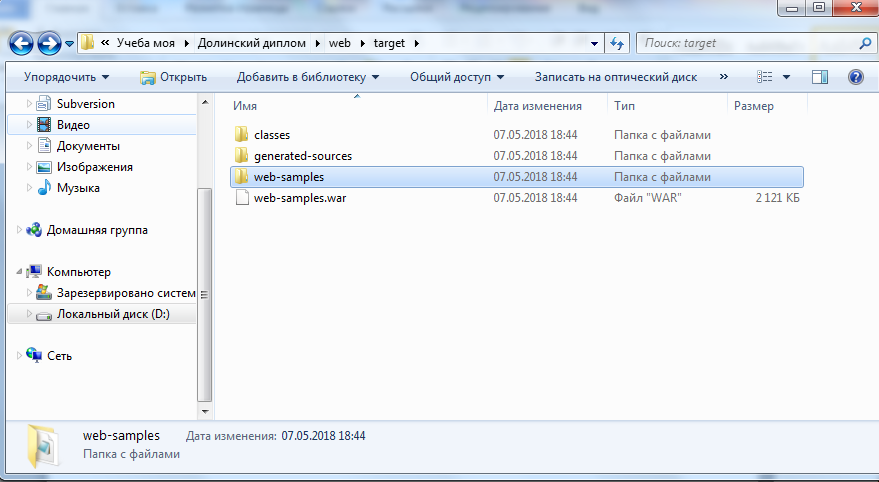
После того как Вы настроите среду разработки и откроете приложение web-samples в Intelij Idea. Вы можете приступать к работе с этим проектом, изменять и добавлять, однако после всех своих изменений и добавлений непременно вы захотите залить все свои изменения на боевую DL. Для этого необходимо:

1) Собрать war файл, делать это удобнее через Intelij Idea.

2) Через терминальный доступ подключиться к серверу DL через сеть университета.

3) Копировать war файл в папку webapps.

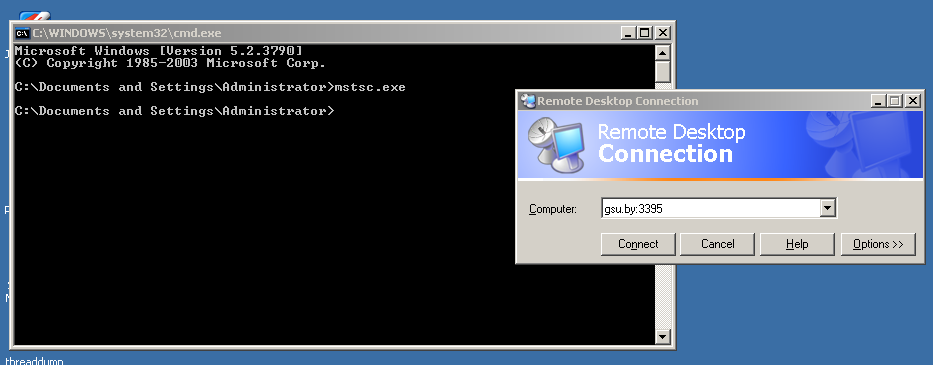
Для того чтобы собрать war файл нужно подключить Tomcat к проекту, как это сделать написано в пункте 2.2. И запустить выполнение. War файл вы можете найти по пути: каталог вашего проекта\target\web-samples.war. Пример каталога с war файлом продемонстрирован на рис.30.



**Рисунок 43 – Каталог приложения с war файлом**

Теперь нужно подключиться через терминальный доступ к серверу DL. Но прежде чем подключиться необходимо получить права у администратора. После того как вы получили права у администратора можно подключиться к сети университета. Удобнее это делать в DLVM.

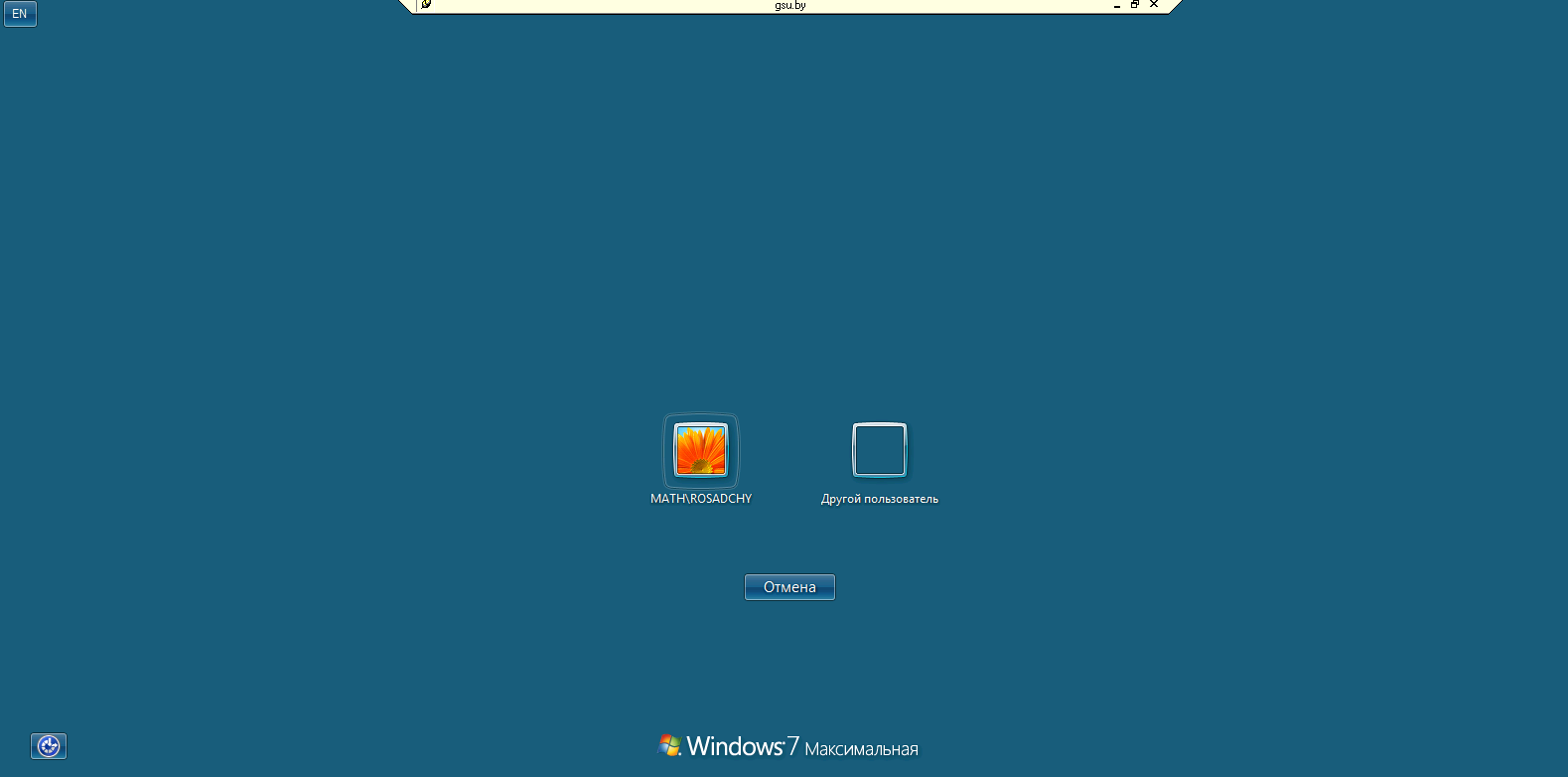
После того как запустили DLVM запустите командную строку и пропишите в ней mstsc.exe.



**Рисунок 44 – Mstsc.exe и указание компьютера для подключения**

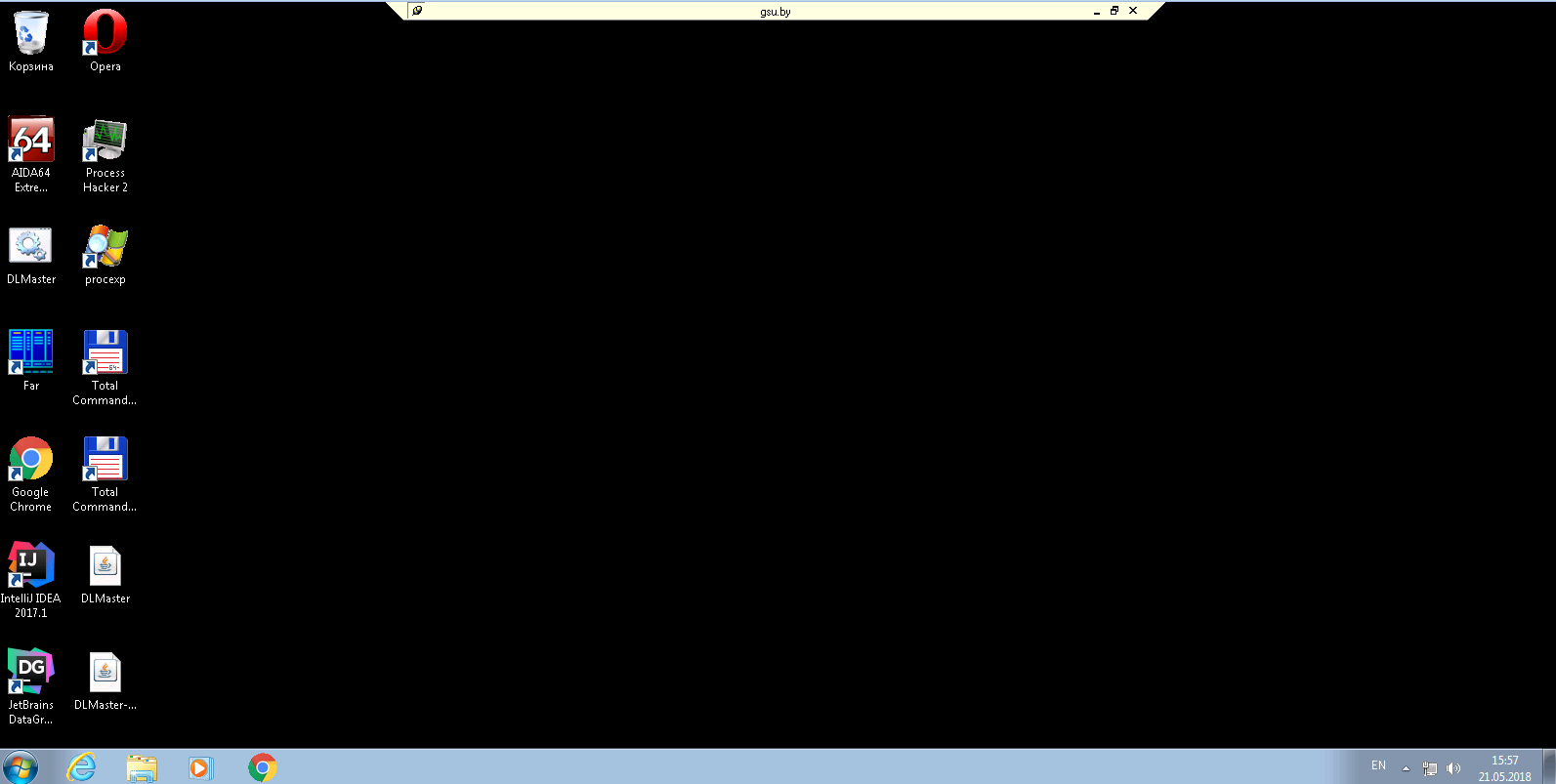
Адреса для удалённого терминального подключения к серверам DL по RDP можно посмотреть по адресу : <http://confluence.newit.gsu.by/display/dldev/RemoteRDP>

Нам нужен nit7. После того как заполните поле Computer нажмите кнопку Connect.



**Рисунок 45 – Выбор пользователя**

Введите свой аккаунт в сети ГГУ, не забудьте прописать домен MATH\.



**Рисунок 46 – Рабочий стол**

Запустите Total Comander и пропишите cd\\DL\_SERVER и зайдите в папку webapps$. В эту папку необходимо поместить war файл web-samples.

У вас может появится проблема с копированием war файла на сервер. Удобно это делать через Google диск. Далее war распакуется и приложение будет доступно по адресу dl.gsu.by/web-samples.

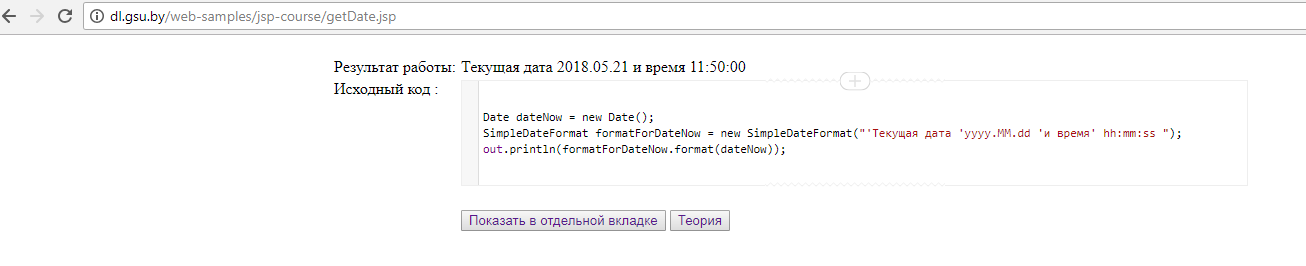
# 5 Примеры JSP страниц

В процессе работы были выполнены полезные JSP примеры. Которые призваны помочь новичкам. К тому же при реализации тестирования подобных задач примеры станут первыми представителями.

