Рубежка №1

* Билет 1
  + LESS / SASS / SCSS. Особенности, отличия, поддержка браузерами
    - Особнность:
      * Можно использовать переменные
      * Можно использовать вложенные правила
      * Более мощные возможности по импорту, наследованию силей
      * Поддержка математических операторов
    - Браузеры могут не поддерживать их таблицы стилей, нужно преобразует в CSS при помощи транслятора
    - Отчличия:
      * В Less нет логики(if, for)
      * в LESS можно включать в определение свойства других классов.
      * Scss новая грамматика Sass. И в Scss нет скобки
  + Java SE, Java EE, Java ME. Платформа Java EE, понятие контейнера и компонента
    - SE - Стандартное издание.Ядро java
    - EE - Корпоративное издание. Новые классы
    - ME- платформа для мобильных устройств
    - Java EE это платформа, построенная на основе Java SE.
    - Контейнер - среда работы комронента. Он управляет жизнь и взаимодействие компонент
    - Компонента - выполняет задание. Они не может взаимодействовать без контейнера
  + Структура HTTP запроса, передающего логин и пароль пользователя

логин и пароль пользователя в URL(login и password)

POST /index.html?username=username&password=password HTTP/1.1

* Билет 2
  + ESMAScript - особенности, типы данных, инструкции.
    - это встраиваемый расширяемый не имеющий io язык программирования, для построения других скриптовых языков
    - примитивные: Number,String,Boolean, Null, Undefined

составные: Object, List, Reference

* + - Блок, переменное, выражение, цикл, исключение и так далее. В ES6 есть Класс
  + Правила трансляции JSP.
    - Обычные html и текст или <%=expression%> будет в out.print()
    - <%java code%> в \_jspService() и наконец вызовываться service()
    - <%! Field/Method Declaration %> переводят в поля или метод servlet
    - <%@ %> управляет процесс трансляции
  + Написать код HTML-страницы, отправляющей номер вопроса и выбранный вариант ответа (латинская буква от "A" до "F") после получения некоего текста.

<body>  
<script>  
function send(){  
let answer = document.getElementById("answer").value;  
let pattern = /^(A|B|C|D|E|F)$/g  
if(pattern.test(answer)){  
let form = document.getElementById("form");  
let data = new FormData(form)  
let request = new XMLHttpRequest();  
request.open("POST","url");  
request.send()  
}  
}  
</script>  
<form id = "form" method = "post" action = "url" onsubmit = send()>  
<input type = text id = "answer">  
</form>  
<body>

* Билет 3
  + методы HTTP
    - get, post, delete, head, put, options
  + жизненный цикл сервлета
    - Загрузка класса
    - Создать экземплар
    - вызов init()
    - service()
    - destroy
  + написать на JS функцию, которая на странице заменяет все тестовые поля ввода на кнопки

function replaceAllTextInputs(){  
  Array.from(document.querySelectorAll("input[type=text], textarea"))  
      .forEach(input => {  
          let button = document.createElement('button').setAttribute("value",input.value);  
          input.replaceWith(button);  
      });  
}

* Билет 4
  + коды состояния HTTP
    - 1 - informational
    - 2 - success
    - 3 - redirection
    - 4 - client error
    - 5 - server error
  + Фильтры запросов. Реализация пред и пост обработки запросов
    - Изменяем его url-pattern. Если в него клиент то пост, если в него servlet то пред
  + написать css правило, которое повернёт все картинки в форме с id=sampleForm на 90 градусов по часовой стрелке

#sampleForm img{

transform: rotate(90deg)

}

* Билет 5
  + Структура Html-документа

​

​

<head> заголовок

</head>

<body>

тело документа

</body>

</html>

* + Servlet - особенности, отличия от CGI/ FastCGI
    - Servlet только занимает один процесс с могими потоками чтобы контролировать запросы. А CGI один запрос, один процесс
  + Написать CSS правило, которое будет обводить все картинки в классе news в рамку при наведении мышю

