

Модуль 5. Основы разработки серверной части мобильных приложений

Тема 5.2. Web сервер. HTTP запросы и ОТВЕТЫ

4 часа

Оглавление

5.2. Web сервер. HTTP запросы и ответы	2
5.2.1. Структура HTTP-запроса	2
Структура запроса.....	2
Заголовки запроса	3
*Протокол HTTPS	4
5.2.2. Строка запроса. Методы GET и POST	4
Утилиты telnet, netstat. Сокеты	5
5.2.3. Web сервер	6
Установка Tomcat в ОС Windows	7
Установка Tomcat в ОС Linux (Ubuntu)	9
Фреймворк Spring и Eclipse	9
5.2.4. Ответы сервера.....	10
Упражнение 5.2.1	11
Практикум	12
Создание html страницы.....	12
Создание в Eclipse Spring проекта	12
Сборщик проекта Maven.....	14
Обработка запросов на сервере	15
*Реализация веб сервера на PHP	17
Источники.....	18
Благодарности	18

5.2. Web сервер. HTTP запросы и ответы

Протокол HTTP — сетевой протокол, с начала 90-х и по сей день позволяющего вашему браузеру загружать веб-страницы.

HTTP — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам) [1].

Аббревиатура HTTP расшифровывается как HyperText Transfer Protocol, «протокол передачи гипертекста». В соответствии со спецификацией OSI, HTTP является протоколом прикладного (верхнего, 7-го) уровня. Актуальная на данный момент версия протокола, HTTP 1.1, описана в спецификации RFC 2616.

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует **HTTP-запрос** и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует **ответ** и передаёт его обратно клиенту. До тех пор, пока Клиент не получит ответ, он не может отправить новый запрос. Принцип работы соединения очень похож на лифт, который ездит между клиентом и сервером.

API многих программных продуктов также подразумевает использование HTTP для передачи данных — сами данные при этом могут иметь любой формат, например, XML или JSON.

5.2.1. Структура HTTP-запроса

Как происходит взаимодействие через HTTP-запросы? Каждый запрос порождает многошаговую последовательность:

1. DNS-запрос — поиск ближайшего DNS-сервера, чтобы из URI (например, yandex.ru) получить реальный IP-адрес.
2. Установка соединения с сервером по полученному IP-адресу;
3. Отправка пакетов данных HTTP-запроса;
4. Ожидание ответа, пока пакеты дойдут до сервера, он их обработает и вернет ответ назад;
5. Получение данных.

Первые две операции, как правило, занимают больше всего времени.

Структура запроса

Каждый HTTP-запрос состоит из трёх частей, которые передаются в указанном порядке:

1. **строка запроса** — указан метод запроса (HTTP-метод), URI, версия протокола;
2. **заголовки** — характеризуют тело сообщения, параметры передачи и прочие сведения;
3. **тело сообщения** — данные сообщения.

HTTP спецификация определяет более строгое описание структуры запроса или ответа:

```
message = <start-line>
        *(<message-header>)
        CRLF
        [<message-body>]

<start-line> = Request-Line | Status-Line

<message-header> = Field-Name ':' Field-Value
```

CRLF - обязательная “новая” строка между секцией заголовка и телом .

Заголовок (**message-header**) и тело (**message-body**) запроса может отсутствовать, но строка запроса (**start-line**) есть всегда.

Заголовки запроса

Заголовки запроса задают его параметры. В зависимости от назначения заголовки HTTP запросов разделяют на 4 группы:

- general
- request specific
- response specific
- entity

Из General заголовков (применяются как для запроса, так и ответа сервера) наиболее распространенные:

- **Date** указывает дату запроса, пример:
Date: Fri, 29 Jun 2016 09:20:45 GMT
- **MIME-version** указывает версию MIME (по умолчанию 1.0), пример:
MIME-version: 1.0
- **Pragma** содержит указания для таких промежуточных агентов как прокси и шлюзы, пример:
Pragma: no-cache

Request заголовки:

- Authorization содержит информацию об аутентификации.
- From - поле, в котором браузер может посылать полный E-mail адрес пользователя серверу.
From: info@myitschool.ru
- User-Agent указывает наименование и версию браузера, ОС. Например:
User-Agent: Mozilla/3.0

Полную справку по всем видам заголовков можно легко найти в интернете.

*Протокол HTTPS

Сам по себе протокол HTTP не предполагает использование шифрования для передачи информации. Тем не менее, для HTTP есть распространённое расширение, которое реализует упаковку передаваемых данных в криптографический протокол SSL или TLS.

Название этого расширения — HTTPS (HyperText Transfer Protocol Secure). Для HTTPS-соединений обычно используется TCP-порт 443. HTTPS широко используется для защиты информации от перехвата, а также, как правило, обеспечивает защиту от атак вида man-in-the-middle — в том случае, если сертификат проверяется на клиенте, и при этом приватный ключ сертификата не был скомпрометирован, пользователь не подтверждал использование неподписанного сертификата, и на компьютере пользователя не были внедрены сертификаты центра сертификации злоумышленника. На данный момент HTTPS поддерживается всеми популярными веб-браузерами [2].

5.2.2. Строка запроса. Методы GET и POST

Вернемся к обязательной части HTTP-запроса - строке запроса. В ней через пробел указывается метод, URI (чаще всего URL необходимого ресурса) и протокол. Например так:



HTTP метод представляет собой последовательность из любых символов, кроме управляющих и разделителей, и определяет операцию, которую нужно осуществить с указанным ресурсом. [1]

Перечислим наиболее популярные методы, которые многие фреймворки и инструменты явно реализуют [3]:

- **GET** - получает существующий ресурс. При этом URI содержит всю необходимую информацию, чтобы найти и вернуть ресурс.
- **POST** - создать новый ресурс. При этом POST запрос обычно содержит всю необходимую информацию для создания нового ресурса.
- **PUT** - обновить существующий ресурс. При этом PUT запрос содержит все необходимые данные для обновления ресурса.
- **DELETE** - удалить существующий ресурс.

