Введение в Java EE. Сервлеты

№ урока: 3 **Курс:** Основы Java EE

Средства обучения: InteliJ Idea

Обзор, цель и назначение урока

Знакомство с J2EE. HTTP. Сервлеты. Создание первых веб приложений

Изучив материал данного занятия, учащийся сможет:

- Поймет разницу между Java SE и Java EE
- Создавать первые веб приложения
- Создавать разметку страницы HTML
- Добавлять стили CSS для страницы
- Научится работать с жизненным циклом сервлета
- Обрабатывать запросы сервлета через get и post

Содержание урока

1. Отличия Java SE и Java EE.
2. Архитектура и состав Java EE.
3. HTTP. Принцип работы.
4. Спецификация Java EE
5. HTML. Основные тэги.
6. CSS. Способы подключения.
7. Типы веб серверов
8. Сервлеты. Преимущества и недостатки
9. Методы HTTP запроса
10. Многопоточность в серлетах

Резюме

- **Java Platform, Enterprise Edition**, сокращенно **Java EE** (до версии 5.0 — **Java 2 Enterprise Edition** или **J2EE**). В 2018 Eclipse Foundation переименовала **Java EE** в **Jakarta EE** — набор спецификаций и соответствующей документации для языка Java, описывающей архитектуру серверной платформы для задач средних и крупных предприятий.
  Спецификации детализированы настолько, чтобы обеспечить переносимость программ с одной реализации платформы на другую. Основная цель спецификаций — обеспечить масштабируемость приложений и целостность данных во время работы системы. Java EE во многом ориентирована на использование её через веб, как в интернете, так и в локальных сетях. Вся спецификация создаётся и утверждается через JCP (Java Community Process) в рамках инициативы Sun Microsystems Inc.
  Java EE является промышленной технологией и в основном используется в высокопроизводительных проектах, в которых необходима надежность, масштабируемость, гибкость.
  Популярности Java EE также способствует то, что Sun предлагает бесплатный комплект разработки, SDK, позволяющий предприятиям разрабатывать свои системы, не тратя больших средств. В этот комплект входит сервер приложений GlassFish с лицензией для разработки.
- **HTTP** (англ. *HyperText Transfer Protocol* — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов в формате «HTML», в настоящий момент используется для передачи

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

произвольных данных). Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

Потребителей (клиентов), которые инициируют соединение и посылают запрос;

Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с веб-сайтов. В 2006 году — в Северной Америке доля HTTP-трафика превысила долю P2P-сетей и составила 46 %, из которых почти половина — это передача потокового видео и звука[1].

HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня, таких как SOAP, XML-RPC, WebDAV.

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. (в частности, для этого используется HTTP-заголовок). Именно благодаря возможности указания способа кодирования сообщения, клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

HTTP — протокол прикладного уровня; аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI. В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос-ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера). Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

- **HTML** (от англ. *HyperText Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

  Язык HTML до 5-й версии определялся как приложение SGML (стандартного обобщённого языка разметки по стандарту ISO 8879). Спецификации HTML5 формулируются в терминах DOM (объектной модели документа).

  Язык XHTML является более строгим вариантом HTML, он следует синтаксису XML и является приложением языка XML в области разметки гипертекста.

  Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования.
- **CSS** (англ. *Cascading Style Sheets — каскадные таблицы стилей*) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

  Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

  CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.