.news img:hover{

border:5px solid black;

}

* Билет 6
  + Super agent
    - api for ajax. Easy to learn and use
  + Шаблоны проектирования, архитектурные, для чего нужны, разница
    - проектирования - архитектурная контрукция как решение проблемы
    - архитекрурные: Более высокий уровень. Дело обычно с целыми компонентами или модуль. Значит могут быть построены с различными проектирования
  + Jquery ajax запрос на сервлет, ответ от сервлета - объект json, вывести на страницу firstname, lastname, img\_url

$.ajax({

url: "url of servlet"

data: {img\_url: 'img\_url',lastname:'lastname',firstname:'firstname'}

dataType: "json"

})

* Билет 7
  + DOM и BOM
    - Платфор-независимые интерфейс, который позволяет допуск к содержимым HTML страницы. Структура как дерево
    - Прослйка между ядром и DOM. Интерфейс позволяет взаймодействовать с окном бразуер
  + Управление сессией. HttpSession.
    - Создание: request.getSession(true)
    - Получение: Получить из Cookie
    - Destroy: invalidate() или в элементе <session-timeout> определяет время жизни
  + Написать JS-функцию, которая запрещает вводить любые символы, кроме цифр и букв латинского алфавита

**function** fixValue(){

**if** (**this**.value.match(*/[ ^ a-z0-9]/ig*) !== **null**)

**this**.value = '';

}

**function** setValidation(){

let inputs = document.querySelectorAll('input[type=text], input[type=email], input[type=password], textarea');

Array.from(inputs).forEach(

input => {

input.addEventListener('input', fixValue)

}

);

* Билет 8
  + CSS : назначение, правила, приоритеты
    - определяет как страница выглядит
    - Selector{

key : value

}

* + - ！important > в <style = ""> > #id > колчество class(тем больше, тем выше) > количество имени
  + MVC : назначение, элементы, примеры реализации
    - Основная цель применения этой концепции состоит в отделении [бизнес-логики](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) (*модели*) от её визуализации.За счёт такогповышается возможность [повторного использования кода](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0).
    - Сontroller(управляет другие), Model(как работает), View(как выглядит)
  + Реализовать функцию на JavaScript, которая будет закрывать текущее окно, если в нем открыт [https://www.google.ru](https://www.google.ru/) ([Возможный ответ](http://pastebin.com/72gLeMH7))

**function** close(){

let isGoogle = documet.URL.match(*/^https?:\/\/(www*.*)?google*.*com(\/.*\**)?$/ig*);

**if** (isGoogle)

window.close();

}

* Билет 9
  + AJAX и DHTML - описание, сходства и различия
    - DHTML: Css, JS, Dom, HTML. Динамичеки изменяет страницу при операции позователя
    - AJAX: XmlHttpRequest(). Отправить запрос в север, и потом динамически изменять страницу на основе ответа.
    - Все могут динамичеки изменять страницу
    - DHTML все операции совершенны в клиенте. AJAX взаймодействует с сервером
  + Какие проблемы возникают при параллельной обработке запросов в JSP, как этого можно избежать?

Может быть конфликты. Непредсказуемые процесс программы и результаты. Можем делать асихронно или savebuy().

* + Написать js функцию, которая заменяет содержимое с именем класса "nyan" на изображение по ссылке: <http://www.example.com/nyancat.gif> ([Возможный ответ](http://pastebin.com/HFiFU850))

**function** replaceWithNyan(){

let img = document.createElement('img');

img.src = '<http://www.example.com/nyancat.gif>';

Array.from(document.querySelectorAll('.nyan')).forEach(

div => {

div.innerHTML = '';

div.appendChild(img);

}

);