Рассмотрим два наиболее распространенных метода более подробно.

- **Метод GET** предназначен для получения требуемой информации и передачи данных в адресной строке. Пары «имя=значение» присоединяются в этом случае к адресу после вопросительного знака и разделяются между собой амперсандом (символ &). Удобство использования метода get заключается в том, что адрес со всеми параметрами можно использовать неоднократно, сохранив его, например, в закладки браузера, а также менять значения параметров прямо в адресной строке.
- **Метод POST** посылает на сервер данные в запросе браузера. Это позволяет отправлять большее количество данных, чем доступно методу get, поскольку у него установлено

ограничение в 4 Кб. Большие объемы данных используются в форумах, почтовых службах, заполнении базы данных, при пересылке файлов и др.

Как следствие из выше сказанного между POST и GET следующие различия:

Характеристика	GET	POST
Тело запроса	отсутствует	содержит данные
Максимальный объём	255 символов	8 Кб
Кэширование	Да	Нет

Если пользователь хочет сохранить сгенерированную страницу в закладки, то генерация должна происходить путём GET-запроса, иначе добавить страницу в закладки не получится.

Пример GET-запроса:

GET /path/resource?param1=value1¶m2=value2 HTTP/1.1

На рисунке разбор примера адресной строки в результате GET запроса [3]:



Отсюда вполне понятно, почему при передаче логина и пароля нельзя ставить метод GET! Иначе после нажатия кнопки "Submit" на форме в браузере в адресной строке может появиться что-то наподобии этого: "`http://myitschool.ru/login.php?log=User&pass=123456`", - и пароль виден всем, чего, разумеется, допускать нельзя. Поэтому здесь надо использовать метод POST.

Утилиты telnet, netstat. Сокеты

Самый простой способ разобраться с протоколом HTTP — это попробовать обратиться к какому-нибудь веб-ресурсу вручную. Для этого воспользуемся любой подходящей утилитой командной строки. Например, telnet. В ОС Windows, возможно, придется его установить. Как это сделать описано здесь <http://windows.microsoft.com/ru-ru/windows/telnet-faq>

ОС Windows

```
Нажмите WIN+R ->
cmd <ENTER> telnet <ENTER>
open myitschool.ru 80 <ENTER>
Мы подключились по 80 порту к хосту example.com и теперь
сервер ожидает от нас запрос, введите следующее:
GET / HTTP/1.1
host: myitschool.ru
Если вы выполнили все верно, то будет выведен ответ
сервера в виде html кода странички.
Для получения справки по всем ключам команды:
telnet /?
```

ОС Linux

```
Откройте любой терминал ->
telnet myitschool.ru 80
```

Соединение между нами и сервером по протоколу TCP невозможно без сокетов.

Сокет - это программно реализованный интерфейс для обеспечения обмена данными между процессами.

Воспользуемся командой **netstat**, чтобы увидеть текущие TCP/IP соединения. Наберем в командной строке Windows:

```
netstat -f -o
```

Для ОС Linux аналогичная команда:

```
netstat -p | grep tcp
```

Будет выведен список соединений, причем, в столбце Local Address будет указан ваш IP адрес и через двоеточие программный порт, выделенный ОС, с которым связан идентификатор работающего программного процесса (последний столбец).

TCP	192.168.1.36:61421	65.55.2.82:https	ESTABLISHED	5464
TCP	192.168.1.36:61422	kiks.yandex.ru:https	ESTABLISHED	1344
TCP	192.168.1.36:61423	kiks.yandex.ru:https	ESTABLISHED	1344
TCP	192.168.1.36:61426	pass.yandex.ru:https	ESTABLISHED	1344
TCP	192.168.1.36:61428	awaps.yandex.ru:https	ESTABLISHED	1344

Эта пара IP-адреса и порт задают клиентский сокет. Когда с сервера на указанный порт приходит ответ, операционная система ищет связанный с ним сокет и помещает в него данные.

На сервере сокет работает по другому механизму. Приложение имеет возможность создать “слушающий” сокет и, привязав его к порту ОС сервера, получать адресованные ему пакеты данных.

5.2.3. Web сервер

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-поток или другими данными.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, 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-поток или другие данные, которые необходимы клиенту. В ответ веб-сервер передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP.

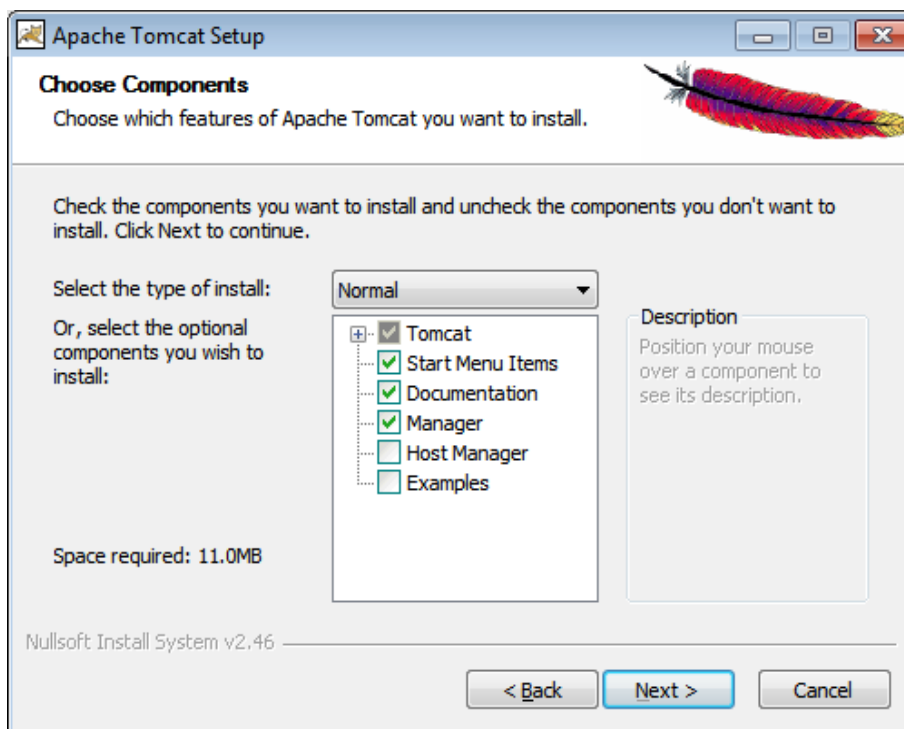
Существует множество веб-серверов, написанных на разных языках. Самыми распространёнными являются Apache и nginx. Так же существует сервера поддерживающие сервлеты написанные на языке Java. Таким решением является Tomcat.

