

AI Tools for Jakarta EE



Empowering Developers with Next-Gen AI Capabilities

Gaurav Gupta,
@jGauravGupta, github.com/jGauravGupta
Senior Software Engineer @ Payara Service Limited,
Apache Committer, NetBeans Dream Team Member,
Jeddict Creator <https://jeddict.github.io/>

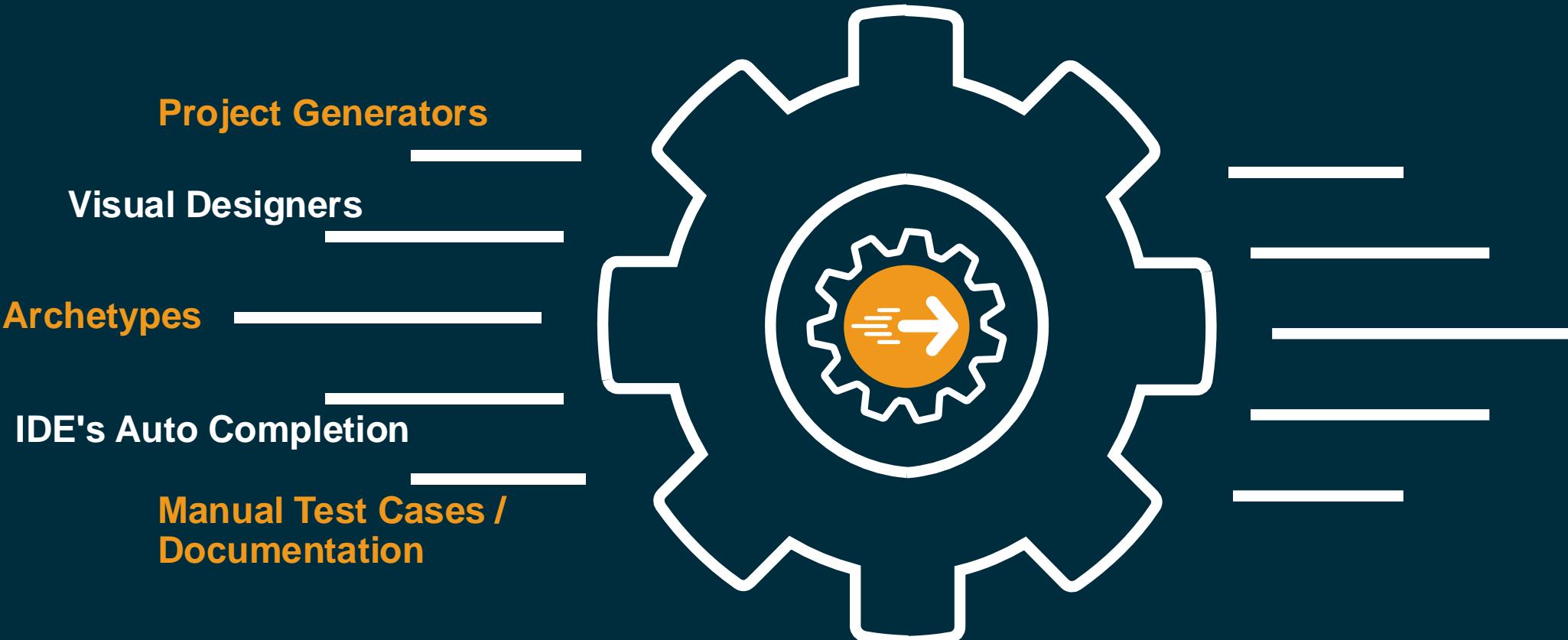
Agenda

Creating SFSCON Conference App Using AI Tools

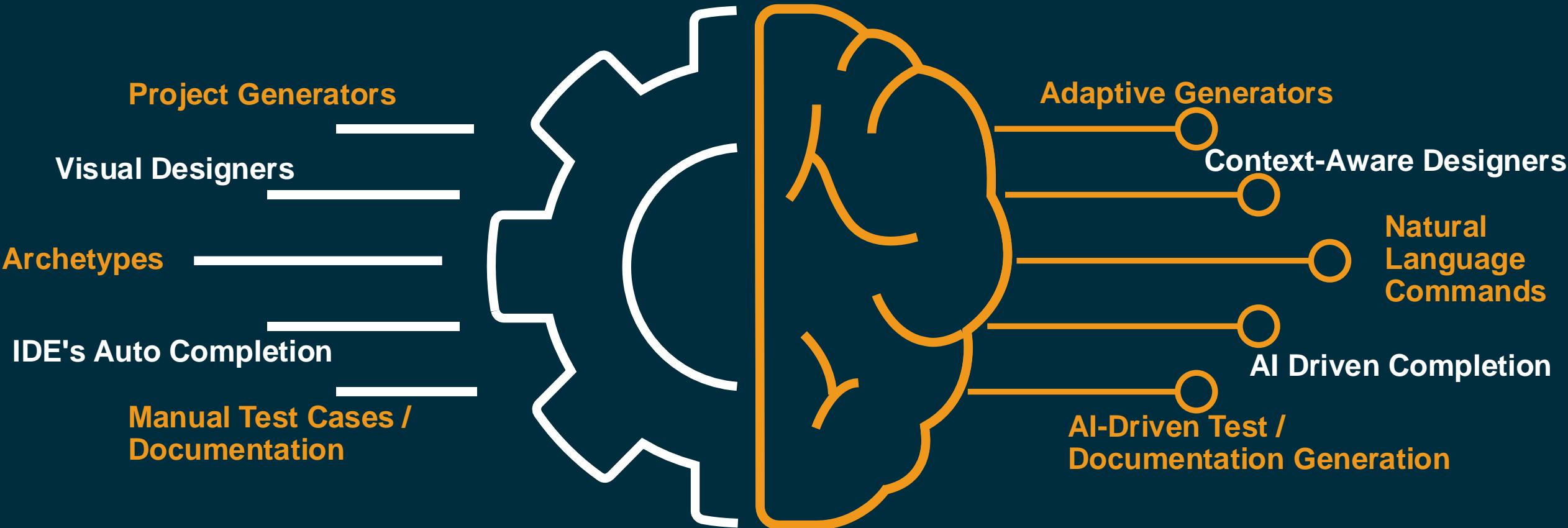
Disclaimer

*The application generated in this demonstration utilizes **AI-powered tools** and serves as a foundational prototype. AI-generated code may contain compilation errors, inconsistencies, and is inherently variable with each generation attempt. This code is not production-ready and should be treated as a preliminary draft. Comprehensive review, refinement, and testing by experienced developers are essential to ensure functionality, performance, security, and adherence to production standards.*

