

[CS304] Tutorial 10 - Software Documentation

Part 1 JavaDoc

JavaDoc tool is a document generator tool in Java programming language for generating standard documentation in HTML format. It generates API documentation. It parses the declarations and documentation in a set of source file describing classes, methods, constructors, and fields.

Before using JavaDoc tool, you must include JavaDoc comments `/**.....*/` providing information about classes, methods, and constructors, etc. For creating a good and understandable document API for any java file you must write better comments for every class, method, constructor.

Example :

Syntax	Parameter	Description
@author	author_name	Describes an author
@param	description	provide information about method parameter or the input it takes
@version	version-name	provide version of the class, interface or enum.
@return	description	provide the return value

More Tag information links(Official links):

<https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html>

Part 1-1 Start a simple Java program

Create a simple Java program (int)a+b, please refer attachment(example_01.java), run command-line below the java path:

```
javadoc -d .\javadoc -author -version -encoding UTF-8 -charset UTF-8
example_01.java
```

public class `example_01`
extends `Object`

版本:
jdk1.8.0
作者:
CS304

构造器概要

构造器	说明
<code>example_01()</code>	

方法概要

所有方法	静态方法	具体方法
修饰符和类型	方法	说明
static int	<code>add(int n, int m)</code>	This is a program for adding two numbers in java.
static void	<code>main(String[] args)</code>	This is the main method which is very important for execution for a java program.

从类继承的方法 `java.lang.Object`

About command you can refer oracle document below:

<https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html>

`javadoc {packages|source-files} [options] [@argfiles]`

Part 1-2 Create apidocs for Teedy Project

Generate standalone javadocs for the project

You could add the **Javadoc** Plugin in the section of your pom (if no configuration defined, the plugin uses default values).

Add the following into the pom.xml of Teedy, Then, run `mvn javadoc:javadoc`, which generates apidocs in html format in `target/site/apidocs/index.html` in each module directory.

```
<project>
  ...

  <build>
    <plugins>

      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <version>3.6.3</version>
        <configuration>
          <failOnError>>false</failOnError>
        </configuration>
        <executions>
          <execution>
```

```

        <id>aggregate</id>
        <goals>
            <goal>aggregate</goal>
        </goals>
        <phase>site</phase>
    </execution>
</executions>
</plugin>

</plugins>
</build>

...
</project>

```

reference :