Tomcat (в старых версиях — Catalina) — контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java.

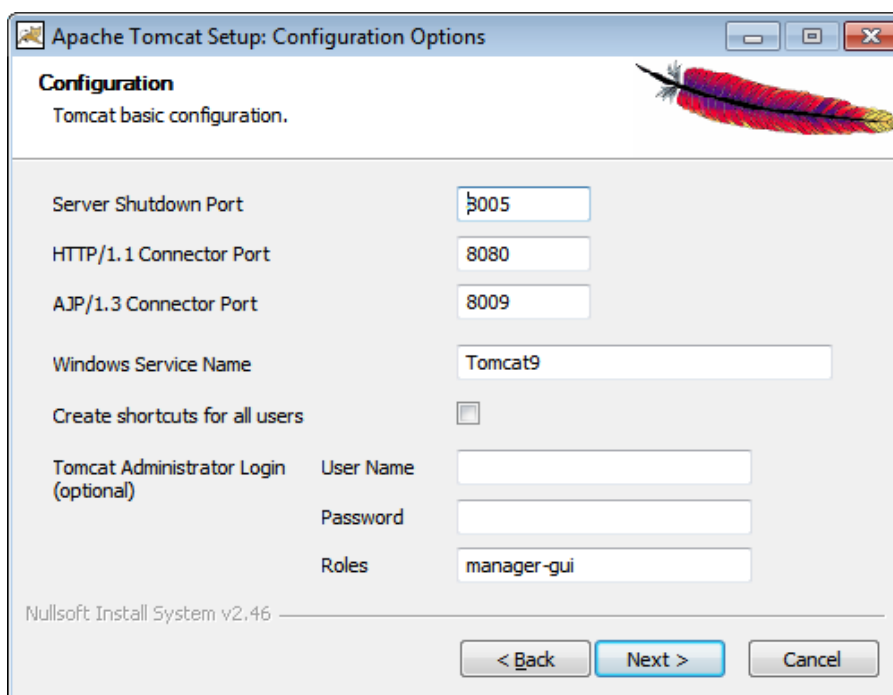
Установка Tomcat в ОС Windows

Чтобы установить в ОС Windows нужно зайти на сайт <http://tomcat.apache.org/> из раздела Downloads выбрать последнюю актуальную версию (на данный момент это Tomcat 9.0) нажать на ссылку **32-bit/64-bit Windows Service Installer** в разделе Binary Distributions.

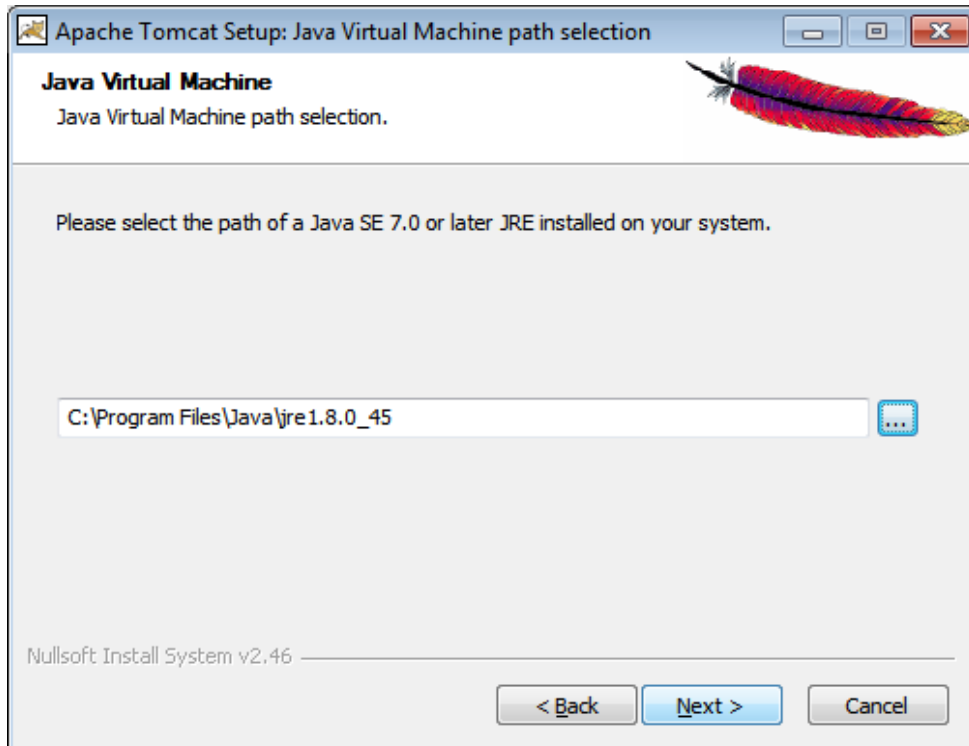
Запускаем скачанный установщик. На приветственной странице, следуя инструкциям нажимаем на кнопку Next, в следующем окне принимаем лицензионное соглашение I Agree. В третьем окне выбираем элементы:



