به نام خدا

در ادامه، پروژه میان‌ترم در چهار فاز آمده است. ضمن این‌که کماکان به موارد ذکر شده در فایل توضیحات تمرین‌ها (موجود در سایت درس) توجه می‌کنید، در این پروژه، برای هر فاز یک پوشه به نام آن فاز بسازید. در هر پوشه، کل فایل‌های مربوط به آن فاز را گذاشته و یک فایل فشرده‌شده در قالب zip مطابق الگوی زیر بارگذاری کنید. دقت کنید که پروژه مربوط به هر فاز باید در درگاه آپلود جداگانه‌ای که برای آن فاز در نظر گرفته شده، بارگذاری شود.

StudentNumber_FirstName_LastName.zip

مثال: 9031066_Ehsan_Edalat.zip

آخرین مهلت برای کسب نمره کامل فاز **اول** تا **ساعت ۲۳:۵۵ روز ۱۶ اردیبهشت ۱۳۹۹** و آخرین مهلت برای کسب نمره کامل فازهای **دوم و سوم** تا **ساعت ۲۳:۵۵ روز ۵ خرداد ۱۳۹۹** خواهد بود. پس از زمان‌های تعیین‌شده برای هر سوال، می‌توانید تمرین خود را با احتساب ۱۵٪ کسر نمره به ازای هر روز تأخیر، تا ساعت ۲۳:۵۵ روز ۲۱ اردیبهشت ۱۳۹۹ برای فاز اول و تا ساعت ۲۳:۵۵ روز ۱۰ خرداد ۱۳۹۹ برای فاز دوم و سوم ارسال کنید. فاز چهارم پروژه که کاملا امتیازی است را می‌توانید تا ساعت ۲۳:۵۵ روز ۱۰ خرداد ۱۳۹۹ ارسال کنید.

* استفاده مناسب از مفاهیم و مطالب تدریس‌شده ضروری است. طراحی خوب و منطقی کلاس‌ها و اینترفیس‌ها باید متناسب با اصول برنامه‌نویسی شی‌گرا باشد. رعایت اصول پنهان‌سازی اطلاعات (information hiding)، سلسله‌مراتب مناسب ارث‌بری جهت استفاده مجدد از کدها (code reusability)، استفاده از چندریختی (polymorphism) و سایر نکات تدریس‌شده الزامی است. سعی کنید پیش از شروع پیاده‌سازی، تعداد و نام کلاس‌ها، فیلدها و متدهای مورد نیاز برنامه را تحلیل کنید و مطابق با تحلیل و طراحی انجام‌شده، برنامه را پیاده‌سازی کنید.

* رعایت اصول کدنویسی خوانا، کامنت‌گذاری و مستندسازی در قالب JavaDoc برای هر ۴ فاز الزامی است.

* برنامه‌ها را پیش از بارگذاری به خوبی تست و اشکال‌زدایی کنید! همه کلاس‌ها و همه متدهای موجود در برنامه‌ها کاملا مورد بررسی و آزمون قرار بگیرند تا از درستی عملکرد برنامه اطمینان حاصل کنید. حالت‌های مختلف ورودی توسط کاربر باید بررسی شود و در صورت لزوم، پیغام خطای مناسب نمایش داده شود.

* در قسمت‌های مختلف پروژه باید خطاهای مختلف بررسی شوند و در قبال آن رفتار مناسبی از برنامه دریافت شود. پس شما باید برای تمامی قسمت‌ها عملیات Exception Handling را برای استثناهای Checked و همچنین در مواقع لزوم Unchecked باید پیاده‌سازی کنید.

* این تمرین، در صورت مساعد بودن شرایط، تحویل حضوری خواهد داشت. تحویل حضوری در اولین جلسه کارگاه‌ها بعد از مهلت نهایی بارگذاری پروژه‌ها برگزار می‌شود و فقط پروژه‌های بارگذاری‌شده در سایت درس تحویل گرفته می‌شوند. برای رعایت مساوات بین دانشجویان گروه‌های مختلف کارگاه، امکان تغییر کدها بعد از بارگذاری تا زمان تحویل حضوری وجود ندارد.

* این تمرین برای یک‌روز کار طراحی نشده است! لطفاً در زمان‌بندی انجام تمرین دقت کافی را داشته باشید.

* در صورتی که کد هر کدام از سوالات را در اینترنت بیابید و از آن استفاده کنید، تقلب محسوب می‌شود و طبق قوانین تقلب با شما رفتار می‌شود.

* تعدادی از دانشجویان ممکن است درگیر بیماری شده و با مشکلاتی روبه‌رو شده باشند، تعدادی هم در خانه سیستم کامپیوتر شخصی ندارند، این دسته از افراد برای هماهنگی حتماً با آی‌دی @deepmine_admin در پیام‌رسان تلگرام مکاتبه کنند. بر اساس شرایط هر فرد برای آنها تصمیم‌گیری می‌شود که چگونه پروژه را تحویل دهند. در صورتی که در پنج روز آینده مکاتبه‌ای صورت نگیرد، فرض بر این است که مشکلی برای انجام پروژه مثل باقی دانشجویان وجود ندارد و دیگر بهانه‌ای پذیرفته نخواهد شد. توجه کنید که برای درستی ادعای این افراد مکانیزم راستی‌آزمایی صورت خواهد گرفت. در صورتی که اثبات شود، فردی صادقانه رفتار نکرده است، متقلب محسوب شده و طبق قوانین تقلب به صورت جدی با او برخورد خواهد شد.

* همان طور که در دستور کار شماره ۲ کارگاه آمده است، شما باید تمامی پروژه‌ها و تمرین‌های خود را در گیت قرار دهید. تاکید می‌کنیم، صرف قرارگیری و آپلود پروژه در گیت کفایت نمی‌کند. شما باید کامیت‌های مناسب و مستمر داشته باشید. **پروژه میانترم هم از این قاعده مستثنی نیست.** برای استفاده از گیت، می‌توانید از امکانات گیت دانشکده استفاده کنید. به یاد داشته باشید ریپازیتوری پروژه خود را در حالت خصوصی قرار دهید

و به استاد کارگاه خود دسترسی بدهید تا بتواند بر روند اجرای پروژه نظارت داشته باشد. بارگذاری یکباره پروژه در سرور گیت پذیرفته نیست و شما باید اصول مربوط به کنترل نسخه[۱] را رعایت نمایید.

* فازهای پروژه طراحی شده، به ترتیب تدریس مطالب آمده‌اند. وزن هر فاز در نمره نهایی به ترتیب ۲۰، ۴۰ و ۴۰ از مجموع ۱۰۰ نمره خواهد بود. در نظر داشته باشید که حداکثر ٪۲۰ نمره امتیازی برای فازهای اول تا سوم در نظر گرفته می‌شود. فاز چهارم به اندازه ۱۵ واحد نمره امتیازی خواهد داشت. در نهایت نمره پروژه میان‌ترم با احتساب نمره‌های امتیازی حداکثر ۱۳۵ از ۱۰۰ خواهد بود. با توجه به نمره هر پروژه و همچنین، حداکثر نمره امتیازی، در انجام پروژه‌ها برنامه‌ریزی کنید.