Ниже приведен список разработанных примеров.

**Пример получения текущей даты getDate.jsp.**

Внешний вид :



**Рисунок 46 – Пример getDate.jsp**

Пример демонстрирует получение текущей даты и времени в JSP. Для этого использовался объекты класса Date и SimpleDateFormat. Объект Date предоставляет текущую дату и время. Объект класса SimpleDateFormat служит для форматирования даты и вывода его в адекватном виде. В конструктор посылается пример форматирование в данном случае это строка ” 'Текущая дата 'yyyy.MM.dd 'и время' hh:mm:ss” . Этим мы задали форматирование даты. Необходимо так же применить к объекту класса Date

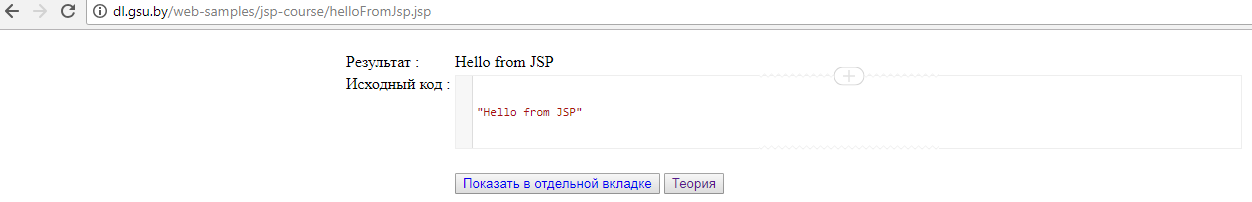
заданное нами форматирование, для объекта класса SimpleDateFormat вызываем метод format(объекта класса Date) и направляем результат в поток вывода.

Прежде чем работать с классами Date, SimpleDateFormat необходимо их импортировать. Делаем это следующим образом:

<%@ page import="java.util.Date" %>   
 <%@ page import="java.text.SimpleDateFormat" %>

**Пример на вывод приветственной строки, helloFrom.jsp.**

Внешний вид:



**Рисунок 47 – Пример helloFromJsp.jsp**

Пример демонстрирует обычный вывод строки. Для того чтобы вывести приветственную строку использовалось выражение.

Выражение в странице JSP - это исполняемое выражение, написанное на языке скрипта, указанного в объявлении language (как правило Java). Результат выражения JSP, имеющий обязательный тип String, направляется в стандартный поток вывода out с помощью текущего объекта JspWriter. Если результат выражения не может быть приведен к типу String, возникает либо ошибка трансляции, если проблема была выявлена на этапе трансляции, либо возбуждается исключение ClassCastException, если несоответствие было выявлено в процессе выполнения запроса. Выражение имеет следующий синтаксис:

<%= текст выражения %>

Альтернативный синтаксис для выражений JSP при использовании XML:

<jsp:expression> текст выражения </jsp:expression>

Порядок выполнения выражений в странице JSP слева-направо. Если выражение появляется более чем в одном атрибуте времени выполнения, то оно выполняется слева-направо в данном теге. Выражение должно быть полным выражением на определенном скрипте (как правило Java).

Выражения выполняются во время работы протокола HTTP. Значение выражения преобразуется в строку и включается в соответствующую позицию файла JSP.

Выражения обычно используются для того, чтобы вычислить и вывести на экран строковое представление переменных и методов, определенных в блоке объявлений страницы JSP или полученных от компонентов JavaBeans, которые доступны из JSP.

**Пример jspInclude.jsp.**

Внешний вид:



**Рисунок 48 – Пример jspInclude.jsp**

Пример демонстрирует знаменитый элемент Jsp Action, jsp:include.

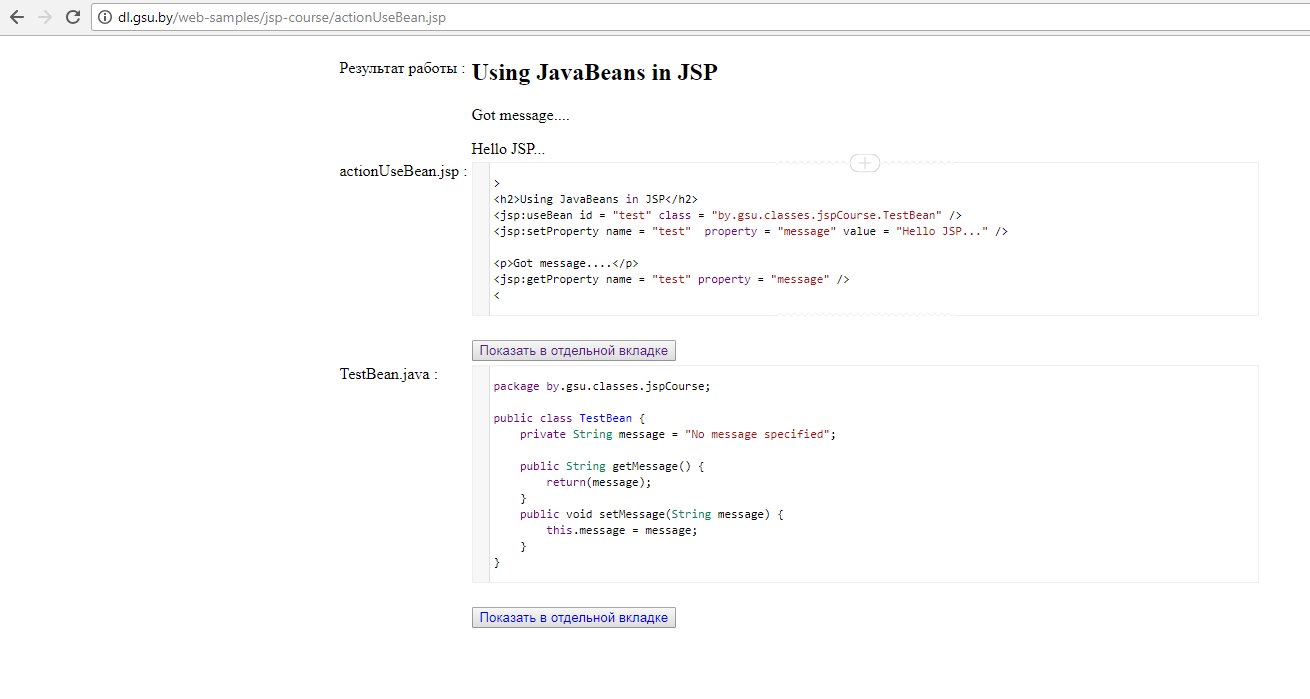
Действие jsp:include позволяет подключать статические и динамические ресурсы в контекст текущей страницы JSP. Так, например, выходной поток сервлета может быть включен в содержимое страницы JSP. Тогда при вызове страницы JSP выходной поток сервлета будет встроен в выходной поток JSP. Ресурс определяется по его относительному URL-адресу, который интерпретируется в контекст Web-сервера. В отличие от директивы include, которая вставляет файл на этапе трансляции страницы JSP, действие jsp:include вставляет файл при запросе страницы. Это приводит к некоторой потере эффективности и исключает возможность наличия во вставляемом файле кода JSP, но дает существенное преимущество в гибкости. Если рассмотреть в качестве примера JSP-страницу, которая вставляет четыре различных отрывка в Web страницу с новостями сайта. Каждый раз когда меняются заголовки автору достаточно изменить содержимое четырех файлов, тогда как главная JSP страница остается неизменной.

Тег jsp:include имеет атрибуты:  
 page : Атрибут page определяется относительно текущей страницы JSP. Включаемая страница имеет доступ только к объекту JspWriter и не может устанавливать заголовки.

flush : Необязательный атрибут flush управляет переполнением. Если этот атрибут имеет значение true и выходной поток страницы JSP буферизуется, то буфер освобождается при переполнении, в противном случае - не освобождается. По умолчанию значение атрибута flush равно false.

**Пример actionUseBean.jsp.**

Внешний вид:



**Рисунок 49 – Пример actionUseBean.jsp**

Действие useBean довольно разносторонне. Сначала он ищет существующий объект, используя переменные id и scope. Если объект не найден, он пытается создать указанный объект.

Самый простой способ загрузки компонента состоит в следующем:

<jsp: useBean id = "name" class = "package.class" />

После загрузки класса bean-компонента вы можете использовать jsp: setProperty и jsp: действия getProperty для модификации и извлечения свойств компонента.

Теперь давайте рассмотрим действия jsp: setProperty и jsp: getProperty, прежде чем приводить действительный пример, связанный с этими действиями.

Действие setProperty устанавливает свойства Bean. Перед этим действием Bean должен был быть определен ранее. Существует два основных способа использования действия setProperty -

Вы можете использовать jsp: setProperty после, но вне элемента jsp: useBean.

<jsp:useBean id = "myName" ... />

...

<jsp:setProperty name = "myName" property = "someProperty" .../>

В этом случае jsp: setProperty выполняется независимо от того, был ли создан новый компонент или существующий компонент.

Второй контекст, в котором может отображаться jsp: setProperty, находится внутри тела элемента jsp: useBean, как указано ниже -

<jsp: useBean id = "myName" ...>

   ...

   <jsp: setProperty name = "myName" property = "someProperty" ... />

</ JSP: useBean>

Здесь jsp: setProperty выполняется только в том случае, если был создан экземпляр нового объекта, а не если существующий найден.

Действие getProperty используется для извлечения значения данного свойства и преобразования его в строку и, наконец, вставляет его в вывод.

Действие getProperty имеет только два атрибута, оба из которых необходимы. Синтаксис действия getProperty выглядит следующим образом:

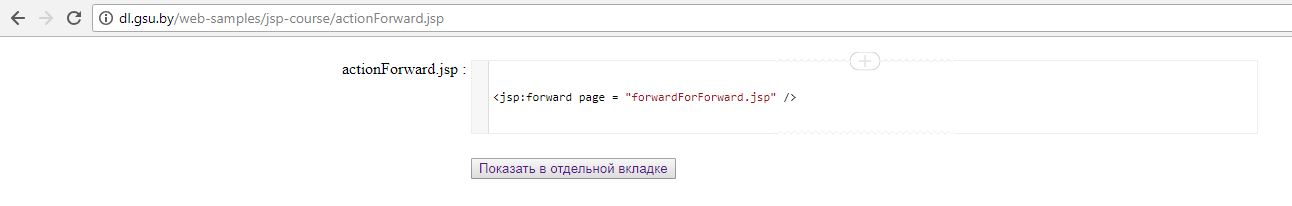
<jsp: useBean id = "myName" ... />

...

<jsp: getProperty name = "myName" property = "someProperty" ... />

**Пример actionForward.jsp.**

Внешний вид:

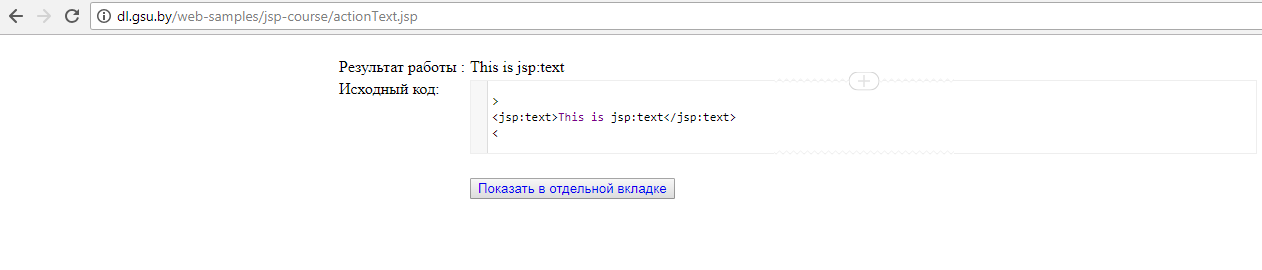


**Рисунок 50 – Пример actionForward.jsp**

Действие jsp:forward позволяет во время выполнения страницы JSP перенаправлять текущий запрос на другую страницу JSP, некоторый статический ресурс или класс Java-сервлета, находящийся в том же контексте, что и текущая страница JSP.

