Javadoc, JSdoc, PHPdoc...

- Documentació XML vs Javadoc
- Exemple classe C# en Javadoc

Documentació XML vs Javadoc

La **Documentació XML de C#** (amb `///`) s'inspira en el format **Javadoc** (utilitzat principalment en Java, i amb variants com **JSDoc** per a JavaScript i **PHPDoc** per a PHP). Tots aquests formats tenen l'objectiu comú d'incrustar la documentació directament al codi mitjançant etiquetes especials per a la generació automàtica de documentació.

Tot i que la sintaxi és diferent (XML amb `///` vs. `@etiqueta` amb `/**`), les funcions de les etiquetes principals es corresponen directament:

Funció de Documentació	C# (Documentació XML)	Java/JS/PHPDoc (Javadoc Style)
Descripció Principal	`<summary>`	Text lliure abans de la primera `@etiqueta`
Paràmetre	`<param name="nom">`	`@param {Tipus} nom`
Valor Retornat	`<returns>`	`@return {Tipus}`
Excepcions/Errors	`<exception cref="Tipus">`	`@throws Tipus` o `@exception Tipus`
Informació Addicional	`<remarks>`	Text lliure addicional o `@note`
Referència Creuada	`<see cref="Element">`	`@see Element`
Descripció de Propietat/Valor	`<value>`	Generalment inclòs a la descripció principal.

Correspondències

Mètodes/Funcions

C# (Documentació XML)	Javadoc / JSDoc / PHPDoc

`csharp`
`/// <summary>`
`/// Calcula el volum d'una esfera.`
`/// </summary>`
`/// <param name="radi">El radi de l'esfera.</param>`
`/// <returns>El volum calculat.</returns>`
`/// <seealso cref="System.Math.PI"/>`
 	
`public double CalcularVolum(double radi)`
	`java`
	`/**`
	`* Calcula el volum d'una esfera.`
	`*`
	`* @param radi El radi de l'esfera.`
	`* @return El volum calculat.`
	`* @see Math#PI`
	`*/`
	
	`public double calcularVolum(double radi)`

Les etiquetes

Etiqueta	Funció	C#	Javadoc/JSDoc/PHPDoc
Paràmetre	Identifica un argument d'entrada.	`<param name="radi">`	`@param radi`
Retorn	Descriu el valor que surt.	`<returns>`	`@return`
Referència	Enllaç a un altre element.	`<see cref="Tipus">`	`@see Tipus`

Aquesta similitud en el disseny fa que sigui relativament fàcil per a un desenvolupador canviar entre llenguatges, ja que el concepte de documentació de codi basada en etiquetes és consistent.

Exemple classe C# en Javadoc

Exemple d'una classe desenvolupada en C#, però **documentada utilitzant la sintaxi de Javadoc/JSDoc** (amb `/** ... */` i etiquetes amb `@`).

> ⚠ **RECORDA**: Encara que el codi sigui C#, la sintaxis del comentari no és l'estàndard de C# (Documentació XML).

C#

```
/**
 * Classe que representa un Lector (o Usuari) en un sistema de gestió de biblioteca.
 *
 * @author George Write
 * @version 1.1
 * @see GestorBiblioteca
 */
public class Lector
{
    /**
     * Identificador únic del lector.
     * <p>Aquest camp és privat i es llegeix a través de la propietat <c>ID</c>.</p>
     *
     * @type {int}
     */
    private int lectorID;

    /**
     * El nombre màxim de llibres que un lector pot tenir prestats simultàniament.
     * Aquest és un valor estàtic per a tots els lectors.
     *
     * @type {int}
     */
    public static int LIMIT_PRESTEC_LLIBRES = 5;

    /**
     * Constructor per inicialitzar un nou Lector.
     *
     * @param nomEl nom complet del lector.
     * @param identificador L'ID únic assignat.
     * @throws IllegalArgumentException Si l'identificador és negatiu.
     */
    public Lector(string nom, int identificador)
```

```
    {
        if (identificador < 0)
        {
            // Noteu que en C# normalment es llançaria ArgumentOutOfRangeException,
            // però aquí documentem l'equivalent de Javadoc.
            throw new ArgumentException("L'ID no pot ser negatiu.");
        }
        this.Nom = nom;
        this.lectorID = identificador;
    }

    /**
     * Propietat que obté el nom del lector.
     *
     * @return {string} El nom complet de l'usuari.
     */
    public string Nom { get; }

    /**
     * Mètode per simular el préstec d'un llibre.
     *
     * @param llibre El títol del llibre que es vol prestar.
     * @param dataLimit La data màxima de retorn del llibre.
     * @param {bool} haNotificacio Si s'ha d'enviar una notificació per correu
electrònic.
     * @returns {bool} Retorna 'true' si el préstec ha estat satisfactori.
     * @throws PrestecExcesLimitException Quan el lector ja ha superat el
<c>LIMIT_PRESTEC_LLIBRES</c>.
     * @see #LIMIT_PRESTEC_LLIBRES
     */
    public bool PrestarLlibre(string llibre, DateTime dataLimit, bool haNotificacio)
    {
        // ... (lògica de C#)
        return true;
    }
}

/**
 * Excepció personalitzada que es llança quan el lector intenta prestar
 * més llibres dels que el seu límit permet.
 *
 * @extends Exception
 */
public class PrestecExcesLimitException : Exception
{
    // ...
}
```

Documentació Javadoc en C#

A continuació, s'explica com s'han utilitzat les etiquetes Javadoc per als diferents elements de C#:

Element C#	Etiqueta Javadoc Utilitzada	Explicació de la Correspondència
Classe (`Lector`)	`@author`, `@version`, `@see`	Proporcionen metadades i referències creuades.
Atribut (Camp) (`lectorID`)	`@type {int}`	S'utilitza la sintaxi de **JSDoc/ PHPDoc** per especificar el tipus de dades d'un camp.
Propietat (`Nom`)	`@return {string}`	Es documenta com si fos una funció que només retorna un valor.
Mètode (`PrestarLlibre`)	`@param`, `@returns`, `@throws`	Les etiquetes principals són idèntiques en funció a les de C# XML (`<param>`, `<returns>`, `<exception>`).
Excepció (`PrestecExcesLimitException`)	`@extends Exception`	Indica la classe de la qual hereta l'excepció (equivalent a l'etiqueta **JSDoc**).
Especificar Tipus (Paràmetre)	`@param {bool} haNotificacio`	S'utilitza la sintaxi de **JSDoc/ PHPDoc** per especificar el tipus entre claus (`{}`).
Referència Creuada	`@see #LIMIT_PRESTEC_LLIBRES`	S'utilitza la sintaxi de Javadoc per referenciar membres dins de la mateixa classe.

> **ATENCIÓ:** Utilitzar aquest tipus de documentació està generalment **desaconsellada** en un projecte de C#, ja que el compilador i les eines no podran generar el fitxer XML de documentació ni oferir la funcionalitat d'IntelliSense completa de la mateixa manera que ho farien amb la Documentació XML.

Com seria amb Documentació XML de C#?

La mateixa classe `Lector` desenvolupada en C#, però aquesta vegada documentada correctament utilitzant l'estàndard de **Documentació XML de C#** (amb `///`).

Aquesta és la manera en què s'ha de documentar el codi de C# per garantir la compatibilitat total amb l'IntelliSense de Visual Studio i les eines de generació de documentació.

C#

```
/// <summary>
/// Representa un Lector (o Usuari) en un sistema de gestió de biblioteca.
/// </summary>
/// <remarks>
/// Aquesta classe encapsula la informació bàsica d'un usuari i les regles de préstec.
/// </remarks>
/// <seealso cref="GestorBiblioteca"/>
public class Lector
{
    /// <summary>
    /// Identificador únic del lector.
    /// </summary>
    /// <remarks>
    /// Aquest camp és privat i es llegeix a través d'una propietat pública si és
necessari.
    /// </remarks>
    private int lectorID;

    /// <summary>
    /// El nombre màxim de llibres que un lector pot tenir prestats simultàniament.
    /// </summary>
    /// <value>
    /// Un <see cref="int"/> que representa el límit per a tots els lectors.
    /// </value>
    public static int LIMIT_PRESTEC_LLIBRES = 5;

    /// <summary>
    /// Constructor per inicialitzar un nou <see cref="Lector"/>.
    /// </summary>
    /// <param name="nom">El nom complet del lector.</param>
    /// <param name="identificador">L'ID únic assignat.</param>
    /// <exception cref="ArgumentOutOfRangeException">Es llança si l'identificador és
negatiu.</exception>
    public Lector(string nom, int identificador)
    {
        if (identificador < 0)
        {
            throw new ArgumentOutOfRangeException(nameof(identificador), "L'ID no pot
ser negatiu.");
        }
        this.Nom = nom;
        this.lectorID = identificador;
    }

    /// <summary>
    /// Obté el nom complet del lector.
    /// </summary>
    /// <value>
    /// El nom complet de l'usuari com a <see cref="string"/>.
    /// </value>
```

```
    public string Nom { get; }

    /// <summary>
    /// Mètode per simular el préstec d'un llibre al lector.
    /// </summary>
    /// <param name="llibre">El títol del llibre que es vol prestar.</param>
    /// <param name="dataLimit">La data màxima de retorn del llibre, de tipus <see
cref="DateTime"/>.</param>
    /// <param name="haNotificacio">Indica si s'ha d'enviar una notificació per correu
electrònic.</param>
    /// <returns>
    /// <c>true</c> si el préstec ha estat satisfactori; <c>false</c> en cas d'error o
límit.
    /// </returns>
    /// <exception cref="PrestecExcesLimitException">Quan el lector ja ha superat el
<see cref="LIMIT_PRESTEC_LLIBRES"/>.</exception>
    public bool PrestarLlibre(string llibre, DateTime dataLimit, bool haNotificacio)
    {
        // ... (lògica de C#)
        return true;
    }
}

/// <summary>
/// Excepció personalitzada que es llança quan el lector intenta prestar
/// més llibres dels que el seu límit permet.
/// </summary>
/// <seealso cref="Lector.LIMIT_PRESTEC_LLIBRES"/>
public class PrestecExcesLimitException : Exception
{
    // ...
}
```

Que canvia?

Element	Format Javadoc	Format XML Documentation (C#)
Delimitador	`/** ... */`	`///`
Descripció general	Text lliure	Etiqueta `<summary>`
Paràmetre	`@param nom`	Etiqueta `<param name="nom">`
Valor retornat	`@return`	Etiqueta `<returns>`
Tipus (Referència)	`{bool}` (sintaxi JSDoc)	Etiqueta `<see cref="bool"/>` o `<see cref="DateTime"/>`
Excepció	`@throws Tipus`	Etiqueta `<exception cref="Tipus">`

Element	Format Javadoc	Format XML Documentation (C#)
Valor de propietat	S'implica a `@return`	Etiqueta `<value>`

Totes les etiquetes de C# són elements XML ben formats, cosa que permet que el compilador pugui processar-los amb èxit.