<https://maven.apache.org/plugins/maven-javadoc-plugin/index.html>

Generate Javadocs As Part Of Project Reports

To generate javadocs as part of the site generation, you should add the **Javadoc** Plugin in the section of your pom: add the following into the pom.xml of Teedy, Then, run **mvn site**, which generates apidocs in html format in target/site/apidocs/index.html in root and module directory.

```

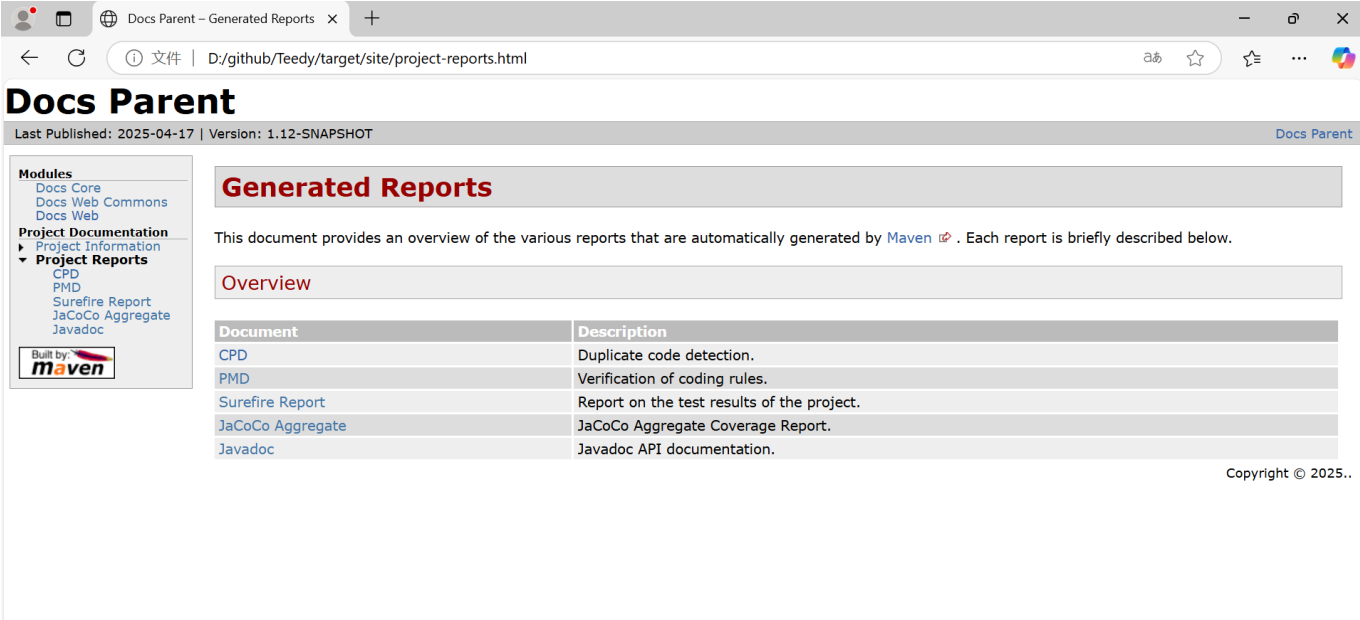
<project>
    ...

    <reporting>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-javadoc-plugin</artifactId>
                <version>3.6.3</version>
                <configuration>
                    <failOnError>>false</failOnError>
                </configuration>
                <reportSets>
                    <reportSet>
                        <id>aggregate</id>
                        <inherited>>false</inherited>
                        <reports>
                            <report>aggregate</report>
                        </reports>
                    </reportSet>
                    <reportSet>
                        <id>default</id>
                        <reports>
                            <report>javadoc</report>
                        </reports>
                    </reportSet>
                </reportSets>
            </plugin>
        </plugins>
    </reporting>