**Пример actionText.jsp.**

Внешний вид:

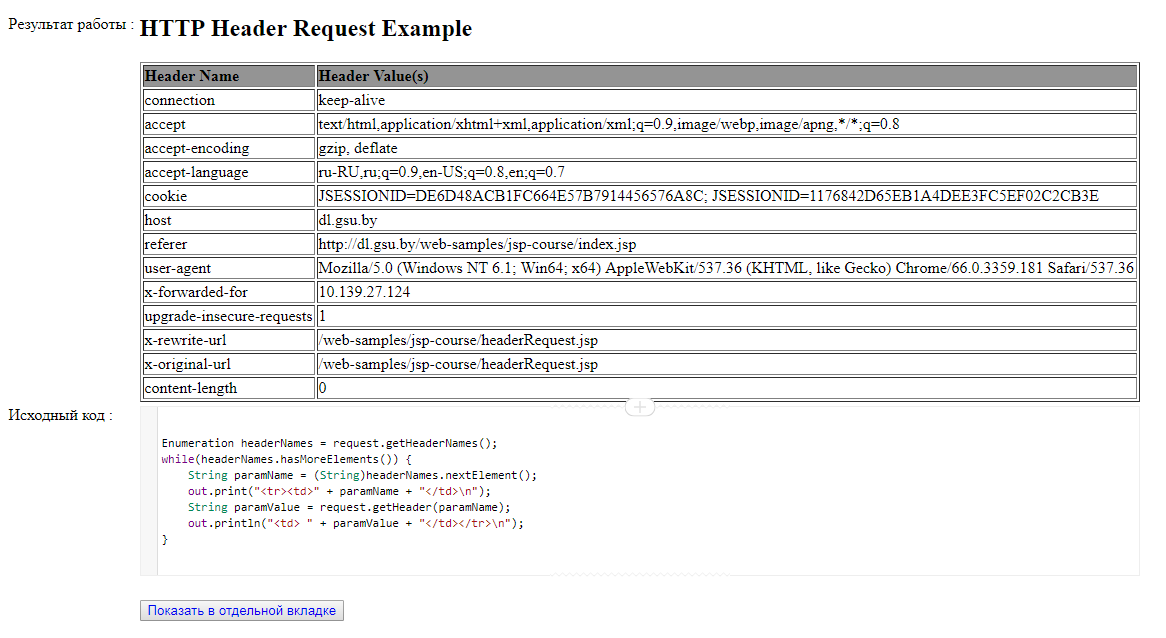


**Рисунок 51 – Пример actionText.jsp**

Действие jsp:text служит для вывода тектса, альтернатива примеру helloFromJsp.jsp

**Пример HTTP Header Request , headerRequest.jsp.**

Внешний вид:



**Рисунок 52 – Пример headerRequest.jsp**

Пример, который использует метод getHeaderNames() HttpServletRequest для чтения информации заголовка HTTP. Этот метод возвращает перечисление, содержащее информацию заголовка, связанную с текущим HTTP-запросом.

После того, как мы перечислили, мы выполняем цикл перечисления стандартным образом. Использовался метод hasMoreElements(), чтобы определить, когда завершить цикл, и метод nextElement (), чтобы получить имя каждого имени параметра.

**Пример HTTP Header Response , headerRequest.jsp.**

Внешний вид:



**Рисунок 53 – Пример headerResponse.jsp**

В данном примере реализовано обновление странице через заданное число секунд. Мы уже писали дату в предыдущих примерах, возьмем кусочек кода из getDate.jsp. И вызовем для неявного объекта response метод setIntHeader(’Refresh’, 10). Число секунд задано цифрой 10.

**Пример HTTP Status Codes, forStatusCode.jsp.**

Внешний вид:

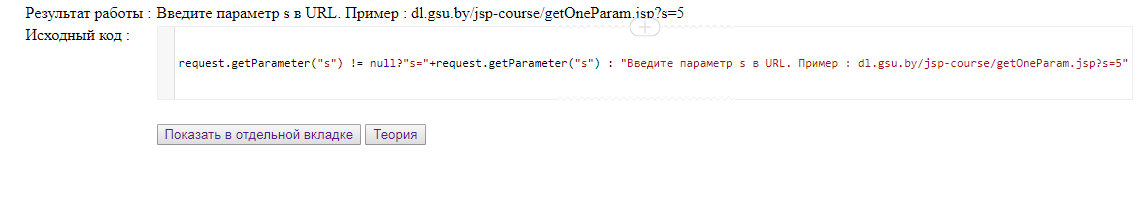


**Рисунок 54 – Пример forStatusCode.jsp**

С помощью неявного объекта response и метода sendError(номер ошибки, ”сообщение”). Можно переопределять сообщение в встроенных страницах ошибок Tomcat. Обычно в веб приложениях создаются свою собственные страницы ошибок. В данном примере мы вызвали страницу 404 обычно она выскакивает если запрашиваемая страница не найдена, мы же ее переопределили как Need authentication!!!.

**Пример получение конкретного параметра из Url, getParam.jsp.**

Внешний вид:

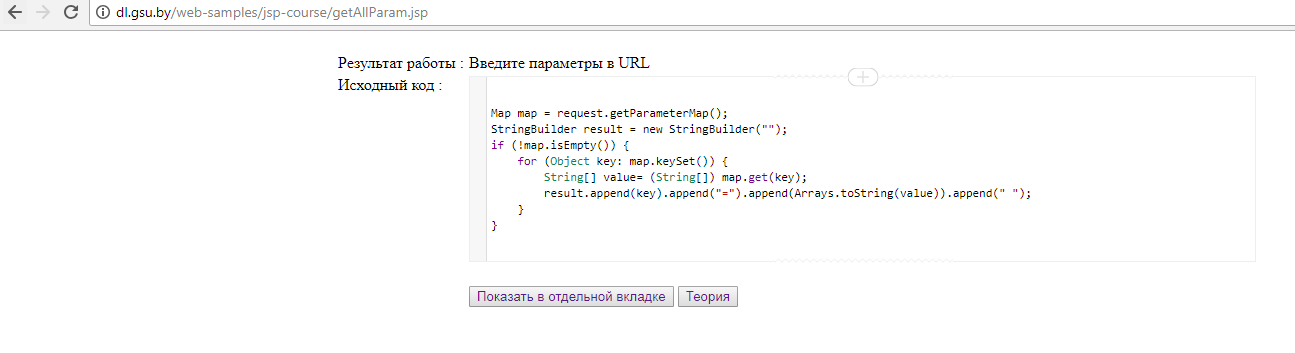


**Рисунок 55 – Пример getParam.jsp**

Для реализации данного примера использовался неявный объект request и метод getParametr(”s”). Так как по условию примера требуется обработать конкретный параметр , в данном случае параметром является s.

**Пример получение всех параметров из Url, getAllParam.jsp.**

Внешний вид:

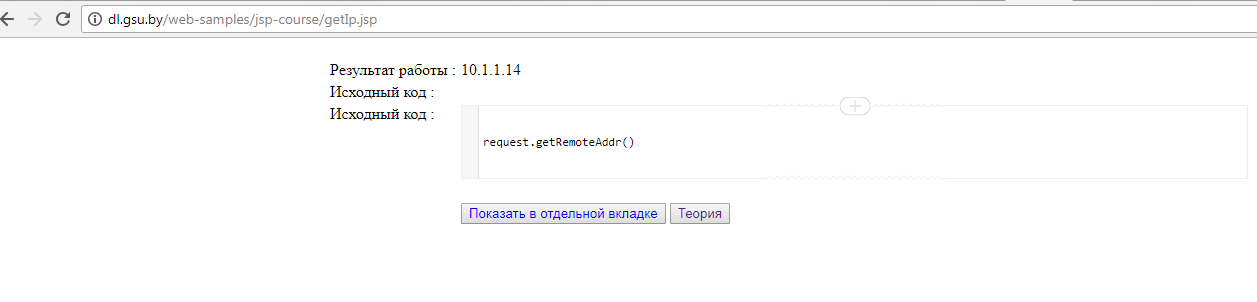


**Рисунок 56 – Пример getAllParam.jsp**

В данном примере использовался объект класса Map. В контейнерах Map (отображение) хранятся два объекта: ключ и связанное с ним значение. Иногда используют термин "ассоциативный массив". Map позволяет искать объекты по ключу. Объект, ассоциированный с ключом, называется значением. И ключи, и значения являются объектами. Ключи могут быть уникальными, а значения могут дублироваться. Некоторые отображения допускают пустые ключи и пустые значения.

Для начала нужно получить Map с помощью метода request.getParametrMap(). Для вывода ключей и параметров Map, я прошелся по всем ключам map и формировал строку , с ключом и значениями присущих данному ключу и далее направил строку в поток вывода.  
 **Пример получения IP пользователя, getIp.jsp.**

Внешний вид:

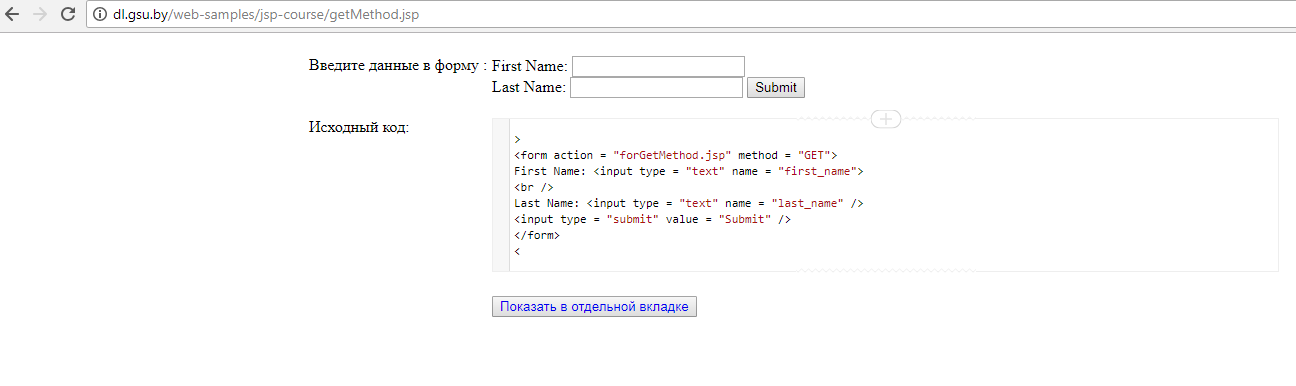


**Рисунок 57 – Пример getIp.jsp**

В данном примере я использовал неявный объект request и метод getRemoteAdr(), который предоставляет IP пользователя.

**Пример обработка формы метод GET, getMethod.jsp.**

Внешний вид:



**Рисунок 58 – Пример getMethod.jsp**

И файл обработки формы:

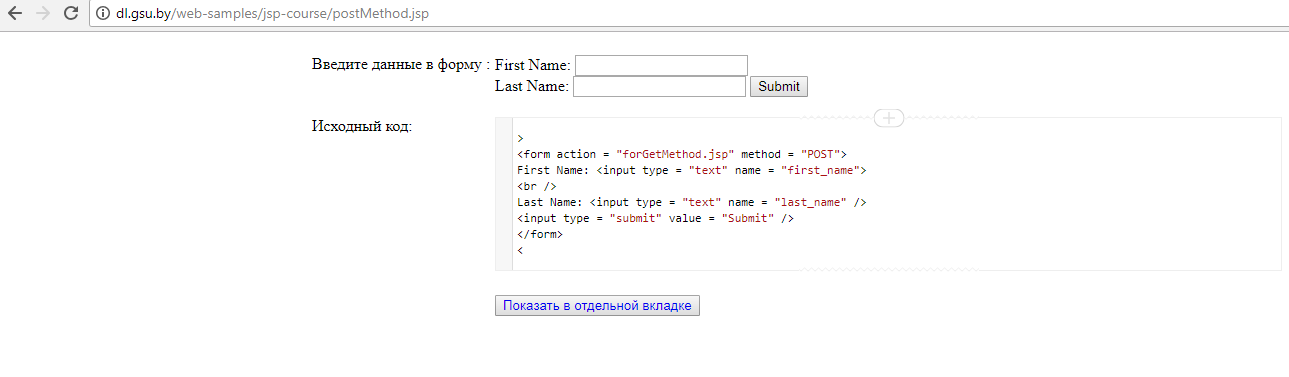


**Рисунок 59 – файл-обработчик, forGetMethod.jsp**

Метод GET отправляет скрипту всю собранную информацию формы как часть URL. После того как вы введете данные в форму, необходимо эти данные обработать, данные обрабатывались при помощи метода getParameter().

**Пример обработка формы метод POST, postMethod.jsp.**

Внешний вид:

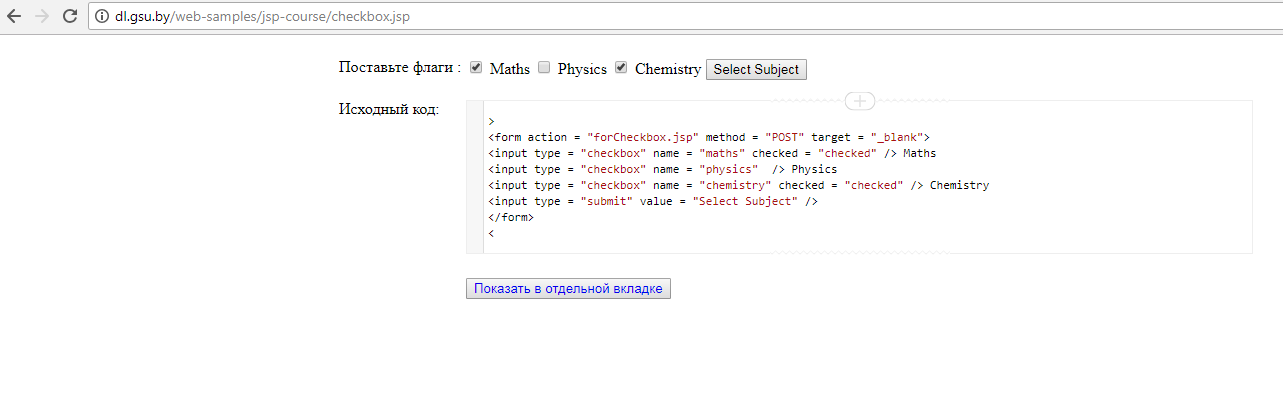


**Рисунок 60 –Пример postMethod.jsp**

Метод POST передает данные таким образом, что пользователь сайта уже не видит передаваемые скрипту данные. После того как вы введете данные в форму, необходимо эти данные обработать , данные обрабатывались при помощи метода getParameter(). Обычно POST метод используется для отправки данных на сервер, в отличии от метода GET.

