Вариант 1:

- Загружаем WildFly в свою папку
- В 316 строчке в standalone.sh убираем скобки AGENT_SET = $(...)
- В файле standalone.xml листаем вниз и ставим свой offset. Ваш порт будет 8080 + offset
- В папку deployments загружаем war файл
- Запускаем bin/standalone.sh
- Заходим в командную строку и пробрасываем порты:
  ssh -L PORT:localhost:PORT sXXXXXX@helios.se.ifmo.ru -p 2222
- Вводим пароль и заходим на localhost:PORT/var_name

Вариант 1, подробно:

1) Открываете WinSCP, создаете директорию для своей лабы, перетаскиваете туда папку с разархивированным WildFly

2) В этой папке есть директория bin, находите в ней файл standalone.sh. В его содержимом в стоке 316 убираете скобки, должно получится:
   =$echo “MODULE_OPTS" | $GREP "\-javaagent:

   возвращаемся в главную папку WildFly, открываем standalone/configuration/standalone.xml, в самом конце есть тег socket-binding-group, у него есть атрибут port-offset, которому присваивается смещение, там нужно написать свое число:

   port-offset="${jboss.socket.binding.port-offset:<ваше смещение>
   }
   Тогда ваш номер вашего порта будет равен значению port из дочернего тега <socket-binding name="http" port="${jboss.http.port:8080}"/>(по дефолту это 8080) плюс смещение.

   Например, здесь номер порта будет 8080+1020 = 9100:

   <socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:1020}">
   <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
   <socket-binding name="http" port="${jboss.http.port:8080}"/>
   <socket-binding name="https" port="${jboss.https.port:8443}"/>
   *тут другие дочерние теги *
   </socket-binding-group>

3) возвращаемся в главную директорию WildFly, переходим в каталог standalone/deployments, перетаскиваем сюда из идеи ваш собранный варник(убедитесь, что вы загружаете нормальный вариант))0)
4) bash bin/standalone.sh
5) возвращаемся в главную директорию WildFly, прописываем команду , это запускает ваш сервер. Могут быть ошибки, например, если этот порт уже занят, придется вернуться на шаг 3) и указать другое смещение

6) Открываете командную строку вашей локальной машины!!!, не с гелиосом!, и прописываете команду:
ssh -L PORT:localhost:PORT sXXXXXX@helios.se.ifmo.ru -p 2222
вместо PORT без пробелов подставляете номер вашего порта, полученного на шаге 3)
Далее вам нужно будет ввести пароль от хелиоса

7) Остается открыть браузер и ввести
http://localhost:18100/web3-1.0-SNAPSHOT.
Если все прошло нормально, то на этом localhost:<номер порта> адресе должна быть странична WildFly

Вариант 2:

1) скачать zip-архив с wildfly с его сайта;

2) этот архив распаковать, полученную папку поместить на гелиос;

2*) на гелиос заходить, пробрасывая порты на тот порт, на котором будете смотреть лабу (дальше про номер порта и portbase чуть подробнее)

3) в этой папке (которая уже на гелиосе) найти файл standalone.xml по пути standalone/configuration/standalone.xml

В этом файле заменить

```
<interface name="public">
 <inet-address value="${jboss.bind.address:127.0.0.1}"/>
 </interface>
```

на

```
<interface name="public">
 <any-address/>
 </interface>
```

а также строчку

```
<socket-binding name="http" port="${jboss.http.port:8080}"/>
```

на

ваш Portbase из гугл-таблички по вебу (+ не обязательно добавить к нему число от 0 до 99, т.к. Portbase выдают с расчётом, что 100 портов, идущие после него, тоже ваши, и вы их можете спокойно использовать}"/>

5) чтобы задеплоить лабу, первое - запускаете сервер:

bash <путь к wildfly>/bin/standalone.sh

второе - копируете war-архив с вашей лабой в папку <путь к wildfly>/standalone/deployments

6) Если пробрасывали порты, то лаба доступна по адресу localhost:<ваш порт>/lab2

ВОПРОСЫ К ЗАЩИТЕ

Тут хорошо расписано про веб на сервлетах

1. Java-сервлеты. Особенности реализации, ключевые методы, преимущества и недостатки относительно CGI и FastCGI

Сервлеты - серверные сценарии, написанные на Java. Жизненным циклом сервлетов управляет веб-контейнер. В отличие от CGI, запросы обрабатываются в отдельных потоках (а не процессах) на веб-контейнере.

Преимущества:
- Выполняются быстрее, чем CGI-сценарии
- Хорошая масштабируемость
- Надежность и безопасность (реализованы на Java)
- Платформенно-независимый
- Множество инструментов мониторинга и отладки

Недостатки:
- Слабое разделение уровня представления и бизнес-логики
- Возможны конфликты при параллельной обработке запросов

2. Контейнеры сервлетов. Жизненный цикл сервлета.

