Система META

Аутентификация и авторизация

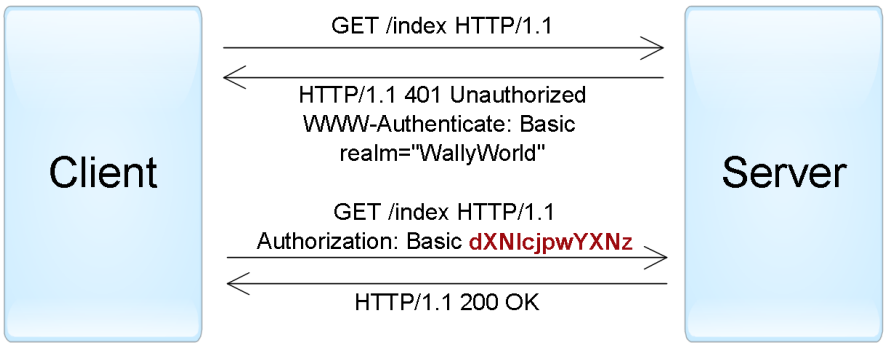
Немного терминологии:

Аутентификация — предоставление доказательств, что вы на самом деле есть тот, кем идентифицировались. Проверка, что вы знаете пароль от этой учетной записи. Для вэб-интерфейсов существует два основных вида аутентификации: это аутентификация, основаная на заголовке Authorization протокола HTTP, и аутентификация на формах, использующая заголовки Cookie протокола HTTP.

Авторизация — проверка, что вам разрешен доступ к запрашиваемому ресурсу.

HTTP authentication

1. Сервер, при обращении неавторизованного клиента к защищенному ресурсу, отсылает HTTP статус “401 Unauthorized” и добавляет заголовок “WWW-Authenticate” с указанием схемы(например Basic или Digest) и параметров аутентификации(например, realm, domain, nonce и прочее).
2. Браузер, при получении такого ответа, автоматически показывает диалог ввода логин и пароль. Пользователь вводит детали своей учетной записи. В диалоге будет написана схема аутентификации, в данном случае WWW-Authenticate : Basic с параметром realm="WallyWorld". Введенные логин:пароль при Basic аутентификации в кодированной строке base64 отправляются на сервер в заголовке Authorization запроса.



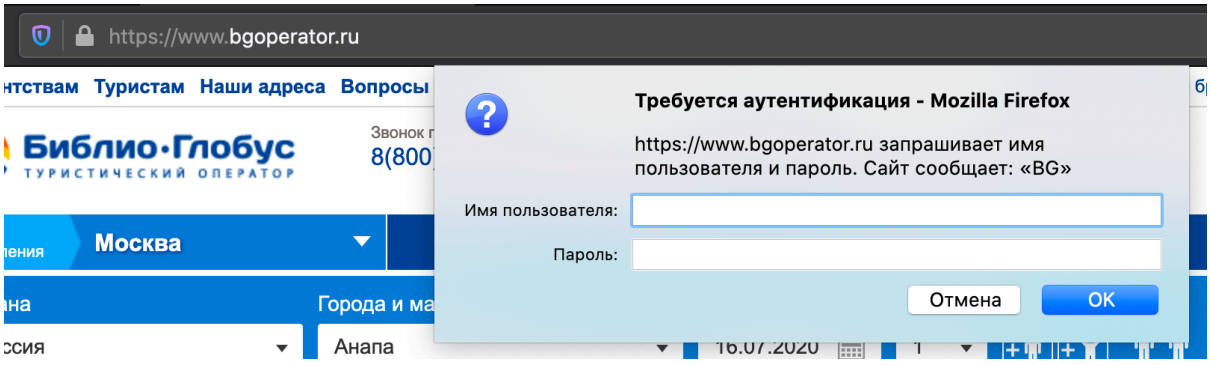
3. В случае успешной авторизации, во всех последующих запросах к этому веб-сайту браузер автоматически добавляет HTTP заголовок “Authorization”, в котором передаются данные пользователя для аутентификации сервером.