```

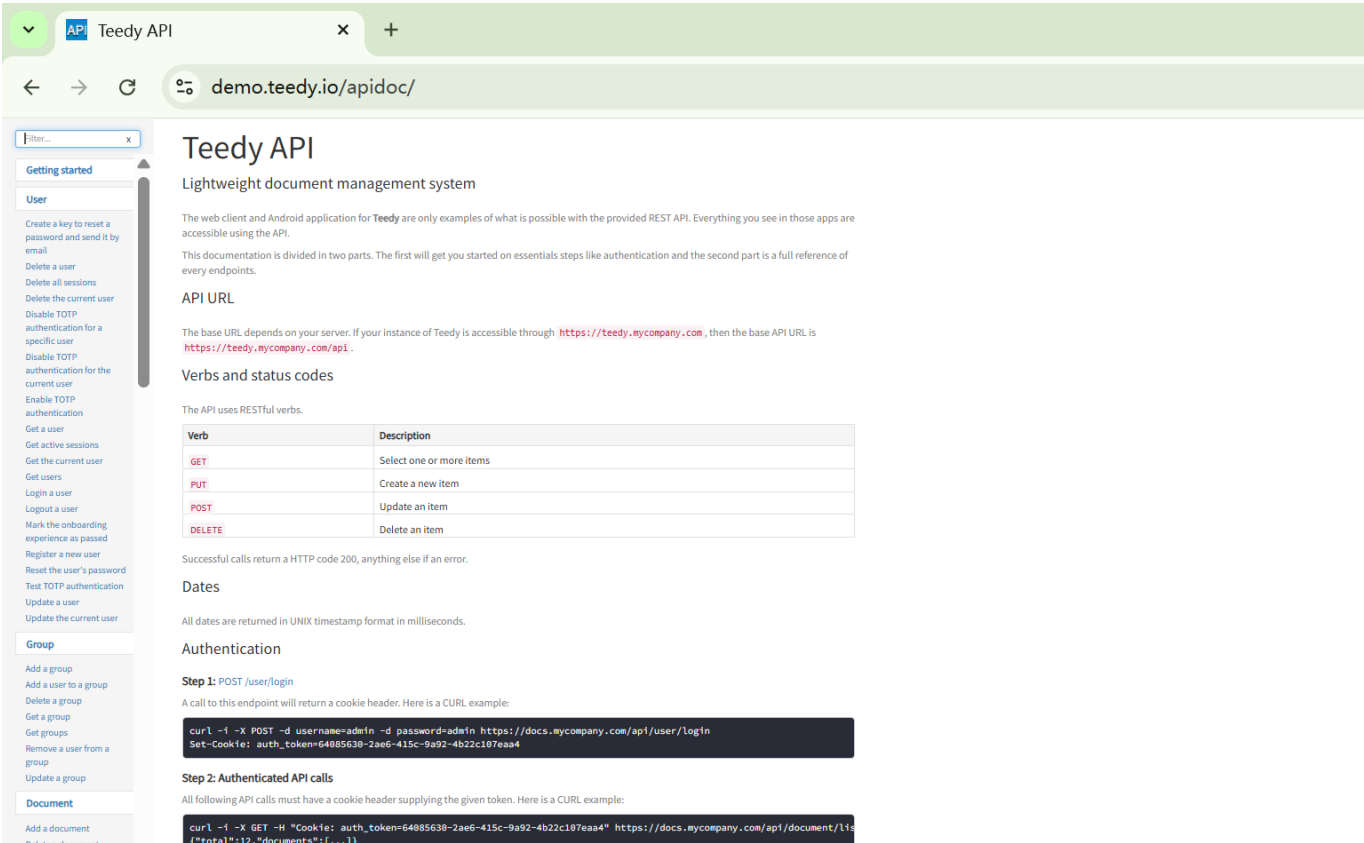
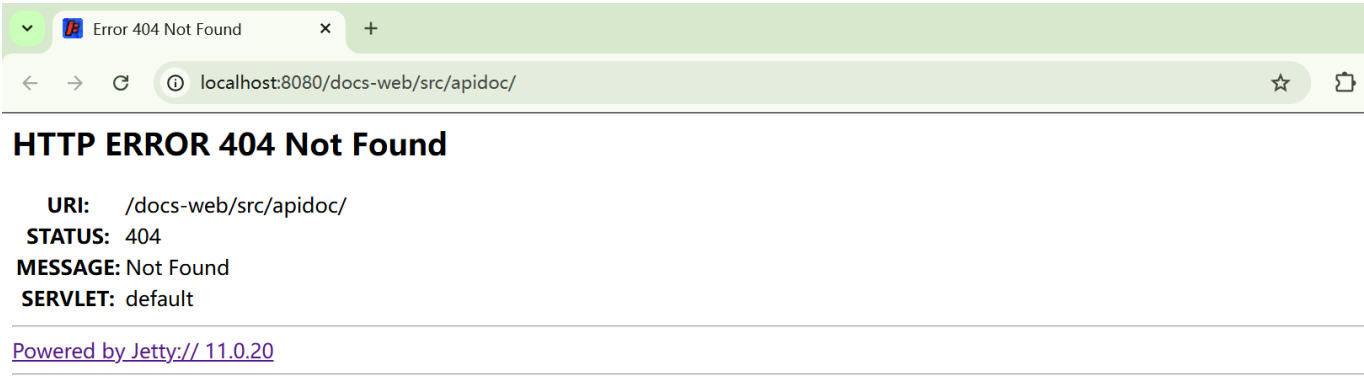
```
        </reportSets>
        </plugin>
    </plugins>
</reporting>

...
</project>
```



Part 2: Deploying Teedy in Production Mode

In our previous practices, we deployed Teedy in development mode. When we clicked the API documentation link in that setup, it returned a 404 error. However, the demo version of Teedy displays the REST API documentation successfully. Why is there such a difference?



This is because our earlier deployment was done in **development mode**, while the demo runs in **production mode**. Defining these two modes provides several benefits:

- Improves development efficiency
- Enhances system security
- Avoids production incidents
- Allocates system resources properly

In short, **Dev mode is designed for developers**, while **Prod mode is meant for users and operational stability**.