منتظر پروژه‌های جذاب شما هستیم! ☺

[۱] Version Controlling

تعریف پروژه میان‌ترم

پروژه پیش‌رو شامل ۳ بخش اساسی است که به ترتیب مباحث تدریس شده، تعریف شده است. طبیعتاً «تمام» مباحث این پروژه تا این لحظه به شما تدریس‌نشده است، اما برنامه‌ریزی شده با توجه به برنامه درس، ددلاین‌های فازهای پروژه عملیاتی شوند.

در این تعریف پروژه هر رنگ در متن (و نه در عکس ها) معنی خاص خود را دارد.

رنگ نارنجی تیره به معنی پیاده‌سازی اجباری است.

رنگ سبز به معنی پیاده‌سازی امتیازی است.

رنگ آبی کمرنگ به معنی توضیحاتی برای بهتر انجام دادن پروژه است. مواردی که در فازهای دوم و سوم توضیحات آنها آمده است، عموماً درباره توضیحات پروتکل HTTP هستند.

پروژه شما در نهایت یک HTTP Client خواهد بود که در حالت ایده آل مشابه برنامه‌های Postman و یا Insomnia خواهد بود. در مورد مباحث شبکه و توضیحات مربوط به پروتکل HTTP، در محتوای درس توضیحات کامل گفته خواهد شد. علاوه بر آن، در انتهای این فایل و در قسمت پیاده‌سازی کارکردهای پروژه (فاز دوم و سوم)، توضیحات کامل این قسمت آورده شده است.

در توضیحات پروژه از کلمات «درخواست»، «ریکوئست» و «request» به جای هم استفاده شده است. همچنین، از کلمات «پاسخ» و «جواب» و «ریسپانس» و «response» جای یکدیگر استفاده شده است. دلیل این کار آشنایی شما با تمام این کلمات است.

این پروژه شما ۴ فاز دارد که در زیر هر فاز را توضیح می‌دهیم. فاز ۴ تماما امتیازی است.

فاز اول: پیاده‌سازی رابط کاربری گرافیکی(GUI) نرم‌افزار.

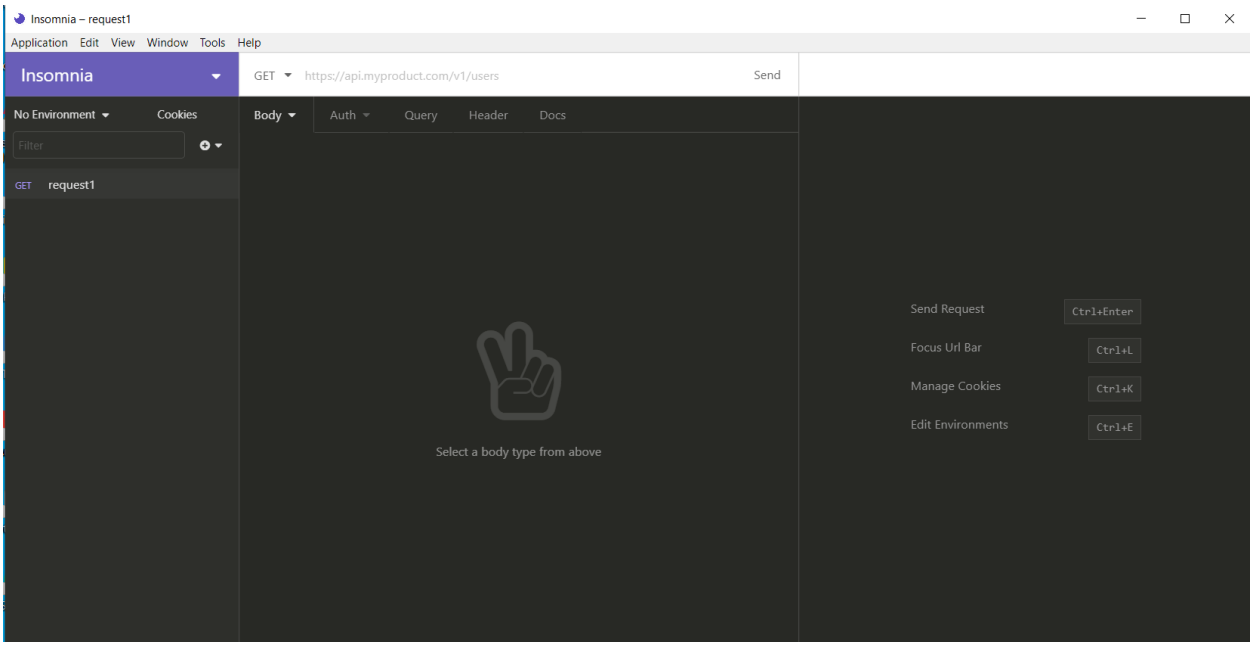
در این فاز شما باید رابط گرافیکی مشابه نرم‌افزار Insomnia داشته باشید. تصاویری که در زیر آمده در حالت «ایده آل» است. فقط اطلاعاتی که در این تعریف پروژه آمده است (آن هم با رعایت امتیازی و اجباری) پیاده‌سازی خواهند شد. نیازی به پیاده‌سازی دقیق این طراحی و زیباسازی در این سطح با داشتن انیمیشن و موارد مشابه نیست. پیاده‌سازی هر چه شبیه‌تر به نرم‌افزار اصلی و داشتن افکت‌های انیمیشنی، امتیازی خواهد بود. توجه کنید بخش مهمی از GUI که شما در نرم‌افزار اصلی می‌بینید، صرفاً رنگ‌بندی دکمه‌ها و پنل‌ها و غیره است، شما

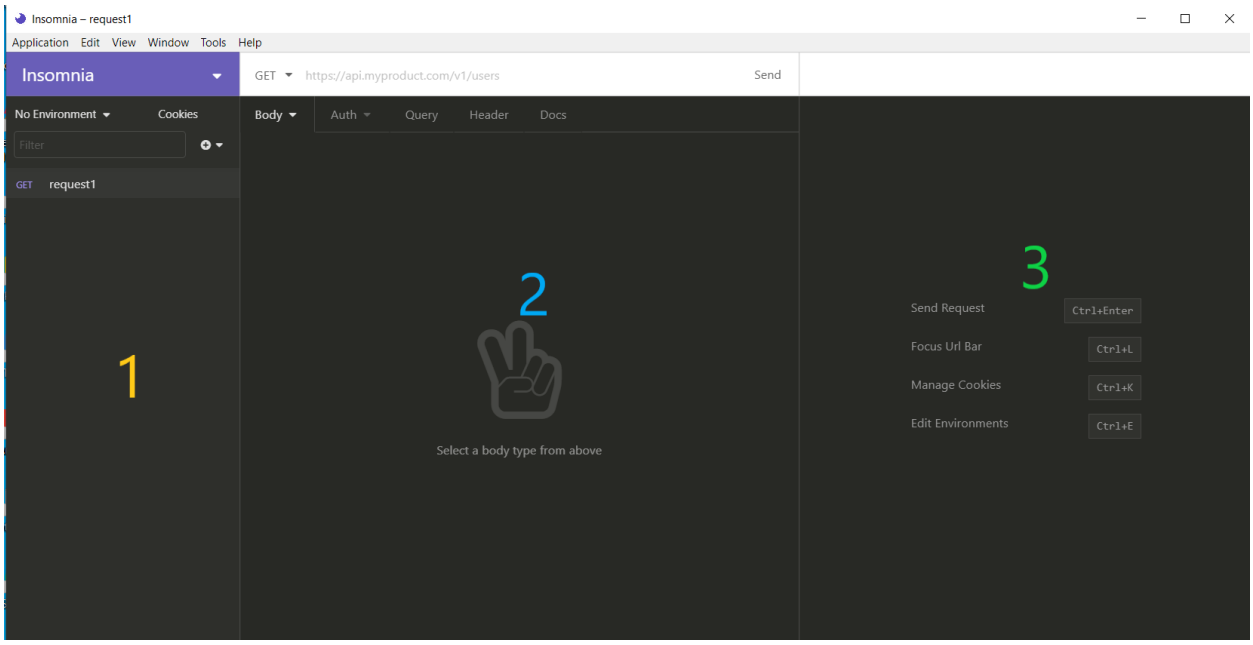
می‌توانید به راحتی آنها را در جاوا پیاده‌سازی کنید. پیاده‌کردن افکت hover نیز به راحت از طریق تعریف یک listener قابل پیاده‌سازی است. پس نگران نباشید، شما حتماً می‌توانید این نیازمندی‌ها و حتی قسمت‌های امتیازی پروژه را هم با مطالب تدریس‌شده پیاده‌سازی کنید. ممکن است در توضیحات زیر به برخی از ویژگی‌های HTTP اشاره کنیم، اگر آنها را فعلا متوجه نمی‌شوید، مشکلی نیست، در فاز های بعدی و در درس، آنها به طور کامل شرح داده می‌شوند. در این مرحله فقط پیاده‌سازی رابط گرافیکی مد نظر است.