}

* Билет 10
  + Rest и RPC
    - REST будет использовать все методы HTTP, а RPC только будет использовать Get и Post
    - Rest - подход к архитектуре сетевых протоколов, обеспечивающих доступ к ресурсам
    - RPC - подход, основенный на вызове удаленных процедур
  + RequestDispatcher
    - Полученны из контекста(абсолютная путь) или request(абсолютная или относительная)
    - dispatch the request to another resource with forward();
  + Правило css, меняющее цвет фона на желтый, если ссылка посещена и не лежит в классе "news" ([Возможный ответ](http://pastebin.com/Y9Crhmaq))

not(.news)>a:visited {

**color**: yellow;

}

* Билет 11
  + javascript, особенности, что для чего и тд
    - объектно-ориентированный скриптовый язык программирования для придания интерактивность странтицам.
    - Из ECMAscript, DOM, BOM
  + JSP actions
    - XML-элемент, позволяющие управлять поведением сервлет
    - <jsp : action attribute = "value"/>
  + php скрипт, который достаёт из get запроса имя и фамилию и приветствует пользователя, выводя html страницу ([Возможный ответ](http://pastebin.com/pWbJWbM5))

**<?php**

$name = $\_GET['name'];

echo 'Hello ' . ($name?$name:'noname');

**?>**

* Билет 12:
  + структура протокола http, характеристики

Сообщение(запрос и ответ, они все состоят из стартовой строки, заголовка и тело) и соединение. И каждый запрос имеет свой TCP соединение.

* + жизненный цикл jsp
    - Трансляция в сервлет
    - компиляция сервлета
    - загразка класса сервлета
    - создать экземпляра сервлета
    - вызов jspinit()
    - вызов jspService()
    - вызов jspDestroy
  + написать на php класс ([Возможный ответ](http://pastebin.com/qy4x8CTa))

**class** User{

**private** $userName;

**private** $age;

**function** \_\_constuctor($userName, $age){

$this->userName = $userName;

$this->age = $age;

}

**function** whois(){

return 'Name: ' . $this->userName . '\nAge: ' . $this->age;

}

}

* Билет 13
  + Преимущества и недостатки ajax
    - Экономия трафика, уменшение нагрузка на сервер, ускорение реакции интерфейса
    - усложение проекта, требуется javascript, отсутствие интеграции со стандартными инструментами браузера
  + Директива page: назначение, особенности, атрибуты
    - управлять параметр JSP страницы
    - <% @ directive attribute = "value" %>, может в любом месте
    - атрибуты определяют что делать
  + Написать конфигурацию сервлета (org.xxx.MyServlet) с помощью аннотации. Сервлет должен принимать все запросы от файлов .html .xhtml

@WebServlet(name = "org.xxx.MyServlet", value = "/servlet",)

* Билет 14
  + структура http запроса
  + типы данных php

interger , float , double, boolean , string , NULL

* + сервлет перенаправляющий все запросы на страницу google ([Возможный ответ](http://pastebin.com/EnTVLpNG))

**class** ToGoogleServlet **extends** HttpServlet{

**public** **void** service(HttpServletRequest request, HttpServletResponse response) **throws** IOException, ServletException{

response.sendRedirect("<http://google.com>;")

}

}

* Билет 15
  + Php, особенности и запуск

скриптовый язык, часто используемый для написания веб-приложений. ООП

* + Long Polling vs WebSockets

Long polling is much more resource intensive on servers whereas WebSockets have an extremely lightweight footprint on servers. Long polling also requires many hops between servers and device

* + Написать JSP страницу, которая будет возвращать количество сессий, обратившихся к ней за последние 60 секунд и формировать вывод в HTML

в jsp

<%Enumeration e = session.getAttributeNames();  
  int count = 0;  
  while(e.hasMoreElements()) {  
      count = count+1;  
      e.nextElement();  
  }  
  response.getWriter().println(e);  
%>

в xml

<session-config>  
  <session-timeout>60</session-timeout>  
</session-config>

* Билет 16:
  + элементы JSP

<%-- omment --%> <%@ directive %> <%! decl%> <% code %> <%= expression%>

* + CGI - обработка запроса, преимущества и недостатки

Вызов программ на севере использователем. Каждый запрос обрабатывается одельным процессом СGI-прогораммы