Configuration Differences Between Dev and Prod

Let's briefly look at how the `docs-web` submodule of Teedy defines its `dev` and `prod` profiles in `pom.xml`:

Maven Profiles Overview

```
<profiles>
  <!-- Development profile -->
  <profile>
    <id>dev</id>
    <activation>
      <activeByDefault>true</activeByDefault>
      <property>
        <name>env</name>
        <value>dev</value>
      </property>
    </activation>
    <build>
      <resources>
        <resource>
          <directory>src/dev/resources</directory>
          <filtering>false</filtering>
          <excludes>
            <exclude>*/config.properties</exclude>
          </excludes>
        </resource>
        <resource>
          <directory>src/dev/resources</directory>
          <filtering>true</filtering>
          <includes>
            <include>*/config.properties</include>
          </includes>
        </resource>
      </resources>
      <plugins>
        <plugin>
          <groupId>org.eclipse.jetty</groupId>
          <artifactId>jetty-maven-plugin</artifactId>
          <configuration>
            <systemProperties>
              <application.mode>dev</application.mode>
              <docs.home>../data/docs</docs.home>
            </systemProperties>
            <webApp>
              <contextPath>/docs-web</contextPath>
              <overrideDescriptor>${project.basedir}/src/dev/main/webapp/web-
override.xml</overrideDescriptor>
            </webApp>
          </configuration>
        </plugin>
      </plugins>
    </build>
  </profile>
</profiles>
```

```

        </configuration>
      </plugin>
    </plugins>
  </build>
</profile>

<!-- Production profile -->
<profile>
  <id>prod</id>
  <build>
    <resources>
      <resource>
        <directory>src/prod/resources</directory>
        <filtering>false</filtering>
        <excludes>
          <exclude>**/config.properties</exclude>
        </excludes>
      </resource>
      <resource>
        <directory>src/prod/resources</directory>
        <filtering>true</filtering>
        <includes>
          <include>**/config.properties</include>
        </includes>
      </resource>
    </resources>
    <plugins>
      <!-- Launch NPM & Grunt -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-antrun-plugin</artifactId>
        <executions>
          <execution>
            <phase>generate-sources</phase>
            <configuration>
              <target name="building">
                <exec executable="npm" dir="${project.basedir}/src/main/webapp"
osfamily="unix" failonerror="true">
                  <arg line="install" />
                </exec>
                <exec executable="grunt"
dir="${project.basedir}/src/main/webapp" osfamily="unix" failonerror="true">
                  <arg line="--apiurl=api" />
                </exec>
              </target>
            </configuration>
            <goals>
              <goal>run</goal>
            </goals>
          </execution>
        </executions>
      </plugin>

      <!-- WAR generation -->

```

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-war-plugin</artifactId>
  <configuration>

<warSourceDirectory>${project.basedir}/src/main/webapp/dist</warSourceDirectory>
  <webXml>src/main/webapp/WEB-INF/web.xml</webXml>
  </configuration>
</plugin>
</plugins>
</build>
</profile>
</profiles>
```

Key Differences

1. Resource Paths

- **Dev** uses `src/dev/resources`.
- **Prod** uses `src/prod/resources`.

Both allow filtering of `config.properties`.

2. Plugin Configuration

- **Dev**: Uses `jetty-maven-plugin` to quickly launch an embedded Jetty server.
- **Prod**: Uses `maven-antrun-plugin` to run `npm install` and `grunt` to build frontend resources. Uses `maven-war-plugin` to package a WAR file.

3. System Properties

- **Dev**: Sets `application.mode=dev` and `docs.home=../data/docs`.
- **Prod**: Relies on default production properties.

4. Build Goals

- **Dev**: Optimized for fast development and testing.
- **Prod**: Optimized for bundling, minification, and deployment.

Deploying Teedy in Production Mode Locally

1. Prerequisites

Make sure the Teedy environment is properly installed. For details, refer to **Tutorial 2 → Part 2: Teedy Native Installation → Environment Setup**.

Run the following command to globally install the Grunt CLI:


```
sudo npm install -g grunt-cli
```

This installs the Grunt command-line interface, allowing you to run Grunt tasks defined in the `Gruntfile.js`, which include:

- Cleaning temporary directories
- Annotating AngularJS files
- Concatenating and minifying JS/CSS
- Compiling LESS
- Handling Angular templates
- Generating REST API documentation

Refer to `docs-web/src/main/webapp/Gruntfile.js` for full task details. Make sure the both npm and Grunt are working correctly:

```
ao_hao@DESKTOP-4JVIQPU:/mnt/d/github/Teedy$ npm --version
9.2.0
ao_hao@DESKTOP-4JVIQPU:/mnt/d/github/Teedy$ grunt --version
grunt-cli v1.5.0
grunt v1.6.1
```

2. Build and Run Production Mode

Activate the `prod` profile explicitly:

```
mvn clean install -Pprod -DskipTests
mvn jetty:run-war -Pprod -DskipTests
```

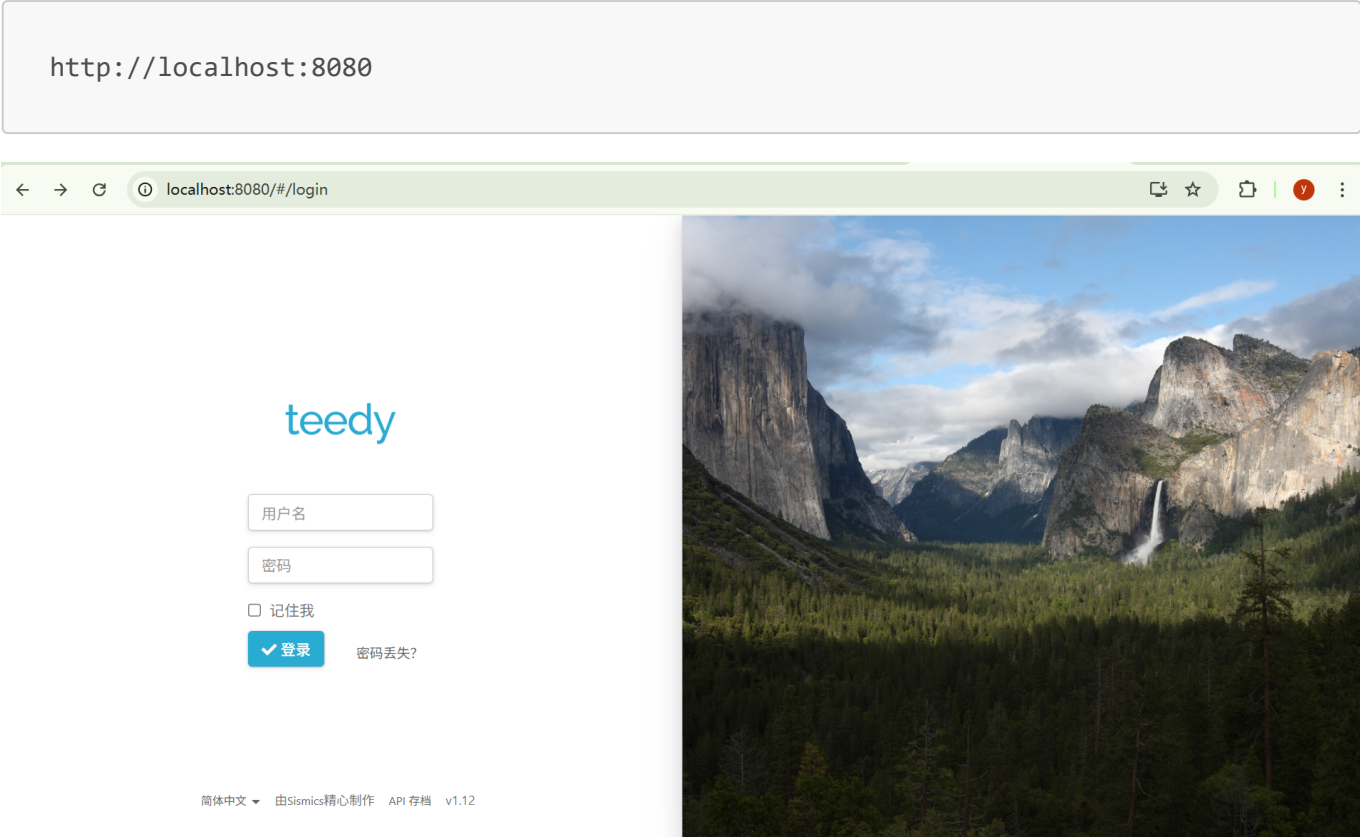
```
base. Only suitable for testing purpose, not for production!
15 Apr 2025 20:05:45,673 INFO com.sismics.util.ClasspathScanner.findClasses(ClasspathScanner.java:48) Found 1 classes for
r IndexingHandler
15 Apr 2025 20:05:45,967 INFO com.sismics.docs.core.util.indexing.LuceneIndexingHandler.initLucene(LuceneIndexingHandler
.java:132) Using file Lucene storage: /var/docs/lucene
15 Apr 2025 20:05:46,556 INFO com.sismics.docs.core.util.indexing.LuceneIndexingHandler.initLucene(LuceneIndexingHandler
.java:144) Checking index health and version
15 Apr 2025 20:05:46,676 INFO com.sismics.docs.core.service.FileService.startUp(FileService.java:39) File service starti
ng up
15 Apr 2025 20:05:46,713 INFO com.sismics.docs.core.service.InboxService.startUp(InboxService.java:52) Inbox service sta
rting up
15 Apr 2025 20:05:46,723 INFO com.sismics.docs.core.service.FileSizeService.startUp(FileSizeService.java:30) File size s
ervice starting up
15 Apr 2025 20:05:48,496 INFO com.sismics.docs.core.service.FileSizeService.lambda$runOneIteration$0(FileSizeService.jav
a:50) No more file to process, stopping the service
15 Apr 2025 20:05:48,504 INFO com.sismics.docs.core.service.FileSizeService.shutdown(FileSizeService.java:35) File size
service shutting down
[INFO] Started o.e.j.m.p.MavenWebApplicationContext@73a49597{Teedy,/,file:///mnt/d/github/Teedy-doc/docs-web/target/docs-web-1.1
2-SNAPSHOT/,AVAILABLE} {/mnt/d/github/Teedy-doc/docs-web/target/docs-web-1.12-SNAPSHOT.war}
[INFO] Started ServerConnector@7764db24{HTTP/1.1,(http/1.1)}{0.0.0.0:8080}
[INFO] Started Server@4db46344{STARTING}[11.0.20,sto=0] @291612ms