- **Веб-сервер** — сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.
  Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер (см.: Сервер (аппаратное обеспечение)), на котором это программное обеспечение работает.
  Клиент, которым обычно является веб-браузер, передаёт веб-серверу запросы на получение ресурсов, обозначенных URL-адресами. Ресурсы — это HTML-страницы, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ веб-сервер передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP.
- **Сервлет** является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.
  Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP-специфичные сервлет классы.
  Пакеты `javax.servlet` и `javax.servlet.http` обеспечивают интерфейсы и классы для создания сервлетов.
  Первая спецификация сервлетов была создана в Sun Microsystems (версия 1.0 была закончена в июне 1997). Начиная с версии 2.3, спецификация сервлетов разрабатывалась под руководством Java Community Process. Стандарт JSR 53 определял как Servlet 2.3, так и спецификацию JavaServer Page 1.2. JSR 154 включает в себя спецификации Servlet 2.4 и 2.5. Текущая спецификация на 13 июня 2013 года — Servlet 3.1 (описана в JSR-340).
- **Метод HTTP** (англ. *HTTP Method*) — последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово, записанное заглавными буквами. Обратите внимание, что название метода чувствительно к регистру.
  Сервер может использовать любые методы, не существует обязательных методов для сервера или клиента. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус `501` (Not Implemented). Если серверу метод известен, но он неприменим к конкретному ресурсу, то возвращается сообщение с кодом `405` (Method Not Allowed). В обоих случаях серверу следует включить в сообщение ответа заголовок `Allow` со списком поддерживаемых методов.
  Кроме методов `GET` и `HEAD`, часто применяется метод `POST`.
  OPTIONS
  Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок `Allow` со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.
  Предполагается, что запрос клиента может содержать тело сообщения для указания интересующих его сведений. Формат тела и порядок работы с ним в настоящий момент не определён; сервер пока должен его игнорировать. Аналогичная ситуация и с телом в ответе сервера.
  Для того, чтобы узнать возможности всего сервера, клиент должен указать в URI звёздочку — «`*`». Запросы «`OPTIONS * HTTP/1.1`» могут также применяться для проверки работоспособности сервера (аналогично «пингованию») и тестирования на предмет поддержки сервером протокола HTTP версии 1.1.
  Результат выполнения этого метода не кэшируется.
  GET
  Используется для запроса содержимого указанного ресурса. С помощью метода `GET` можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа « ? »:

`GET /path/resource?param1=value1&param2=value2 HTTP/1.1`

Согласно стандарту HTTP, запросы типа `GET` считаются идемпотентными[4]

Кроме обычного метода `GET`, различают ещё

Условный `GET` — содержит заголовки `If-Modified-Since`, `If-Match`, `If-Range` и подобные;

Частичный `GET` — содержит в запросе `Range`.

Порядок выполнения подобных запросов определён стандартами отдельно.

HEAD

Аналогичен методу `GET`, за исключением того, что в ответе сервера отсутствует тело. Запрос `HEAD` обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше — копия ресурса помечается как устаревшая.

POST

Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода `POST` обычно загружаются файлы на сервер.

В отличие от метода `GET`, метод `POST` не считается идемпотентным[4], то есть многократное повторение одних и тех же запросов `POST` может возвращать разные результаты (например, после каждой отправки комментария будет появляться очередная копия этого комментария).

При результате выполнения `200` (Ok) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ `201` (Created) с указанием URI нового ресурса в заголовке `Location`.

Сообщение ответа сервера на выполнение метода `POST` не кэшируется.

PUT

Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существует ресурс, то сервер создаёт его и возвращает статус `201` (Created). Если же был изменён ресурс, то сервер возвращает `200` (Ok) или `204` (No Content). Сервер не должен игнорировать некорректные заголовки `Content-*`, передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или не допустим при текущих условиях, то необходимо вернуть код ошибки `501` (Not Implemented).

Фундаментальное различие методов `POST` и `PUT` заключается в понимании предназначений URI ресурсов. Метод `POST` предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя `PUT`, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.

Сообщения ответов сервера на метод `PUT` не кэшируются.

PATCH

Аналогично PUT, но применяется только к фрагменту ресурса.

DELETE

Удаляет указанный ресурс.

TRACE

Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.

CONNECT

Преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищённого SSL-соединения через нешифрованный прокси.

- **Код состояния HTTP** (англ. *HTTP status code*) — часть первой строки ответа сервера при запросах по протоколу HTTP. Он представляет собой целое число из трёх десятичных цифр. Первая цифра

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

указывает на ***класс состояния***. За кодом ответа обычно следует отделённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа.