تصاویر زیر از نرم‌افزار Insomnia است. می‌توانید این نرم‌افزار را از لینک زیر دانلودکرده و با قابلیت‌های آن آشنا شوید.

https://insomnia.rest/download/

تصویر کلی برنامه به صورت زیر است:





رابط گرافیکی به صورت کلی به ۳ قسمت عمده تقسیم می‌شود:

۱ . لیست ریکوئست‌های قبلی ذخیره‌شده، در صورت وجود گروه‌بندی موضوعات که امتیازی است، این قسمت خود به چند زیر بخش تقسیم می‌شود. با کلیک بر روی هر گروه، لیست ریکوئست‌های ذخیره شده در آن گروه نمایش داده می‌شود. در اینجا تا زمانی که نمی‌دانید ریکوئست چیست، برای هر ریکوئست یک آیتم مستطیلی در نظر بگیرید.

۲ . این قسمت محل تنظیم ریکوئست جدید است.

در این قسمت یک فیلد برای نوشتن آدرس URL، برای ارسال درخواست، وجود دارد.

کنار این فیلد، یک لیست وجود دارد که از داخل آن یکی از متدهای GET, DELETE, POST, PUT, PATCH را می‌توان انتخاب کرد. از این اطلاعات در فاز های بعدی برای ارسال درخواست های HTTP استفاده می‌کنیم.

یک دکمه تحت عنوان Send و یا ارسال، برای ارسال درخواست وجود دارد.

در قسمت پایین تعدادی تب (tab) وجود دارد.

تب اول برای پرکردن request body (مجموعه هدرها و message body) است. بدیهی است با توجه به نوع message body، باید امکان انتخاب بین Form Data و JSON (امتیازی) وجود داشته باشد.

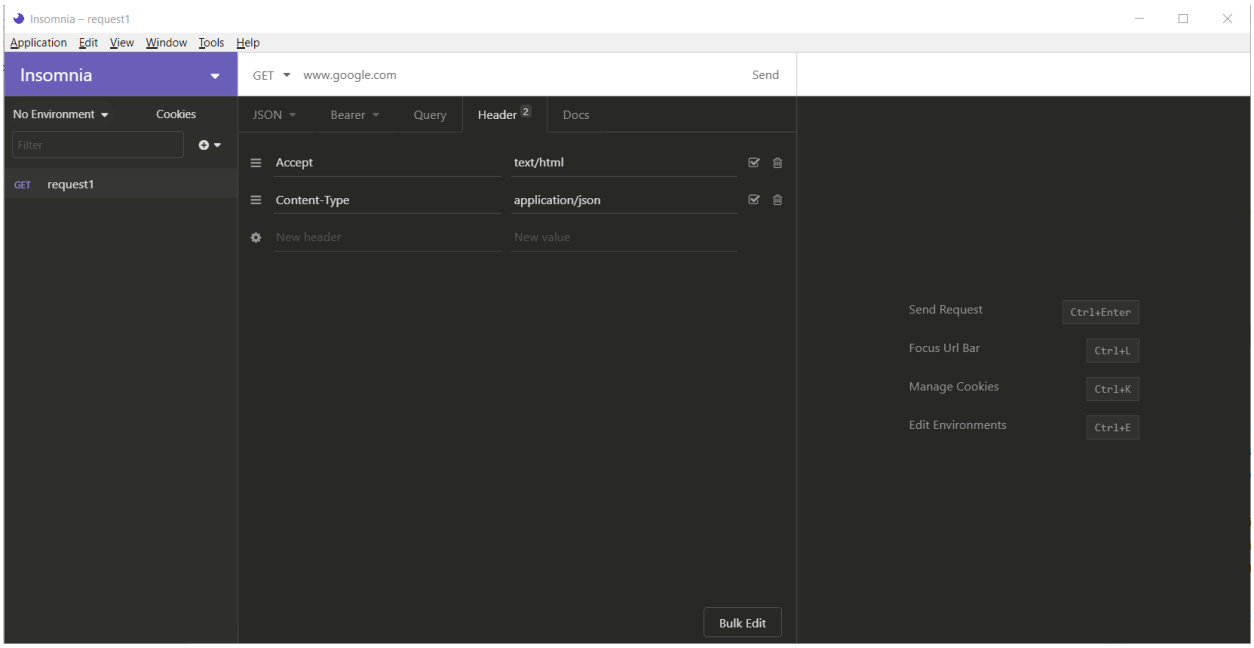
اگر امکان آپلود فایل نیز دارید، یک گزینه دیگر در ادامه ۲ آیتم قبلی به نام Binary Data اضافه خواهد شد که یک فایل انتخاب می‌کند (امتیازی است).

تب بعدی به عنوان Header برای تنظیم هدرهای درخواست وجود دارد. تصویر تب هدر در بخش زیر قابل مشاهده است.

در این بخش تعدادی جفت فیلد وجود دارد. هر جفت فیلد شامل ۲ فیلد کنار هم، یکی برای Key و یکی برای Value است.

در کنار هر جفت ۲ گزینه وجود دارد، یک گزینه با علامت "سطل زباله" یا "ضربدر" که با کلیک روی آن جفت لیست حذف می‌شود.