ОБРАБОТКА HTTP-запроса

1. Браузер формирует HTTP-запрос и отправляет его на сервер
2. Веб-контейнер создает объекты HttpServletRequest (обработка данных запроса клиента) и HttpServletResponse (ответ сервера, содержит поток вывода TCP-соединения)
3. Веб-контейнер вызывает метод service сервлета
4. Сервлет формирует ответ и записывает его в поток вывода HttpServletResponse

ЖИЗНЕННЫЙ ЦИКЛ

- Жизненным циклом сервлета управляет веб-контейнер
- Методы, управляющие жизненным циклом, должен вызывать только веб-контейнер

1. Загрузка класса. Проверяются пути /WEB-INF/classes/, WEB-INF/lib/*.jar, стандартные классы Java SE и классы веб-контейнера
2. Создание экземпляра
3. Вызов метода init()

4. Ожидание запроса. Вызов метода service()
5. Вызов метода destroy()

3. Диспетчеризация запросов в сервлетах. Фильтры сервлетов.

- Сервлеты могут делегировать обработку запросов другим ресурсам (сервлетам, JSP и HTML-страницам)
- Диспетчеризация осуществляется с помощью реализаций интерфейса javax.servlet.RequestDispatcher
- Два способа получения RequestDispatcher - через ServerRequest (абсолютный или относительный URL) и ServletContext (только абсолютный URL)
- Два способа делегирования обработки запроса - forward, include

ФИЛЬТРЫ ЗАПРОСОВ

- Фильтры позволяют осуществлять пред- и пост обработку запросов до и после передачи их ресурсу (сервлету, JSP или HTML-странице)
- Пример предобработки - допуск к странице только авторизованных пользователей
- Пример постобработки - запись в лог времени обработки запроса
- Реализуют интерфейс javax.servlet.Filter
- Ключевой метод - doFilter
- Метод doFilter класса FIlterChain передаёт управление следующему фильтру или целевому ресурсу; таким образом, возможна реализация последовательностей фильтров, обрабатывающих один и тот же запрос

4. HTTP-сессии - назначение, взаимодействие сервлетов с сессией, способы передачи идентификатора сессии.

- HTTP - stateless-протокол
- javax.servlet.HttpSession - интерфейс, позволяющий идентифицировать ContexScрконкретного клиента (браузер) при обработке множества HTTP-запросов от него
- Экземпляр HttpSession создаётся при первом обращении клиента к приложению и сохраняется некоторое (настраиваемое) время после последнего обращения
- Идентификатор сессии либо помещается в cookie, либо добавляется к URL. Если его удалить, то сервер создаст новую сессию.
- В экземпляр HttpSession можно помещать общую для этой сессии информацию (get/setAttribute)
- Сессия привязана к конкретному приложению, у разных приложений разные сессии
- В распределенном окружении обеспечивается сохранение целостности данных в HTTP-сессии (независимо от количества экземпляров JVM)

5. Контекст сервлета - назначение, способы взаимодействия сервлетов с контекстом.

- API, с помощью которого сервлет может взаимодействовать со своим контейнером
- Доступ к методам осуществляется через интерфейс javax.servlet.ServletContext
- В контекст можно помещать общую для всех сервлетов информацию (методы getAttribute, setAttribute)
- Если приложение - распределенное, то на каждом экземпляре JVM контейнером создаётся свой контекст

6. JavaServer Pages. Особенности, преимущества и недостатки по сравнению с сервлетами, область применения.

- Страницы JSP - текстовые файлы, содержащие статический HTML и JSP-элементы
- JSP-элементы позволяют формировать динамическое содержимое
- При загрузке в веб-контейнер страницы JSP транслируются компилятором jasper в сервлеты
- Позволяют отделить бизнес-логику от уровня представления (если их комбинировать с сервлетами)

ПРЕИМУЩЕСТВА:

- Высокая производительность - транслируются в сервлеты
- Не зависят от используемой платформы - код пишется на Java
- Позволяют использовать Java API
- Простые для понимания - структура похожа на обычный HTML

НЕДОСТАТКИ:

- Трудно отлаживать, если приложение целиком основано на JSP
- Возможны конфликты при параллельной обработке нескольких запросов

7. Жизненный цикл JSP.

1. ф
2. Хуй
3. Компиляция сервлета
4. Загрузка класса сервлета
5. Создание экземпляра сервлета
6. Вызов метода jsPinit()

7. Ожидание запроса. Вызов метода _jspService()
8. Вызов метода jspDestroy()

8. Структура JSP-страницы. Комментарии, директивы, объявления, скриптлеты и выражения.