- **Информационные** - этот класс выделены коды, информирующие о процессе передачи. При работе через протокол версии 1.0 сообщения с такими кодами должны игнорироваться. В версии 1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но серверу отправлять что-либо не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-сервера подобные сообщения должны отправлять дальше от сервера к клиенту.
  100 Continue — сервер удовлетворён начальными сведениями о запросе, клиент может продолжать пересылать заголовки. Появился в HTTP/1.1.
  101 Switching Protocols — сервер предлагает перейти на более подходящий для указанного ресурса протокол; список предлагаемых протоколов сервер обязательно указывает в поле заголовка `Upgrade`. Если клиента это заинтересует, то он посылает новый запрос с указанием другого протокола. Появился в HTTP/1.1.
  102 Processing — запрос принят, но на его обработку понадобится длительное время. Используется сервером, чтобы клиент не разорвал соединение из-за превышения времени ожидания. Клиент при получении такого ответа должен сбросить таймер и дожидаться следующей команды в обычном режиме.
- **Успех** - сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.
  200 OK — успешный запрос. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке и/или теле сообщения. Появился в HTTP/1.0.
  201 Created — в результате успешного выполнения запроса был создан новый ресурс. Сервер может указать адреса (их может быть несколько) созданного ресурса в теле ответа, при этом предпочтительный адрес указывается в заголовке `Location`. Серверу рекомендуется указывать в теле ответа характеристики созданного ресурса и его адреса, формат тела ответа определяется заголовком `Content-Type`. При обработке запроса, новый ресурс должен быть создан до отправки ответа клиенту, иначе следует использовать ответ с кодом `202`. Появился в HTTP/1.0.
  202 Accepted — запрос был принят на обработку, но она не завершена. Клиенту не обязательно дожидаться окончательной передачи сообщения, так как может быть начат очень долгий процесс. Появился в HTTP/1.0.
  203 Non-Authoritative Information — аналогично ответу `200`, но в этом случае передаваемая информация была взята не из первичного источника (резервной копии, другого сервера и т. д.) и поэтому может быть неактуальной. Появился в HTTP/1.1.
  204 No Content — сервер успешно обработал запрос, но в ответе были переданы только заголовки без тела сообщения. Клиент не должен обновлять содержимое документа, но может применить к нему полученные метаданные. Появился в HTTP/1.0.
  205 Reset Content — сервер обязывает клиента сбросить введённые пользователем данные. Тела сообщения сервер при этом не передаёт и документ обновлять не обязательно. Появился в HTTP/1.1.
  206 Partial Content — сервер удачно выполнил частичный GET-запрос, возвратив только часть сообщения. В заголовке `Content-Range` сервер указывает байтовые диапазоны содержимого. Особое внимание при работе с подобными ответами следует уделить кэшированию. Появился в HTTP/1.1. (*подробнее...*)
  207 Multi-Status — сервер передаёт результаты выполнения сразу нескольких независимых операций. Они помещаются в само тело сообщения в виде XML-документа с объектом `multistatus`. Не рекомендуется размещать в этом объекте статусы из серии `1xx` из-за бессмысленности и избыточности. Появился в *WebDAV*.
  члены привязки DAV уже были перечислены в предыдущей части (multistatus) ответа и не включаются снова.
  226 IM Used — заголовок `A-IM` от клиента был успешно принят и сервер возвращает содержимое с учётом указанных параметров. Введено в RFC 3229 для дополнения протокола HTTP поддержкой дельта-кодирования.