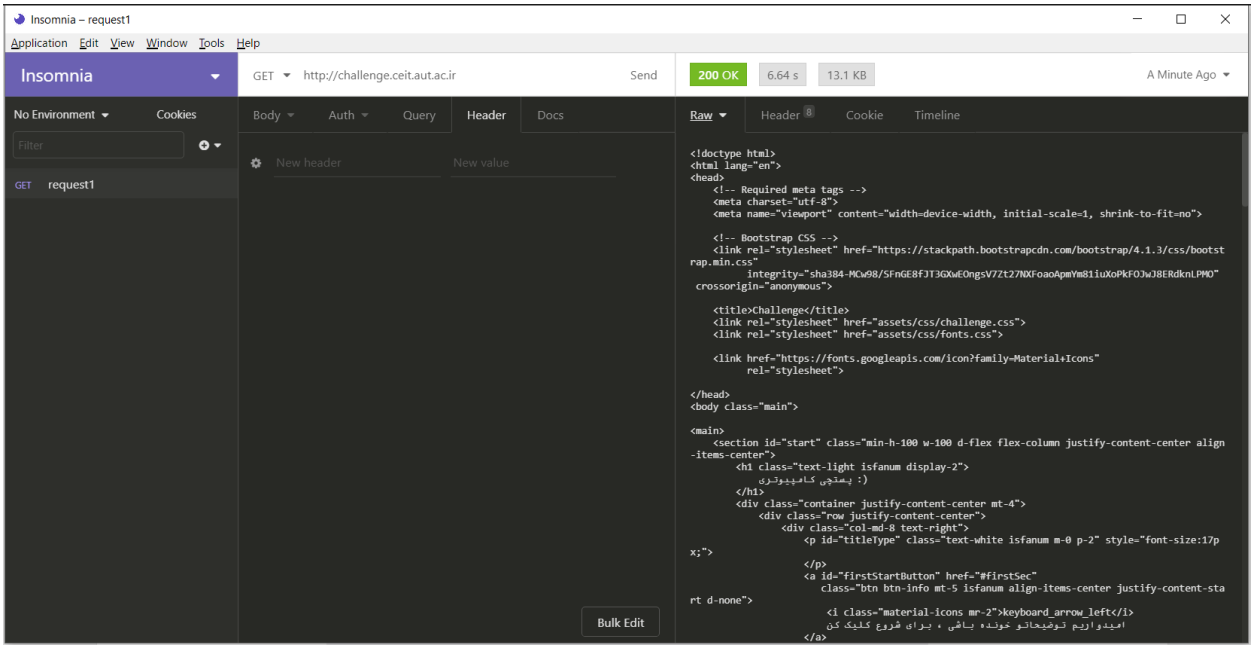
یکی هم یک checkbox که با کلیک روی آن می‌توان وضعیت فعال بودن یا نبودن آن هدر را مشخص کرد.



در حالت امتیازی می‌توانید در این قسمت به صورتی طراحی کنید که همواره حین پر کردن یکی از جفت فیلدها، یک عدد جفت فیلد خالی در زیر آن همواره نمایش داده شود. در غیر این صورت دکمه‌ای مبنی بر اضافه‌کردن یک جفت فیلد جدید به Header قرار دهید.

پیاده سازی تب Query و Auth (فقط شامل مدل Bearer) امتیازی است. در فازهای بعدی با این موارد بیشتر آشنا می‌شوید. تب Query، از نظر ظاهر، ویژگی‌هایی مشابه همان تب Header خواهد داشت.

۳. قسمت پاسخ درخواست و یا response



در این قسمت، پاسخی که از ارسال درخواست، دریافت می‌کنید را نمایش می‌دهید.

در این قسمت پیاده‌سازی ۲ بخش message body و Header اجباری است.

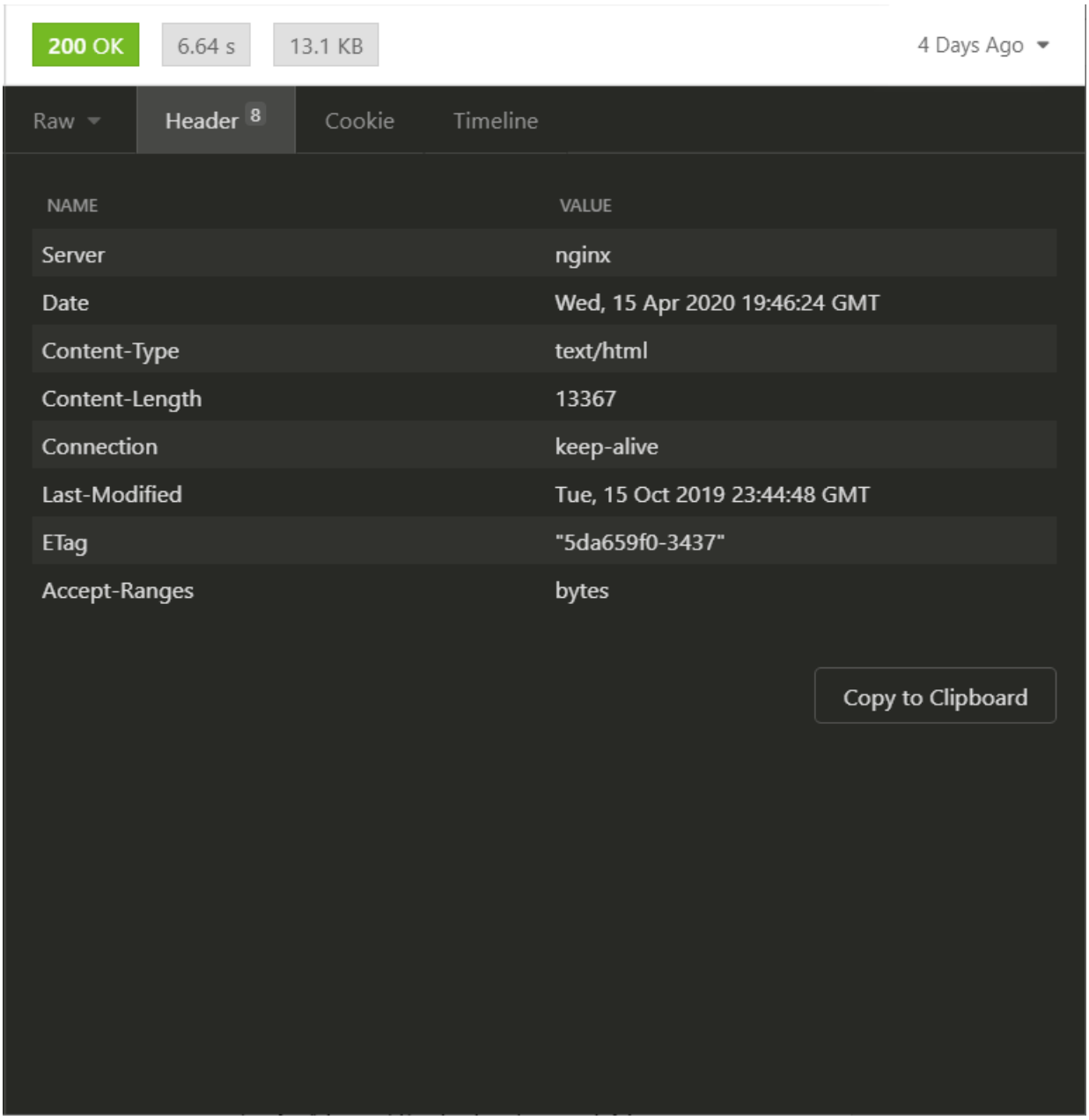
در تب Header، دقیقاً یک صفحه مشابه تب Header در قسمت قبلی نمایش داده می‌شود که این بار دکمه‌های حذف و غیرفعال‌کردن را ندارد و فیلدهای آن قابل ویرایش نیست.