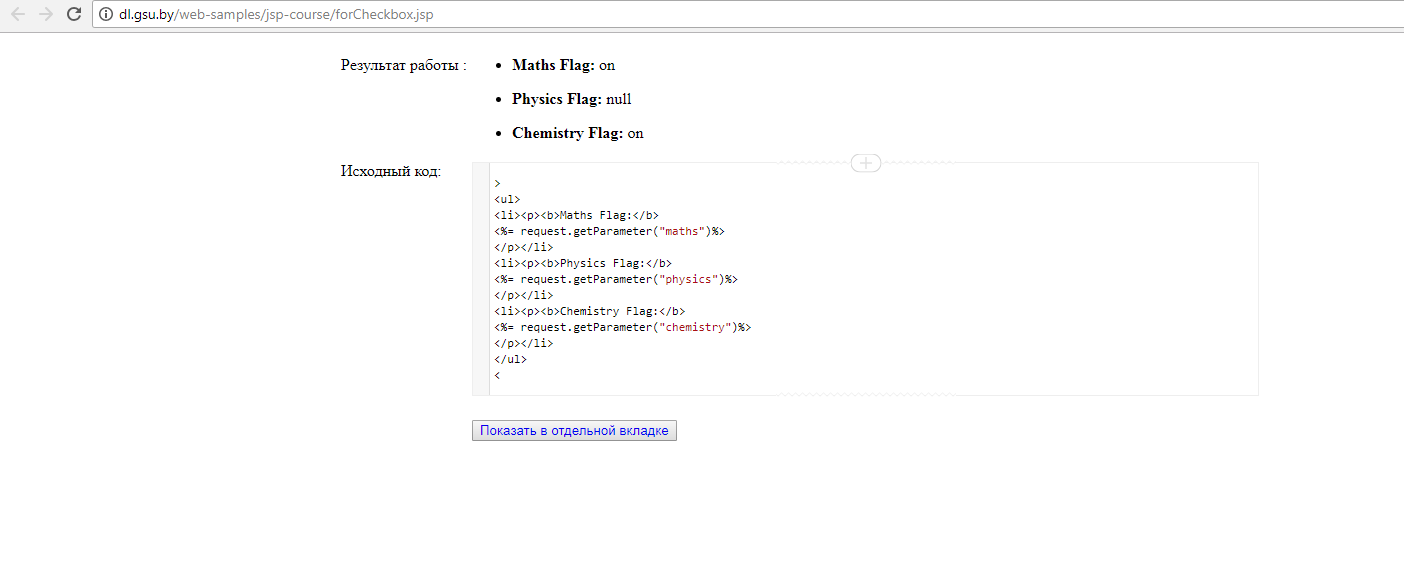
**Пример передачи флажков, checkbox.jsp.**

Внешний вид:



**Рисунок 61 –Пример checkbox.jsp**

Файл-обработчик:

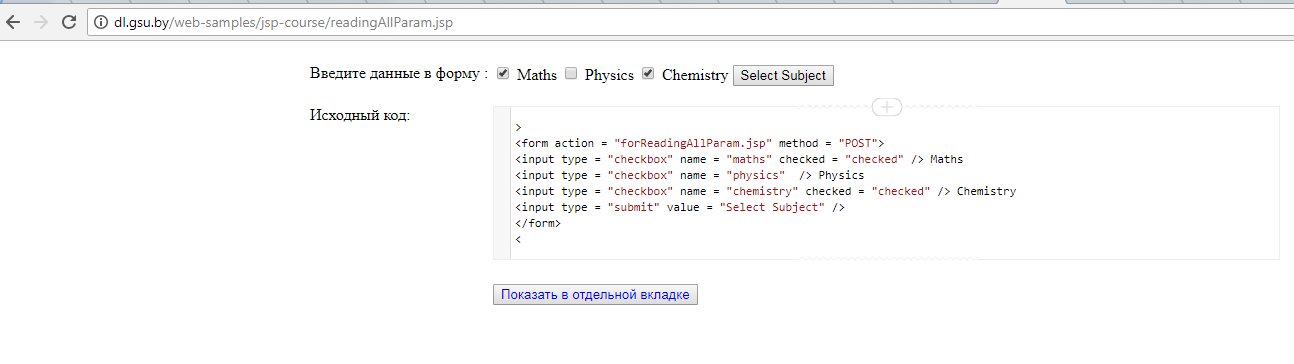


**Рисунок 62 –Файл-обработчик forCheckbox.jsp**

В данном примере с помощью формы, методом POST посылается информация об активных флажках. В последующем в файле-обработчике эти параметры обрабатываются и выводятся на страницу. Получаются параметры с помощью метода getParameter(), и направляются в поток вывода.

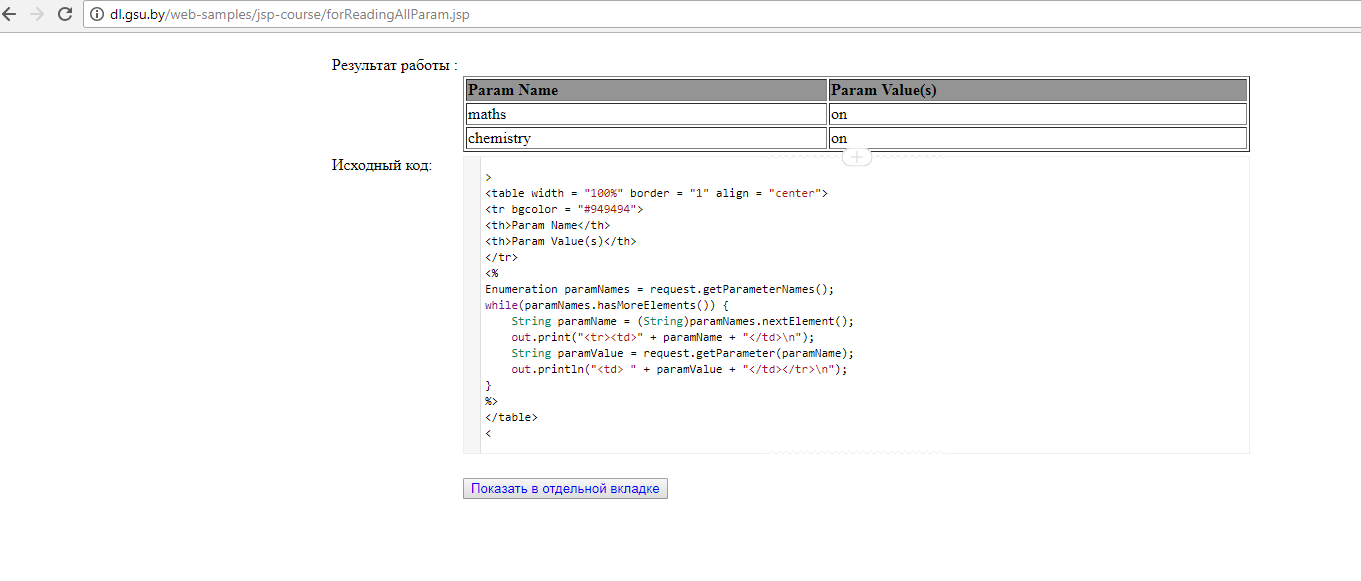
**Пример обработки всех параметров формы, readingAllParam.jsp.**

Внешний вид:



**Рисунок 62 –Пример readingAllParam.jsp**

Файл-обработчик:

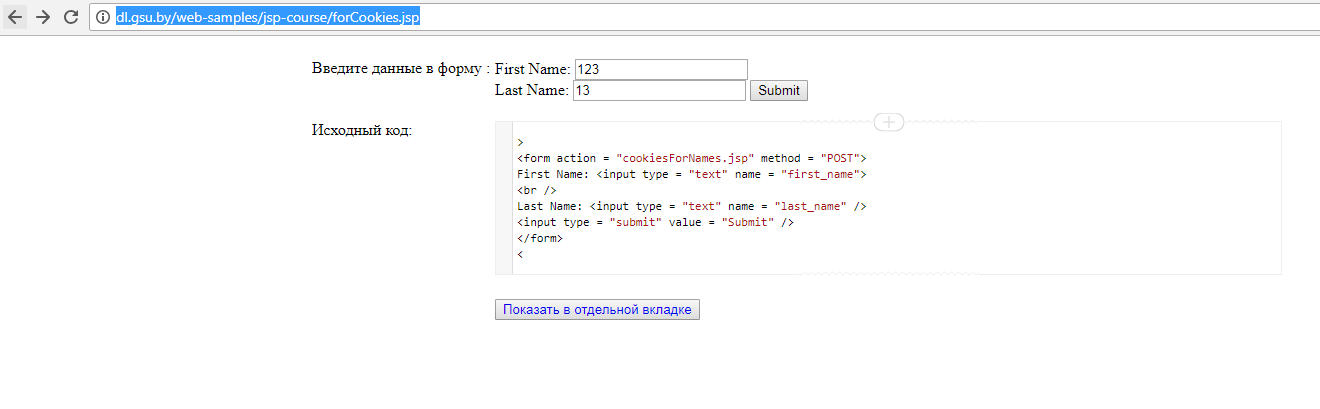


**Рисунок 63 – Файл-обработчик forReadingAllParam.jsp**

Данный пример является альтернативой примеру: получения всех параметров из URL. В данном примере с помощью формы, методом POST посылается информация об активных флажках. В последующем в файле-обработчике эти параметры обрабатываются и выводятся на страницу. Выполнил цикл по параметрам while(paramNames.hasMoreElements()) определяет есть ли следующий параметр и с помощью метода getParameter() , далее направлял параметр в поток вывода.

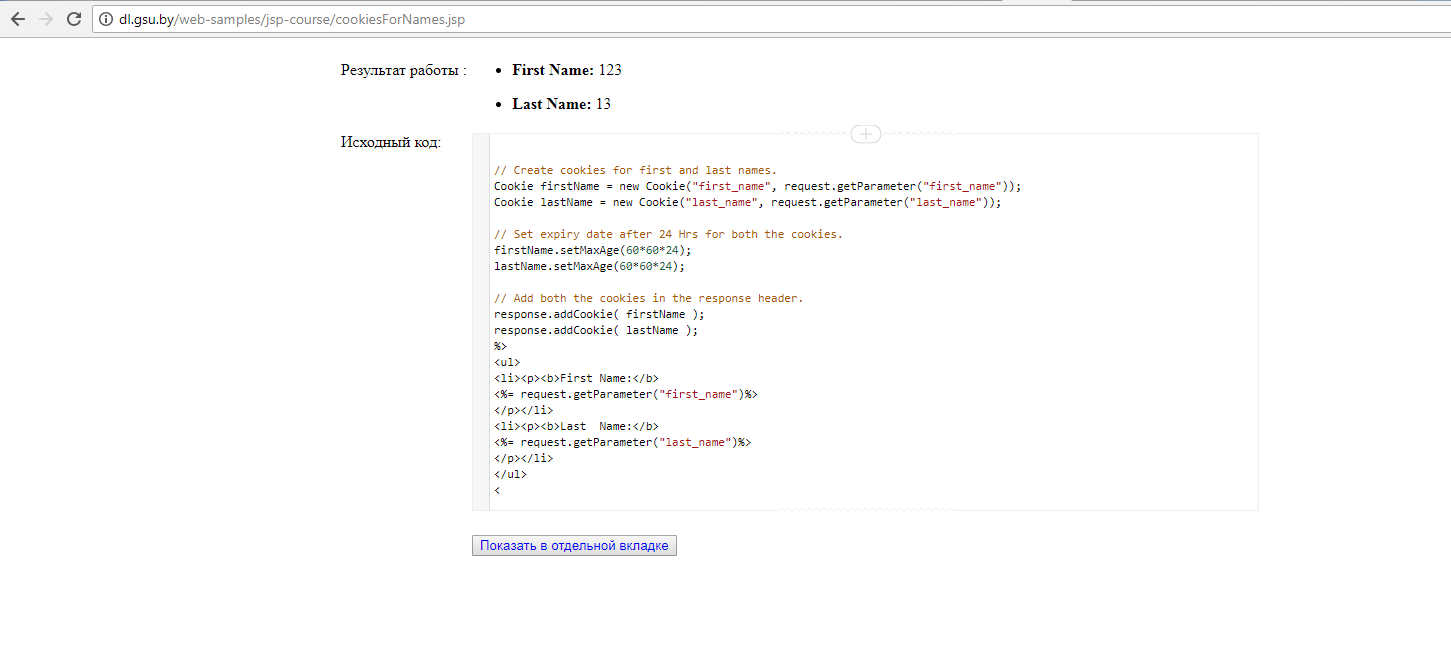
**Пример установки Cookie, forСookies.jsp.**

Внешний вид:



**Рисунок 64 –Пример forCookies.jsp**

Файл-обработчик:



**Рисунок 65 – Файл-обработчик cookiesForNames.jsp**

Куки-файлы - это текстовые файлы, хранящиеся на клиентском компьютере, и они хранятся для различных целей отслеживания информации. JSP прозрачно поддерживает HTTP-файлы cookie, используя базовую сервлет-технологию.