4. Сервер аутентифицирует пользователя по данным из этого заголовка. Решение о предоставлении доступа (авторизация) производится отдельно на основании роли пользователя, списка, кому можно заходить на сайт, или других данных учетной записи и зависит от реализации софта.

Применительно к веб-приложениям, существует несколько стандартных схем для аутентификации по паролю.

Basic — самая простая схема, при которой логин и пароль пользователя передаются в заголовке Authorization в незашифрованном виде (base64-encoded). И при использовании https она становится чуть более безопасной.

Попробуйте зайти в личный кабинет на сайте https://www.bgoperator.ru/



Digest — вызов-ответ-схема, при которой сервер посылает уникальное значение nonce, а браузер передает MD5 хэш пароля пользователя, вычисленный с использованием указанного nonce. Немного более безопасная альтернатива Basic схемы при незащищенных соединениях.

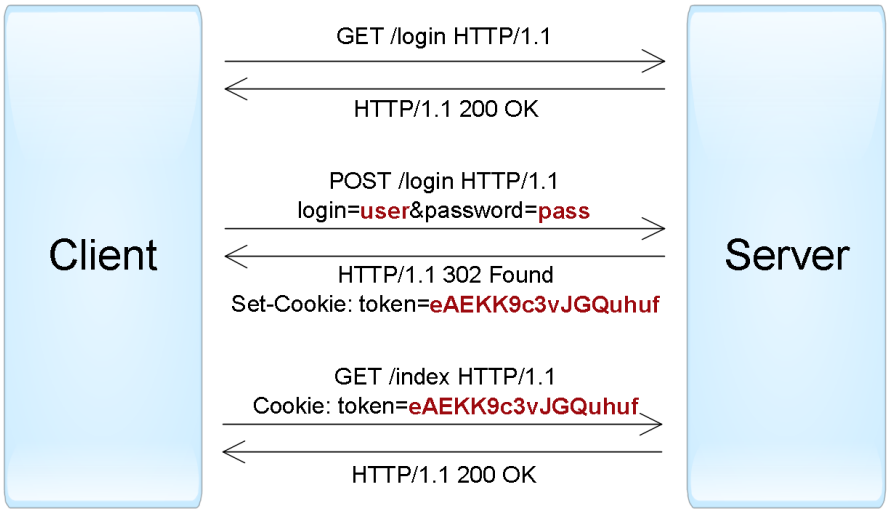
Forms authentication

Самым распространенным и имеющим множество реализаций является метод аутентификации на формах. Он позволяет стилизовать диалог ввода аутентификационных данных, позволяет расширять и усложнять его, использовать совместно с механизмами сессий, заложенных в приложении.

Для этого протокола нет определенного стандарта, поэтому все его реализации специфичны для конкретных систем.

Для аутентификации пользователей в приложении создается html-форма для ввода данных аутентификации (обычно это логин и пароль, но набор полей может отличаться), данные формы отправляются на сервер HTTP-методом POST. Полученные данные проверяются сервером и в случае успешной аутентификации пользователю возвращается ответ, содержащий в заголовках Set-Cookie набор токенов или данных, позволяющих при их правильной передаче браузером пользователя в