نمایش status code و status message نیز اجباری است. (مثلا در تصویر همان 200 OK).

نمایش مدت زمانی که طول کشیده تا درخواست، جواب داده شود اجباری است. (مثل در تصویر 6.64s).

نمایش حجم داده‌ی دریافتی در message body امتیازی است. (مثلا در تصویر 13.1KB).

یک دکمه به عنوان Copy to Clipboard نیز در تب هدر وجود دارد.



در تب message body، وضعیت فعال پیش‌فرض به صورت Raw است. اگر درخواستتان طوری بوده که در جواب یک تصویر آمده، باید یک وضعیت دیگر قابل انتخاب باشد به عنوان Preview که تصویر رسیده را در این وضعیت به صورت خوانا نشان دهد. پیاده سازی وضعیت JSON به عنوان message body نیز امتیازی است.

قسمت نوار بالایی برنامه ۳ منو Application و View و Help وجود دارد.

منو Application شامل ۲ زیر منو Options و Exit است.

با کلیک روی گزینه Options یک پنجره باز می‌شود که در آن شما می‌توانید یک تنظیم follow redirect را با استفاده از یک checkbox فعال یا غیر فعال کنید.

یک checkbox دیگر نیز وجود دارد که با استفاده از آن می‌توانید مشخص کنید برنامه هنگام خروج، به طور کلی بسته شود یا نه در System Tray مخفی شود.

در همین پنجره نیز قسمتی وجود دارد که شما با انتخاب از یک لیست پیشفرض می‌توانید رنگ‌بندی رابط گرافیکی را از بین light theme و dark theme انتخاب کنید.

توجه کنید تمام این تنظیمات باید در یک فایل ذخیره شود و بعد از اجرای نرم افزار، آخرین تنظیمات ذخیره‌شده اعمال شود.

با کلیک بر روی گزینه Exit نیز برنامه عملیات خروجی که برایش تعریف شده (Exit / Hide on system tray) را اجرا می‌کند.

منو View شامل ۲ گزینه Toggle Full Screen و Toggle Sidebar است.

با کلیک بر روی گزینه Toggle Full Screen، برنامه به صورت full screen در می‌آیند و با کلیک دوباره، برنامه به حالت قبلی برمی‌گردد.

با کلیک بر روی گزینه، Toggle Sidebar، قسمت لیست درخواست‌ها (همان قسمت ۱ در تصاویر بالا) مخفی و با کلیک دوباره، نشان داده می‌شود.

منو Help شامل ۲ گزینه About و Help است.

با کلیک بر روی گزینه About، اطلاعات شما به عنوان توسعه‌دهنده مانند ایمیل و شماره دانشجویی در یک پنجره نمایش داده می‌شود.

با کلیک بر روی گزینه Help، پنجره‌ای باز می‌شود که در این پنجره، توضیحات کمکی دستورات (در واقع خروجی دستور help– در فاز بعدی) نمایش داده می‌شود.

برای تمام این گزینه‌ها و منوها، باید پیاده‌سازی Accelerator و Mnemonic انجام شود. برای آشنایی می‌توانید به این لینک مراجعه کنید.

اطلاعات نمایش داده شده در تمام قسمت‌ها در این فاز از پروژه، با داده‌های ساختگی می‌تواند باشد. لازم نیست داده‌ها معنی‌دار باشند.

توجه کنید در این قسمت، به هیچ عنوان پیاده‌سازی کارایی برنامه و اینکه با این اطلاعات وارد شده چه کار انجام می‌شود، مد نظر نیست و پیاده‌سازی GUI صرفا مد نظر است. قسمت‌های امتیازی در منوها مثل باز و بسته‌کردن sidebar کنار برنامه طبیعتاً وقتی دارای نمره امتیازی هستند که کار بکنند و کارکردن این موارد باید در این فاز انجام شود. در این فاز اگر در مواردی مانند کار با فایل، هنوز تدریس نشده است، نیازی به پیاده‌سازی کارایی نیست و فقط ظاهر را پیاده‌سازی کنید. هر آنچه در این تعریف پروژه (و نه در تصاویر) به عنوان موارد اجباری آمده است، باید پیاده‌سازی شود. هر قابلیت اضافه‌تری می‌تواند در دسته تغییرات امتیازی قرار گیرد. مثلا اگه ویژگی ظاهری خاصی در اصل برنامه وجود دارد و مایل به پیاده‌سازی آن هستید، می‌توانید آن را به عنوان بخش امتیازی پیاده‌سازی کنید.

فازهای دوم و سوم: پیاده‌سازی رابط کنسولی و متصل کردن رابط کنسولی به رابط گرافیکی.

در این فاز شما باید اولین سری کارکردهای برنامه را از طریق یک رابط کنسولی پیاده‌سازی کنید. در این فاز ابتدا شما تمام قابلیت‌های نرم‌افزار را تحت یک رابط کنسولی (مشابه تمرینات و پروژه‌های قبلی) پیاده‌سازی می‌کنید که به صورت ایده آل مشابه نرم افزار cURL می‌شود و در قسمت بعدی رابط گرافیکی خود را روی این برنامه‌ای که نوشتید، سوار می‌کنید. (لینک دانلود https://curl.haxx.se/)

برای پیاده سازی این فاز، بخش‌های کار با فایل، Exception Handling و بخش‌های ابتدایی شبکه مورد نیاز هستند.

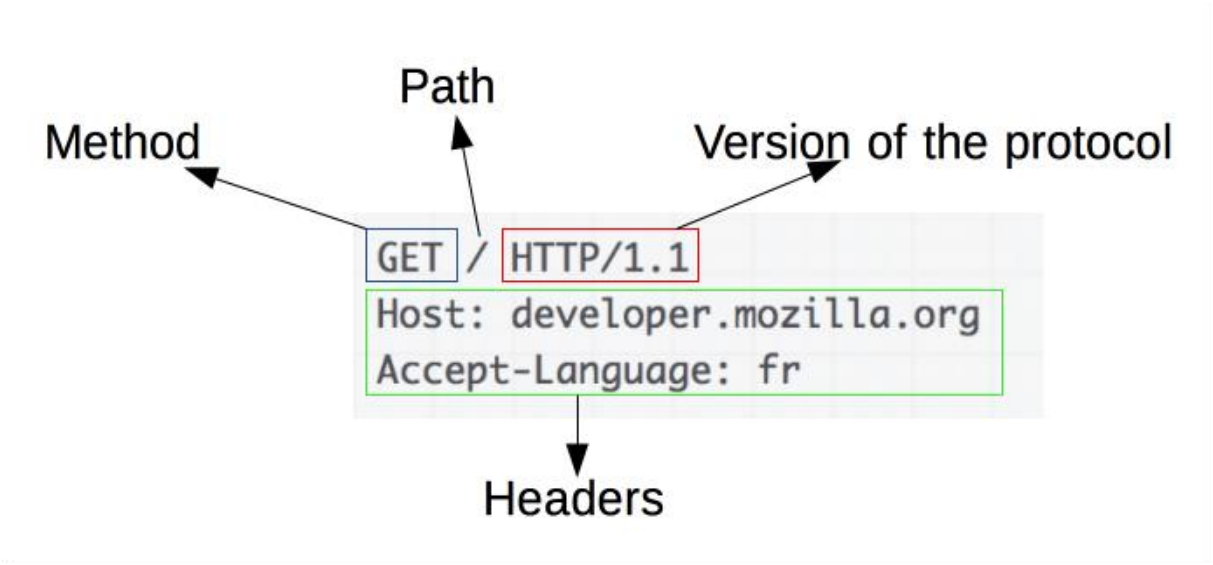
در این قسمت، اول با معرفی پروتکل HTTP شروع می‌کنیم و در ادامه به بررسی ویژگی‌های اجرایی نرم‌افزار HTTP Client می‌پردازیم.