* + - Преимущества: программы могут быть написаы на множестве языков

исключены конфликты при параллельной обработке

нескольких запросов

* + - Недостатки: Высокие расходы. Слабое разделение уровня представления и бизнес-логики
  + Написать сервлет, который принимает из http запроса параметр name и выводит его. Если параметр не обнаружен то вывести Anonymous user ([Возможный ответ](http://pastebin.com/tHgjJSLz))

**class** HelloServlet **extends** HttpServlet{

**public** **void** service(HttpServletRequest request, HttpServletResponse response) **throws** IOException, ServletException{

response.setContentType("text/html; charset=UTF-8");

String name = request.getParameter("name");

PrintWriter out = response.getWriter();

if(name==**null**){

out.println(<h1>"Anonymous user</h1>);

}else{

out.println("<h1>Hello " + name + "</h1>");

out.close();

}

}

}

* Билет 17:
  + FastCGI. Плюсы, минусы, отличия от CGI
    - Безопасность, высокая стабильность, быстрее
    - Не может работать долго, иначе север его убирает.
    - Все запросы могут обрабатываться одним процессом
  + Суперглобальные массивы в PHP (SuperGlobal massive)

Предопределённые массивы, имеющие глобальную область видимости

* + - $\_GLOBALS - массив всех глобальных переменных
    - $\_SERVER - параметры, которые ОС передаёт серверу при его запуске
    - $\_ENV - переменные среды ОС

$\_ POST, $ \_GET, $ \_COOKIE, $ \_ REQUEST, $ \_ FILE, $ \_SESSION

* + Страница JSP, проверяющая есть ли /какой-то параметр/ в запросе и если нету - выводящая сообщение об ошибке ([Возможный ответ](http://pastebin.com/9yVFXZku))

<%= request.getParameter("parameterName")==null?"400 - your did not set parameter":request.getParameter("parameterName") %>

* Билет 18
  + ООП в PHP

Как java в PHP тоже есть класс. Наследование тоже не запрещено. Могут решить проблемы при создании экземплар.

* + Предопределенные переменные JSP

Суперглобальные массивы в PHP (SuperGlobal massive)

* + Код фильтра запросов, запрещающий доступ к приложению неавторизированным пользователям(у неавт пол в запросе отсутствует заголовок x-application-user) ([Возможный ответ](http://pastebin.com/qTuT55Fg))

**class** AuthFilter **extends** ServletFilter{

**public** **void** init(FilterConfig config){}

**public** **void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain) **throws** IOException{

String auth = request.getHeader("x-application-user");

**if** (auth == **null** || auth.equals("")){

PrintWriter out = response.getWriter();

response.setContentType("text/html; charset=UTF-8");

out.println("WTF?! You are not authorized");

}**else**{

chain.doFilter(request, response);

}

}

}

* Билет 19
  + Конфигурация PHP-интеррпретатора, способы интеграции PHP-интеррпретатора с веб-сервером
    - интеграции PHP-интеррпретатора и веб-сервер являются средой PHP.
  + ServletContext - особенности, для чего нужен
    - контейнер, где сохраняет все данные в одном севере. Он общое пространство для всех клиенте
    - Получения глобальные параметры. Читения файла ресурсов проектов. Переводить запрос
  + Код jsp-страницы показывающий содержимое корзины юзера. Содержимое корзины - коллекциия объектов класса ShoppingItem который содержит имя, стоимость и количество заказанного товара - хранится в отдельном managed bean. ([Возможный ответ](http://pastebin.com/pBA7f4Zt))