- Два варианта синтаксиса - на базе HTML и XML
- Обозначаются тегами <% %> (HTML-вариант)
- Существует 5 типов JSP-элементов
    1. Комментарий <%-- Comment --%>
        - HTML-комментарии
          <!-- Comment →
        - - JSP-комментарии
          <%-- Comment →
        - Java-комментарии
          <% /* Java comment */ %>
    2. Директива <%@ directive %>. Управляют процессом трансляции страницы в сервлет.

       <%@ page session="false" %>
       <%@ include file="incl/copyright.html" %>

    3. Объявление <%! decl %>. Объявить поля и методы
    4. Скриптлет <% code %>. JavaCode, исполняемый при вызове метода _jspService
    5. Выражение <%= expr %>. Результат вычисления выражения

9. Правила записи Java-кода внутри JSP. Стандартные переменные, доступные в скриптлетах и выражениях.

ПРЕДОПРЕДЕЛЕННЫЕ ПЕРЕМЕННЫЕ:

- application
- config
- exception
- out
- page
- PageContext
- request
- response
- session

Директива Page позволяет задавать параметры, используемые контейнером при управлении жизненным циклом страницы. На одной странице может быть

задано несколько директив с разными указаниями контейнеру.

Атрибуты:

- buffer. Параметры буферизации и размер буфера для потока вывода
- autoFlush. Автоматически ли выгружается содержимое буфера при его переполнении
- contentType. Задать Content Type и кодировку страницы
- errorPage. Задать страницу, на которую будет осуществлено перенаправление при возникновении RE
- isErrorPage. Является ли страница Error Page
- extends. Имя родительского класса, от которого наследуется сервлет
- import
- info
- isThreadSafe
- language
- session
- isELIgnored
- isScriptingEnabled

JSP Actions - XML-элементы, позволяющие управлять поведением сервлеты
<jsp:action_name attribute="value"/>

- include. Включение внешнего файла
- useBean. Добавляет экземпляр JavaBean с заданным контекстом
- getProperty, setProperty. Установка свойств JavaBean
- forward. Перенаправление на другую страницу

10. /Bean-компоненты и их использование в JSP.

https://javatutor.net/articles/ejb-components-in-jsp-pages

11. Стандартные теги JSP. Использование Expression Language (EL) в JSP.

.	Access a bean property or Map entry
[]	Access an array or List element
( )	Group a subexpression to change the evaluation order
+	Addition
-	Subtraction or negation of a value
*	Multiplication
/ or div	Division
% or mod	Modulo (remainder)
== or eq	Test for equality
!= or ne	Test for inequality
< or lt	Test for less than
> or gt	Test for greater than

<= or le	Test for less than or equal
>= or ge	Test for greater than or equal
&& or and	Test for logical AND
|| or or	Test for logical OR
! or not	Unary Boolean complement
empty	Test for empty variable values

12. Параметры конфигурации JSP в дескрипторе развертывания веб-приложения.

- Задается в web.xml (дескриптор развертывания)
- Находится внутри элемента jsp-config

13. Шаблоны проектирования и архитектурные шаблоны. Использование в веб-приложениях.

- Шаблон проектирования или паттерн - повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста
- Описывает подход к решению типовой задачи
- Одну и ту же задачу часто можно решить с использованием разных шаблонов
- Существует много литературы с описанием различных шаблонов проектирования
- Позволяют избежать “типовых” ошибок при разработке типовых решений
- Позволяют кратко описать подход к решению задачи - программистам, знающим шаблоны, проще обмениваться информацией
- Легче поддерживать код - его поведение более предсказуемо
- В книге GoF (Gang of Four) описаны 23 классических паттерна

ПОРОЖДАЮЩИЕ:
- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

СТРУКТУРНЫЕ:
- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight

- Proxy

ПОВЕДЕНЧЕСКИЕ:
- Chain of responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template
- Visitor

АРХИТЕКТУРНЫЕ ШАБЛОНЫ
- Более высокий уровень по сравнению с шаблонами проектирования
- Описывают архитектуру всей системы или приложения
- Обычно имеют дело не с отдельными классами, а с целыми компонентами или модулями
- Компоненты и модули могут быть построены с использованием различных шаблонов проектирования

14. Архитектура веб-приложений. Шаблон MVC. Архитектурные модели Model 1 и Model 2 и их реализация на платформе Java EE.

3 УРОВНЯ АРХИТЕКТУРЫ:
- Клиент
- Бизнес-логика
- Данные

АРХИТЕКТУРА Model 1
- Предназначена для проектирования приложений небольшого масштаба и сложности
- За обработку данных и представления отвечает один и тот же компонент

ШАБЛОН MVC
- Model
- View
- Controller

АРХИТЕКТУРА Model 2
- Предназначена для проектирования достаточно сложных веб-приложений (корпоративных)

- За обработку и представление данных отвечают разные компоненты

Вторую модель реализуют Apache Struts, Apache Velocity и JavaServer Faces (в составе Java EE)