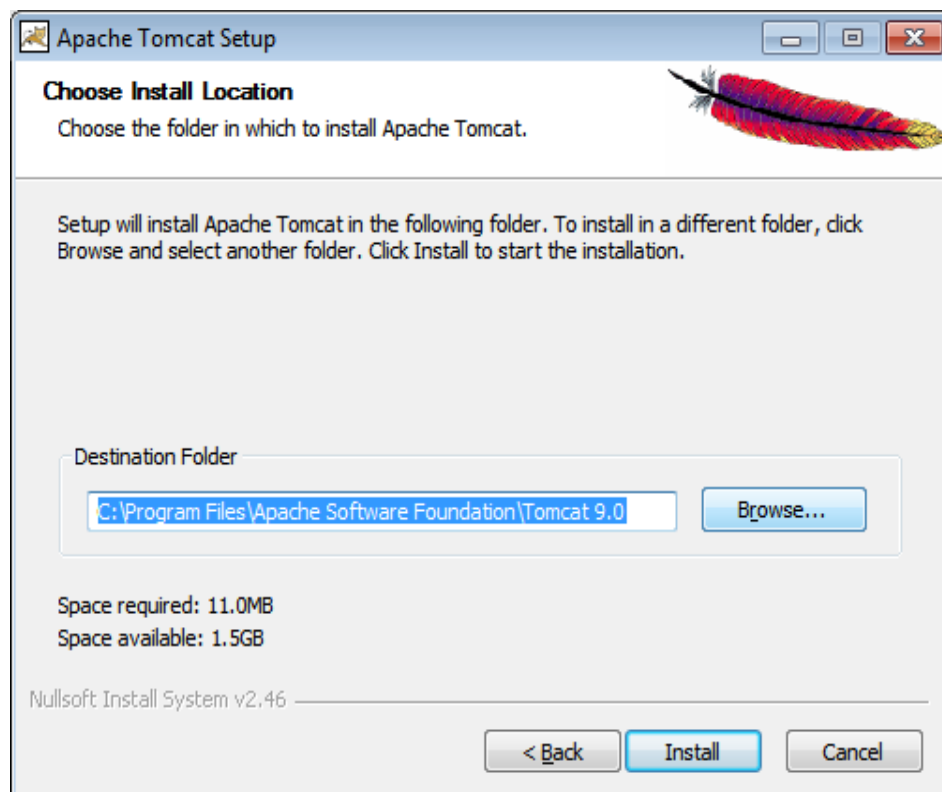
В большинстве случаев можно просто оставить настройки по умолчанию:



Просит нас указать путь к установленному в системе JRE:



И наконец, папку установки:



В последнем окне выбираем опцию Run Apache Tomcat и нажимаем Finish, ждем конца процесса установки. И по окончании установки в трее появится иконка Tomcat.

Установка Tomcat в ОС Linux (Ubuntu)

Чтобы установить Tomcat под Linux нужно запустить терминал и набрать команду:

```
sudo apt-get install tomcat7
```

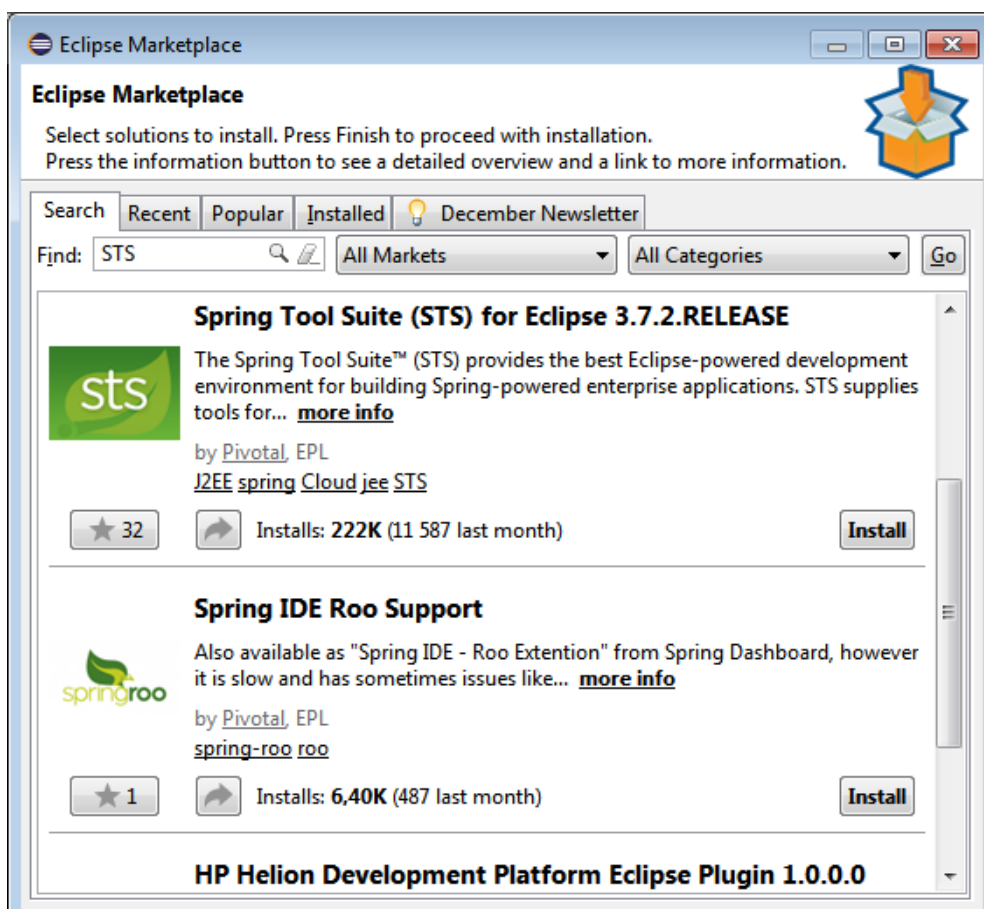
Далее в браузере в адресной строке запускаем <http://localhost:8080> и видим страницу, которую выдает вам на ваш запрос сервер Tomcat.