Traditional Development

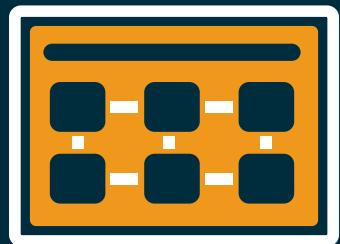


AI-Powered Development

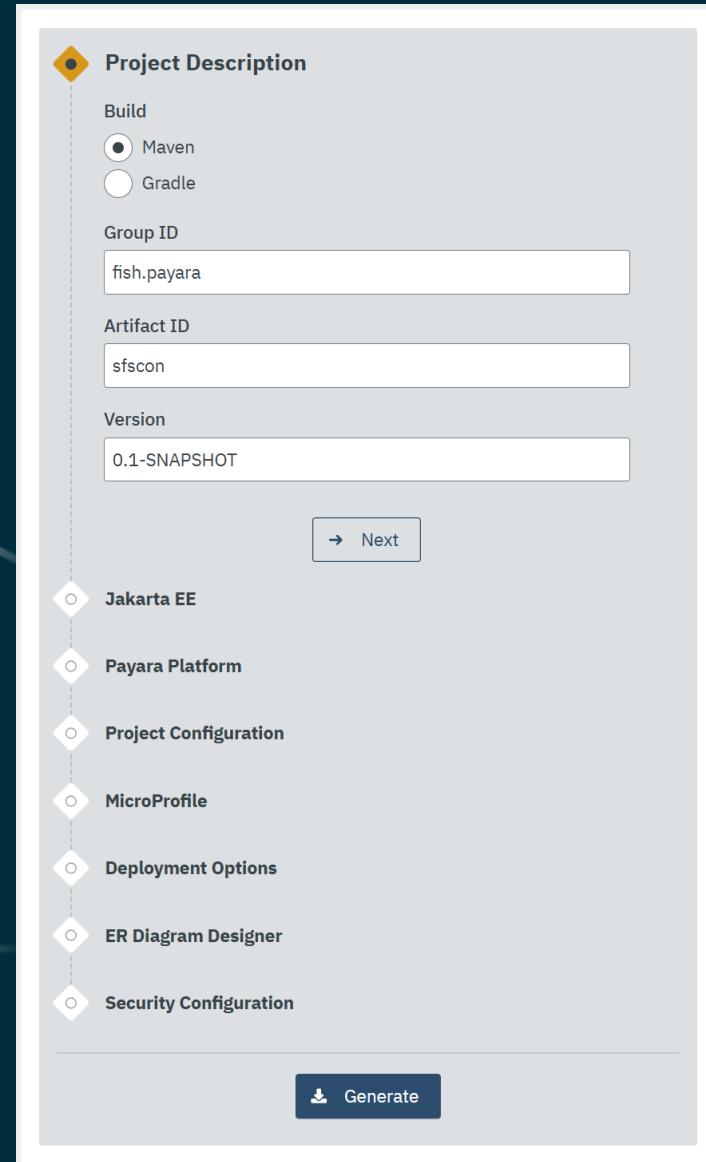


Introducing Payara Starter

<https://start.payara.fish/>



ER Diagram
Designer



Project Description

Build

Maven

Gradle

Group ID

fish.payara

Artifact ID

sfscon

Version

0.1-SNAPSHOT

→ Next

Jakarta EE

Payara Platform

Project Configuration

MicroProfile

Deployment Options

ER Diagram Designer

Security Configuration

Generate



MicroProfile



Secured



ER Diagram Designer

Zero Domain Knowledge

Source SFSCON

```
1 erDiagram
2   CONFERENCE ||--o{ ATTENDEE : has
3   CONFERENCE {
4     int conferenceID PK
5     string name
6     date date
7     string location
8   }
9   ATTENDEE {
10    int attendeeID PK
11    string firstName
12    string lastName
13    string email
14 }
```

Create new Diagram

Live Preview

The diagram illustrates a relationship between two entities: CONFERENCE and ATTENDEE. The CONFERENCE entity is represented by a purple rectangle containing attributes: conferenceID (PK), name, date, and location. The ATTENDEE entity is represented by a purple rectangle containing attributes: attendeeID (PK), firstName, lastName, and email. A line connects the two entities, with a hollow circle at the CONFERENCE end and a plus sign (+) at the ATTENDEE end, indicating a many-to-one relationship labeled 'has'.

Tell us how you want to change the diagram (e.g., 'Add customer address') and press Enter

← → + - []

ER Diagram Designer

Natural Language Command

Source SFSCON

```
1 erDiagram
2   CONFERENCE ||--o{ ATTENDEE : has
3   CONFERENCE {
4     int conferenceID PK
5     string name
6     date date
7     string location
8   }
9   ATTENDEE {
10    int attendeeID PK
11    string firstName
12    string lastName
13    string email
14  }
15   ATTENDEE ||--o{ REGISTRATION : registers
16   REGISTRATION {
17     int registrationID PK
18     date registrationDate
19     string paymentStatus
20   }
21   REGISTRATION }o--|| CONFERENCE : for
22   ATTENDEE ||--o{ FEEDBACK : provides
23   FEEDBACK {
24     int feedbackID PK
```

Natural Language Command

Tell us how you want to change the diagram (e.g., 'Add customer address') and press Enter

Live Preview

