

Javadoc

Jeszenszky Péter

Debreceni Egyetem, Informatikai Kar

jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2024. március 2.

Dokumentációs megjegyzés (1)

- A forráskódban `/**` és `*/` határolók között elhelyezett megjegyzés.
 - Nem hivatalosan Javadoc megjegyzésnek is nevezik.
- Modul, csomag, osztály, interfész, konstruktor, metódus, enum tag vagy adattag deklarációja előtt kerülnek figyelembevételre.
 - Egyéb helyeken figyelmen kívül lesznek hagyva.
- A megjegyzés szövege HTML-ben adható meg.
 - A szövegben ügyelni kell a `<` és `&` karakterek használatára (megadásuk `<` és `&` módon történhet).

Dokumentációs megjegyzés (2)

- Tartalmazhatja a forráskódban egyetlen sor, mint például:

```
/** Egysoros dokumentációs megjegyzés. */
```
- Megadható több sorban, mint például:

```
/** Ez pedig egy több sorból  
 * álló dokumentációs  
 * megjegyzés.  
 */
```
- A feldolgozás során a határolók között a sorok elejéről elhagyásra kerülnek a whitespace és * karakterek.
- Ha egy sor elején nincs * karakter, akkor nem kerülnek elhagyásra a whitespace karakterek a sor elejéről.
 - Ez hasznos például programkód pre elemében történő megadása esetén.

Dokumentációs megjegyzés felépítése

- Az alábbi két részből áll:
 - **Fő leírás (main description):** a `/**` határoló után kezdődik és a címke szakasz elejéig tart, annak hiányában a `*/` határolóig.
 - **Címke szakasz (tag section):** az első olyan `@` karakterrel kezdődik, amely a sor elején jelenik meg, ha elhagyjuk annak elejéről a vezető `*` karaktereket, a whitespace karaktereket és a `/**` határolót.
 - Ha a fő leírásban sor elején kell olyan `@` karaktert megadni, amely nem a címke szakasz elejét jelzi, használjuk az `@` karakterhivatkozást.
- Megadható fő leírást nem tartalmazó, csak címke szakaszból álló dokumentációs megjegyzés.

Fő leírás (1)

- Az első mondat a dokumentált entitás (például osztály, metódus) tömör összefoglalását kell, hogy tartalmazza.
 - Angol nyelv esetén az első mondat végét az a pont karakter jelzi, melyet egy *whitespace* karakter vagy egy HTML címke követ.
 - Ha egy ilyen pont karakter nem az első mondat végét jelezi, akkor egy szóköz helyett használjunk egy ` `; egyedhivatkozást vagy helyezzünk el egy megjegyzést `<!-- és -->` határolók között közvetlenül a pont karakter után.

Fő leírás (2)

- A felhasználó nyelvi beállításai határozzák meg a dokumentációs megjegyzések nyelvét.
 - Linux környezetben lásd a LANG környezeti változó értékét.
- Az operációs rendszer szintű nyelvi beállítások felülbírálása:
 - A Javadoc program parancssorból történő futtatásakor adjuk meg a `-locale` opciót, mint például `-locale en_US`
 - A *Maven Javadoc Plugin* esetén használjuk a `locale` paramétert vagy az azonos nevű rendszertulajdonságot.

Fő leírás (3)

- A JDK 10 bevezeti a `{@summary}` szövegközi címkét a fő leírás első mondatának explicit jelzéséhez.
 - Akkor hasznos, amikor a javadoc eszköz nem képes helyesen kikövetkeztetni, hogy hol végződik az első mondat.
 - A címke csak akkor számít, ha a leírás elején szerepel.

Több adattag dokumentálása

- Kerülendő az alábbi:

```
/**  
 * Az intervallum eleje és vége.  
 */  
public int start, end;
```

- A dokumentációs megjegyzés ilyenkor le lesz másolva mindkét adattaghoz.

Címkék (1)

- A dokumentációs megjegyzésekben használható speciális kulcsszavak, melyekben a kis és nagybetűk különbözőek!
- Kétfajta címke használható: blokk címke és szövegekőzi címke.
 - Szövegekőzi címkék esetén megszorítások lehetnek arra, hogy a címke hol megengedett a megjegyzésben.
 - Blokk címkék esetén megszorítások lehetnek az előfordulások számára.