Фреймворк Spring и Eclipse

Здесь мы не будем подробно останавливаться на преимуществах этого фреймворка, которые признают множество разработчиков. Tomcat так же входит в Java-фреймворк Spring. Подробнее о проекте можно узнать на его сайте <http://spring-projects.ru>.

Для разработки Web приложений используется Spring MVC. Однако для того, чтобы развернуть такое приложение от программиста требуется написать конфигурационный код, причем, как правило для типовых проектов он одинаковы. Поэтому у начинающих разработчиков в особенности пользуется популярностью инструмент Spring Boot, который автоматизирует и тем самым облегчает этот процесс.

Spring Boot в Eclipse можно установить с помощью плагина Spring Tools Suite (STS), который легко подключается через Help -> Eclipse Marketplace. Набираем в строке поиска STS. Жмём Install на найденном плагине STS (первый на рис.).



Дожидаемся поиска всех компонентов в окне Confirm Selected Futures, выбираем все, нажимаем Confirm. В следующем окне соглашаемся с лицензионным соглашением и нажимаем Finish.

Теперь мы готовы к разработке простого клиент-серверного приложения на базе Spring Boot.

5.2.4. Ответы сервера

С помощью URL и метода клиент может инициировать запрос, на который сервер отвечает определенным сообщением + кодом состояния.

Этот код помогает клиенту понять как именно интерпретировать ответ сервера, а также предусмотреть обработку нестандартных ситуаций.

Спецификация HTTP определяет трехзначные коды, описывающие состояния HTTP-ответа, которое получено от сервера. Причем они объединены в 5 групп, начинающихся с 1 до 5. В таблице ниже приведены наиболее часто используемые коды.

Код	Описание кода состояния HTTP
1xx	Информационные коды Этот класс состояний был добавлен в HTTP/1.1 и носит условный характер. Например, сервер может послать заголовок Expect: 100-continue, что будет означать для клиента сигнал к продолжению отправки оставшейся части запроса. Если запрос отправлен полностью, то клиент проигнорирует этот заголовок. HTTP/1.0-клиенты проигнорируют его в любом случае.
2xx	Успешное выполнение запроса Эта группа состояний говорит клиенту, что все прошло хорошо и запрос успешно обработан. Чаще всего встречается код 200 OK. В случае GET-метода вместе с таким кодом в теле HTTP ответа отправляется запрашиваемый ресурс.
200	Запрос был обработан успешно
201	Объект создан
202	Информация принята
203	Информация, которая не заслуживает доверия
204	Нет содержимого
205	Сбросить содержимое
206	Частичное содержимое (например, при "докачке" файлов)
3xx	Перенаправление (чтобы выполнить запрос, нужны какие-либо действия) Коды этого диапазона означают, что клиенту необходимо сделать другой запрос, чтобы получить требуемые ресурсы. Наиболее частый сценарий – запрос на другой URL
300	Несколько вариантов на выбор
301	Ресурс перемещен на постоянной основе
302	Ресурс перемещен временно
303	Смотрите другой ресурс
304	Содержимое не изменилось
305	Используйте прокси-сервер
4xx	Проблема связана не с сервером, а с запросом Эти коды используются, когда сервер полагает, что в результате запроса клиент допустил ошибку, например, запрашивает несуществующий ресурс. Наиболее распространенный код – 404. С ним сталкивались, пожалуй, все. Он говорит клиенту, что запрошенный ресурс не существует на сервере.
400	Некорректный запрос
401	Нет разрешения на просмотр ресурса
402	Требуется оплата

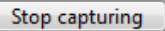
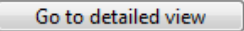
403	Доступ запрещен
404	Ресурс не найден
405	Недопустимый метод
406	Неприемлемый запрос
407	Необходима регистрация на прокси-сервере
408	Время обработки запроса истекло
409	Конфликт
410	Ресурса больше нет
411	Необходимо указать длину
412	Не выполнено предварительное условие
413	Запрашиваемый элемент слишком велик
414	Идентификатор ресурса (URI) слишком длинный
415	Неподдерживаемый тип ресурса
5xx	Ошибки на сервере Этот класс состояний говорит о серверной ошибке в процессе обработки клиентского запроса. Чаще всего можно встречается ошибка 500
500	Внутренняя ошибка сервера
501	Функция не реализована
502	Дефект шлюза
503	Служба недоступна
504	Время прохождения через шлюз истекло
505	Неподдерживаемая версия HTTP

Упражнение 5.2.1

Познакомимся с тем, как внутри устроены запросы и ответы к серверу.

Открыть страницу ya.ru (или любую другую на выбор). Открыть в настройках браузера “Developer Tools” (ctrl+shift+i в Chrome, F12 в IE). Перейти в вкладку “Network” (в IE еще необходимо нажать кнопку Start Capture на панели). Обновите текущую страницу, чтобы появилась информация.

