COURSEWARE

JavaDoc

Contents

Overview

Java provides a tool called **JavaDoc** that can automatically create HTML-based documentation based on comments in your source files.

All you have to do is add a comment for each public class, field, and method; then run the source files through the `javadoc` command, which will create professional-looking web-based documentation for all your classes.

Tutorial

JavaDoc Comments

JavaDoc comments begin with `/**` and end with `*/`

Each subsequent line of a multiline JavaDoc comment usually begins with an asterisk (`*`). JavaDoc ignores this asterisk and any white space between it and the first word on the line.

You can place JavaDoc comments in any of three different locations in a source file:

- Immediately before the declaration of a public class
- Immediately before the declaration of a public field
- Immediately before the declaration of a public method or constructor

The text in a JavaDoc comment can include HTML markup, although, you should avoid using heading tags (`<h1>`, `<h2>`, etc.) as JavaDoc will add these for you.

You can use bold and italics (`<b>`, `<i>`) and to show code examples use the `<pre>` tag.

You can also include special doc tags that provide specific information used by JavaDoc to format the documentation pages.

Tag	Explanation
`@author`	Provides information about the author, typically the author's name, e-mail address, website information, and so on.
`@version`	Indicates the version number.
`@since`	Used to indicate the version with which this class, field, or method was added.
`@param`	Provides the name and description of a method or constructor.

Tag	Explanation
`@return`	Provides a description of a method's return value.
`@throws`	Indicates exceptions that are thrown by a method or constructor.
`@deprecated`	Indicates that the class, field, or method is deprecated and shouldn't be used.

```java
/**
 * @author
 * Your Name Here
 */
@Service
public class NoteService {

    @Autowired
    private NoteRepository noteRepository;

    /**
     * Get all notes in the database
     * @return
     * List of all Note objects
     */
    public List<Note> getNotes(){
        return noteRepository.findAll();
    }

    /**
     * Used to save a Note into the database
     * @param note
     * Whether id is present makes no difference
     * @return
     * a Note, with a given id from the database.
     */
    public Note createNote(Note note){
        note.setId(null);
        return noteRepository.saveAndFlush(note));
    }

    /**
     * Updates a note text.
     * @param note
     * A note . ID must be present
     * @return
     * Returns the updates object
     * @throws NotFoundException
     */

    /**
     * Used to save a Note into the database
     * @param note
     * a Note a DTO. Whether id is present makes no difference
     * @return
     * a Note DTO, with a given id from the database.
     */
    public Note createNote(Note note){
        note.setId(null);
        Note note = new Note();
        note.setText(note.getText());

        return new Note(noteRepository.saveAndFlush(note));
    }

    /**
     * Updates a note text.
     * @param note
     */
    public Note updateNote(Note note) throws NotFoundException{
        Note note = noteRepository.findById(note.getId()).orElseThrow(() -> new
NotFoundException());
        note.setText(note.getText());
        noteRepository.flush();
        return new Note(note);
    }

    /**
     * Deletes a note text.
     * @param id
     * id of the note which must be deleted.
```

```
     * @return
     * returns an empty object.
     * @throws NotFoundException
     * If id is not found in database it will throw the exception
     */
    public Note deleteNote(Long id) throws NotFoundException{
        Note note = noteRepository.findById(id).orElseThrow(() -> new
NotFoundException());
        Note note = new Note(note);
        noteRepository.deleteById(id);
        return note;
    }
}
```

JavaDoc Command

The format of the JavaDoc command is:

```
javadoc [options] [package_names] [source_files]
```

So, for us to generate documentation for the above code the command would be:

```
javadoc /*path*/*to*/*file*/NoteService.java
```

For a full explanation please see the official Oracle documentation.

Exercises

1. Take a previous Java class you have created and add JavaDoc comments to it. Make sure to include:
    - `@author` information
    - `@param`
    - `@return` (If needed)
    - `@throws` (If needed)
2. Run the JavaDoc command on your newly commented code and watch the magic.