# Tartalomjegyzék



# Grizzly életre keltése

- Az elsõ dolog, hogy töltsük le a szükséges fájlokat: grizzly_dependencies.zip (konzolban az *unzip* paranccsal tömöríthetjük ki)

- Majd kezdjünk egy új projectet. A projectet kijelölve, jobb klikk és *properties*, majd *java build path*, itt a *libraries* fül, végül *add external jars* gomb. Itt jelöljük ki az összes letöltött file-t, aminek NINCS a végén a *sources* szó.

- Ezek után már minden mûködni fog, ez a lépés opcionális. Nyissátok le a projectet, majd *referenced libraries* itt jobb klikk valamelyikre, *properties*, gyõzõdjetek meg, hogy a *java source attachment* van kijelölve bal oldalt. Ha igen, akkor *external jar* gomb, és jelöljétek ki azt a *source* file-t, ami az épp kiválasztott library-hez tartozik. Majd végezzétek el ezt a maradék 2-vel.

- Az utolsó lépés által fogunk látni hasznos információkat amikor valami nem mûködik, vagy ha az egérrel egy metódus fölé megyünk, amit a szerver definiál. Valamilyen szintû dokumentációt ad, így megkönnyítheti a munkánkat nagyban.

- Ezeket a lépéseket mindig újra el kell végezni amikor új projectet hoztok létre. (Természetesen újra letölteni nem kell.)

# Elsõ próba

Hozzatok létre egy *Main* nevû osztályt a korábban létrehozott projectben, majd másoljátok bele ezt a kódot:

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import org.glassfish.grizzly.http.server.*;

public class Main {
    public static void main(String[] args) {
        HttpServer server = new HttpServer();
        NetworkListener nl = new NetworkListener("main-listener", "localhost", );
        server.addListener(nl);

        server.getServerConfiguration().addHttpHandler(
            new HttpHandler() {
                public void service(Request request, Response response) throws Exception {
                    final SimpleDateFormat format = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:s
```

```
                    final String date = format.format(new Date(System.currentTimeMillis()));
                    response.setContentType("text/plain");
                    response.setContentLength(date.length());
                    response.getWriter().write(date);
                }
            },
            "/time");
        try {
            server.start();
            System.out.println("Press any key to stop the server...");
            System.in.read();
        } catch (Exception e) {
            System.err.println(e);
        }

    }
}
```

Egy hibát fog jelezni az eclipse, méghozzá a HttpServer létrehozásakor. Direkt nem adtam meg ott egy paramétert. Mindenki a táblázatból nézze meg, melyiket portot rendeltem hozzá, majd ezt írja be az üresen hagyott helyre.

Így már jó lesz a kód és megpróbálhatjátok futtatni.

Ha a következõ oldalra mentek, akkor látjátok az elsõ oldalatokat (a *a portotok*at ne felejtsétek el beírni):

http://localhost:portotok/time

Ha megnézitek az oldal forrását láthatjátok, hogy szimplán a dátumot tartalmazza. Semmilyen HTML kód nincs beleírva, ez csak a dátum és az idõ.

# Feladatok

## HTML-esítés

Használjuk a korábban bemásolt kódot. De tegyük egy kicsit szabványosabbá, HTML-esítsük, használhatjátok ezt a mintát:

```
<!DOCTYPE html>
<html lang="http://wiki.math.bme.huhu"http://wiki.math.bme.hu>
    <head>
        <meta charset="http://wiki.math.bme.huutf-8"http://wiki.math.bme.hu>
        <title>
            Oldal neve
        </title>
    </head>
    <body>
        Weboldal.
    </body>
</html>
```

Az oldalon továbbra is csak a dátum jelenjen meg, de legyen körülötte szépen a HTML kód. A *response.setContentType("http://wiki.math.bme.hutext/plain"http://wiki.math.bme.hu);* sorban, a *text/plain*-t, át kell majd írnunk *text/html*-re hogy jól mûködjön.

Próbáljunk rögtön okosak lenni, mentsük valamilyen változóba azon részeit a HTML kódnak amiket egyértelmûen gyakran fogunk használni.

## Még egy oldal

Rakjunk ki mégegy oldalt. Mondjuk a gyökérbe ("http://wiki.math.bme.hu/"http://wiki.math.bme.hu) amin jelenleg legyen csak annyi kiírva, hogy Fõoldal.

## Kényelmesítés

Vegyük észre, hogy ha minden oldalt így behányunk a *Main* osztályba, akkor nagyon átláthatatlan lesz a kódunk. Biztosan meg tudjuk oldani okosabban. Gondolkozzunk el ezen, majd ha megelégeltük a gondolkozást akkor kövessük a szívünket, vagy ezeket a lépéseket:

- Hozzunk létre egy *HtmlUtilities* osztályt, melybe olyan statikus metódusokat és változókat teszünk, melyek minden oldalon jól fognak jönni. A következõ metódusai/változói legyenek:
    - *public static String createSimpleHead(String title)* visszaadja a html head részét, egészen a <head>-tõl a </head><body>-ig.
    - *public static String top* tartalmazza a <head> elõtti dolgokat.
    - *public static String bottom* tartalmazza a </bod></html>-t
    - *public static String createSimplePage(String title, String content)* visszaad egy olyan html oldal kódját, aminek a címe a megadott cím és a tartalma az adott tartalom. Itt már mindenképp használjatok *StringBuilder*t a String építéséhez és használjátok az elõzõ metódust / változókat.
    - A késõbbiekben még kiegészítjük olyan metódusokkal amiket hasznosnak vélünk, most legyen elég ennyi.
- Most írjuk át az eddigi kódunkat, hogy ezeket használja.

- Gondoljuk meg mennyivel szebb lenne, ha minden oldalt külön osztályban írnánk meg. Gondolkodjunk el hogyan lehetne ezt megoldani az eddigi példa és tanulmányaink alapján, majd kövessük az instrukciókat:
    - Készítsünk egy *TimePage* osztályt, aminek az õse legyen a *HtmlHandler*.
    - Azonnal szólni fog, hogy nincs implementálva egy metódus. Mondjuk neki hogy *add unimplemented...* és hozzá is adja a *service* metódust.
    - A *service*-be másoljuk át a Mainbõl ami az idõs oldal *service* metódusa volt.
    - Töröljük a Mainbõl az idõs oldal hozzáadását, és helyette használva az *addHttpHandler*t, de úgy, hogy benne elsõ paraméternek egy *TimePage*-et hozunk létre kapcsoljuk a "http://wiki.math.bme.hu/time"http://wiki.math.bme.hu oldalra a TimePage objektumunkat.
    - Mindez azért mûködik, mert az *addHttpHandler* egy *HttpHandler*t vár, de a *TimePage* is az, így elfogadja.
    - Végezzük el ugyanezt a fõoldallal *IndexPage* legyen mondjuk az osztály neve.

- Örüljünk hisz sokkal tisztább lett a Mainünk és mostantól tudunk szépen új oldalakat beilleszteni a webszerverünkbe.

## Form elküldése

Az elõzõ gyakorlaton megtanultuk hogyan néznek ki a formok. Ezt használva csináljatok egy egyszerû regisztráló oldalt *RegistrationPage* névvel.

- Legyen egy gomb amit megnyomva továbbküldi az adatokat a **regcheck** oldalra. (a form *action* tulajdonsága legyen **regcheck**re állítva, valamint a formon belül legyen egy <button type="http://wiki.math.bme.husubmit"http://wiki.math.bme.hu>

- Ne felejtsük el a Mainben hozzáadni a webszerver oldalaihoz ezt az osztályt.

## Adatok fogadása

Az elõzõ oldallal elküldjük az adatokat a *regcheck* oldalra. Most ezt az oldalt kellene úgy megírni, hogy attól függõen milyen adatokat kap, kiírja, hogy sikeres regisztráció, vagy kiírja, hogy nem egyezik a két jelszó.

- Használjátok a *request.getParameter("http://wiki.math.bme.huparameterNeve"http://wiki.math.bme.hu)* metódust az elõzõ oldalon elküldött paraméterek értékének a lekérésére.
- Ne felejtsétek el, hogy Stringeket is az *equals* metódussal kell összehasonlítani.
- Esetleg visszajelzésként, ha sikeres a regisztrálás, akkor a beküldött adatokat *System.out*al írjuk ki a konzolra.

## Integrálás az eddigi rendszerbe

Ha ezekkel megvagyunk akkor próbáljuk meg a korábban megírt Facebook osztályokkal összekapcsolni a webszerverünket. Szóval a regisztrációs oldal tényleg regisztráljon egy embert a Facebookos adatszerkezetbe.

Ha ez is megvan, akkor írjunk egy bejelentkezõ oldalt, ahol a Facebook bejelentkezését használva megpróbálhatunk belépni regisztrált felhasználóval.

Sajna most még csak addig élnek a felhasználók amíg újra nem indítjuk a szervert, de a késõbbiekben adatbázisban tároljuk majd õket így megmaradnak majd.