Будет выведена табличка, в которой каждая строчка это некоторый запрос к серверу. В современном мире даже самая простая страница отправляет десяток запросов к серверу:

File Find Disable View Images Cache Tools Validate Browser Mode: IE9 Document Mode: IE9 standards							
HTML CSS Console Script Profiler Network							
 							
URL	Method	Result	Type	Received	Taken	Initiator	Timings
https://ya.ru/	GET	200	text/html	12,96 KB	250 ms	refresh	
https://yastatic.net/www/_t/f/LT2KjI4uj...	GET	304	text/css	302 B	< 1 ms	<link rel="style...	
https://yastatic.net/www/_j/k/kNvXxWXf...	GET	304	text/css	301 B	< 1 ms	<link rel="style...	
https://yastatic.net/jquery/1.8.3/jquery....	GET	304	application/x-java...	311 B	< 1 ms	<script>	
https://yastatic.net/www/2.666/white/pa...	GET	304	application/x-java...	319 B	< 1 ms	<script>	
https://mc.yandex.ru/metrika/watch.js	GET	304	application/x-java...	254 B	< 1 ms	<script>	
https://yastatic.net/www/_c/Q/WXXr7gj...	GET	304	image/svg+xml	307 B	< 1 ms	background-image	
https://yastatic.net/islands/_K4e4uv8u9I...	GET	304	image/svg+xml	280 B	< 1 ms	background-image	

Кликнуть на ya.ru для просмотра основного запроса инициализированного вводом ссылки в браузер. Щелкнув на каждый запрос можно увидеть более детальную информацию.

Практикум

1. Создадим простейшую html форму, которая с помощью браузера отправляет значения двух полей *Имя* и *Фамилия* на сервер по кнопке *Отправить*. Например:

Фамилия:

Имя:

2. Сервер в ответ должен вернуть в браузер одну строку, сформированную из исходных полей, согласно примеру это должно быть “Иванов П.”
3. Эта строка должна появиться на нашей странице.

Создание html страницы

Выполним первый пункт. Напишем html код простейшей формы и сохраним в файл 5.2.2.html:

```
<html>
<form name='test' method='post' action='http://localhost:8080/'>
  <p><b>Фамилия:</b><br>
  <input type='text' name='lastname' size='40'>
</p>

  <p><b>Имя:</b><br>
  <input type='text' name='firstname' size='40'>
</p>

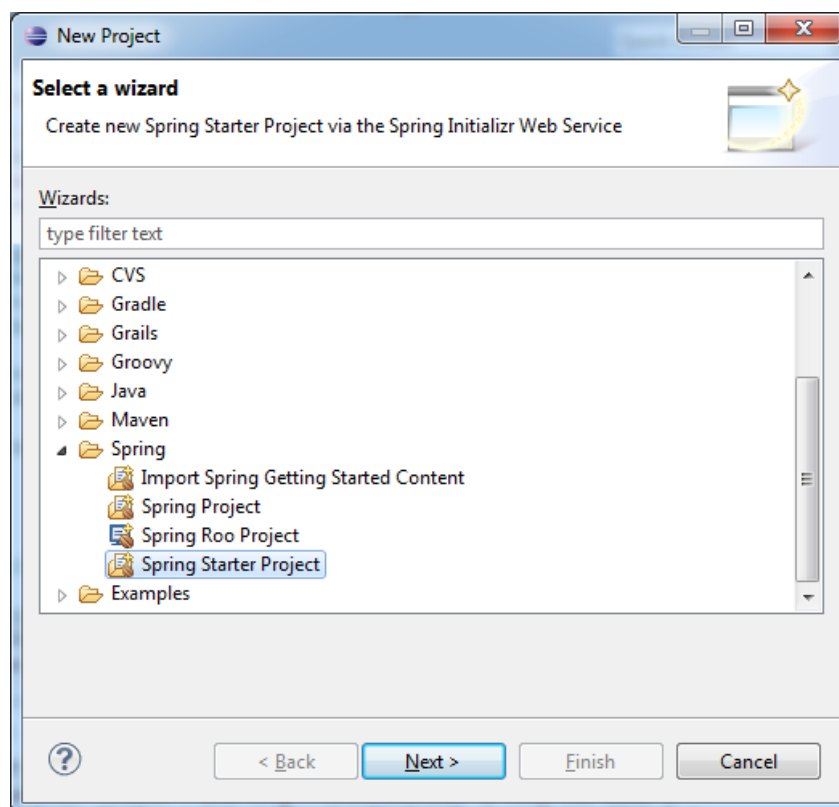
  <p><input type='submit' value='Отправить'>
</p>
</form>
</html>
```

Мы не будем останавливаться на разборе этого кода, полагая, что языком html мы владеем. Если необходимо освежить знания о нем, то можно посмотреть значение инструкций языка здесь [4].