В идентификации и возврате пользователей задействованы три этапа -

Серверный скрипт отправляет набор файлов cookie в браузер. Например, имя, возраст или идентификационный номер и т. д.

Браузер сохраняет эту информацию на локальном компьютере для будущего использования.

Когда в следующий раз браузер отправит любой запрос на веб-сервер, он отправит эти файлы cookie на сервер, и сервер использует эту информацию для идентификации пользователя или может быть для какой-то другой цели.

В данном примере мы лишь просто установим Cookie, в следующем примере показано как обработать параметры и вывести их.

Прежде необходимо создать объект Cookie , делается это следующим образом:

Cookie firstName = new Cookie("first\_name", request.getParameter("first\_name"));

Cookie lastName = new Cookie("last\_name", request.getParameter("last\_name"));

Необходимо указать так же время хранения Cookies , для моего примера это 24 часа .

firstName.setMaxAge(60\*60\*24);

lastName.setMaxAge(60\*60\*24);

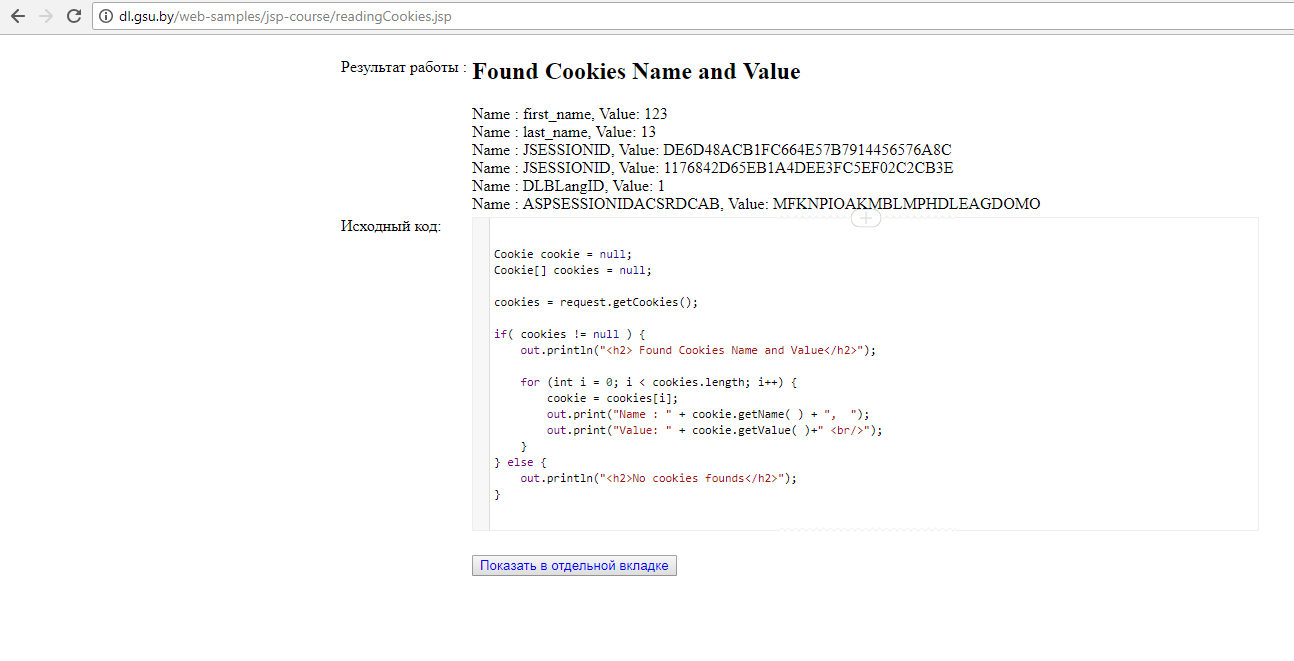
Далее добавляем Cookies в response header:

response.addCookie( firstName );

response.addCookie( lastName );

**Пример чтения Cookies, readingCookies.jsp.**

Внешний вид:



**Рисунок 66 –Пример readingCookies.jsp**

На экран выведется все имеющееся на данный момент Cookies, как видно из рис.57 там и есть параметры которые мы установили в прошлом примере.

Сначала я зарезервировал имена cookie и массив cookies. Далее из request.getCookies() получил все Cookies.

Cookie cookie = null;

Cookie[] cookies = null;

cookies = request.getCookies();

Далее мы прошлись по массиву и вывели все cookies.

if( cookies != null ) {

out.println("<h2> Found Cookies Name and Value</h2>");

for (int i = 0; i < cookies.length; i++) {

cookie = cookies[i];

out.print("Name : " + cookie.getName( ) + ", ");

out.print("Value: " + cookie.getValue( )+" <br/>");

}

} else {

out.println("<h2>No cookies founds</h2>");

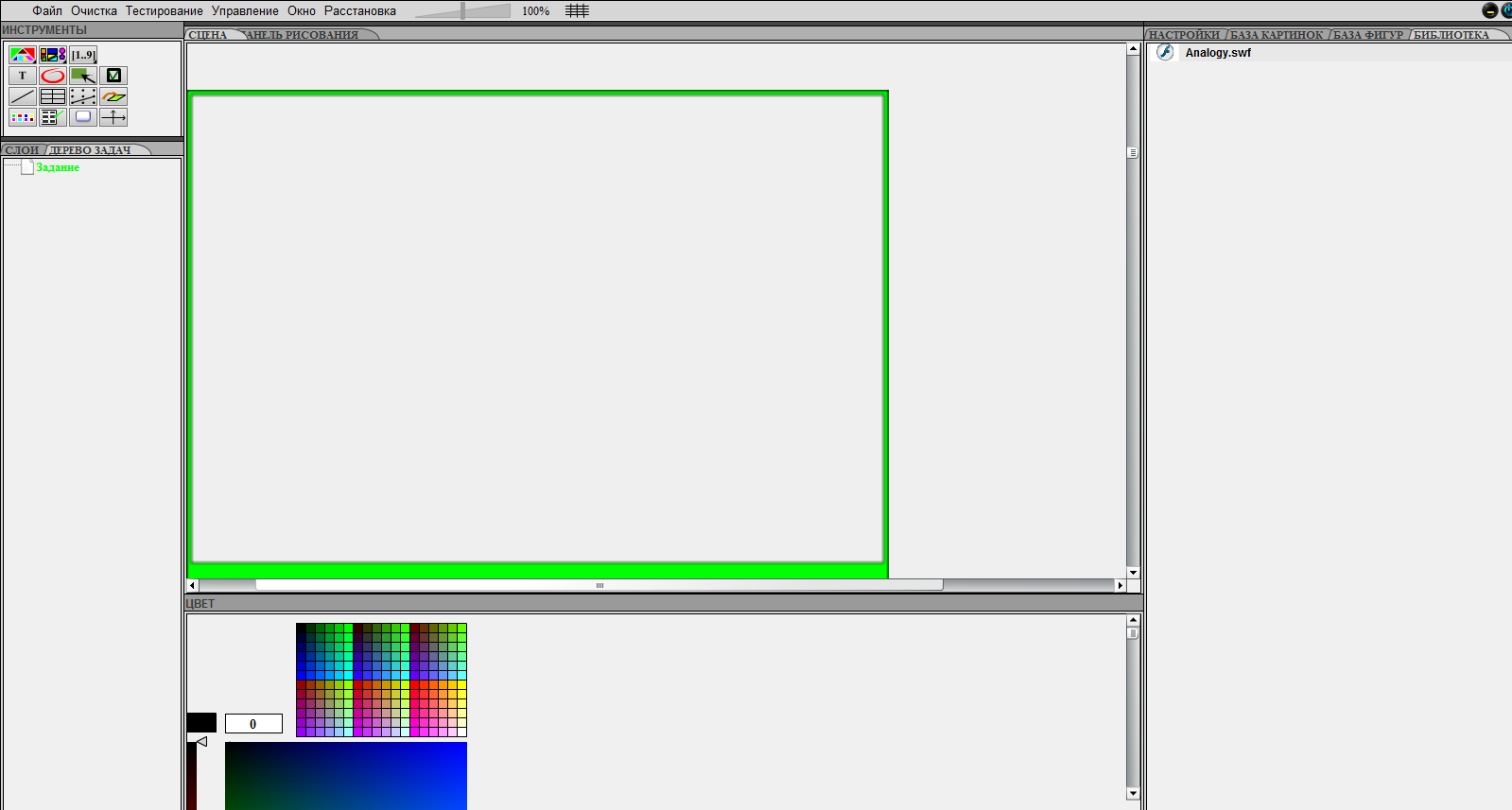
}

# 6 Развитие интерактивных возможностей сайта DL

Flash задания предоставляют огромные возможности для разработчиков, возможность создавать новые , интересные интерактивные задачи.

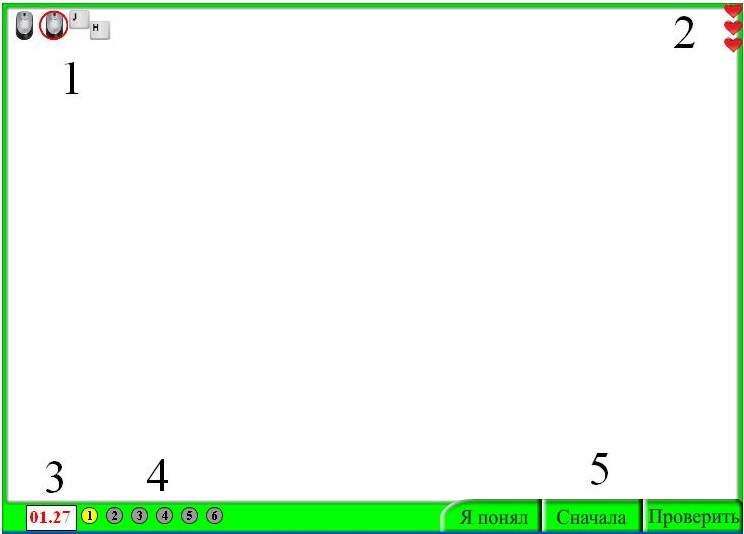
Комплекс для конструирования flash заданий для DL содержит набор программ:

Конструктор (Envelope.exe) - AIR приложение для конструирования заданий.



**Рисунок 67 – Конструктор**

Плеер (Pleer.swf) - программа для FlashPlayer для воспроизведения заданий.



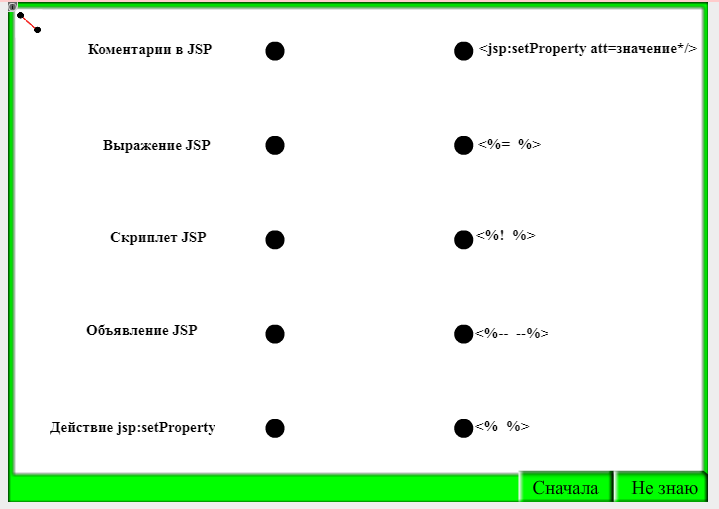
**Рисунок 68** – **Плеер**

DLManager - программа для скачивания flash заданий из базы DL.



**Рисунок 70** – **DLManager**

Пример Flash-задания точки соединения.