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- **Перенаправление** - коды этого класса сообщают клиенту, что для успешного выполнения операции необходимо сделать другой запрос, как правило, по другому URI. Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям. Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.
  По последним стандартам клиент может производить перенаправление без запроса пользователя только если второй ресурс будет запрашиваться методом GET или HEAD [7]. В предыдущих спецификациях говорилось, что для избежания круговых переходов пользователя следует спрашивать после 5-го подряд перенаправления[16]. При всех перенаправлениях, если метод запроса был не HEAD, то в тело ответа следует включить короткое гипертекстовое сообщение с целевым адресом, чтобы в случае ошибки пользователь смог сам произвести переход.
  Разработчики HTTP отмечают, что многие клиенты при перенаправлениях с кодами 301 и 302 ошибочно применяют метод GET ко второму ресурсу, несмотря на то, что к первому запрос был с иным методом (чаще всего PUT)[17]. Чтобы избежать недоразумений, в версии HTTP/1.1 были введены коды 303 и 307 и их рекомендовано использовать вместо 302. Изменять метод нужно только если сервер ответил 303. В остальных случаях следующий запрос производить с исходным методом.
  300 Multiple Choices — по указанному URI существует несколько вариантов предоставления ресурса по типу MIME, по языку или по другим характеристикам. Сервер передаёт с сообщением список альтернатив, давая возможность сделать выбор клиенту автоматически или пользователю. Появился в HTTP/1.0.
  301 Moved Permanently — запрошенный документ был окончательно перенесен на новый URI, указанный в поле Location заголовка. Некоторые клиенты некорректно ведут себя при обработке данного кода. Появился в HTTP/1.0.
  302 Found, 302 Moved Temporarily — запрошенный документ временно доступен по другому URI, указанному в заголовке в поле Location. Этот код может быть использован, например, при управляемом сервером согласовании содержимого. Некоторые[*какие?*] клиенты некорректно ведут себя при обработке данного кода. Введено в HTTP/1.0.
  303 See Other — документ по запрошенному URI нужно запросить по адресу в поле Location заголовка с использованием метода GET несмотря даже на то, что первый запрашивался иным методом. Этот код был введён вместе с 307-ым для избежания неоднозначности, чтобы сервер был уверен, что следующий ресурс будет запрошен методом GET. Например, на веб-странице есть поле ввода текста для быстрого перехода и поиска. После ввода данных браузер делает запрос методом POST, включая в тело сообщения введённый текст. Если обнаружен документ с введённым названием, то сервер отвечает кодом 303, указав в заголовке Location его постоянный адрес. Тогда браузер гарантировано его запросит методом GET для получения содержимого. В противном случае сервер просто вернёт клиенту страницу с результатами поиска. Введено в HTTP/1.1.
  304 Not Modified — сервер возвращает такой код, если клиент запросил документ методом GET, использовал заголовок If-Modified-Since или If-None-Match и документ не изменился с указанного момента. При этом сообщение сервера не должно содержать тела. Появился в HTTP/1.0.
  305 Use Proxy — запрос к запрашиваемому ресурсу должен осуществляться через прокси-сервер, URI которого указан в поле Location заголовка. Данный код ответа могут использовать только исходные HTTP-сервера (не прокси). Введено в HTTP/1.1.
  306 (зарезервировано) — использовавшийся раньше код ответа, в настоящий момент зарезервирован. Упомянут в RFC 2616 (обновление HTTP/1.1).
  307 Temporary Redirect — запрашиваемый ресурс на короткое время доступен по другому URI, указанный в поле Location заголовка. Метод запроса (GET/POST) менять не разрешается. Например, POST запрос должен быть отправлен по новому URI тем же методом POST. Этот код

CyberBionic systematics
CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

был введён вместе с 303 вместо 302-го для избежания неоднозначности. Введено в RFC 2616 (обновление HTTP/1.1).

308 Permanent Redirect — запрашиваемый ресурс был окончательно перенесен на новый URI, указанный в поле `Location` заголовка. Метод запроса (GET/POST) менять не разрешается. Например, POST запрос должен быть отправлен по новому URI тем же методом POST. Этот код был введён вместо 301-го для избежания неоднозначности. Введено в RFC 7238 (RFC 7538).

- **Ошибка клиента** - Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

  400 Bad Request — сервер обнаружил в запросе клиента синтаксическую ошибку. Появился в HTTP/1.0.

  401 Unauthorized — для доступа к запрашиваемому ресурсу требуется аутентификация. В заголовке ответ должен содержать поле `WWW-Authenticate` с перечнем условий аутентификации. Иными словами для доступа к запрашиваемому ресурсу, клиент должен представиться, послав запрос, включив при этом в заголовок сообщения поле `Authorization` с требуемыми для аутентификации данными.

  402 Payment Required — предполагается использовать в будущем. В настоящий момент не используется. Этот код предусмотрен для платных пользовательских сервисов, а не для хостинговых компаний. Имеется в виду, что эта ошибка не будет выдана хостинговым провайдером в случае просроченной оплаты его услуг. Зарезервирован, начиная с HTTP/1.1.

  403 Forbidden — сервер понял запрос, но он отказывается его выполнять из-за ограничений в доступе для клиента к указанному ресурсу. Иными словами, клиент не уполномочен совершать операции с запрошенным ресурсом. Если для доступа к ресурсу требуется аутентификация средствами HTTP, то сервер вернёт ответ `401`, или `407` при использовании прокси. В противном случае ограничения были заданы администратором сервера или разработчиком веб-приложения и могут быть любыми в зависимости от возможностей используемого программного обеспечения. В любом случае клиенту следует сообщить причины отказа в обработке запроса. Наиболее вероятными причинами ограничения может послужить попытка доступа к системным ресурсам веб-сервера (например, файлам `.htaccess` или `.htpasswd`) или к файлам, доступ к которым был закрыт с помощью конфигурационных файлов, требование аутентификации не средствами HTTP, например, для доступа к системе управления содержимым или разделу для зарегистрированных пользователей либо сервер не удовлетворён IP-адресом клиента, например, при блокировках. Появился в HTTP/1.0.