در نظر داشته باشید تمامی محتوای این متن از روی منابع معتبری نظیر Mozilla Developer Network (MDN) خلاصه برداری شده است و تلاش شده است تا با نکات کاربردی این بحث آشنایی لازم صورت بگیرد. حتما پیشنهاد می‌شود با مراجعه به این منبع، پست‌های کامل آن را مطالعه کنید.

HTTP مخفف Hyper Text Transfer Protocol است، پروتکل اساساً یک سری قوانین قراردادی برای نحوه ارتباط بین سرور و کلاینت است. دراین مدل، کلاینت یک درخواست (Request)، برای سرور می‌فرستد و سرور بعد از پردازش، به آن یک پاسخ (Response) می‌دهد. دنیای وب بر مبنای این request و response کار می‌کند، پروتکل HTTP یک قرارداد برای تعریف انواع مختلفی از این request و responseها است. برای فهمیدن بهتر این موضوع، به ارسال یک بسته پستی فکر کنید.

نکته مهم سادگی این پروتکل است. به این صورت که به راحتی توسط انسان قابل شناسایی و خواندن است. در بحث شبکه، این پروتکل از «اتصال» برای ارسال request و response استفاده می‌کند، به این صورت که برای ارسال هر درخواست، یک اتصال برقرار می‌شود و این request و response از طریق آن بین کلاینت و سرور منتقل می‌شوند که در اینجا توضیحات و نحوه این اتصال که این ارتباط بین سرور و کلاینت از طریق آن انجام می‌شود از حوصله بحث دور است و در طراحی و پیاده‌سازی هم مد نظر نیست.

برای شروع بیایید از مشاهده ساختار یک درخواست HTTP استفاده کنیم. گفتیم که این پروتکل توسط انسان قابل خواندن و فهمیدن است، تصویر زیر گویای این موضوع است. یک درخواست ساده HTTP را که برای یک سرور (مثلا سرورهای MDN) ارسال می‌شود را نشان می‌دهد.



همانطور که در تصویر بالا مشخص است، هر درخواست HTTP از ۴ بخش تشکیل شده است. اولین مرحله برای فرستادن یک درخواست، دانستن آدرس مقصد است، در اینترنت این آدرس‌ها را با URL (مخفف Uniform Resource Locator) مشخص می‌کنند. پس در یک درخواست HTTP باید مشخص شود که مقصد درخواست کجاست. مقصد درخواست می‌تواند هر جایی باشد.

قسمت بعدی نوع پروتکل و نسخه را مشخص می‌کند که به صورت پیش‌فرض HTTP و نسخه ۱,۱ است، اخیرا نسخه جدیدتری از HTTP معرفی و استفاده می‌شود که مورد بحث نیست. هر درخواست HTTP از دو بخش مهم دیگری تشکیل شده است به اسم Method و Headers.

HTTP یک سری متد استاندارد برای کاربر تعریف کرده که با استفاده از آنها می‌توان نوع عملیاتی که روی آن URL مقصد قرار است انجام دهیم را مشخص کنیم.

متد ها (یا verb)های استاندارد HTTP عبارتند از :

GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH

در HTTP، semantics اهمیت دارد. برای مثال شما می‌توانید درخواستی را که صرفاً یک سری اطلاعات از سرور تقاضا می‌کند هم با متد GET ارسال کنید و هم با PUT و یا با DELETE. ولی به شدت تاکید شده است تا از طریق متدها، مفهوم درخواست را به درستی انتقال دهیم. اگر قرار است صرفاً اطلاعاتی دریافت کنید از GET و اگر قرار است اطلاعاتی برای سرور بفرستید تا یک پردازشی انجام شود، از PATCH، POST و یا DELETE

استفاده کنید. مثلاً اگر از نوع ویرایش یک داده فعلی است، از PATCH یا اگر از نوع حذف یک داده فعلی است از DELETE و یا اگر از نوع ایجاد یک داده است که وجود نداشته از POST استفاده کنید. البته اجباری روی این مدل استفاده نیست و فقط بسیار تاکید شده، برای خوانایی بیشتر از هرکدام در مفهوم درست خود استفاده کنیم.

قسمت بعدی در یک درخواست HTTP، هدرها (Headers) هستند. هدرها در یک درخواست HTTP این امکان را به ما می‌دهند که ویژگی‌های بیشتری برای درخواستمان مشخص کنیم. هدرها به صورت جفت های key:value هستند. در مورد هدرها، تعدادی هدر استاندارد وجود دارد که می‌توانیم از آنها استفاده کنیم، علاوه بر آن می‌توان هدرهای مخصوص خودمان (که فقط برای سرور هدف ما معنی‌دار است) نیز استفاده کنیم. لیست کامل هدرها و توضیحات هر کدام در لینک مرجع زیر از وب‌سایت MDN آمده است:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