Hit <enter> to redeploy:

15 Apr 2025 20:06:33,563 INFO com.sismics.util.ClasspathScanner.findClasses(ClasspathScanner.java:48) Found 2 classes for
r AuthenticationHandler
```

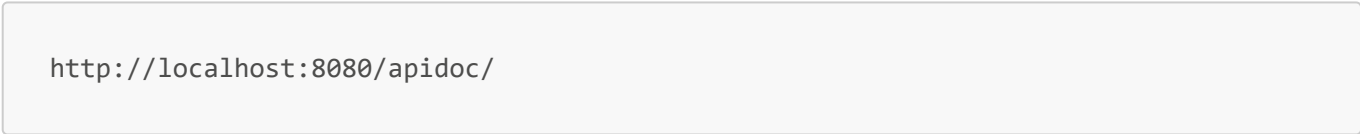
3. Access the Production Instance

Open your browser and go to:

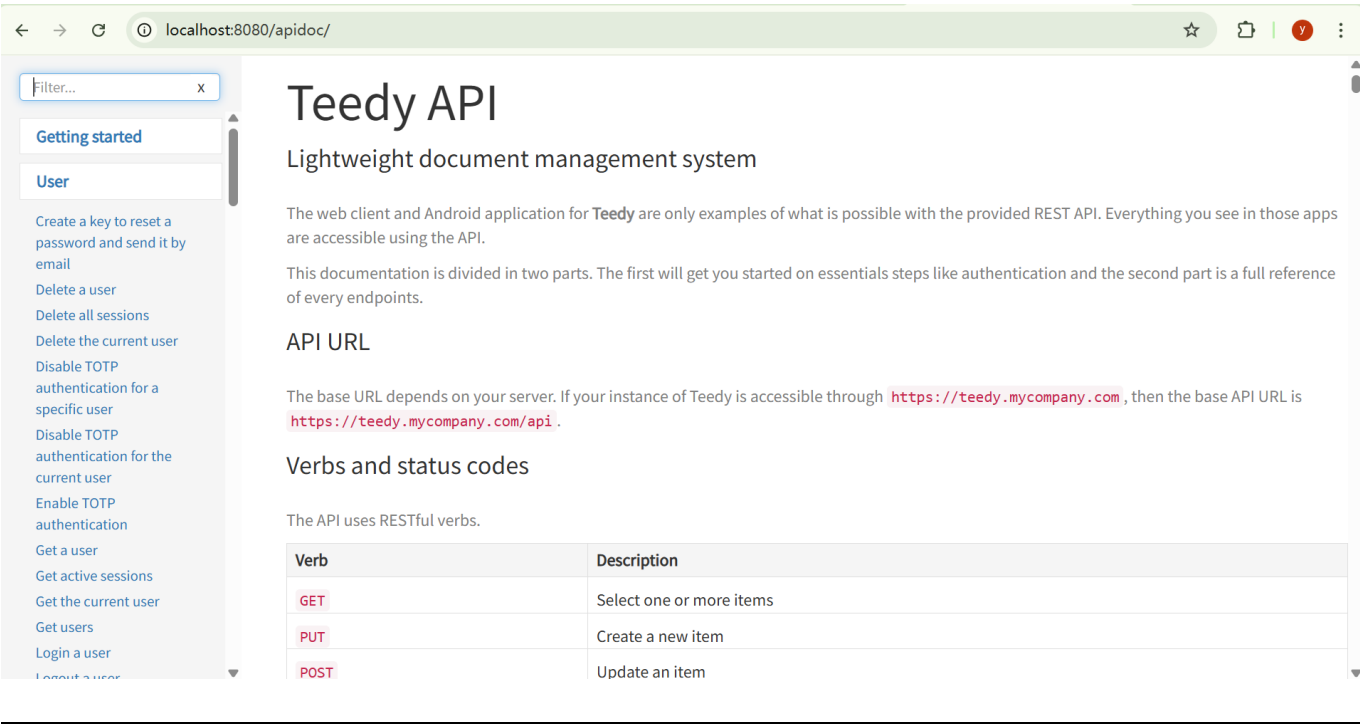


4. Access the REST API Documentation

Click the **API 存档** link at the bottom, or visit:



You should now see the Grunt-generated API documentation.



Part 3: Teedy REST API Documentation Generation

1. Setup and Build

In the `docs-web` directory, the `pom.xml` refers to two core commands used to install packages and trigger the build process:

```
# Run from the docs-web directory
npm install
# Build the project
grunt --apiurl=api
```

Running `npm install` without arguments automatically installs all dependencies listed under both **dependencies** and **devDependencies** in the `package.json` file into the `node_modules` directory.

The `grunt` command initiates the build process according to the configuration defined in `Gruntfile.js`, which includes generating the REST API documentation.

2. Key configuration

`package.json` Excerpt

```
"devDependencies": {
  "grunt-apidoc": "^0.11.0"
},
"apidoc": {
  "name": "Teedy API",
  "title": "Teedy API",
  "url": "/api",
  "template": {
    "withCompare": false,
    "withGenerator": false
  },
  "order": [
    "User",
    "Group",
    "Document",
    "File",
    "Tag",
    "Comment",
    "Share",
    "Acl",
    "Auditlog",
    "App",
    "Theme",
    "Vocabulary"
  ],
  "header": {
    "title": "Getting started",
```

```

        "filename": "header.md"
    }
},

```

Teedy uses a `Gruntfile.js` located at `docs-web/src/main/webapp/` to define build tasks. The key part for API doc generation is:

Gruntfile.js Excerpt

```

apidoc: {
  generate: {
    src: '../java/',
    dest: 'dist/apidoc/'
  }
}

```

This tells Grunt to run the `apidoc` tool on the Java source files in `../java/`, outputting documentation to `dist/apidoc/`.

The full build pipeline (default task) also includes compiling JavaScript, LESS, and templates, followed by optimization steps like minification and string replacements.

3. apidoc Introduction

The documentation comments for `apidoc` are written directly above controller methods in the Java code using special annotations. Here's a simple example:

```

/**
 * @api {get} /user/:id Get user by ID
 * @apiName GetUser
 * @apiGroup User
 *
 * @apiParam {Number} id User's unique ID.
 *
 * @apiSuccess {String} id User ID.
 * @apiSuccess {String} name Name of the User.
 */
public User getUser(int id) {
    // implementation
    ...
}

```

Common `apidoc` Annotations:

Annotation	Description
<code>@api {method} path description</code>	Defines the endpoint

Annotation	Description
@apiName	Internal identifier
@apiGroup	Logical grouping (e.g., User, Document)
@apiParam	Parameters passed to the API
@apiSuccess	Response structure

The output is a clean HTML page showing all endpoints, parameters, and example responses.

EXPLORER

Gruntfile.js

AclResource.java

docs-web > src > main > java > com > sismics > docs > rest > resource > J AclResource.java > AclResource >

TEEDY

docs-core

pom.xml 1

docs-importer

docs-web

src

dev

main

java\com\...

constant

resource

AclResource.java

AppResource.j...

AuditLogReso...

BaseResource.j...

CommentReso...

DocsMessage...

32 /**

33 * ACL REST resources.

34 *

35 * @author bgamard

36 */

37 @Path("/acl")

38 public class AclResource extends BaseResource {

39 /**

40 * Add an ACL.

41 *

42 * @api {put} /acl Add an ACL

43 *

44 * @apiName PutAcl

45 * @apiGroup Acl

46 * @apiParam {String} source Source ID

47 * @apiParam {String="READ","WRITE"} perm Permission

48 * @apiParam {String} target Target ID

49 * @apiParam {String="USER","GROUP","SHARE"} type Target type

50 * @apiSuccess {String} id Acl ID

51 * @apiSuccess {String} perm Permission

52 * @apiSuccess {String} name Target name

53 * @apiSuccess {String="USER","GROUP","SHARE"} type Target type

54 * @apiError (client) ForbiddenError Access denied

55 * @apiError (client) ValidationError Validation error

56 * @apiError (client) InvalidTarget This target does not exist

57 * @apiPermission user

58 * @apiVersion 1.5.0

59 *

60 * @param sourceId Source ID

61 * @param permStr Permission

62 * @param targetName Target name

63 * @param typeStr ACL type

64 * @return Response

65 */

66 @PUT

67 public Response add(@FormParam("source") String sourceId,

Acl - Add an ACL

PUT

/api/acl

权限: user ⓘ

参数

字段	类型	描述
source	String	Source ID
perm	String	Permission 允许值: "READ", "WRITE"
target	String	Target ID
type	String	Target type 允许值: "USER", "GROUP", "SHARE"

Success 200

字段	类型	描述
id	String	Acl ID
perm	String	Permission
name	String	Target name
type	String	Target type 允许值: "USER", "GROUP", "SHARE"

Client error

名称	描述
ForbiddenError	Access denied
ValidationError	Validation error
InvalidTarget	This target does not exist

4. Alternative: Swagger(OpenAPI) with Grunt

Teedy uses `apidoc`, but `grunt` can also work with `Swagger` (now called OpenAPI). Swagger provides interactive API documentation and uses a different annotation format in Java:

OpenAPI Annotation Example

Here's a simple example using **OpenAPI 3** annotations::

```
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/user")
```

```
@Tag(name = "User", description = "Operations related to users")
public class UserController {

    @Operation(summary = "Get a user by ID", description = "Returns a single
user")
    @GetMapping("/{id}")
    public User getUser(
        @Parameter(description = "ID of the user", required = true)
        @PathVariable Integer id) {
        // implementation
        ...
    }
}
```

Common **OpenAPI** Annotations:

Annotation	Description
<code>@Operation</code>	Describes a single API operation (e.g., summary, description).
<code>@Parameter</code>	Documents parameters (e.g., path, query, body).
<code>@Tag</code>	Groups endpoints for better organization.
<code>@ApiResponse</code>	(Optional) Describes possible HTTP responses.

These annotations are processed at runtime or compile-time to generate an **OpenAPI spec** (YAML or JSON). You can then:

- Use `swagger-ui` to create an interactive HTML interface.
- Or use a **Grunt task** to copy/move static docs into your distribution folder (`dist/`).