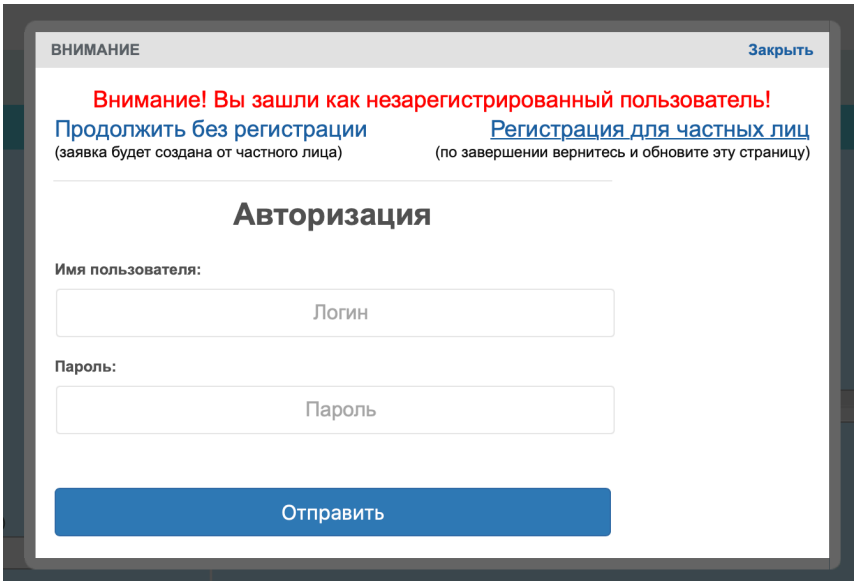
заголовке Cookie считать пользователя аутентифицированым. Cookie - это стандартный заголовок HTTP, поддерживаемый всеми вэб-клиентами, имеющий набор параметров, позволяющий защитить его от перехвата. Процедура работы клиента и сервера с заголовками Cookie/Set-Cookie и количество токенов/данных определяется только реализацией вэб-приложения, это позволяет обезопасить пользователя даже в случае их перехвата.



Пример forms authentication.

Необходимо понимать, что перехват токена при "слабых" реализациях дает аналогичный уровень доступа, что и знание логин/пароль. Но, например, наши сессионные/авторизационные куки не будут работать у другого клиента после перехвата.

Примеры окон http аутентификации в системе БГ, они могут быть с разным набором полей в отличие от стандартного диалога бейсика:



Организация | Organization:

Организация | Organization

Имя пользователя | Login:

Логин | Login

Пароль | Password:

Пароль | Password

Отправить | Send

Авторизация | Authorization

Организация | Organization:

Организация | Organization

Имя пользователя | Login:

Логин | Login

Код проверки | Code:

Код проверки | Code

Отправить | Send

В проекте за аутентификацию отвечает appt.meta3.servlet.AuthServlet (считывает как forms, так и basic авторизацию)

User whoami = AuthServlet.*isAuth*(req, res, mains, "meta");

В функцию передаем:

req - это HttpServletRequest, вся необходимая информация о запросе

res - это HttpServletResponse, здесь можно записать ответ

mains - объект с параметрами настройки меты

"meta" - AuthServlet поддерживает возможность аутентификации пользователя не только по логину и паролю, но еще и по полю организации внутри одного проекта, по умолчанию в туроператоре мы используем "meta" и аутентификацию только по логину и паролю. Количество полей в форме авторизации настраивается при помощи rights.properties, а не в вызове isAuth и обычно в мету используется больше полей + одноразовый пароль на вход, а строка в isAuth позволяет один и тот же софт в одном и том же приложении разделять для авторизации при уровне 1 (3 поля).

В случае успешной аутентификации метод возвращает объект класса appt.meta3.User, у которого есть поля:

int id - это идентификатор аутентифицировавшегося пользователя, id объекта этого пользователя 9006 типа

int status - статус пользователя от 0 до 9

String login - логин пользователя

String mail - почта пользователя

String phone - номер телефона пользователя

Вы уже знакомы с id пользователя - каждый раз, когда редактировали объект в него передавали ваш id пользователя.