Címkék (2)

- **Blokk címke (block tag):** *@címke* formában adhatók meg a sor elején, opcionálisan megengedettek előttük whitespace és * karakterek.
 - Példa: @author, @param
 - A címkéhez a kulcsszót követő szöveg tartozhat, melyet akár több sor is tartalmazhat.
 - A szöveg a következő blokk címkéig vagy a dokumentációs megjegyzés végét jelző */ határolóig tart.
- **Szövegközi címke (inline tag):** {*@címke*} formában adhatók meg a megjegyzésben bárhol, ahol szöveg megengedett.
 - Példa: {@code}, {@link}

Példa

- Az OpenJDK 21 `java.util.LinkedList` osztályának forrásából:

```
/**
 * Pops an element from the stack represented by this list.
 * In other words, removes and returns the first element of
 * this list.
 *
 * <p>This method is equivalent to {@link #removeFirst()}.
 *
 * @return the element at the front of this list (which is
 *         the top of the stack represented by this list)
 * @throws NoSuchElementException if this list is empty
 * @since 1.6
 */
public E pop() {
    return removeFirst();
}
```

Csomagokhoz tartozó megjegyzés állományok

- Csomagok dokumentálásához az ajánlott megoldás:
 - Helyezzünk el a csomagban egy `package-info.java` nevű állományt, melynek tartalmát az alábbi példa szemlélteti:

```
/**
 * Segédosztályokat tartalmazó csomag.
 */
package util;
```

 - A dokumentációs megjegyzésben csak bizonyos blokk és szövegekői címkék megengedettek.

Áttekintő megjegyzés állományok (1)

- Megadható a teljes alkalmazásra vagy több csomagra vonatkozó dokumentáció, melyet külön állomány tartalmaz.
 - Ennek neve tetszőleges lehet, de tipikusan `overview.html` használt a gyakorlatban.
 - Tipikusan a csomagokat tartalmazó könyvtárszerkezet tetején helyezik el.
 - Az állomány elérési útvonalát a Javadoc eszköznek a `-overview` opcióval kell megadni, a Maven Javadoc Plugin-nek pedig az azonos nevű paraméterrel.

Áttekintő megjegyzés állományok (2)

- Az állományt HTML-ben kell megírni.
 - Kötelező a body elem megadása, melynek tartalma kerül másolásra a dokumentáció generálásakor.

- Példa az állomány tartalmára:

```
<html>
  <body>
    <p>Nyílt forrású osztálykönyvtár racionális
      számok kezeléséhez. A fő funkcionalitást a
      {@link math.fraction} csomag tartalmazza.</p>
  </body>
</html>
```

- Bizonyos blokk és szövegekzi címkék is használhatók.

Megjegyzések automatikus másolása

- Metódusok dokumentációs megjegyzései örökölhethők a szülőosztálytól és az implementált interfészekről.
 - Ez nem vonatkozik a konstruktorokra!
 - Ha egy dokumentációs megjegyzésben hiányzik a fő leírás, a `@return`, `@param` vagy `@throws` címkék valamelyike, akkor ezek automatikus másolása.
 - `@throws` címke másolása csak akkor, ha a kivétel dobását deklarálja a metódus!
 - A dokumentáció másolása explicit módon kérhető az `{@inheritDoc}` szövegekőzi címkével.
 - A fő leírásban, `@return`, `@param` és `@throws` címkékhez tartozó szövegben használható.

Rendelkezésre álló címkék

- Szövegekőzi címkék:

- `{@code}`
- `{@docRoot}`
- `{@index}`
- `{@inheritDoc}`
- `{@link}`
- `{@linkplain}`
- `{@literal}`
- `{@return}` (JDK 16)
- `{@snippet}` (JDK 18)
- `{@summary}`
- `{@systemProperty}`
- `{@value}`

- Blokk címkék:

- `@author`
- `@deprecated`
- `@exception`
- `@hidden`
- `@param`
- `@provides`
- `@return`
- `@see`
- `@serial`
- `@serialData`
- `@serialField`
- `@since`
- `@throws`
- `@uses`
- `@version`