**Рисунок 69** – **Пример Flash задания точки соединения**

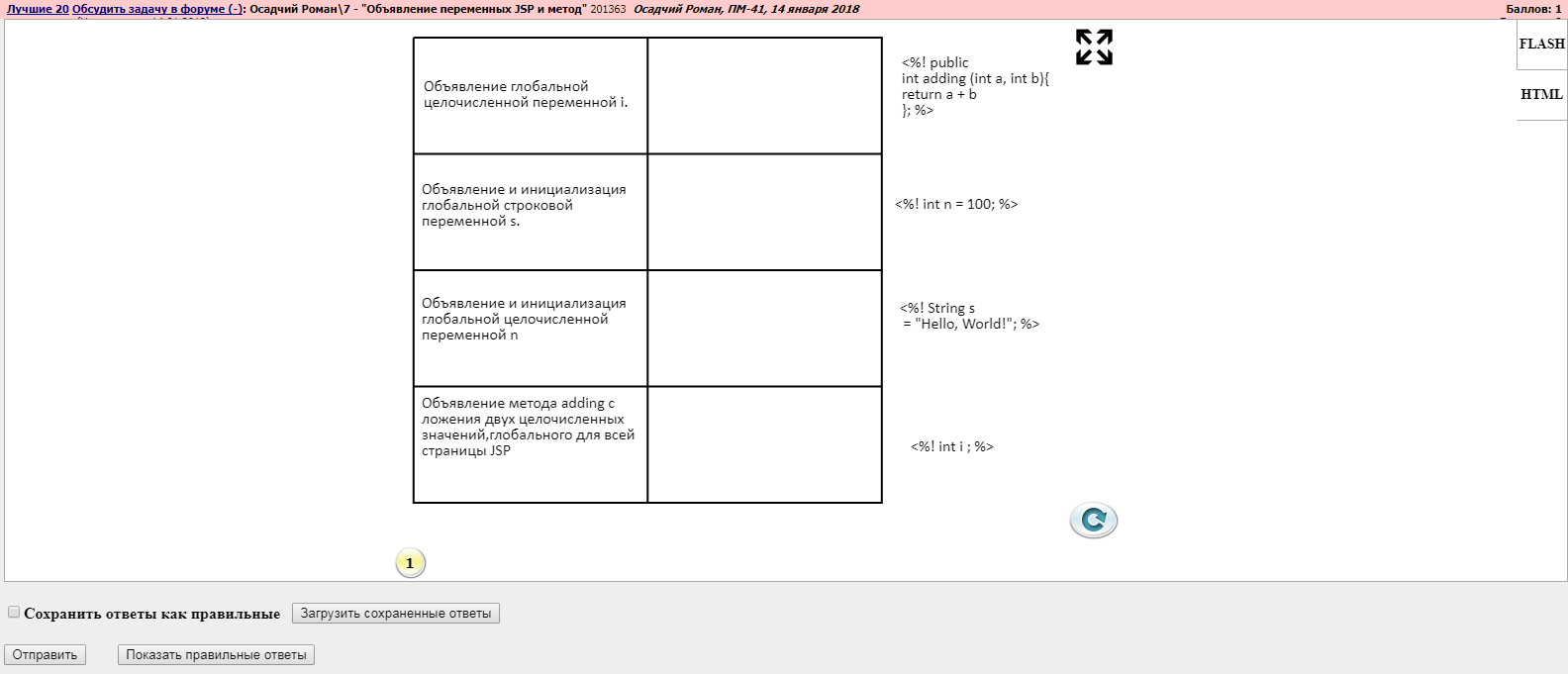
Задание на точки соединения проверяют знания пользователя, в данном примере пользователю предлагается соединить точки из левой колонки с точками из правой колонки. Тем самым проверяется знание пользователся относительно синтаксиса JSP страниц.

## 6.1 Модификация HTML плеера для flash заданий

В связи с тем что современные браузеры постепенно отказываются от технологии Flash, появляется необходимость замены этой технологии на DL.

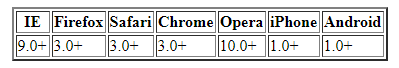
HTML5 плеер базируется на технологии Canvas и языка JavaScript.Canvas — элемент HTML5, предназначенный для создания растрового двухмерного изображения при помощи скриптов, обычно на языке JavaScript.

Внешний вид HTML5 плеера :



**Рисунок 71 – HTML5 плеер**

Так как плеер не поддерживает задачи с полями ввода было решено выполнить кнопки перехода от flash плеера к HTML5 плееру. Необходимо также выполнить проверку браузера пользователя на поддержку Canvas , если он не поддерживается, автоматически подключать flash плеер. На рисунке 20 продемонстрированы браузеры которые поддерживают технологию Canvas:



**Рисунок 72** – **Браузеры которые поддерживают технологию Canvas**

Чтобы разместить Canvas на страничке ,необходимо прописать :

<canvas id="drawingCanvas" width="500" height="300"></canvas>

id - идентификатор элемента. По этому имени JavaScript будет обращаться к холсту.

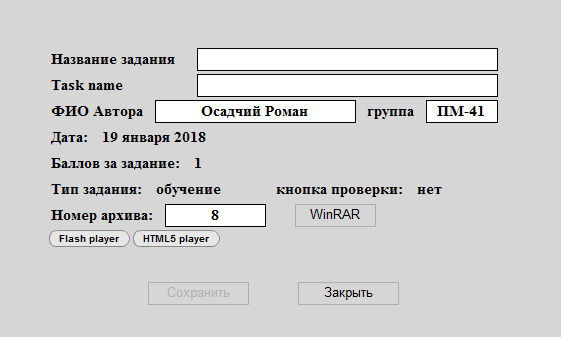
width - ширина холста.

height - высота холста.

Установка задач ничем не отличается от установки flash заданий, изменениям был подвергнут конструктор flash заданий, для удобства пользователя.

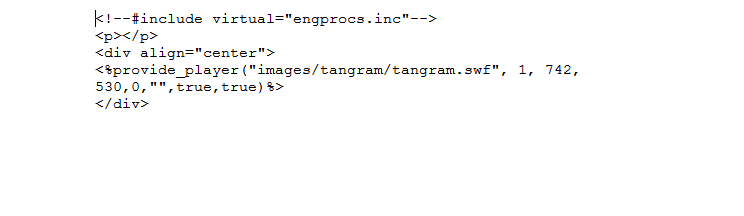
## 6.2 Обновление конструктора для flash заданий

Для облегчения создания flash заданий был придуман конструктор ,который помог бы разработчикам создавать их для дальнейшего обучения пользователей. С внедрением HTML5 плеера необходимо было вносить правки и в плеер. В плеер была добавлена кнопки для создания задания для flash плеера , для HTML5 плеера, для HTML5 и flash плеера.



**Рисунок 73** – **Кнопки для flash заданий и для HTML5 заданий**

Так же претерпел изменения и файл task.asp в задачах . Теперь в файлах task.asp вместо функции flash() будет создаваться функция provide\_player() с двумя новыми параметрами – флагами isUseFlash и isUseHTML. Эти два параметра необходимы ,так как в HTML5 плеере есть некоторый функционал который не реализован, например текстовые поля. В связи с этим было принято решение предоставить выбор плеера пользователю, посредством кнопок переключения, пока не будет реализована проверка в конструкторе ,которая не дала бы поставить флаг isUseHTML.

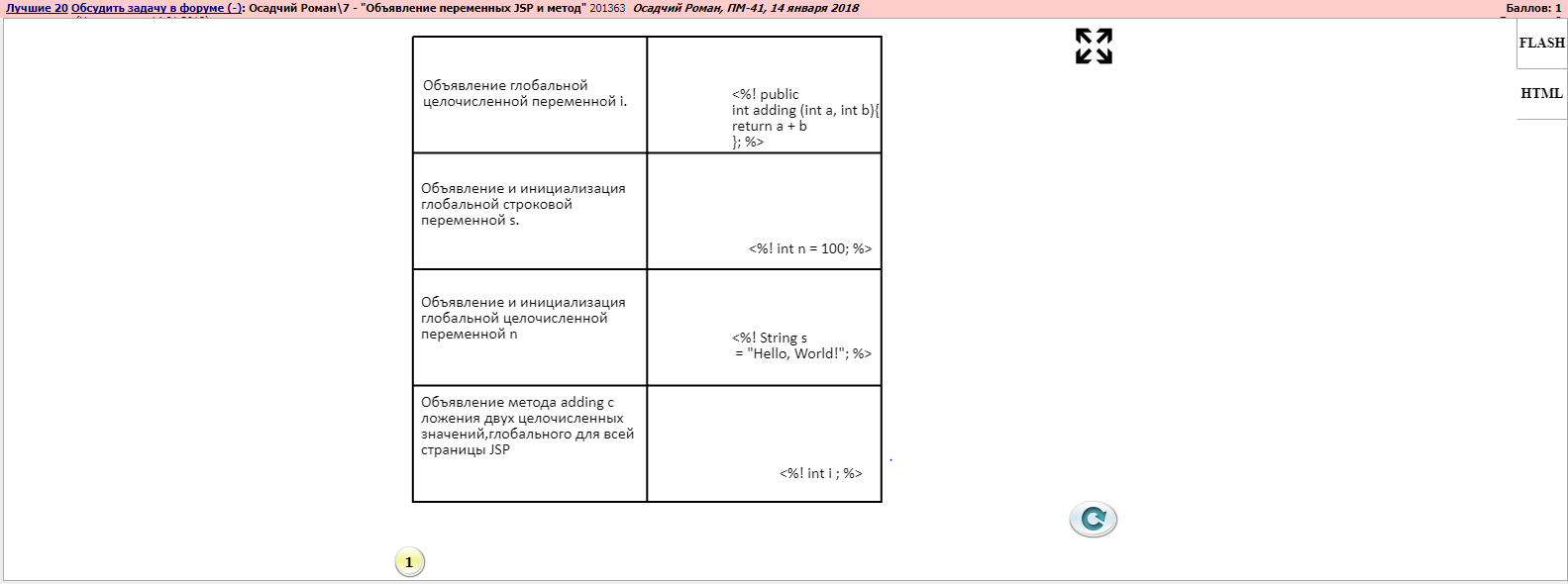
 **Рисунок 74** – **Файл task.asp с обновленной функцией.**

На стороне сервера были внесены изменения в файл engproc.inc и которые описаны в приложении.

## 6.3 Примеры интерактивных заданий на тему JSP

**Задача на область внесения.**

Задачи на область внесения обычно требуют от пользователя сопоставления рисунков или текста.



**Рисунок 75** – **Задача на область внесения.**

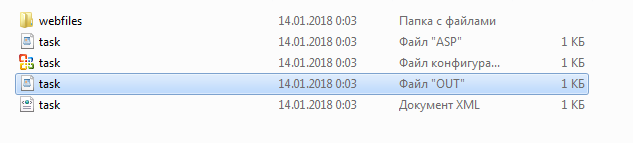
В данной задаче необходимо сопоставить опеределение и синтаксис на тему JSP.

Для создания подобных задач необходимо:

1)Нарисовать табличку в конструкторе.

2)Для нарисованной таблички поставить метку ”Область внесения”

3) Загрузить изображения и сделать их танами.



**Рисунок 76** – **Каталог файлов.**

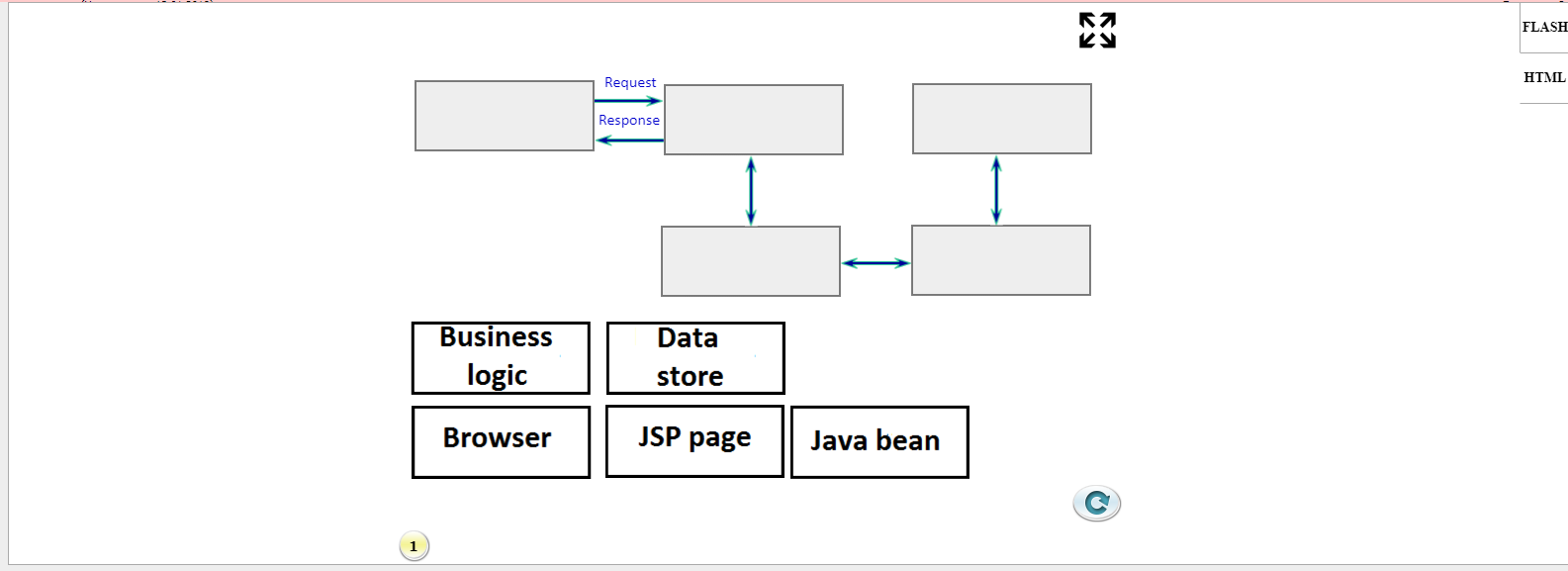
В каталоге webfiles лежат файлы необходимые для задачи , картинки и файл Position.txt. Position.txt главный файл отвечающий за задачу.



**Рисунок 77** – **Файл Position.txt.**

**Задачи на перенос рисунков.**

В задачах на перенос рисунков необходимо перенести рисунки в нужные места по смыслу или по заданному примеру.



**Рисунок 78** – **Пример задания.**

В данном задании необходимо правильно внести рисунки в схему, схема демонстрирует работу JSP страниц.

Каталог файлов тот же что и у задачи на область внесения.

