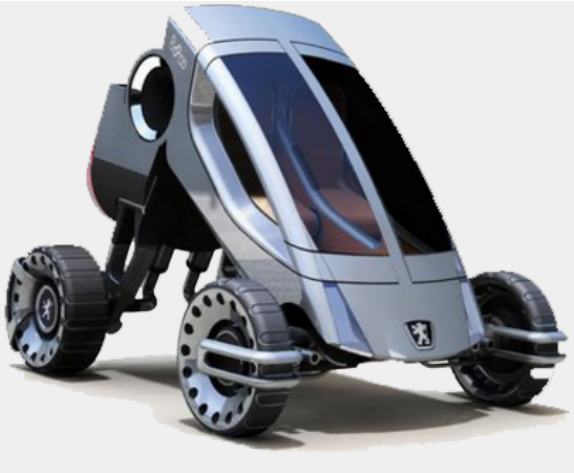
Ex4: Internet 2015/2016



- **In this exercise your job is to create a dynamic web server (AKA application server or middleware server)**
    - notice that dynamic web server can respond to static requests as well.
    - dynamic servers support dynamic functionality as well as cookies:
        - Read the Cookie's wikipedia page: http://en.wikipedia.org/wiki/HTTP_cookie focus on the implementation details http://en.wikipedia.org/wiki/HTTP_cookie#Implementation
- your job is to extend the Ex3's HTTP Static server in addition to redesigning it.
    - change the hujiwebserver API to work as follows:
        - .start() receives *port* and *callback* (and returns a server object)
        - The server object should get extended:
            - .use(resource,requestHandler) method:
                - resource is the prefix of the resource that you had like to handle
                    - for example the resource '/x/y' should 'catch' any the '/x/y' resource or any resource that starts with /x/y/ including /x/y/z/sd/f/
                    - resource can be parametrized, the parameters must be between two '/'s or at the end of the resource. the parameter name starts with the colon symbol
                        - e.g. the resource /x/:y/z handles resources that looks like /x/*/z including /x/dsfsdg/z/sfdgfg and /x/e/z
                    - the resource argument of the use method is **optional,** in case the .use function receives only

one argument, you will set the resource to be '/' i.e. it **should handle any request**.
    - e.g. .use(function(rq,rs,nxt){...}) should handle any request.
- The requestHandler is a function that receives 3 arguments, request, response and next
    - request is the object that represents the HTTP request (via hujiparser)
        - it should have the following properties. params, query, method, cookies, path, host, protocol, get(), param() and is() . the description of each property can be found here: http://expressjs.com/api.html#req.params
        - in addition to a .body property that consists of the http request body, in case there is no body, it should equal null.
        - **You should support all the HTTP methods or at least the minimal list I set at the Students forum.It's very easy to do . the rest of the HTTP message is exactly the same.All you need to do is to detect the HTTP method for a specific request and set the request.method to be equals that method.**
    - response is the object that represents the HTTP response
        - it should have the following properties. set(), status(), get() , cookie(), send() and json() . the description of each property can be found here: http://expressjs.com/api.html#res.status
    - next() is a function that hints the module to lookup for the next requestHandler

- In general upon calling the .use() function, your server job is to register the resource+handler in some data model.
    - upon HTTP request you should walk the registered handlers one by one by the order of their insertion. the first handler that matches the real request resource to the handler's resource should get called.
        - while executing the first handler, only if the next() function has been called you should start walking again on the handlers data model in order to find the next handler to execute if there is any.
        - in case non of the handlers match the resource or none of them call .send() or .json() the response you should return 404
        - in case of any exception that is not treated you should return 500
  - You server should not serve static resources out of the box. The user can enable static serving easily by registering the static handler like this:
    - server.use('<some route>',hujiwebserver.static(rootFolder))
    - You should implement the hujiwebserver.static(rootFolder) method to enable your server to serve static resources out of the rootFolder.
  - Add one more method as hujiwebserver.myUse(). It should do something that many web-servers might need like the hujiwebserver.static method (some functionality that it will be a waste to reimplement for each server)
    - hujiwebserver.myUse().toString() should explains what myUse is doing, why is it needed, and what are the arguments of the method if there are any (like rootFolder for static())
- **Testing**
  - Add a tester.js that test your server, add documentation (doc.html) that explains exactly what you are testing (you should use the HTTP module)
    - you can publish your tester on the Students Forum
- **Security instructions**
  - Do your best in order to avoid DOS attack
    - it means that you should make sure that exceptions does not crash your server.
  - In addition, make sure that the user can't get any files that are not under the root folder (in case of a non-resourceMap request)
- **Some important instructions**
  - It's OK to assume that the dynamic functions will only add ASCII characters to response.body (via send())
    - It means that you can use response.body.length() in order to evaluate the content-length (ASCII char persist 8 bits of memory, and the content-length measure-unit is 8 bits)
  - You must use your own Ex3 code

- **Ex4's (yet still important) instructions**
  - When you are generating a response, you must include the content-length header.
  - Your HTTP server should support the same content type your Ex3 supports
  - You are not allowed to utilize the 'http' module, you should use the 'net' and the 'fs' module as your infrastructure.
  - You are not allowed to use any external node.js files nor plug-ins without asking us first.
  - Try to minimize the number of global javascript variables as possible
  - Keep in mind that this is a web-server, it should be able to serve thousands of concurrent requests. Pay attention to performance issues.
    - Don't do any I/O operation in a synchronized manner.
    - Try to write a tool that will load this server(send as many concurrent requests as possible) in order to test it. (include this load.js file)
  - You are allowed to do this exercise in pairs
  - **Compress all your files and submit fullName.ID9digists.ex4.zip**
    - Add a partner.<partnerID-firstNameEng-lastNameEng/NoPartner).txt file to the zip.
    - Submit once per group of students.
    - Add a readme.txt file to the zip the describes (1) What was hard in this ex? (2) What was fun in this ex? (We won't reduce points in case this part is empty)
    - Add Ex2 to the zip under "www" folder, make sure your server can serve it perfectly. (where 'www' is the root folder)
  - **Deadline: 19/1/2015 23:55 pm**
- Extra reading
  - http://en.wikipedia.org/wiki/Nodejs
  - http://en.wikipedia.org/wiki/C10k_problem
  - http://oreilly.com/openbook/webclient/ch03.html
- Good luck