Что такое статус необходимо рассмотреть подробнее - статус используется для ограничения/разрешения доступа пользователю к частям системы и данным как в интерфейсе меты, так и внутри приложения. Сейчас у вас на тестовом сервере статус 6. Статусы зависят от реализации софта, расскажу про некоторые статусы в рамках проекта Туроператора:

1 - частник или пользователь партнера (гид)

2 - агентство или менеджер агентства

3 - пользователь партнера с большими правами, чем гид

4 - 9 - сотрудники туроператора

Статус пользователя обычно ограничивает доступ пользователя глобально в системе каким-то большим блоком или блоками, для локальных ограничений/разрешений внутри больших блоков используется более тонкая и широкая система прав, реализация которой так же зависит от реализации приложения. Например, если взять разделение по статусам, частник видит только свои заявки, агентство видит только свои заявки и заявки своих менеджеров, партнер видит только заявки, которые должен обслужить, а пользователь партнера (гид) только заявки партнера для отеля, который обслуживает гид, сотрудник туроператора с низким статусом видит все заявки своего отдела, а с высоким - всех отделов.

Кроме статуса у пользователя есть некий набор прав, по которым можно более тонко настроить уровень доступа. Отображение одной и той же страницы для сотрудников с одним статусом, но с разными правами может отличаться, как пример: сотрудник со статусом 4 из отдела занесения цен будет видеть в интерфейсе функционал добавления и редактирования цен, в то время как сотрудник из отдела работы с партнерами увидит только занесенные данные, и не сможет их отредактировать. А сотрудник из отдела маркетинга вообще не сможет зайти на эту страницу, потому что у него нет прав ни на чтение, ни на запись.

Чтобы проверить наличие права у аутентифицированного пользователя системы необходимо получить его объект. Функция *getUserByComm* вернет объект класса appt.meta3.Obb (в рамках системы Библио-Глобус это может быть пользователь - 111 тип, агентство - 112 тип, контрагент партнера - 317 тип и др.).

```
Obb obUser = PersonalPageServlet.getUserByComm(mains, whoami);
```

mains - объект с параметрами настройки меты

whoami - аутентифицированный объект класса appt.meta3.User (тот, кто зашел на страницу)

У объектов типа 111, 112, 317 и др. есть атрибут, в котором лежат права, соответсвующие этому объекту. Проверить наличие права можно с помощью функции Base.*userHasRole:*

```
boolean hasRoleNewPrice = Base.userHasRole(mains, conname, obUser, "100710000748");
```

Функция возвращает true, если у объекта есть данное право, и false - если нет.

В нее передаем:

mains - объект с параметрами настройки меты

conname - имя коннекшена к базе

obUser - объект типа (111, 112, 317и др.)

"100710000748" - право, наличие которого необходимо проверить у объекта.

Так же существует более сложная расширенная версия метода проверки прав, она была сделана для большей безопасности на критических частях системы. Когда нам

важно дополнительно удостовериться в том, что пользователь не скомпрометирован, если он логинится в систему не из "обычного места", например, не из офиса. Дополнительной проверкой мы заставляем "доказать" пользователя что он - это действительно он, и только тогда разрешаем ему доступ. Это очень актуально, когда сотрудники с высокими статусами и широкими правами работают удаленно, ведь безопасность сети в этом случае никто не может гарантировать.

В коде вы столкнетесь как с простой версией *userHasRole,* так и с расширенной.

Расширенный метод Base.*userHasRole* возвращает значение int от -5 до 4, в зависимости от этого определяется, какую из проверок пользователь прошел или не прошел.

-1 у пользователя нет данного права

-2 IP и User-Agent нет в рабочем списке

-3 IP и User-Agent нет в домашнем списке

-5 неизвестная проблема

0 у пользователя есть данное право, но статус 3 или больше, но проверку на IP и User-Agent пользователь не прошел

1 у пользователя есть данное право и статус меньше 3, проверку на IP и User-Agent пользователь не прошел

2 у пользователя есть данное право, но статус 3 или больше, IP и User-Agent есть в рабочем списке

3 у пользователя есть данное право, но статус 3 или больше, IP и User-Agent есть в домашнем списке