Szövegek közti címkék használata

Címke	Áttekintő megjegyzés állomány	Modul deklaráció	Csomag megjegyzés állomány	Típus-deklaráció	Konstruktor deklaráció	Metódus deklaráció	Adattag deklaráció
{@code}	☑	☑	☑	☑	☑	☑	☑
{@docRoot}	☑	☑	☑	☑	☑	☑	☑
{@index}	☑	☑	☑	☑	☑	☑	☑
{@inheritDoc}						☑	
{@link}	☑	☑	☑	☑	☑	☑	☑
{@linkplain}	☑	☑	☑	☑	☑	☑	☑
{@literal}	☑	☑	☑	☑	☑	☑	☑
{@return}						☑	
{@snippet}	☑	☑	☑	☑	☑	☑	☑
{@summary}	☑	☑	☑	☑	☑	☑	☑
{@systemProperty}	☑	☑	☑	☑	☑	☑	☑
{@value}	☑	☑	☑	☑	☑	☑	☑

Blokk címkék használata

Címke	Áttekintő megjegyzés állomány	Modul deklaráció	Csomag megjegyzés állomány	Típus- deklaráció	Konstruktor deklaráció	Metódus deklaráció	Adattag deklaráció
@author	☑	☑	☑	☑			
@deprecated		☑		☑	☑	☑	☑
@exception					☑	☑	
@hidden				☑		☑	☑
@param				☑	☑	☑	
@provides		☑					
@return						☑	
@see	☑	☑	☑	☑	☑	☑	☑
@serial			☑	☑			☑
@serialData						☑	
@serialField							☑
@since	☑	☑	☑	☑	☑	☑	☑
@throws					☑	☑	
@uses		☑					
@version	☑	☑	☑	☑			

Szövegek közti címkék (1)

- `{@code szöveg}`

- A szöveg megjelenítése fix szélességű betűtípussal (programkódként).
- A szövegben szabadon használható a `<` és `&` karakter.
- Példa:

```
{@code true},  
{@code e1.hashCode() == e2.hashCode() }
```

- `{@docRoot}`

- A generált állományokat tartalmazó könyvtárstruktúra gyökerének relatív elérési útvonalát szolgáltatja.
- Példa:

```
<a href="{@docRoot}/../copyright.html">Copyright</a>
```

Szövegek közti címkék (2)

- `{@index szó leírás}`
`{@index "kifejezés" leírás}`
 - Annak jelzésére szolgál, hogy az adott szó vagy kifejezés az adott opcionális leírással együtt meg kell, hogy jelenjen a standard doclet által létrehozott indexekben.
 - Példa:
`{@index JVM}`
`{@index JVM Java Virtual Machine}`
`{@index "RFC 3986"}`

Szövegek közti címkék (3)

- `{@inheritDoc}`
 - Dokumentáció másolása a „legközelebbi” őssztályból vagy implementált interfészből.
 - Csak metódushoz tartozó dokumentációs megjegyzésben használható az alábbi helyeken:
 - Fő leírásban (csak a fő leírás másolását eredményezi).
 - A `@return`, `@param` és `@throws` blokk címkékhez tartozó szövegben (csak a megfelelő címkehez tartozó szöveg másolását eredményezi).
 - Példa:

```
/**
 * {@inheritDoc}
 *
 * <p>This implementation always throws an
 * {@code UnsupportedOperationException}.
 */
```

Szövegeközi címkék (4)

- `{@link csomag.osztály#tag címke}`
 - Hiperhivatkozás az adott csomag, osztály, interfész, metódus, konstruktor vagy adattag dokumentációjára (megjelenítése fix szélességű betűtípussal).
 - Példa:
 - `{@link math}`
 - `{@link math.Fraction}`
 - `{@link math.Fraction#ZERO}`
 - `{@link math.Fraction#Fraction(long, long)}`
 - `{@link math.Fraction#add(long)}`
- `{@linkplain csomag.osztály#tag címke}`
 - Ugyanaz, mint a `{@link}`, csak normál betűtípussal történik a megjelenítés.

Szövegekőzi címkék (5)

- `{@literal szöveg}`
 - Ugyanaz, mint a `{@code}`, csak a szöveg megjelenítése normál betűtípussal történik.
- `{@return szöveg}`
 - A JDK 16-ban került bevezetésre szövegekőzi címkéként.
 - Szövegekőzi címkéként kizárólag metódus fő leírásának elején fordulhat elő.
 - Ebben az esetben a metódus fő leírásának első mondatát szolgáltatja, és egyben egy *Returns* szakaszt is, mintha a `@return` leírás is meg lenne adva.