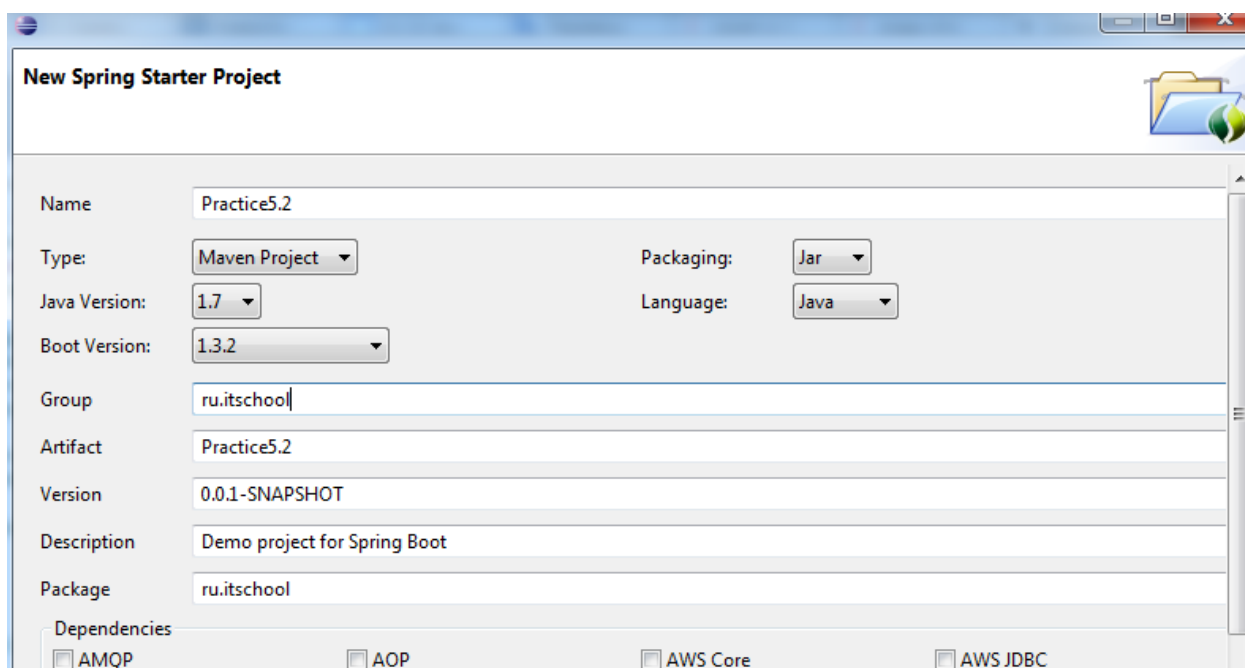
В результате нажатия на кнопку Отправить произойдет переход на страницу localhost, на которой затем и будет выведен результат работы серверной части приложения.

Создание в Eclipse Spring проекта

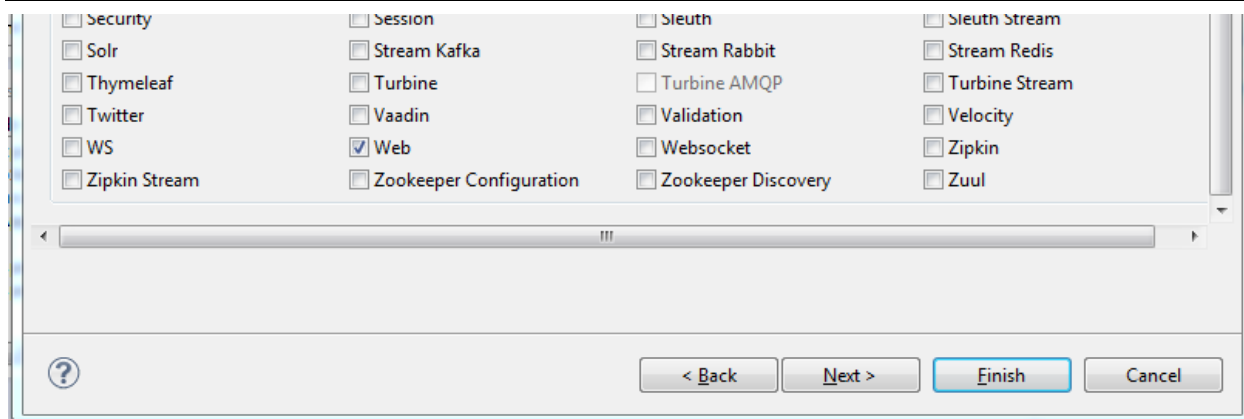
Для выполнения пункта 2 задания запускаем Eclipse (STS плагин должен быть установлен) и создаем новый проект Spring, воспользовавшись мастером:



Далее заполняем необходимые поля для создания нашего проекта, который назовем Practice5.2:



В нижней части окна Dependencies необходимо отметить галкой Web, чтобы в проекте появились необходимые зависимости, подключающие библиотеки для обработки POST и GET запросов.



Нажав Finish получаем подготовленный к дальнейшей разработке проект.

Сборщик проекта Maven

Обратите внимание, что при создании проекта в Type мы указали Maven. Что это такое?

Maven - это один из самых распространенных инструментов автоматической сборки Java проекта. Maven входит в набор решений, который поддерживает Apache Foundation.

Чем он интересен для нашей задачи? Maven помимо компилирования, тестирования и пакетирования позволяет использовать Java сервер Tomcat посредством плагина tomcat-maven-plugin.

Вся структура проекта описывается в файле **pom.xml** (POM – Project Object Model), который можно найти в корневой папке проекта.

Рассмотрим ключевые понятия Maven [5]:

Артефакт — это, по сути, любая библиотека, хранящаяся в репозитории. Это может быть какая-то зависимость или плагин.

Архетип — это некая стандартная компоновка файлов и каталогов в проектах различного рода (веб, swing-проекты и прочие). Другими словами, Maven знает, как обычно строятся проекты и в соответствии с архетипом создает структуру каталогов.

Зависимости (dependencies)— это те библиотеки, которые непосредственно используются в вашем проекте для компиляции кода или его тестирования.

Изучим pom.xml нашего проекта:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>ru.itschool</groupId>
  <artifactId>Practice5.2</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Practice5.2</name>
  <description>Demo project for Spring Boot</description>
```

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.3.2.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>

<properties>
  <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
  <java.version>1.7</java.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

В разделе зависимостей мы видим подключение артефактов spring-boot-starter-web и spring-boot-starter-test, необходимых для работы нашего приложения.

Обработка запросов на сервере

В приложении Maven уже сгенерировал главный класс Application с кодом запуска сервера:

```
package ru.itschool;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Аннотация `@SpringBootApplication` позволяет Spring Boot сообщить о необходимости выполнения ряда действий для генерации другого кода, конфигураций и т. д.