Для установки области внесения необходимо :

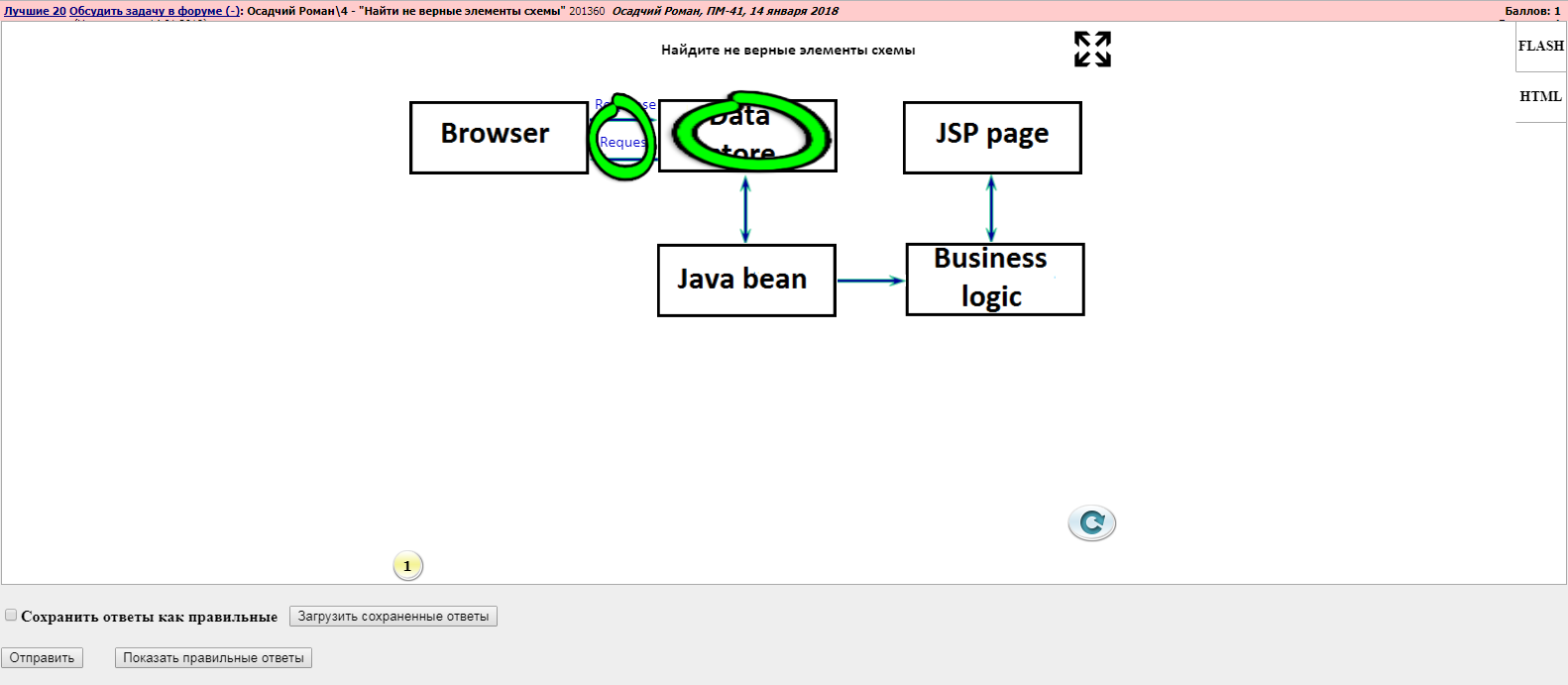
1. Необходимо загрузить картинку.
2. Сделать картинку таном.
3. Сделать область прозрачной.



**Рисунок 79** – **Файл Position.txt.**

**Задачи на область выделения.**

Задачи на область выделения эффективны там где необходимо создать задание где необходимо найти ошибки в чем-нибудь, например в схемах.



**Рисунок 80** – **Пример задачи на область выделения.**

В данной задаче необходимо найти неверные элементы схемы работы JSP страницы.

Каталог файлов тот же что и у задачи на область внесения.

Для того чтобы создать задачу на область выделения необходимо:

1) Разместить в нужном месте область выделения.

2) Проверить задачу.



**Рисунок 81** – **Файл Position.txt.**

# Заключение

В процессе выполнения дипломной работы были решены следующие проблемы:

1) Проблема с исполнением приложения с примерами по JSP независимо от основного сайта. При начальной разработке подобная проблема очень остро стояла, так как задействовать надо было сторонние ресурсы и людей

2) Поддержка проекта системой контроля версий. Для групповой разработки приложения.

3) Для облегчения обучения потенциальных разработчиков был выполнен начальный курс полезных примеров по технологии JSP.

4) Оформление теории.

5) Создание возможности дублирования исходного кода на страницу примера.

6) Оформление исходного кода, выделения нужных частей.

7) Внедрение HTML плеера.

8) Создание интерактивных заданий на тему JSP в HTML плеере.

Полезные примеры создавались с целями:

1) Показать интересные приемы с JSP.

2) Побудить новичков к изучению чего-то нового.

3) Заинтересовать к улучшению и развитию DL.

4) Предложить варианты развития после того как обучающиеся узнают основы HTML, JavaScript и CSS.

5) Создавать свои уникальные решения.

В процессе разработки был приобретен незаменимый опыт. Опыт работы с исходным кодом на проекте. В последующем каждое из решений, несомненно, будет развиваться. Для этого я максимально подробно постарался изложить все проделанные мной настройки, с которым зачастую у большинства возникают проблемы.

В процессе написания дипломной работы были описаны механизмы работы DLVM, способы создания заявок и отправка решений на сервер. Дабы познакомить начинающих разработчиков с механизмом серьезной разработки большой команды. Каждый год над DL трудятся множество разработчиков, хотелось бы, чтобы их было больше и решение задач обсуждалось более активно.

При внедрении HTML плеера так же возник ряд проблем связанный с непониманием кода, который писал другой человек, однако подобную проблему удалось решить благодаря советам старших разработчиков и непосредственно автором этого кода.

# Список используемых источников

1 Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015. - **689** c.

2 Берд, Барри Java для чайников / Барри Берд. - М.: Диалектика / Вильямс, 2013. – **521** c.

3 Гарнаев, Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. - Москва: **СПб. [и др.] : Питер, 2017. - 718**c.

4 Гонсалвес, Энтони Изучаем Java EE 7 / Энтони Гонсалвес. – М.: Питер, 2016. – 640 c.

5 Гупта, Арун Java EE 7. Основы / Арун Гупта. - М.: Вильямс, 2014. - 336 c.

6 Монахов, В. Язык программирования Java и среда NetBeans (+ CD-ROM) / В. Монахов. - М.: БХВ-Петербург, **2012**. - 720 c.

7 Савитч, Уолтер Язык Java. Курс программирования / Уолтер Савитч. – М.: Вильямс, **2015**. – 928 c.

8 Хабибуллин, Ильдар Самоучитель Java / Ильдар Хабибуллин. - М.: БХВ-Петербург, **2014**. - 768 c.

9 Шилдт, Герберт Java 8. Руководство для начинающих / Герберт Шилдт. - М.: Вильямс, 2015. - 720 c.

10 Эккель, Брюс Философия Java / Брюс Эккель. - М.: Питер, 2016. - **809** c.

# Приложение А

##### **Разработанный программный код**

/\*Содержание файла TestBean.java\*\

package by.gsu.classes.jspCourse;  
  
public class TestBean {  
 private String message = "No message specified";  
  
 public String getMessage() {  
 return(message);  
 }  
 public void setMessage(String message) {  
 this.message = message;  
 }  
}

/\*Содержание файла CORSFilter.java\*\

package by.gsu.listner;  
  
  
  
import javax.servlet.\*;  
 import javax.servlet.http.HttpServletRequest;  
 import javax.servlet.http.HttpServletResponse;  
 import java.io.IOException;  
  
public class CORSFilter implements Filter {  
 public void destroy() {  
 }  
 public static String *VALID\_METHODS* = "DELETE, HEAD, GET, OPTIONS, POST, PUT";  
  
 public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws ServletException, IOException {  
 HttpServletRequest httpReq = (HttpServletRequest) req;  
 HttpServletResponse httpResp = (HttpServletResponse) resp;  
  
 // No Origin header present means this is not a cross-domain request  
 String origin = httpReq.getHeader("Origin");  
 if (origin == null) {  
 // Return standard response if OPTIONS request w/o Origin header  
 if ("OPTIONS".equalsIgnoreCase(httpReq.getMethod())) {  
 httpResp.setHeader("Allow", *VALID\_METHODS*);  
 httpResp.setStatus(200);  
 return;  
 }  
 } else {  
 // This is a cross-domain request, add headers allowing access  
 httpResp.setHeader("Access-Control-Allow-Origin", origin);  
 httpResp.setHeader("Access-Control-Allow-Methods", *VALID\_METHODS*);  
  
 String headers = httpReq.getHeader("Access-Control-Request-Headers");  
 if (headers != null)  
 httpResp.setHeader("Access-Control-Allow-Headers", headers);  
  
 // Allow caching cross-domain permission  
 httpResp.setHeader("Access-Control-Max-Age", "3600");  
 }  
 // Pass request down the chain, except for OPTIONS  
 if (!"OPTIONS".equalsIgnoreCase(httpReq.getMethod())) {  
 chain.doFilter(req, resp);  
 }  
 }  
  
 public void init(FilterConfig config) throws ServletException {  
  
 }  
  
}

/\*Содержание файла ServletCode.java\*\

package by.gsu.servlets;  
  
import com.google.common.io.ByteStreams;  
  
import javax.servlet.ServletException;  
import javax.servlet.ServletOutputStream;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.PrintStream;  
  
  
public class ServletCode extends HttpServlet {  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {  
 String uri = request.getRequestURI();  
 response.setContentType("text/plain; charset = utf-8");  
 ServletOutputStream out = response.getOutputStream();  
 try (InputStream is = getClass()  
 .getClassLoader()  
 .getResourceAsStream(uri.substring(uri.lastIndexOf("sources/")+8, uri.length()))) {  
 ByteStreams.*copy*(is, out);  
 } catch (Exception e) {  
 e.printStackTrace(new PrintStream(out));  
 }  
 }  
}

/\*Содержание файла actionForward.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>The include Action Example</title>  
</head>  
  
<body>  
 <h2>The include action Example</h2>  
 //sampleStart  
 <**jsp:forward** page = "forwardForForward.jsp" />  
 //sampleEnd  
</body>  
</html>

/\*Содержание файла actionInclude.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы:</td>  
 <td>  
 **<%**//sampleStart**%>** <**jsp:include** page = "dateForInclude.jsp" flush = "true" />  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only folded-button="true">  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv">  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/actionInclude.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td>  
 <button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/actionInclude.jsp" target="\_blank">Показать в отдельной вкладке</a></button>  
 <button class="button-source"><a href="http://confluence.newit.gsu.by/pages/viewpage.action?pageId=35586074" target="\_blank">Теория</a></button>  
 </td>  
</tr>  
</table>  
</body>  
</html>

/\*Содержание файла actionText.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%**//sampleStart**%>** <**jsp:text**>This is jsp:text</**jsp:text**>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/actionText.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/actionText.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла actionUseBean.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
  
 **<%**//sampleStart**%>** <h2>Using JavaBeans in JSP</h2>  
 <**jsp:useBean** id = "test" class = "by.gsu.classes.jspCourse.TestBean" />  
 <**jsp:setProperty** name = "test" property = "message" value = "Hello JSP..." />  
  
 <p>Got message....</p>  
 <**jsp:getProperty** name = "test" property = "message" />  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>actionUseBean.jsp :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only folded-button="true">  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv">  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/actionUseBean.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/actionForward.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
 <tr>  
 <td>TestBean.java :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre >  
 <code class="hljs language-text" id="ajaxDiv2">  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/by/gsu/classes/jspCourse/TestBean.java",  
 success: **function**(data){  
 $("#ajaxDiv2").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/by/gsu/classes/jspCourse/TestBean.java" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла checkbox.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Поставьте флаги :</td>  
 <td>  
 **<%**//sampleStart**%>** <form action = "forCheckbox.jsp" method = "POST" target = "\_blank">  
 <input type = "checkbox" name = "maths" checked = "checked" /> Maths  
 <input type = "checkbox" name = "physics" /> Physics  
 <input type = "checkbox" name = "chemistry" checked = "checked" /> Chemistry  
 <input type = "submit" value = "Select Subject" />  
 </form>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/checkbox.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/checkbox.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла coockiesForName.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%**//sampleStart  
 // Create cookies for first and last names.  
 Cookie firstName = new Cookie("first\_name", request.getParameter("first\_name"));  
 Cookie lastName = new Cookie("last\_name", request.getParameter("last\_name"));  
  
 // Set expiry date after 24 Hrs for both the cookies.  
 firstName.setMaxAge(60\*60\*24);  
 lastName.setMaxAge(60\*60\*24);  
  
 // Add both the cookies in the response header.  
 response.addCookie( firstName );  
 response.addCookie( lastName );  
 **%>** <ul>  
 <li><p><b>First Name:</b>  
 **<%=** request.getParameter("first\_name")**%>** </p></li>  
 <li><p><b>Last Name:</b>  
 **<%=** request.getParameter("last\_name")**%>** </p></li>  
 </ul>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/cookiesForNames.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/cookiesForNames.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла dateForInclude.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
</head>  
<body>  
 <p>Today's date: **<%=** (new java.util.Date())**%>**</p>  
</body>  
</html>