Szövegközi címkék (6)

- `{@summary szöveg}`
 - A fő leírás első mondatát jelzi.
 - A címke csak akkor számít, ha a leírás elején szerepel.
 - Minden más esetben úgy történik a szöveg kezelése, mintha az a címke nélkül került volna megadásra.

Szövegek közti címkék (7)

- `{@systemProperty szöveg}`
 - Egy rendszertulajdonság nevének megadására szolgál.
 - Példa: `{@systemProperty java.version}`
- `{@value csomag.osztály#adattag}`
 - Az adott konstans adattag értékét szolgáltatja, csak `final` módosító esetén használható.
 - Példa:

```
/**
 * The value of this constant is {@value}.
 */
public static final int DEFAULT_VALUE = 7;
```

Szövegek közti címkék (8)

- `{@snippet attribútumok }`
`{@snippet attribútumok :`
 törzs `}`
- Kódrészlet beillesztése, a `<pre>` elem használatának fejlettebb alternatívájaként szolgál.
- Lehetőségek: külső állományok részének beillesztése, szöveg egy részének kiemelése, ...
- Példák:
 - ```
{@snippet :
public class Hello {
 public static void main(String... args) {
 System.out.println("Hello, World!");
 }
}}
```
  - `{@snippet class=Hello range=main}`

# Általános irányelvek a blokk címkék használatára

- A többször is előforduló blokk címkéket egy csoportban ajánlott megadni.
- Az alábbi sorrendben ajánlott a megadás:
  - `@author`, `@version`, `@param`, `@return`,  
`@exception/@throws`, `@see`,  
`@serial/@serialField/@serialData`,  
`@deprecated`

# Blokk címkék (1)

- `@author szöveg`
  - Szerzők nevének megadására szolgáló címke, melyből egy dokumentációs megjegyzésben több is előfordulhat.
  - Példa: `@author Péter Jeszenszky`
- `@deprecated szöveg`
  - Azt jelzi, hogy a dokumentált entitás elavult, használata kerülendő.
  - A szöveg első mondatában jelezni, hogy mióta, és a helyettesítőt is.
  - Használata esetén a dokumentált entitáshoz adjuk meg a `@Deprecated` annotációt is!
  - Egy dokumentációs megjegyzésben egyszer használni.
  - Példa:  
`@deprecated As of version 1.1, replaced by {@link #newMethod(int)}.`

# Blokk címkék (2)

- `@exception` *kivételosztály leírás*
  - Ugyanaz, mint a `@throws` címke.
- `@hidden`
  - A programelem elrejtése a generált API dokumentációból.
  - Példa:  
`/** @hidden */`

# Blokk címkék (3)

- `@param` *paraméternév leírás*
  - Az adott nevű paraméter leírást szolgáltatja.
  - Típusparaméter esetén a nevet `<` és `>` határoló karakterek között kell megadni.
  - Egy dokumentációs megjegyzésben egy adott nevű paraméterhez csak egyszer megadni.
  - A `@param` címkéket a paraméterek sorrendjének megfelelően ajánlott megadni.
  - Példa:  
`@param e the element to add`

# Blokk címkék (4)

- `@return` *leírás*
  - Metódus visszatérési értékét szolgáltatja.
  - Egy dokumentációs megjegyzésben csak egyszer megadni.

# Blokk címkék (5)

- *@see hivatkozás*
  - Hivatkozás elhelyezésére szolgál, többféle módon használható, egy dokumentációs megjegyzésben akár többször is előfordulhat:
    - *@see "szöveg"* módon, mint például:  
`@see "The Java Programming Language"`
    - *@see <a href="URI-hivatkozás">szöveg</a>* módon, mint például:  
`@see <a href="http://jcp.org/en/jsr/detail?id=173">JSR 173: Streaming API for XML</a>`
    - *@see csomag.osztály#tag* címke módon, a {@link} címkéhez hasonlóan, mint például:  
`@see java.lang.System#currentTimeMillis()`



# Blokk címkék (6)

- *@since szöveg*

- A szoftver azon verziójának megadására szolgál, melyben megjelent a dokumentált entitás.

- Példa:

`@since 1.2`

- *@throws kivételosztály leírás*

- Metódus vagy konstruktor által dobott kivétel dokumentálására szolgál.
- Több @throws címke esetén ajánlott a kivételosztályok neve alapján ábécé sorrendbe rendezni ezeket.