Для запуска Spring приложения в методе `main` используется метод `SpringApplication.run()`, который автоматически формирует web-страницу из имеющихся в приложении классов:

```
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        ApplicationContext ctx = SpringApplication.run(Application.class,
args);
    }
}
```

Реализация страницы ввода ФИ и ответа сервера:

```
package ru.samsung.itschool;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HttpControllerREST extends HttpServlet {

    @RequestMapping("/")
    public String index(HttpServletRequest request, HttpServletResponse
response) {
        if (request.getParameter("lastname") != null ||
request.getParameter("firstname") != null) {
            if (!request.getParameter("lastname").equals("")
&& !request.getParameter("firstname").equals("")) {
                String lastname = request.getParameter("lastname");
                String firstname = request.getParameter("firstname");
                firstname = firstname.substring(0, 1); //Первая буква
                return lastname + " " + firstname + ".";
            }
            else
                return "No POST data lastname or firstname";
        }
        return "Not POST data ";
    }
}
```

Флаг `@RestController` указывает на то, что класс готов принимать web запросы используя Spring MVC. `@RequestMapping("/")` указывает по какому адресу будет доступна страница. Т.к. мы запускаем проект на своём компьютере, то Tomcat будет доступен по адресу localhost на порту 8080.

Запускаем сервер - в появившемся диалоге "Run As" выбираем SpringBootApplication.

*Реализация веб сервера на PHP

PHP (англ. PHP: Hypertext Preprocessor — «PHP: препроцессор гипертекста»; произносится пи-эйч-пи) — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов [6].

PHP — язык программирования, исполняемый на стороне веб-сервера, спроектированный Расмусом Лерддорфом (Rasmus Lerdorf). Язык оказался достаточно гибким и мощным, поэтому приобрёл большую популярность и используется в проектах любого масштаба: от простого блога до крупнейших веб-приложений в Интернете.

Он получил широкое распространение, благодаря множеству преимуществ, начиная с того, что является свободно распространяемым под особой лицензией (PHP license), легок в освоении, имеет развитую поддержку баз данных, библиотеки и расширения языка, может быть развёрнут почти на любом сервере, на широком спектре аппаратных платформ и операционных систем.

Среди недостатков PHP разработчики отмечают несогласованность его синтаксиса, он не подходит для создания десктопных приложений или системных компонентов и, самое главное, сложилось мнение, что веб-приложения, написанные на PHP, зачастую имеют проблемы с безопасностью. Поэтому во многих крупных компаниях, банках существует политика запрета на использование PHP.

Однако, это не мешает его широчайшему распространению на миллионах веб-серверов. К крупнейшим сайтам, использующим PHP, относятся Facebook, Wikipedia и др.

Наиболее популярный PHP сервер - это HTTP сервер **Apache**. Apache - свободно распространяемый кроссплатформенный сервер, полностью разрабатываемый Apache Software Foundation. Его основными достоинствами считаются надёжность и гибкость конфигурации, поддерживает IPv6. Помимо PHP Apache поддерживает языки программирования: Python, Ruby, Perl, ASP, Tcl. По некоторым данным Apache установлен на более чем 60% серверах в мире.

Как установить Apache и PHP5?

В ОС Windows 7 этот процесс подробно описан в статье [здесь: http://www.q2w3.ru/2011/01/12/3093/](http://www.q2w3.ru/2011/01/12/3093/)

Для установки Apache и PHP5 в ОС Linux (Ubuntu) можно воспользоваться руководством отсюда: <http://help.ubuntu.ru/wiki/apachemysqlphp>

Функциональность, аналогичная ранее рассмотренной на Java сервере Tomcat, может быть реализована с помощью кода:

```
<?php
if (isset($_POST["lastname"]) || isset($_POST["firstname"]))
    if ($_POST["lastname"] != "" && $_POST["firstname"] != "") {
        $lastname = $_POST["lastname"];
        $firstname = $_POST["firstname"];
        $firstname = iconv('UTF-8', 'windows-1251', $firstname); // перевод
        кодировки нужен для того, чтобы корректно русские буквы отображались
        $firstname = substr($firstname, 0, 1);
        $firstname = iconv('windows-1251', 'UTF-8', $firstname); // перевод
        кодировки нужен для того, чтобы корректно русские буквы отображались
        echo $lastname . " " . $firstname . ".";
```

```
} else
    echo "No POST data lastname or firstname";
else
    echo "Not POST data";

?>
```

Источники

- [1] Простым языком об HTTP [Электронный ресурс]/ Habrahabr.— Режим доступа: <http://habrahabr.ru/post/215117/> – Загл. с экрана. – (Дата обращения: 03.02.2016).
- [2] Протокол HTTP/HTTPS [Электронный ресурс]/ НОУ Интуит.— Режим доступа: <http://www.intuit.ru/studies/courses/4455/712/lecture/21291?page=2> – Загл. с экрана. – (Дата обращения: 03.02.2016).
- [3] Протокол HTTP: все что нужно для веб-разработки. Часть 1 [Электронный ресурс] /l-paper.ru: Бложек о вебдеве и не только. – Режим доступа: <http://aabramoff.ru/protokol-http-vse-chto-nuzhno-dlya-veb-razrabotki-chast-1/> – Загл. с экрана. – (Дата обращения: 03.02.2016).
- [4] Справочник по HTML [Электронный ресурс] – Режим доступа: <http://htmlbook.ru/html> – Загл. с экрана. – (Дата обращения: 03.02.2016).
- [5] Ермолов И. Apache Maven — основы [Электронный ресурс]/ Habrahabr.— Режим доступа: <http://habrahabr.ru/post/77382/>– Загл. с экрана. – (Дата обращения: 03.02.2016).
- [6] PHP // Википедия. [2016—2016]. Дата обновления: 28.01.2016. URL: <http://ru.wikipedia.org/?oldid=76073132> (дата обращения: 28.01.2016).

Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателю ИТ ШКОЛЫ SAMSUNG Исаеву Руслану Рамилевичу.