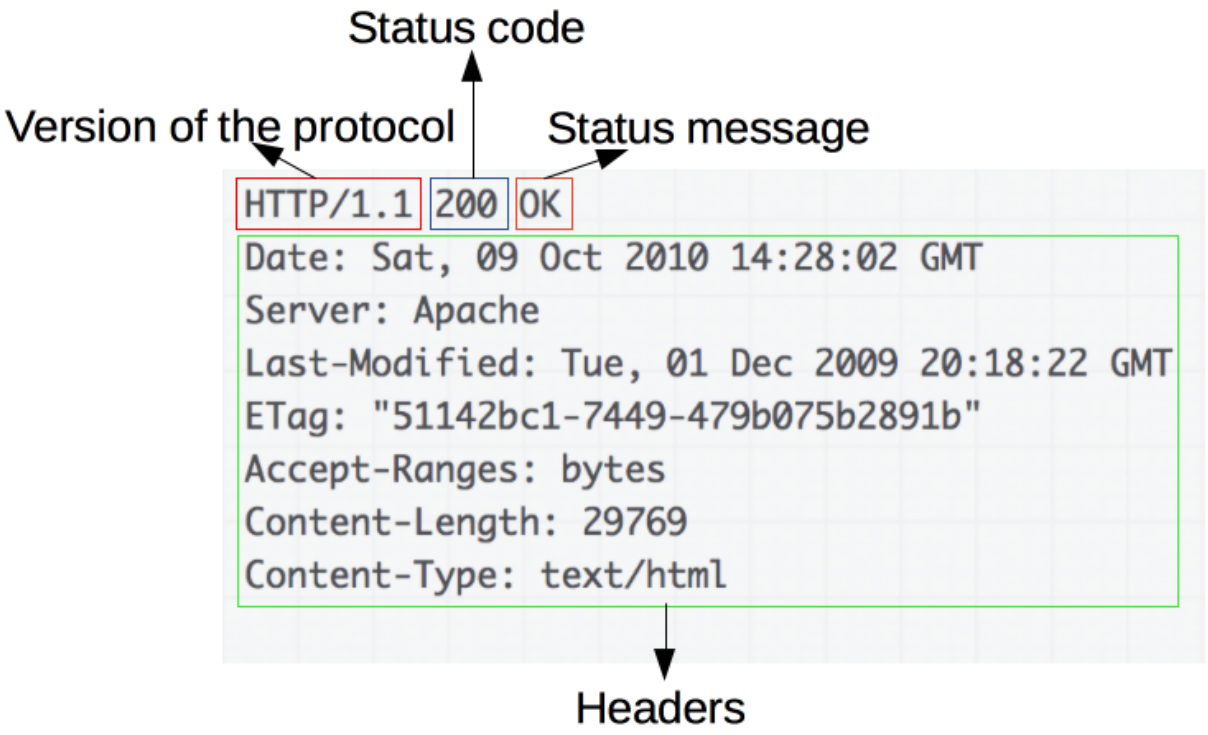
نیازی به یادگیری هدرها و دانستن ریز جزئیات همه آنها نیست. حال در یک درخواست HTTP با توجه به نوع متد و درخواستمان گاهی لازم است اطلاعات بیشتری در درخواست خود قرار دهیم، مثلا اگر قرار است یک ریکوئست POST بفرستیم که یک داده جدیدی در سرور ایجاد شود، باید آن اطلاعات را برای سرور مورد نظر بفرستیم، یا اگر قرار است در ریکوئستی با متد PATCH مشخص کنیم که داده‌ای نیازمند ویرایش است، باید ویرایش مد نظر خود را نیز برای سرور بفرستیم. همینطور برای اینکه بخواهیم یک داده‌ای در سرور حذف شود با استفاده از ریکوئست DELETE باید مشخص کنیم که داده‌ای که می‌خواهیم حذف کنیم، چیست. از این رو در برخی ریکوئست‌ها علاوه بر Header و Method بخش دیگری نیز تحت عنوان body یا message یا message body و یا payload وجود دارد که به ما این امکان را می‌دهد با هر قالبی که دوست داریم برای سرور داده ارسال کنیم.

روش‌های معروف‌تر استفاده از message body عبارتست از Form Data و JSON در Form Data ما اطلاعات را به صورت جفت‌های key=value نگهداری می‌کنیم (شبیه HTTP Headers) و در message body قرار می‌دهیم. هر جفت هم با & از هم جدا می‌شوند.

در مدل JSON، نوع message body (چه در request و چه در response) از نوع شی‌هایی مشابه شی‌هایی در جاوا-اسکریپت است. مطالعه و پیاده‌سازی این مورد در پروژه به طوری که مدل JSON به طور کامل پشتیبانی شود، امتیازی است.

حال وقتی این ریکوئست را فرستادیم، سرور بعد از بررسی درخواستمان، به ما یک HTTP Response باز میگرداند که نتیجه درخواست ما است.

قالب این ریسپانس دقیقاً مشابه قالب ریکوئست است تنها با یک تفاوت که در HTTP Response یک قسمت جدید تحت عنوان Status Code و Status Message اضافه شده که سرور با استفاده از آنها نتیجه درخواست را برای ما مشخص می‌کند. در تصویر زیر نمونه یک HTTP Response را می‌بینید :



همانطور که می‌بینید قالب یک response، مشابه request است با این تفاوت که به جای method حالا status داریم.

احتمالا با برخی از status code ها تا کنون برخورد کرده‌اید: مثلا 200، 404، 403، 500 و غیره. در مرجع MDN به طور کامل تمامی status codeها توضیح داده شده است. به برخی از آنها اشاره می‌کنیم: کد های سری 2** از نوع موفق بودن درخواست هستند. مثلا 200 OK به معنی موفقیت‌آمیز بودن درخواست است. مثلا اگر درخواستی از نوع GET بوده یعنی داده‌های مورد نظر با موفقیت از سمت سرور دریافت شده است. پیشفرض جواب درخواست‌های موفقیت‌آمیز همین کد ۲۰۰ است.

حال کمی مثال ملموس‌تر ببینیم، وقتی شما یک آدرسی را در نوار بالایی مرورگر وارد می‌کنید، مرورگر بعد از ایجاد اتصال با سرور وب‌سایت، یک درخواست HTTP به آن می‌فرستد، حال با توجه به پاسخی که می‌آید، حرکت بعدی خود را مشخص می‌کند. پاسخ می‌تواند یک متن ساده باشد، می‌تواند عکس باشد یا می‌تواند یک فایل HTML و یا یک فایل CSS باشد که مرورگر آنها را در صفحه نشان می‌دهد.

پس عملاً مرورگر در درون خود یک HTTP client دارد که درخواست‌های HTTP می‌فرستد و با توجه به جواب، عملیات‌های بعدی را انجام می‌دهد. برای اینکه ببینید مرورگرتان چه درخواست‌هایی می‌فرستد و چه جوابی می‌گیرد، از طریق ابزارهای developerایی که مرورگرها در اختیارتان قرار می‌دهند، می‌توانید این request و response ها را مشاهده کنید.

کافی‌ست برای مثل در مرورگر کروم در ویندوز، Ctrl + Shift + i را بزنید و در پنجره باز شده به تب Network بروید و حال صفحه را رفرش کنید تا ریکوئست‌ها را ببینید.

حال در مورد پروژه میانترم، شما باید یک HTTP client با زبان جاوا بنویسید، جاوا کلاس های متعددی برای کار با ریکوئست های HTTP دارد که کار شما را راحت‌تر می‌کند، می‌توانید از آنها استفاده کنید، به عنوان مثال می‌توانید از کلاس HttpUrlConnection استفاده کنید. به یاد داشته باشید که شما تنها مجاز هستید که از کتابخانه‌های مربوط به زبان جاوا استفاده کنید.

شما در این فاز ابتدا باید HTTP client بنویسید که به طور کامل بتواند از طریق رابط کاربری کنسولی کار کند و ریکوئست‌ها را نمایش دهد. در فاز بعدی، شما باید رابط گرافیکی که در مرحله قبل پیاده‌سازی شد را روی این رابط کنسولی خود سوار کنید.

در فاز چهارم نیز باید بحث شبکه با استفاده از socket programming را نیز اضافه کنید که توضیحاتش در انتها آمده است.

اول پیشنهاد می‌شود ابزار cURL را نصب کنید و با آن کمی کار کنید، قرار است رابط کنسولی شما، مشابه رابط کنسولی این نرم افزار باشد که در زیر با کمی تغییر آن را توضیح می‌دهیم.

در ساده ترین حالت نرم افزار شما در محیط کنسول باید از آرگومان های زیر پشتیبانی کند:

۱ . آدرس مقصدی که ریکوئست قرار است به آن فرستاده شود.

۲ . متد ریکوئست شما با استفاده از `--method` و یا `-M` (که به طور پیشفرض GET است).

۳ . هدرهای ریکوئست‌های شما با استفاده از `--headers` و یا `-H`

۴ . آرگومان `-i` که مشخص می‌کند هدرهای response نمایش داده بشوند یا خیر.

۵ . آرگومان `--help` و یا `-h` که اطلاعات مربوط به همین لیست را به صورت مناسب در کنسول چاپ می‌کند.

