Information systems- design & development
Computer science degree

What's Javadoc?

Javadoc is a tool included in the Java Development Kit (JDK) used to generate HTML documentation directly from Java source code. It is designed to help developers document their classes, methods, and other program elements clearly and structurally, following a standard.

Javadoc comments follow a standard structure that makes them easy to read and convert into HTML.

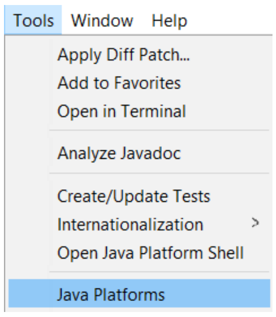
- Javadoc documentation should start with a slash and double asterisk (/**) and end with an asterisk and single slash (*/).
- Reserved words (tags) preceded by the @ character are used to create the documentation.
- The comment must always be declared immediately before the class, method, or function it is associated with.

How to comment javadoc:

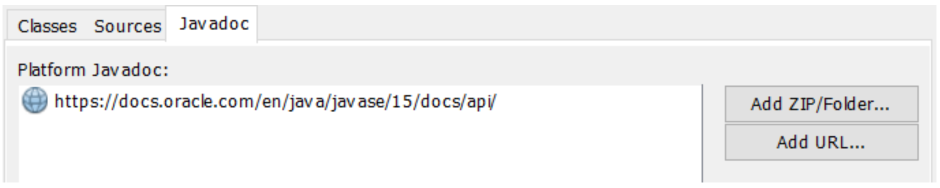
```
/**
* This is a Javadoc comment.
* @param explains information related to the parameters of the class, method, etc.
* @return describes information about what is returned.
* @throws provides details about the exceptions that may be thrown.
*/
```

Loading Javadoc

Javadoc is loaded by default after installing JDK in NetBeans. You can verify its presence by selecting **Tools -> Java Platforms.**

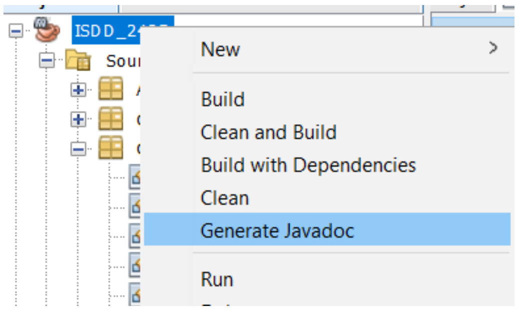


If Javadoc is not installed, simply type the URL manually in the window shown in the Java Platforms section.



Platform Javadoc:
https://docs.oracle.com/en/java/javase/15/docs/api/

How to generate Javadoc documentation for the project?
To generate Javadoc, right-click on the project and select "Generate Javadoc".

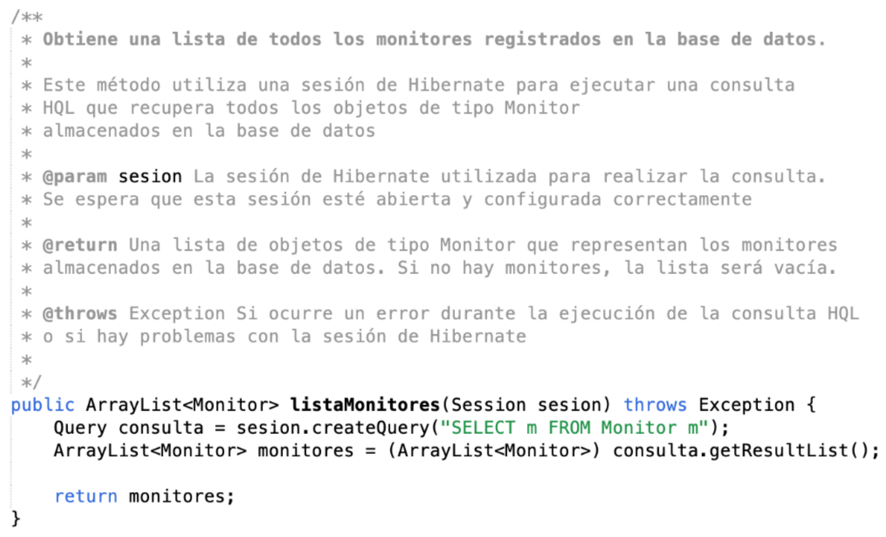


For projects generated with Maven, the documentation is generated in the following path:

<project_path>\target\reports\apidocs

Example of Javadoc for a method or function:

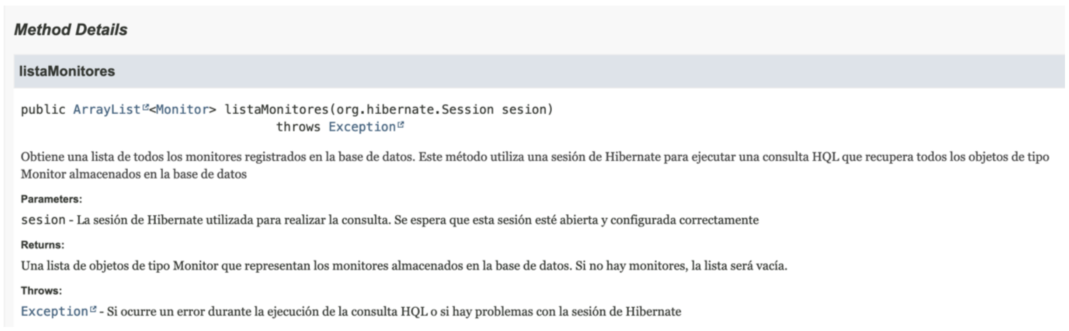
NetBeans Code:



```
/**
 * Obtiene una lista de todos los monitores registrados en la base de datos.
 *
 * Este método utiliza una sesión de Hibernate para ejecutar una consulta
 * HQL que recupera todos los objetos de tipo Monitor
 * almacenados en la base de datos
 *
 * @param sesion La sesión de Hibernate utilizada para realizar la consulta.
 * Se espera que esta sesión esté abierta y configurada correctamente
 *
 * @return Una lista de objetos de tipo Monitor que representan los monitores
 * almacenados en la base de datos. Si no hay monitores, la lista será vacía.
 *
 * @throws Exception Si ocurre un error durante la ejecución de la consulta HQL
 * o si hay problemas con la sesión de Hibernate
 *
 */
public ArrayList<Monitor> listaMonitores(Session sesion) throws Exception {
    Query consulta = sesion.createQuery("SELECT m FROM Monitor m");
    ArrayList<Monitor> monitores = (ArrayList<Monitor>) consulta.getResultList();

    return monitores;
}
```

Generated Javadoc:
In the MonitorDAO class, the "Method Details" section displays the description written in the code.



Method Details

listaMonitores

```
public ArrayList<Monitor> listaMonitores(org.hibernate.Session sesion)
                                  throws Exception
```

Obtiene una lista de todos los monitores registrados en la base de datos. Este método utiliza una sesión de Hibernate para ejecutar una consulta HQL que recupera todos los objetos de tipo Monitor almacenados en la base de datos

Parameters:

`sesion` - La sesión de Hibernate utilizada para realizar la consulta. Se espera que esta sesión esté abierta y configurada correctamente

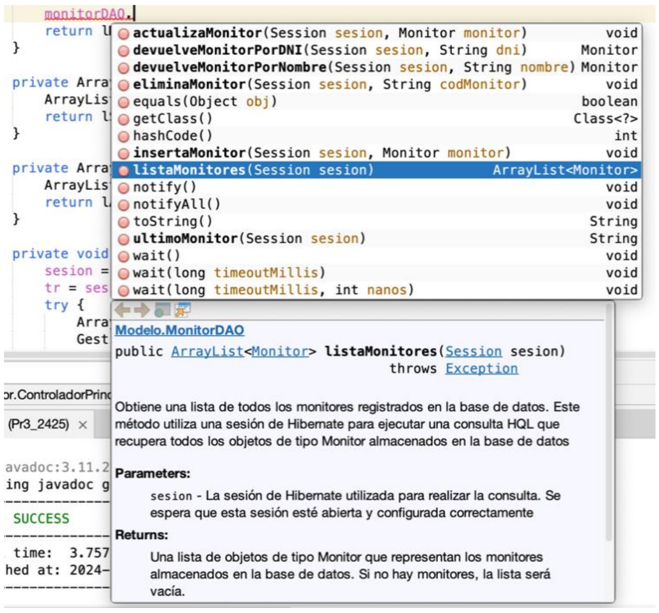
Returns:

Una lista de objetos de tipo Monitor que representan los monitores almacenados en la base de datos. Si no hay monitores, la lista será vacía.