<%@ import="my.package.ShoppingItem" %>

<%@ import="java.util.Collection" %>

<%@ contentType="text/html;charset=UTF-8" language="java"%>

<jsp:useBean id="managed" class="my.package.ManagedBean" scope="session">

<**html**>

<**head**>

        <meta charset="utf-8">

</**head**>

<**body**>

<**tr**>

<**th**>name</**th**>

<**th**>price</**th**>

<**th**>count</**th**>

</**tr**>

<%

Collection<ShoppingItem> basket = mananged.getBasket();

for (ShoppingItem position: basket){%>

<**tr**>

<**td**><%= position.getName() %></**td**>

<**td**><%= position.getPrice() %></**td**>

<**td**><%= position.getCount() %></**td**>

</**tr**>

<%}%>

</**table**>

</**body**>

* Билет 20
  + HTML формы
    - cбор ввод разных видов использователя

text, checkbox, radio, button, password

action = "url" - куда отправитьs

* + конфигурация сервлетов. файл web.xml

<servlet>

<servlet-name>name</servlet-name>

<servlet-class>classname</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>name</servlet-name>

<url-pattern>/url</url-pattern>

</servlet-mapping>

* + написать css правило, которое при клике на ссылку добавляет ей подчеркивание, всем кроме ссылок в теге h1 ([Возможный ответ](http://pastebin.com/Y9Crhmaq))

not(h1) a:active {

**text-decoration**: underline;

}

* Билет 21
  + jQuery.
    - Библиотека Javascript, чтобы легче использовать javascript
  + Архитектуры model 1 и model 2.
    - model 1: клиент отправляет запросы jsp или сервлет. И потом то jsp (или сервлет) завершает все задания
    - model 2: обычно при MVC. Задания завершены нескольких jsp( или сервлетов)
  + Написать php скрипт, формирующий форму для ввода логина и пароля и отправляющий запрос сервису authorize.php с помощью UserAgent. Если пользователь корректный, то скрипт должен редиректить на страницу pagename.php.
* Билет 22
  + Экмаскрипт, преимущества 6й и 7й версии
    - Можно писать класс
    - Новые операции, например, функция с ->(похож на lambda в java), которые упрощают прогромирование
  + ГоФ паттерны. Что такое? Основные виды, примеры
    - Повторимая архитектурная конструкция, которая решает какую-то проблема.
    - Порождающие: builder, factory, одиночка

Структурная: bridge, proxy, adapter, decorator

Поведенческие: command, iterator, observer

* + Написать html страницу и сервлет, возвращающий странице количество активных сессий
* Билет 23
  + обработка http с помощью servlet
    - При помощи HttpServletRequest получить запрос и делать обработку. И ответ сохраняется в HttpServletResponse
  + многоуровневая архитектура приложения
    - Presentatio: самый верхний уровень. Что мы видим
    - logic: Между другими двумя.Получать запрос, читать данные из data, делать обработку, отправить результат клиенту
    - data:Сохранение данных
  + с помощью jquery посчитать количество div с классом lecture содержащие «де-факто»
* Билет 24
  + ECMAScript, типы данных, особенности, инструкции
    - это встраиваемый расширяемый не имеющий средств [ввода-вывода](https://ru.wikipedia.org/wiki/%D0%92%D0%B2%D0%BE%D0%B4-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4) язык программирования, для построения других скриптовых языков
    - примитивные: Number,String,Boolean, Null, Undefined

составные: Object, List, Reference

* + - Блок, переменное, выражение, цикл, исключение и так далее. В ES6 есть Класс
  + цикл жизни jsp

Трансляция в сервлет

компиляция сервлета

загразка класса сервлета

создать экземпляра сервлета

вызов jspInit()

вызов jspService()

вызов jspDestroy()

* + html форма отправляющая номер вопроса и выбор от a до f

<body>

​<script>

function send(){

let form = document.getElementById("form");

let data = new FormData(form)

let request = new XMLHttpRequest();

request.open("POST","url");

request.send()

}

</script>

<form id = "form" method = "post" action = "url" onsubmit = send()>

<input type = radio value = "a" name = "choice">a<br>

<input type = radio value = "b" name = "choice">b<br>

<input type = radio value = "c" name = "choice">c<br>

<input type = radio value = "d" name = "choice">d<br>

<input type = radio value = "e" name = "choice">e<br>

<input type = radio value = "f" name = "choice">f<br>

</form>

<body>