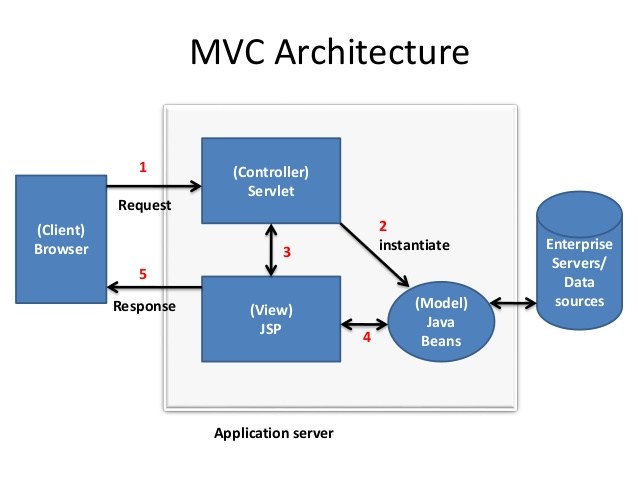
**Servlet guide**

**https://proselyte.net/tutorials/servlets/**

Сервлеты – это Java программы, который работают на сервере и являются обработчиком запросов, который приходят от HTTP клиента и Java приложением (HTTP сервер), либо базой данных (далее – БД).

 Общая архитектура Java приложения, которое использует сервлеты:



От HTTP клиента приходит запрос. Данный запрос обрабатывается сервлетом, далее – идёт работа классов, которые связаны с БД. Если запрос был корректным, то в результате сервер возвращает ответ.

Сервлеты выполняют следующие задачи:

* Чтение и анализ HTTP запроса от клиента
* Обработка данных
* Передача данных HTTP клиенту
* Передача HTTP ответа клиенту

Java классы, которые представляют собой сервлеты, находятся в пакетах javax.servlet и javax.servlet.http.

**Жизненный цикл:**

Жизненный цикл любого сервлета состоит из 4 этапов:

* Инициализация, которая происходит при вызове метода **init()**
* Обработка клиентского запроса после вызова метода **service()**
* Уничтожение сервлета с помощью вызова метода **destroy()**
* Полное уничтожение сервлета сборщиком мусора Java (Garbage Collector).

Метод **init()**

Данный метод вызывается только один раз при создании сервлета.

Обычно, создание происходит в момент перехода по ссылке (URL) на страницу, работу которой обеспечивает определённый сервлет. Это может быть изменено – и создание может происходить при запуске сервера.

После создания сервлета, каждый запрос от клиента приводит к созданию нового потока, который работает с методами doGet или doPost – в зависимости от самого запроса.

**Метод service()**

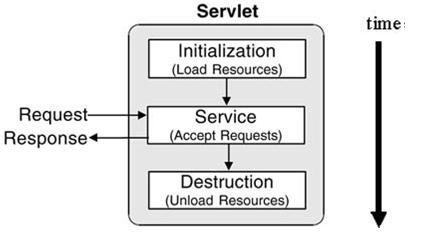
Данный метод является основным  методом, который и содержит саму логику работы сервлета. Он вызывается для обработки полученного запроса и передачи ответа HTTP клиенту.

При получении запроса, сервер создаёт новый поток, который вызывает метод service. Данный метод проверяет тип запроса (POST, GET и т.д.) и вызывает соответствующий метод (doPost, doGet и т.д.)

**Метод destroy()**

Данный метод вызывается после того, как сервлет выполнил свои задачи. Используется, например, для таких задач, как запись данных в cookie, закрытие соединения к базе данных (далее – БД) и т.д.

Схематически, жизненный цикл сервлета можно представить следующим образом:

[](https://i0.wp.com/proselyte.net/wp-content/uploads/2017/03/servletLifeCcycle.jpg)

После получения запроса, для обработки которого необходим определённый сервлет, происходит его инициализация (метод **init()** ). После этого, происходит обработка данного запроса (**service()** ). И, в конце жизненного цикла сервлета происходит вызов метода **destroy()**.

**Пример приложения:** [**https://proselyte.net/tutorials/servlets/simple-application/**](https://proselyte.net/tutorials/servlets/simple-application/)

# HTTP коды:

# Полное руководство <https://proselyte.net/tutorials/http-tutorial/>

HTTP запрос и HTTP ответ имеют следующий формат:

* Срока статуса
* Срока заголовка
* Пустая линия
* Тело сообщения (опционально)

Коды статусов делятся на несколько групп:

|  |  |
| --- | --- |
| № | Код и описание |
| 1 | **1xx: Информационное**  Означает, что запрос был успешно получен и идёт его обработка. |
| 2 | **2xx: Успешное выполнение** Запрос был успешно получен, понят и принят. |
| 3 | **3xx: Перенаправление**  Последующие действия должны быть предприняты для выполнения запроса. |
| 4 | **4xx: Ошибка на стороне клиент**  Запрос содержит синтаксическую ошибку, либо не корректен. |
| 5 | **5xx: Ошибка на стороне сервера** Сервер не может выполнить обработать корректный запрос. |

Список некоторых кодов и их значения:

**1ХХ: Информационные**

|  |  |
| --- | --- |
| Сообщение | Описание |
| 100 Continue | Только часть запроса была получена сервером, но на данный момент запрос не был отклонён. Клиент должен продолжать запрос. |
| 101 Switching Protocols | Сервер выбирает протокол. |

**2ХХ: Успешное выполнение**

|  |  |
| --- | --- |
| Сообщение | Описание |
| 200 OK | Запрос успешен |
| 201 Created | Запрос выполнен, новый ресурс создан. |
| 202 Accepted | Запрос принят, но обработка не завершена. |
| 203 Non-authoritative Information | Информация в заголовке получена из локальной или сторонней копии, а не от оригинального сервера. |
| 204 No Content | Код статуса и заголовок прееданы в ответе, но данные отсутствуют. |
| 205 Reset Content | Браузер должен очистить форму для этой транзакции для дополнительного ввода. |
| 206 Partial Content | Сервер возвращает часть данных. Используется для ответа на запрос, содержащий заголовок Content-Range. |

**3ХХ: Перенаправление**

|  |  |
| --- | --- |
| Сообщение | Описание |
| 300 Multiple Choices | Список ссылок. Пользователь может выбрать ссылку и перейти к локации. Максимальное количество ссылок – 5. |
| 301 Moved Permanently | Запрашиваемая страница была перенесена на новую URL. |
| 302 Found | Запрашиваемая страница была временно перенесена на новую URL. |
| 303 See Other | Запрашиваемая страница может быть найдена по другой URL. |
| 304 Not Modified | Код ответа для заголовка If-Modifier-Since или If-None-Match, где URL не была изменена с крайней даты обновления. |
| 305 Use Proxy | Доступ к запрашиваемой странице должен быть запрошен через прокси, который указан в заголовке Location. |
| 306 *Unused* | Данный код использовался в предыдущей версии, но на данный момент он не используется, а сам код был зарезервирован. |
| 307 Temporary Redirect | Запрашиваемая страница была временно перенесена на новую URL. |

**4ХХ: Ошибка на стороне клиента**

|  |  |
| --- | --- |
| Сообщение | Описание |
| 400 Bad Request | Сервер не понял запрос. |
| 401 Unauthorized | Запрашиваемая страница требует  имя пользователя и пароль. |
| 402 Payment Required | На данный момент клиент не может использовать данный код. |
| 403 Forbidden | Доступ к запрашиваемой странице запрещён. |
| 404 Not Found | Сервер не может найти запрашиваемую страницу. |
| 405 Method Not Allowed | Метод, указанный в запросе, является недопустимым. |
| 406 Not Acceptable | Сервер генерирует ответ, который не может быть принят клиентом. |
| 407 Proxy Authentication Required | Клиент должен авторизироваться с помощью прокси-сервера прежде чем запрос сможеть быть обработан. |
| 408 Request Timeout | Запрос длился дольше, чем сервер был готов ждать. |
| 409 Conflict | Запрос не может быть завершён из-за конфликта. |
| 410 Gone | Запрашиваемая страница больше недоступна. |
| 411 Length Required | Элемент “Content-Length” не определён. Сервер не может принять запрос без данного элемента. |
| 412 Precondition Failed | Сервер признал пред-условия запроса ошибочными. |
| 413 Request Entity Too Large | Сервер не примет данный запрос, так как тело запроса слишком большое. |
| 414 Request-url Too Long | Сервер не примет данный запроса, так как URL слишком большая (длинная). |
| 415 Unsupported Media Type | Сервер не примет данный запрос, так как данный тип медиа-фалйа не поддерживается. |
| 416 Requested Range Not Satisfiable | Запрашиваемый диапазон байтов недоступен и находится за пределами диапазона. |
| 417 Expectation Failed | Ожидания, переданные в заголовке запроса, не могут быть удовлетворены данным сервером. |

**5ХХ: Ошибка на стороне сервера**

|  |  |
| --- | --- |
| Сообщение | Описание |
| 500 Internal Server Error | Запрос не выполнен. Сервер попал в непредвиденные условия. |
| 501 Not Implemented | Запрос не выполнен. Сервер не поддерживает запрашиваемый функционал. |
| 502 Bad Gateway | Запрос не выполнен. Сервер получил недоступный запрос от следущего сервера. |
| 503 Service Unavailable | Запрос не выполнен. На данный момент сервер перегружается, или не работает. |
| 504 Gateway Timeout | Соединение закрыто по времени. |
| 505 HTTP Version Not Supported | Сервер не поддерживает данную версию протокола HTTP. |

Для того, чтобы установить HTTP коды в нашем сервлете у нас есть несколько методов:

|  |  |
| --- | --- |
| № | Метод и его описание |
| 1 | **public void setStatus ( int statusCode )**  Данный метод позволяет нам установить любой код. Принимает целочисленное значение кода статуса в качестве аргумента. |
| 2 | **public void sendRedirect(String url)**  Генерирует ответ 302 с заголовком Location и ссылкой на новый документ. |
| 3 | **public void sendError(int code, String message)**  Отправляет ошибку с коротким сообщением, которое автоматически форматируется внутри HTML документа и отправляется клиенту. |