/\*Содержание файла forCheckbox.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%**//sampleStart**%>** <ul>  
 <li><p><b>Maths Flag:</b>  
 **<%=** request.getParameter("maths")**%>** </p></li>  
 <li><p><b>Physics Flag:</b>  
 **<%=** request.getParameter("physics")**%>** </p></li>  
 <li><p><b>Chemistry Flag:</b>  
 **<%=** request.getParameter("chemistry")**%>** </p></li>  
 </ul>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/forCheckbox.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/forCheckbox.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла forCoockies.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Введите данные в форму :</td>  
 <td>  
 **<%**//sampleStart**%>** <form action = "cookiesForNames.jsp" method = "POST">  
 First Name: <input type = "text" name = "first\_name">  
 <br />  
 Last Name: <input type = "text" name = "last\_name" />  
 <input type = "submit" value = "Submit" />  
 </form>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/forCookies.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/forCookies.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла forGetMethod.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%**//sampleStart**%>** <ul>  
 <li><p><b>First Name:</b>  
 **<%=** request.getParameter("first\_name")**%>** </p></li>  
 <li><p><b>Last Name:</b>  
 **<%=** request.getParameter("last\_name")**%>** </p></li>  
 </ul>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/forGetMethod.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/forGetMethod.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла forPostMethod.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%**//sampleStart**%>** <ul>  
 <li><p><b>First Name:</b>  
 **<%=** request.getParameter("first\_name")**%>** </p></li>  
 <li><p><b>Last Name:</b>  
 **<%=** request.getParameter("last\_name")**%>** </p></li>  
 </ul>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/forPostMethod.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/forPostMethod.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла forReadingAllParam.jsp\*\

<%@ **page** import="java.util.Enumeration" %>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%**//sampleStart**%>** <table width = "100%" border = "1" align = "center">  
 <tr bgcolor = "#949494">  
 <th>Param Name</th>  
 <th>Param Value(s)</th>  
 </tr>  
 **<%** Enumeration paramNames = request.getParameterNames();  
 while(paramNames.hasMoreElements()) {  
 String paramName = (String)paramNames.nextElement();  
 out.print("<tr><td>" + paramName + "</td>\n");  
 String paramValue = request.getParameter(paramName);  
 out.println("<td> " + paramValue + "</td></tr>\n");  
 }  
 **%>** </table>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/forReadingAllParam.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/forReadingAllParam.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла forStatusCode.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 <iframe src="statusCode.jsp"></iframe>  
 </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/statusCode.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/statusCode.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла forwardForforward.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="1">  
 <tr>  
 <td>actionForward.jsp :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/actionForward.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/actionForward.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла getAllParam.jsp\*\

<%@ **page** import="java.util.Map" %>  
<%@ **page** import="java.util.Arrays" %>  
  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
  
**<%** //sampleStart  
 Map map = request.getParameterMap();  
 StringBuilder result = new StringBuilder("");  
 if (!map.isEmpty()) {  
 for (Object key: map.keySet()) {  
 String[] value= (String[]) map.get(key);  
 result.append(key).append("=").append(Arrays.*toString*(value)).append(" ");  
 }  
 }  
 //sampleEnd  
**%>**<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%=**(result.toString().isEmpty()) ? "Введите параметры в URL" : result**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only folded-button="true">  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv">  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/getAllParam.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td>  
 <button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/getAllParam.jsp" target="\_blank">Показать в отдельной вкладке</a></button>  
 <button class="button-source"><a href="http://confluence.newit.gsu.by/pages/viewpage.action?pageId=35586068" target="\_blank">Теория</a></button>  
 </td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла getDate.jsp\*\

﻿<%@ **page** import="java.util.Date" %>  
<%@ **page** import="java.text.SimpleDateFormat" %>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы:</td>  
 <td>  
 **<%** //sampleStart  
 Date dateNow = new Date();  
 SimpleDateFormat formatForDateNow = new SimpleDateFormat("'Текущая дата 'yyyy.MM.dd 'и время' hh:mm:ss ");  
 out.println(formatForDateNow.format(dateNow));  
 //sampleEnd  
 **%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only folded-button="true">  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv">  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/getDate.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td>  
 <button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/getDate.jsp" target="\_blank">Показать в отдельной вкладке</a></button>  
 <button class="button-source"><a href="http://confluence.newit.gsu.by/pages/viewpage.action?pageId=35586070" target="\_blank">Теория</a></button>  
 </td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла getIp.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%=** //sampleStart  
 request.getRemoteAddr()  
 //sampleEnd  
 **%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код :</td>  
 <td></td>  
 </tr>  
 <tr>  
 <td>Исходный код :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only folded-button="true">  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv">  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/getIp.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/getIp.jsp" target="\_blank">Показать в отдельной вкладке</a></button>  
 <button class="button-source"><a href="http://confluence.newit.gsu.by/pages/viewpage.action?pageId=35586072" target="\_blank">Теория</a></button>  
 </td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла getMethod.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Введите данные в форму :</td>  
 <td>  
 **<%**//sampleStart**%>** <form action = "forGetMethod.jsp" method = "GET">  
 First Name: <input type = "text" name = "first\_name">  
 <br />  
 Last Name: <input type = "text" name = "last\_name" />  
 <input type = "submit" value = "Submit" />  
 </form>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/getMethod.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/getMethod.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла getOneParam.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%=** //sampleStart  
 request.getParameter("s") != null?"s="+request.getParameter("s") : "Введите параметр s в URL. Пример : dl.gsu.by/jsp-course/getOneParam.jsp?s=5"  
 //sampleEnd  
 **%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only folded-button="true">  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv">  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/getOneParam.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td>  
 <button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/getOneParam.jsp" target="\_blank">Показать в отдельной вкладке</a></button>  
 <button class="button-source"><a href="http://confluence.newit.gsu.by/pages/viewpage.action?pageId=35586061" target="\_blank">Теория</a></button>  
 </td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла headerRequest.jsp\*\

<%@ **page** import="java.util.Enumeration" %>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
  
<body>  
 <table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 <h2>HTTP Header Request Example</h2>  
 <table width = "100%" border = "1" align = "center">  
 <tr bgcolor = "#949494">  
 <th>Header Name</th>  
 <th>Header Value(s)</th>  
 </tr>  
 **<%** //sampleStart  
 Enumeration headerNames = request.getHeaderNames();  
 while(headerNames.hasMoreElements()) {  
 String paramName = (String)headerNames.nextElement();  
 out.print("<tr><td>" + paramName + "</td>\n");  
 String paramValue = request.getHeader(paramName);  
 out.println("<td> " + paramValue + "</td></tr>\n");  
 }  
 //sampleEnd  
 **%>** </table>  
 </td>  
 </tr>  
 <tr>  
 <td>Исходный код :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/headerRequest.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/headerRequest.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
 </table>  
  
</body>  
</html>

/\*Содержание файла headerResponse.jsp\*\

<%@ **page** import="java.util.Calendar" %>  
<%@ **page** import="java.util.GregorianCalendar" %>  
﻿<%@ **page** import="java.util.Date" %>  
<%@ **page** import="java.text.SimpleDateFormat" %>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Результат работы :</td>  
 <td>  
 **<%** //sampleStart  
 response.setIntHeader("Refresh", 10);  
  
 Date dateNow = new Date();  
 SimpleDateFormat formatForDateNow=new SimpleDateFormat("'Текущая дата 'yyyy.MM.dd 'и время' hh:mm:ss ");  
 out.println(formatForDateNow.format(dateNow));  
 //sampleEnd  
 **%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/headerResponse.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/headerResponse.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла helloFromJsp.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>  
 Результат :   
 </td>  
 <td>  
 **<%=** //sampleStart  
 "Hello from JSP"  
 //sampleEnd  
 **%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код :</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only folded-button="true">  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv">  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/helloFromJsp.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td>  
 <button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/helloFromJsp.jsp" target="\_blank">Показать в отдельной вкладке</a></button>  
 <button class="button-source"><a href="http://confluence.newit.gsu.by/pages/viewpage.action?pageId=35586051" target="\_blank">Теория</a></button>  
 </td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла index.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
</head>  
<body>  
<div align="center">  
 <table class="index-table">  
 <tr>  
 <td>  
 <h3>Полезные примеры</h3>  
 <a href="helloFromJsp.jsp">Вывод строки "Hello from JSP"</a>  
 <br>  
 <a href="getDate.jsp">Получение текущей даты</a>  
 <br>  
 <h3>JSP action</h3>  
 <a href="actionInclude.jsp">Jsp включение(jsp:include)</a>  
 <br>  
 <a href="actionUseBean.jsp">Jsp useBean(jsp:useBean, jsp:setProperty, jsp:getProperty)</a>  
 <br>  
 <a href="actionForward.jsp">Jsp forward(jsp:forward)</a>  
 <br>  
 <a href="actionText.jsp">Jsp text(jsp:text)</a>  
 <h3>HTTP</h3>  
 <a href="headerRequest.jsp">HTTP Header Request</a>  
 <br>  
 <a href="headerResponse.jsp">HTTP Header Response</a>  
 <br>  
 <a href="forStatusCode.jsp">HTTP Status Codes</a>  
 <br>  
 <a href="getOneParam.jsp">Получение конкретного параметра из URL</a>  
 <br>  
 <a href="getAllParam.jsp">Получение всех параметров из URL</a>  
 <br>  
 <a href="getIp.jsp">Получение IP пользователя</a>  
 <h3>HTML</h3>  
 <a href="getMethod.jsp">Обработка формы метод Get</a>  
 <br>  
 <a href="postMethod.jsp">Обработка формы метод Post</a>  
 <br>  
 <a href="checkbox.jsp">Передача флажков</a>  
 <br>  
 <a href="readingAllParam.jsp">Чтение всех параметров формы </a>  
 <h3>Cookies</h3>  
 <a href="forCookies.jsp">Установка cookies</a>  
 <br>  
 <a href="readingCookies.jsp">Чтение cookies</a>  
 </td>  
 <td>  
 <h3>Полезные ссылки</h3>  
 <a href="https://studfiles.net/preview/1029766/">Введение в JSP для новичков</a>  
 <br>  
 <a href="http://orlova.rsue.ru/content/sst/j2ee2.htm">Примеры разработки JSP страниц</a>  
 <br>  
 <a href="http://www.java2ee.ru/jsp/syntax.html">Синтаксис JSP</a>  
 <br>  
 <a href="http://javastudy.ru/interview/jee-jsp-questions-answers/">Вопросы и ответы на тему JSP</a>  
 <br>  
 <a href="http://it-channel.ru/2016/02/29/install-tomcat/">Установка Tomcat и интеграция с Intellij Idea</a>  
 <br>  
 <a href="https://www.tutorialspoint.com/jsp/index.htm">JSP Tutorial</a>  
 <br>  
 <a href="https://github.com/OsadchyRoman/web">Проект с примерами на github.com</a>  
 <h3>Среда разработки</h3>  
 <a href="http://confluence.newit.gsu.by//x/HwAaAg">Среда разработки для пользователей</a>  
 </td>  
 </tr>  
 </table>  
</div>  
</body>  
</html>

/\*Содержание файла postMethod.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Введите данные в форму :</td>  
 <td>  
 **<%**//sampleStart**%>** <form action = "forGetMethod.jsp" method = "POST">  
 First Name: <input type = "text" name = "first\_name">  
 <br />  
 Last Name: <input type = "text" name = "last\_name" />  
 <input type = "submit" value = "Submit" />  
 </form>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/postMethod.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/postMethod.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>

/\*Содержание файла readingAllParam.jsp\*\

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <link rel="stylesheet" type="text/css" href="../resources/css/style.css">  
 <script type="text/javascript" src="../resources/javascript/task.js"></script>  
 <script src="https://unpkg.com/kotlin-playground@1" data-selector=".kotlin-code"></script>  
 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>  
</head>  
<body>  
<table cols="2">  
 <tr>  
 <td>Введите данные в форму :</td>  
 <td>  
 **<%**//sampleStart**%>** <form action = "forReadingAllParam.jsp" method = "POST">  
 <input type = "checkbox" name = "maths" checked = "checked" /> Maths  
 <input type = "checkbox" name = "physics" /> Physics  
 <input type = "checkbox" name = "chemistry" checked = "checked" /> Chemistry  
 <input type = "submit" value = "Select Subject" />  
 </form>  
 **<%**//sampleEnd**%>** </td>  
 </tr>  
 <tr>  
 <td>Исходный код:</td>  
 <td>  
 <div class="kotlin-code" data-highlight-only>  
 <pre>  
 <code class="hljs language-text" id="ajaxDiv" >  
 <script type="text/javascript">  
 $.ajax({  
 url: "**<%=**request.getContextPath()**%>**/sources/jsp-course/readingAllParam.jsp",  
 success: **function**(data){  
 $("#ajaxDiv").text(data);  
 }  
 });  
 </script>  
 </code>  
 </pre>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><button class="button-source"><a href="**<%=**request.getContextPath()**%>**/sources/jsp-course/readingAllParam.jsp" target="\_blank">Показать в отдельной вкладке</a></button></td>  
 </tr>  
</table>  
</body>  
</html>