- Példa:

`@throws IOException if an I/O error occurs`

# Blokk címkék (7)

- `@version` szöveg
  - A szoftver verziójának a jelzésére szolgál, amelynek része a dokumentált entitás.
  - Egy dokumentációs megjegyzésben akár többször is előfordulhat.
  - Példa:  
`@version 4.12`

# API dokumentáció generálása (1): javadoc eszköz

- Az alábbi példa szemlélteti a program használatát:

```
javadoc -d build/apidocs \ célkönyvtár
-encoding UTF-8 \ a források karakterkódolása
-charset UTF-8 \ HTML karakterkódolás
-sourcepath src/main/java \ a forrásokat tartalmazó könyvtár
app.core app.gui csomagok, amelyekből dokumentációt kell generálni
```

# API dokumentáció generálása (2): Apache Maven

- Használjuk a *Maven Javadoc Plugin* javadoc célját, ehhez az `mvn javadoc:javadoc` parancsot kell végrehajtani.
- További információk:
  - <https://maven.apache.org/plugins/maven-javadoc-plugin/>
  - <https://github.com/apache/maven-javadoc-plugin>

# API dokumentáció generálása (2): Apache Maven

- Ha a `reporting` elemben hivatkozunk a bővítményre, akkor az API dokumentáció a webhely része lesz:

```
<reporting>
 <plugins>
 ...
 <plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-javadoc-plugin</artifactId>
 <version>3.6.3</version>
 </plugin>
 ...
 </plugins>
</reporting>
```

# API dokumentáció generálása: Gradle

- A Java Gradle bővítmény biztosítja a javadoc feladatot.
  - A feladat a `gradle javadoc` paranccsal hajtható végre.
- További információk:  
<https://docs.gradle.org/current/dsl/org.gradle.api.tasks.javadoc.Javadoc.html>

# API dokumentáció generálása (3): integrált fejlesztői környezetek

- ***Eclipse IDE:***

- Válasszuk az alábbi: *Project* → *Generate Javadoc...*

- ***IntelliJ IDEA:***

- Válasszuk az alábbi: *Tools* → *Generate Javadoc...*
  - Lásd: <https://www.jetbrains.com/help/idea/javadocs.html>

- ***NetBeans:***

- Válasszuk az alábbi: *Run* → *Generate Javadoc*

# Doclet (1)

- Egy olyan program, melyet a javadoc eszköz használ a kimenet előállításához.
  - Nem is feltétlenül API dokumentációt állít elő, szolgálhat egyéb speciális célokra.
- Doclet API:
  - A JDK `jdk.javadoc` modulja tartalmazza a `jdk.javadoc.doclet` csomagban.  
<https://docs.oracle.com/en/java/javase/21/docs/api/jdk.javadoc/jdk/javadoc/doclet/package-summary.html>
- Standard Doclet:
  - A Javadoc által alapértelmezésben használt doclet.



# Doclet (2)

- További hasznos docletek:
  - UML osztálydiagramokat előállító docletek:
    - *UMLDoclet* (licenc: *Apache License 2.0*)  
<https://github.com/talsma-ict/umldoclet>
    - *UMLGraph doclet* (licenc: *Simplified BSD License*)  
<https://www.spinellis.gr/umlgraph/>  
<https://github.com/dspinellis/UMLGraph>

# Kapcsolódó eszközök és szolgáltatások

- *Checkstyle* <https://checkstyle.org/>  
<https://github.com/checkstyle/checkstyle>
  - Lásd: <https://checkstyle.org/checks/javadoc/>
- *javadoc.io* <https://javadoc.io/>

# További információk

- *Javadoc Guide* <https://docs.oracle.com/en/java/javase/21/javadoc/>
- *How to Write Doc Comments for the Javadoc Tool*  
<https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>
- *Requirements for Writing Java API Specifications*  
<https://www.oracle.com/java/technologies/javase/api-specifications.html>
- *Documentation Comment Specification for the Standard Doclet (JDK 21)*  
<https://docs.oracle.com/en/java/javase/21/docs/specs/javadoc/doc-comment-spec.html>
- *javadoc Architecture*  
<https://openjdk.java.net/groups/compiler/javadoc-architecture.html>
- Jonathan Gibbons, Pavel Rappo. *Programmer's Guide to Snippets*. 2022.  
<https://docs.oracle.com/en/java/javase/19/code-snippet/>