4 проверки на IP и User-Agent не было

```
int hasTitleAddRole = Base.userHasRole(mains, conname, whoami, obUser, "100718258857", req.getHeader("X-Real-IP"), req.getHeader("User-Agent"));
```

mains - объект с параметрами настройки меты

conname - имя коннекшена к базе

whoami - аутентифицированный объект класса appt.meta3.User (тот, кто зашел на страницу)

obUser - объект авторизовавшегося пользователя в системе

"100718258857" - право, наличие которого необходимо проверить у объекта

req.getHeader("X-Real-IP") - проверка IP клиента, что пользователь заходит из разрешенного списка сетей. Данный заголовок специфичен для наших проектов, мы его получаем от системы защиты от ддос-атак, либо эмулируем сами в запросах на бэкенд.

req.getHeader("User-Agent") - в этом заголовке содержится: название и версия браузера, язык, версия операционной системы, программное обеспечение, установленное на используемом устройстве и тип устройства, с которого пользователь зашел на сайт.

Метод Utils.showAuthError вернет текст ошибки, соответствующий статусу (меньше или равно 0) ответа от расширенного Base.*userHasRole* , то есть вход запрещен.

out.println(Utils.*showAuthError*(whoami, hasTitleAddRole);

whoami - аутентифицированный объект класса appt.meta3.User (тот, кто зашел на страницу)

hasTitleAddRole - статус ответа расширенного Base.*userHasRole*

Примеры ответа Utils.*showAuthError*

Login: **E.DENISOVA**
IP нет в списке разрешенных! IP not in the list of allowed! (-3)
Доступ запрещен!
Permission denied!
Email: **support@bgoperator.com**

Login: **E.DENISOVA**
Пользователя нет в списке разрешенных! User not in the list of allowed! (-1)
Доступ запрещен!
Permission denied!
Email: **support@bgoperator.com**

Задания

1. Сделать интерфейс с редактированием объекта Квоты на экскурсии и услуги. Необходима возможность менять дату квоты 1799510027, и количество квот 1799210047. В интерфейсе надо отображать код экскурсии и адрес 1036410028, для которой есть квоты 1799910177 -> 1036423021 + 1036410028. Редактировать поля может только авторизовавшийся пользователь, без авторизации можно только просматривать. (179910003775, 179910003759, 179910003611, 179910003696, 179910003635, 179910003361)
2. Сделать интерфейс занесения стопов на проживание.
   Дата начала действия 1074100142
   Отель 1000538 (ссылка)
   Страна 1000802(ссылка как список)

Дата окончания действия 1074100143
Стоп на квоту 1074200033, 0 - стоп, 1 - стоп на квоту
Только зарегистрированный пользователь может добавлять и удалять стопы.
Стопы можно заносить только на будущие даты.

3. Сделать интерфейс, в котором можно добавить вам право, независимо от того, кто зашел на страницу, есть возможность добавить право только вам, или удалить у вас право. Если у пользователя статус 6 или больше, то он может редактировать ваши права, если статус меньше - не может даже зайти (писать ошибку).
   Права доступа 1001900012
   Название права 1007410000
4. Выведите на экран ваш id пользователя, тип объекта id пользователя, ваш статус, логин, почту, IP и User-Agent
5. Если у пользователя есть право 100727600368, то он может зайти в интерфейс редактирования квот на проживание, но только для просмотрела. Если у пользователя есть право 100710057771, то в интерфейсе он может добавлять квоты, редактировать, так же может удалять. На странице должен быть выбор отеля по городу, потом выбор НСа, и отображение квот на выбранный период.
   Количество квот 1093100638 (число)
   Дата квот 1093100019 (дата)
   Ссылка из квот на объект 21(номер) типа 1093970007
   Название НСа (46 тип) 1000348
   Ссылка из НСа на номер 1000350
   Тип стоимости (0/1/2 - С/БНС/НС) 1046222729
   Название номера (21 тип) 1000168
   Отель (из номера ссылка на отель) 1000169
   Название отеля 1990410000
   Рабочие НСы массивом 1990423125
   Ссылка на город в отеле 1990100059
   Название города 1000098
   Ссылка на страну в городе 1000004
   Название страны 1000000