۶ . آرگومان -f که مشخص می‌کند در صورت جواب ریدارکت از سمت سرور، به صورت خودکار ریکوئست‌های مورد نیاز بعدی را نیز بزند و اصطلاحاً ریدایرکت را follow کند.

۷ . آرگومان `--output` ویا `-O` که با استفاده از آن می‌توان response body را در یک فایل ذخیره کرد. این آرگومان می‌تواند یک نام هم بگیرد که به عنوان نام فایلی استفاده می‌شود که خروجی را در آن می‌ریزد، اگر نام داده نشده بود یک نام پیشفرض در قالب output_[CurrentDate] در نظر می‌گیرد و محتوا را در آن ذخیره می‌کند.

۸ . آرگومان `--save` و یا `-S` که با مشخص کردن آن، request فعلی به همراه تمام تنظیمات آن در یک فایل متنی ذخیره شده و در صورت لزوم می‌تواند توسط برنامه بازخوانی شود.

۹ . آرگومان `--data` و یا `-d` که درون آن message body مشخص می‌شود و قالب آن به صورت Form Data است.

۱۰ . آرگومان `--json` یا `-j` که درون آن message body مشخص می‌شود که قالب آن به صورت یک شی json است بدیهی است که اگر قرار بر پیاده‌سازی این مورد است، هر ریکوئست فقط می‌تواند شامل یکی از انواع message body ها باشد.

۱۱ . آرگومان `--upload` که یک آدرس absolute فایل در سیستم را دریافت می‌کند و آن فایل را به آن آدرس آپلود می‌کند. پیاده‌سازی این مورد می‌تواند در قالب همان Form Data و از طریق Multipart Form Data باشد. برای آشنایی بیشتر به لینک زیر مراجعه کنید:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

این برنامه یک فایل دارد که در آن لیست requestهای ذخیره‌شده از طریق دستور `-save` و یا `-S` در آن ذخیره می‌شود، از این طریق عملا امکان پیاده‌سازی یک کالکشن و گروه از ریکوئست‌ها وجود دارد.

پیاده‌سازی امکان دسته‌بندی ریکوئست‌ها و داشتن بیش از یک گروه امتیازی است.

فرض کنید اسم نرم افزار jurl است. در محیط کنسول مثال‌های زیر را می‌توانیم داشته باشیم:

```
>jurl foo.com
>jurl foo.com -M GET
>jurl foo.com --method GET --headers “key1:value1;key2:value2”
>jurl foo.com -i -M GET
>jurl foo.com -i
>jurl foo.com -i -f -O google.html
>jurl foo.com -i --output
>jurl foo.com -i -O google.html --method GET -H
“Accept:text/html” --save
>jurl foo.com --method POST -d
“firstName=hadi&lastName=Tabatabaei”
>jurl foo.com/hadi --method PATCH -json
“{firstName:hadi,lastName:1 abatabaei}”        8
…
```

توجه داشته باشید هنگامی که آرگومان سیو را به برنامه می‌دهید، باید قبل از اجرای درخواست، آن را ذخیره کند و بعد درخواست را اجرا کند.

برنامه باید با استفاده از دستور زیر لیست درخواست های ذخیره شده را به کاربر نمایش دهد:

```
>jurl list
```

خروجی این دستور باید یک منو باشد به صورت زیر:

```
1 . url: google.com | method: GET | headers: Accept: text/html …
2 . url …
3 . …
```

بعد کاربر می‌تواند با انتخاب یک یا چند درخواست، به ترتیب آنها را صدا بزند. دقت کنید برنامه نباید بعد از نشان دادن لیست برای دریافت ورودی قفل کند و منتظر ورودی کاربر باشد.

مثلاً اگر قبلا ۵ درخواست ذخیره شده، کاربر می‌تواند بعد از دیدن لیست، برای اجرای درخواست های ذخیره شده دوم و سوم دستور زیر را وارد کند:

```
>jurl fire 2 3
```

که خروجی این دستور معادل آن است که درخواست ها را به ترتیب ورودی، در کنسول اجرا کرده باشیم.

برای پیاده سازی قسمت امتیازی و اضافه کردن چندین گروه، یک مرحله به list اضافه می‌شود که ابتدا اسامی گروه‌ها را نمایش می‌دهد و سپس با دستور jurl list listName> می‌توان اگر لیستی با آن اسم وجود دارد، درخواست‌های آن را نشان دهد، در هنگام اجرای درخواست‌های لیست نیز دستور قبلی به jurl fire listName … 1 2 تبدیل می‌شود که یعنی درخواست‌های ۱ و ۲ را از لیست listName اجرا کن. برای ایجاد لیست جدید از دستور jurl create listName استفاده می‌کنید. در هنگام save نیز می‌توانید یک ورودی به آرگومان --save دهید که اسم لیست مورد نظر است.

دستورات بالا به ترتیب به این صورت خواهند بود :

```
>jurl create hadiFavorites
>jurl google.com --save hadiFavotires
>jurl list hadiFavorites
>jurl fire hadiFavorites 1
```

فاز سوم: اتصال فاز اول و دوم به هم

در این فاز، باید رابط گرافیکی که در فاز ۱ طراحی کرده بودید را به رابط کنسولی پیاده‌سازی شده در فاز ۲ متصل کنید. به این صورت که باید در رابط گرافیکی خود برای انجام درخواست‌ها، از رابط کنسولی استفاده کنید. توجه کنید به هیچ وجه هنگام ارسال درخواست از طریق رابط گرافیکی، نباید برنامه قفل و یا freeze کند. (برای این کار باید از چندنخی و از SwingWorker استفاده نمایید.)

فاز ۴: شبکه و socket (امتیازی)

در این فاز برنامه شما باید بتواند لیست درخواست‌های HTTP ذخیره شده و یا گروه‌بندی شده را تحت شبکه برای یک سرور پراکسی بفرستد. این سرور پراکسی را نیز باید شما پیاده‌سازی کنید.

در نرم‌افزارهای insomnia و یا postman این قابلیت وجود ندارد، به سلیقه خودتان قسمتی در رابط گرافیکی برای این قسمت در نظر بگیرید.

در رابط کنسولی از طریق دستور زیر IP و Port سرور پراکسی را مشخص کرده و با استفاده از دستور send-- یک فایل شامل تمام درخواست‌های ذخیره شده را برای آن نرم افزار می‌فرستد:

```
>jurl --send --ip 192.168.1.4 --port 8080
```

برنامه باید این قابلیت را داشته باشد که یک request ذخیره شده و یا جدید را تحت شبکه به سرور پراکسی بفرستد، سرور پراکسی درخواست را اجرا می‌کند و جواب را به برنامه اصلی برمی‌گرداند. با این کار درخواست‌های شما با آدرس کامپیوتر پراکسی ارسال می‌شود.

این ویژگی می‌تواند از طریق دستور proxy-- و در ادامه با دو دستور ip-- و port-- در خود رابط کنسولی این اتفاق بیفتد. با استفاده ازip-- آدرس IP کامپیوتر پراکسی و با port-- پورتی که برنامه پراکسی روی آن اجرا می‌شود مشخص می‌شود.

```
>jurl google.com -i --proxy --ip 192.168.1.1 --port 8080
```