  404 Not Found — самая распространённая ошибка при пользовании Интернетом, основная причина — ошибка в написании адреса Web-страницы. Сервер понял запрос, но не нашёл соответствующего ресурса по указанному URL. Если серверу известно, что по этому адресу был документ, то ему желательно использовать код 410. Ответ `404` может использоваться вместо `403`, если требуется тщательно скрыть от посторонних глаз определённые ресурсы. Появился в HTTP/1.0.

  405 Method Not Allowed — указанный клиентом метод нельзя применить к текущему ресурсу. В ответе сервер должен указать доступные методы в заголовке `Allow`, разделив их запятой. Эту ошибку сервер должен возвращать, если метод ему известен, но он не применим именно к указанному в запросе ресурсу, если же указанный метод не применим на всём сервере, то клиенту нужно вернуть код `501` (Not Implemented). Появился в HTTP/1.1.

  406 Not Acceptable — запрошенный URI не может удовлетворить переданным в заголовке характеристикам. Если метод был не `HEAD`, то сервер должен вернуть список допустимых характеристик для данного ресурса. Появился в HTTP/1.1.

  407 Proxy Authentication Required — ответ аналогичен коду `401` за исключением того, что аутентификация производится для прокси-сервера. Механизм аналогичен идентификации на исходном сервере. Появился в HTTP/1.1.

  408 Request Timeout — время ожидания сервером передачи от клиента истекло. Клиент может повторить аналогичный предыдущему запрос в любое время. Например, такая ситуация может

возникнуть при загрузке на сервер объёмного файла методом POST или PUT. В какой-то момент передачи источник данных перестал отвечать, например, из-за повреждения компакт-диска или потери связи с другим компьютером в локальной сети. Пока клиент ничего не передаёт, ожидая от него ответа, соединение с сервером держится. Через некоторое время сервер может закрыть соединение со своей стороны, чтобы дать возможность другим клиентам сделать запрос. Этот ответ не возвращается, когда клиент принудительно остановил передачу по команде пользователя или соединение прервалось по каким-то иным причинам, так как ответ уже послать невозможно. Появился в HTTP/1.1.

409 Conflict — запрос не может быть выполнен из-за конфликтного обращения к ресурсу. Такое возможно, например, когда два клиента пытаются изменить ресурс с помощью метода PUT. Появился в HTTP/1.1.

410 Gone — такой ответ сервер посылает, если ресурс раньше был по указанному URL, но был удалён и теперь недоступен. Серверу в этом случае неизвестно и местоположение альтернативного документа (например, копии). Если у сервера есть подозрение, что документ в ближайшее время может быть восстановлен, то лучше клиенту передать код 404. Появился в HTTP/1.1.

411 Length Required — для указанного ресурса клиент должен указать Content-Length в заголовке запроса. Без указания этого поля не стоит делать повторную попытку запроса к серверу по данному URI. Такой ответ естественен для запросов типа POST и PUT. Например, если по указанному URI производится загрузка файлов, а на сервере стоит ограничение на их объём. Тогда разумней будет проверить в самом начале заголовок Content-Length и сразу отказать в загрузке, чем провоцировать бессмысленную нагрузку, разрывая соединение, когда клиент действительно пришлёт слишком объёмное сообщение. Появился в HTTP/1.1.

412 Precondition Failed — возвращается, если ни одно из условных полей заголовка (If-Match и др., см. RFC 7232) запроса не было выполнено. Появился в HTTP/1.1.

413 Payload Too Large — возвращается в случае, если сервер отказывается обработать запрос по причине слишком большого размера тела запроса. Сервер может закрыть соединение, чтобы прекратить дальнейшую передачу запроса. Если проблема временная, то рекомендуется в ответ сервера включить заголовок Retry-After с указанием времени, по истечении которого можно повторить аналогичный запрос. Появился в HTTP/1.1. Ранее назывался «Request Entity Too Large».

