




Concurrency in Swing

- 
- ▶ A szálkezelés a swing alkalmazásokban is fontos.
 - ▶ Cél egy olyan felhasználói felület készítése, amely soha nem fagy, mindig válaszol a felhasználói interakciókra, bármit is csináljon éppen.
 - ▶ Swing 3 féle szállal dolgozik:
 - ▶ Kezdeti szálak (initail Threads): az alkalmazást futtató kezdeti szálal.
 - ▶ Event dispatch thread: ahol minden eseménykezelő kód és minden swing-el kapcsolatos interakció fut.
 - ▶ Háttér szálak (worker - threads): időigényes műveletek háttérben futtatására.
 - ▶ A szálakat nem szükséges explicit létrehozni, ezeket a Swing kezeli helyettünk.



Initial Threads

- Minden alkalmazáshoz tartozik néhány szál, ahonnan az alkalmazás elindul.
- Általában ez a main thread.
- Swing alkalmazásokban a kezdeti szál nem végez lát el sok feladatot.
- A legfontosabb feladatai
 - egy Runnable objektum létrehozása, amely inicializálja a GUI-t
 - A Runnable objektum ütemezése az event dispatch thread-re.
- A GUI elindítása után az alkalmazást többnyire az interfész eseményei vezérlik.
- Az események rövid taskok végrehajtását váltják ki az EDT-en.

- Az alkalmazás egyéb taskokat is tud az EDT-re vagy háttér szálra ütemezni.
- A kezdeti szálak a GUI létrehozásának ütemezése
 - `SwingUtilites.invokeLater`: Ütemezi a taskot és visszatér
 - `SwingUtilities.invokeLaterAndWait`: Ütemezi a taskot és megvárja, hogy befejődjön.
- Általában a GUI létrehozás ütemezése az utolsó dolog, amit a main szál végez.
- **Minden Swing componenst használó kódnak az EDT-en kell futnia!**

```
SwingUtilities.invokeLater(new Runnable() {  
    public void run() {  
        createAndShowGUI();  
    }  
});
```

Event Dispatch Thread (EDT)

- ▶ A swing esemény kezelő kódja egy speciális szálon fut.
- ▶ A swing compnenseket használó, létrehozó kód is ezen a szálon fut.
- ▶ Ez szükséges, mert a legtöbb swing objektum nem szálbiztos.
- ▶ Ezeket figyelmen kívül hagyva előre nem látható, nehezen kezelhető hibákba ütközhetünk
- ▶ Az EDT rövid taskok sorozataként fut:
 - ▶ A legtöbb, az eseménykezelő metódusok hívása, mint az `actionPerformed`
 - ▶ Egyéb: az alkalmazás által ütemezett taskok, az `invokeLater` és `invokeAndWait` metódusok használatával.
- ▶ Az EDT taskoknak rövidnek kell lennie.
- ▶ `SwingUtilities.isEventDispatchThread`.




Worker Threads – SwingWorker

- Swing alkalmazásból hosszú futásidejű feladatok végrehajtására használható szálak.
- Minden task egy `SwingWorker` példányként reprezentált.
- Ez egy absztrakt osztály, ennek egy leszármazottját kell definiálni.
- `SwingWorker` a java SE 6 óta.




SwingWorker

- Kommunikációs mechanizmusai
 - `done` metódus definiálható a `SwingWorker` implementációban, a háttér task lefutása után kerül meghívásra, automatikusan az EDT-n.
 - A `SwingWorker` implementálja a `Future` interfészt így a háttértask adhat vissza értéket. Az interfész lehetővé teszi még a task leállítását (`cancel`) és a task állapotának lekérdezését (befejeződött, leállított).
 - A háttér szál elérhetővé tud tenni részeredményeket a futás befejeződése előtt a `publish` metódus hívásával.
 - A részeredményeket a `process` metódus dolgozza fel, amely automatikusan az EDT-n fut.




```
SwingWorker worker = new SwingWorker<ImageIcon[], Void>() {
    public ImageIcon[] doInBackground() {
        final ImageIcon[] innerImgs = new ImageIcon[nimgs];
        for (int i = 0; i < nimgs; i++) {
            innerImgs[i] = loadImage(i+1);
        }
        return innerImgs;
    }

    public void done() {
        try { imgs = get(); } //vagy get(1000);
        catch (InterruptedException ignore) {}
        catch (java.util.concurrent.ExecutionException e) {
            ...
        }
    }
};
```

```
private class FlipTask extends SwingWorker<Void, Integer> {
    protected Void doInBackground() {
        Random random = new Random();
        while (!isCancelled()) {
            publish(random.nextInt(1000));
        }
        return null;
    }
    protected void process(List<Integer> r) {
        ...
    }
}
```

- 
- ▶ Háttér task-nak támogatni kell a megszakítását, ez két módon lehetséges:
 - ▶ Bejezés interrupt hatására (ahogy a szálaknál), `isInterrupted..`
 - ▶ `isCancelled()` metódus hívása periódikusan. `True`-t ad vissza, ha a worker `cancel` metódusát meghívátk.



Swing Timer

- ▶ Egy vagy több Action eventet generál a megadott idő után.
- ▶ Az általános timer-el ellentétben, ez a GUI-val kapcsolatos időzítési feladatok elvégzésére való.
- ▶ Előre létrehozott időzítő szálát használ.
- ▶ A GUI taskok automatikusan az EDT-n hajtódnak végre.
- ▶ Felhasználási módok:
 - ▶ Task végrehajtása egyszer, késleltetés után
 - ▶ Ismétlődő feladatok végrehajtása.



Használata

- ▶ A timer létrehozásakor meg kell adni egy `actionListener`-t amely lefut a megadott időben.
- ▶ A létrehozáskor meg kell adni a végrehajtások közötti várakozási időt.
- ▶ `setRepeats(false)` hívással megadható, hogy a timer ne ismétlődjön.
- ▶ `Start` metódus indítja a timer-t
- ▶ `Stop`-al megállítható.