Throws:

`Exception` - Si ocurre un error durante la ejecución de la consulta HQL o si hay problemas con la sesión de Hibernate

Contextual help in NetBeans:



```
monitorDAO.
actualizaMonitor(Session sesion, Monitor monitor)                void
devuelveMonitorPorDNI(Session sesion, String dni)             Monitor
devuelveMonitorPorNombre(Session sesion, String nombre) Monitor
eliminaMonitor(Session sesion, String codMonitor)                void
equals(Object obj)                                            boolean
getClass()                                                   Class<?>
hashCode()                                                        int
insertaMonitor(Session sesion, Monitor monitor)                  void
listaMonitores(Session sesion)                     ArrayList<Monitor>
notify()                                                         void
notifyAll()                                                      void
toString()                                                     String
ultimoMonitor(Session sesion)                                  String
wait()                                                           void
wait(long timeoutMillis)                                         void
wait(long timeoutMillis, int nanos)                              void
```

Modelo.MonitorDAO

```
public ArrayList<Monitor> listaMonitores(Session sesion)
                          throws Exception
```

Obtiene una lista de todos los monitores registrados en la base de datos. Este método utiliza una sesión de Hibernate para ejecutar una consulta HQL que recupera todos los objetos de tipo Monitor almacenados en la base de datos

Parameters:

`sesion` - La sesión de Hibernate utilizada para realizar la consulta. Se espera que esta sesión esté abierta y configurada correctamente

Returns:

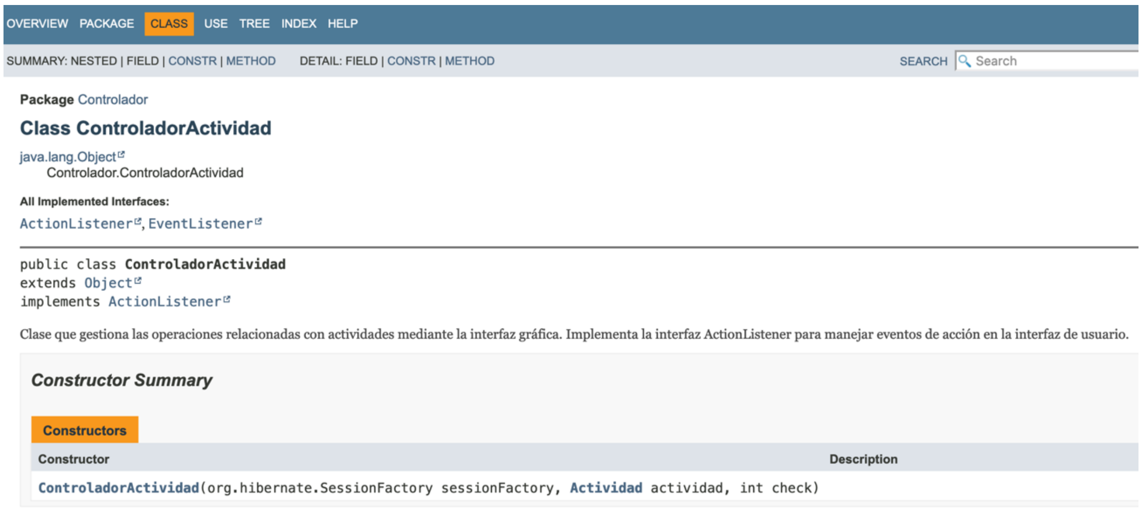
Una lista de objetos de tipo Monitor que representan los monitores almacenados en la base de datos. Si no hay monitores, la lista será vacía.

Example of Javadoc for a class:

NetBeans Code:

```
/**
 * Clase que gestiona las operaciones relacionadas
 * con actividades mediante la interfaz gráfica.
 * Implementa la interfaz ActionListener para manejar
 * eventos de acción en la interfaz de usuario.
 */
public class ControladorActividad implements ActionListener {
```

Generated HTML code:



OVERVIEW PACKAGE CLASS USE TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH Search

Package Controlador

Class ControladorActividad

java.lang.Object
    Controlador.ControladorActividad

All Implemented Interfaces:

`ActionListener`, `EventListener`

```
public class ControladorActividad
extends Object
implements ActionListener
```

Clase que gestiona las operaciones relacionadas con actividades mediante la interfaz gráfica. Implementa la interfaz ActionListener para manejar eventos de acción en la interfaz de usuario.

Constructor Summary

Constructors

Constructor	Description
`ControladorActividad(org.hibernate.SessionFactory sessionFactory, Actividad actividad, int check)`	