414 URI Too Long — сервер не может обработать запрос из-за слишком длинного указанного URI. Такую ошибку можно спровоцировать, например, когда клиент пытается передать длинные параметры через метод GET, а не POST. Появился в HTTP/1.1. Ранее назывался «Request-URI Too Long».

415 Unsupported Media Type — по каким-то причинам сервер отказывается работать с указанным типом данных при данном методе. Появился в HTTP/1.1.

416 Range Not Satisfiable — в поле Range заголовка запроса был указан диапазон за пределами ресурса и отсутствует поле If-Range. Если клиент передал байтовый диапазон, то сервер может вернуть реальный размер в поле Content-Range заголовка. Данный ответ не следует использовать при передаче типа multipart/byterange. Введено в RFC 2616 (обновление HTTP/1.1). Ранее назывался «Requested Range Not Satisfiable».

417 Expectation Failed — по каким-то причинам сервер не может удовлетворить значению поля Expect заголовка запроса. Введено в RFC 2616 (обновление HTTP/1.1).

418 I'm a teapot — Этот код был введен в 1998 году как одна из традиционных первоапрельских шуток IETF в RFC 2324, Hyper Text Coffee Pot Control Protocol. Не ожидается, что данный код будет поддерживаться реальными серверами[20].

421 Misdirected Request — запрос был перенаправлен на сервер, не способный дать ответ.

422 Unprocessable Entity — сервер успешно принял запрос, может работать с указанным видом данных (например, в теле запроса находится XML-документ, имеющий верный синтаксис), однако имеется какая-то логическая ошибка, из-за которой невозможно произвести операцию над ресурсом. Введено в *WebDAV*.

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

423 Locked — целевой ресурс из запроса заблокирован от применения к нему указанного метода. Введено в WebDAV.

424 Failed Dependency — реализация текущего запроса может зависеть от успешности выполнения другой операции. Если она не выполнена и из-за этого нельзя выполнить текущий запрос, то сервер вернёт этот код. Введено в *WebDAV*.

426 Upgrade Required — сервер указывает клиенту на необходимость обновить протокол. Заголовок ответа должен содержать правильно сформированные поля `Upgrade` и `Connection`. Введено в RFC 2817 для возможности перехода к TLSпосредством HTTP.

428 Precondition Required — сервер указывает клиенту на необходимость использования в запросе заголовков условий, наподобие `If-Match`. Введено в черновике стандарта RFC 6585.

429 Too Many Requests — клиент попытался отправить слишком много запросов за короткое время, что может указывать, например, на попытку DDoS-атаки. Может сопровождаться заголовком Retry-After, указывающим, через какое время можно повторить запрос. Введено в черновике стандарта RFC 6585.

431 Request Header Fields Too Large — Превышена допустимая длина заголовков. Сервер не обязан отвечать этим кодом, вместо этого он может просто сбросить соединение. Введено в черновике стандарта RFC 6585.

434 Requested host unavailable — Запрашиваемый адрес недоступен

449 Retry With — возвращается сервером, если для обработки запроса от клиента поступило недостаточно информации. При этом в заголовок ответа помещается поле `Ms-Echo-Request`. Введено корпорацией *Microsoft* для *WebDAV*. В настоящий момент как минимум используется программой *Microsoft Money*.

451 Unavailable For Legal Reasons — доступ к ресурсу закрыт по юридическим причинам, например, по требованию органов государственной власти или по требованию правообладателя в случае нарушения авторских прав. Введено в черновике IETF за авторством Google, при этом код ошибки является отсылкой к роману Рэя Брэдбери «451 градус по Фаренгейту». Был добавлен в стандарт 21 декабря 2015.

- **Ошибка сервера** – коды 5xx выделены под случаи неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

  500 Internal Server Error — любая внутренняя ошибка сервера, которая не входит в рамки остальных ошибок класса. Появился в HTTP/1.0.

  501 Not Implemented — сервер не поддерживает возможностей, необходимых для обработки запроса. Типичный ответ для случаев, когда сервер не понимает указанный в запросе метод. Если же метод серверу известен, но он не применим к данному ресурсу, то нужно вернуть ответ 405. Появился в HTTP/1.0.

  502 Bad Gateway — сервер, выступая в роли шлюза или прокси-сервера, получил недействительное ответное сообщение от вышестоящего сервера. Появился в HTTP/1.0.

  503 Service Unavailable — сервер временно не имеет возможности обрабатывать запросы по техническим причинам (обслуживание, перегрузка и прочее). В поле `Retry-After` заголовка сервер может указать время, через которое клиенту рекомендуется повторить запрос. Хотя во время перегрузки очевидным кажется сразу разрывать соединение, эффективней может оказаться установка большого значения поля `Retry-After` для уменьшения частоты избыточных запросов. Появился в HTTP/1.0.

  504 Gateway Timeout — сервер в роли шлюза или прокси-сервера не дождался ответа от вышестоящего сервера для завершения текущего запроса. Появился в HTTP/1.1.

  505 HTTP Version Not Supported — сервер не поддерживает или отказывается поддерживать указанную в запросе версию протокола HTTP. Появился в HTTP/1.1.

  506 Variant Also Negotiates — в результате ошибочной конфигурации выбранный вариант указывает сам на себя, из-за чего процесс связывания прерывается. Экспериментальное. Введено в RFC 2295 для дополнения протокола HTTP технологией *Transparent Content Negotiation*.

  507 Insufficient Storage — не хватает места для выполнения текущего запроса. Проблема может быть временной. Введено в *WebDAV*.

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

509 Bandwidth Limit Exceeded — используется при превышении веб-площадкой отведённого ей ограничения на потребление трафика. В данном случае владельцу площадки следует обратиться к своему хостинг-провайдеру. В настоящий момент данный код не описан ни в одном RFC и используется только модулем «bw/limited», входящим в панель управления хостингом *cPanel*, где и был введён.

510 Not Extended — на сервере отсутствует расширение, которое желает использовать клиент. Сервер может дополнительно передать информацию о доступных ему расширениях. Введено в RFC 2774 для дополнения протокола HTTP поддержкой расширений.

511 Network Authentication Required — этот ответ посылается не сервером, которому был предназначен запрос, а сервером-посредником — например, сервером провайдера — в случае, если клиент должен сначала авторизоваться в сети, например, ввести пароль для платной точки доступа к Интернету. Предполагается, что в теле ответа будет возвращена Web-форма авторизации или перенаправление на неё. Введено в черновике стандарта RFC 6585.

520 Unknown Error, возникает когда сервер CDN не смог обработать ошибку веб-сервера; нестандартный код CloudFlare,

521 Web Server Is Down, возникает когда подключения CDN отклоняются веб-сервером; нестандартный код CloudFlare.

522 Connection Timed Out, возникает когда CDN не удалось подключиться к веб-серверу; нестандартный код CloudFlare.

523 Origin Is Unreachable, возникает когда веб-сервер недостижим; нестандартный код CloudFlare.

524 A Timeout Occurred, возникает при истечении таймаута подключения между сервером CDN и веб-сервером; нестандартный код CloudFlare.

525 SSL Handshake Failed, возникает при ошибке рукопожатия SSL между сервером CDN и веб-сервером; нестандартный код CloudFlare.

526 Invalid SSL Certificate, возникает когда не удаётся подтвердить сертификат шифрования веб-сервера; нестандартный код CloudFlare.

Закрепление материала

- Что такое Java EE?
- В чем разница между Java SE и Java EE?
- Что такое HTTP?
- Какой принцип работы HTTP?
- Какие есть типы запросов HTTP?
- Что такое HTML?
- Что такое CSS?
- Что такое Сервлеты?
- Как работает многопоточность в веб приложениях и сервлетах?

Дополнительное задание

Задание
Реализовать веб страницу HTML с использованием сервлета. Создать папку images загрузить в нее несколько изображений и загрузить с помощью сервлета.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Создать калькулятор. Параметры: число one, число two, operation (add, substract, multiply, divide). Использовать отдельный объект CalcOperations. Сервлет может использовать другие классы как обычный Java класс. Результат должен быть к примеру 2+2=4, 4*3=12.

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Задание 3*
Ознакомится с HTML со справочника http://htmlbook.ru/
Построить веб страницу по схеме:

Задание 4*
Ознакомится с CSS со справочника http://htmlbook.ru/
Добавить стили к предыдущему проекту:

Рекомендуемые ресурсы

Самоучитель по HTML & CSS
http://htmlbook.ru/

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com