The diagram illustrates the following entities and their associations:

- CONFERENCE**:
 - Attributes: conferenceID (PK), name, date, location.
 - Relationships: "has" (o--o) with ATTENDEE.
- ATTENDEE**:
 - Attributes: attendeeID (PK), firstName, lastName, email.
 - Relationships: "for" (o--o) with REGISTRATION, "about" (o--o) with CONFERENCE.
- REGISTRATION**:
 - Attributes: registrationID (PK), registrationDate, paymentStatus.
 - Relationships: "registers" (o--o) with ATTENDEE.
- FEEDBACK**:
 - Attributes: feedbackID (PK), comments, rating.
 - Relationships: "provides" (o--o) with ATTENDEE, "from" (o--o) with REGISTRATION.

ER Diagram Designer

Context Awareness

Source SFSCON

```

1 erDiagram
2   CONFERENCE ||--o{ ATTENDEE : has
3   CONFERENCE {
4     int conferenceID PK
5     string name
6     date date
7     string location
8   }
9   ATTENDEE {
10    int attendeeID PK
11    string firstName
12    string lastName
13    string email
14  }
15  ATTENDEE ||--o{ REGISTRATION : registers
16  REGISTRATION {
17    int registrationID PK
18    date registrationDate
19    string paymentStatus
20  }
21  REGISTRATION }o--|| CONFERENCE : for
22  ATTENDEE ||--o{ FEEDBACK : provides
23  FEEDBACK {
24    int feedbackID PK

```

Live Preview

The diagram illustrates the entities and their relationships:

- CONFERENCE** entity (purple border):
 - Attributes: conferenceID (PK), name, date, location.
 - Relationships: has (with ATTENDEE), features (with SESSION).
- ATTENDEE** entity (purple border):
 - Attributes: attendeeID (PK), firstName, lastName, email.
 - Relationships: registers (with REGISTRATION), provides (with FEEDBACK), participates (with SESSION).
- REGISTRATION** entity (blue border):
 - Attributes: registrationID (PK), registrationDate, paymentStatus.
 - Relationships: registers (with ATTENDEE).
- FEEDBACK** entity (blue border):
 - Attributes: feedbackID (PK), comments, rating.
 - Relationships: provides (with ATTENDEE).
- SPEAKER** entity (blue border):
 - Attributes: speakerID (PK), name, bio, expertise.
 - Relationships: presents (with SESSION).
- SESSION** entity (blue border):
 - Attributes: sessionID (PK), title, description, duration.
 - Relationships: includes (with CONFERENCE), during (with NETWORKING), presents (with SPEAKER).
- NETWORKING** entity (orange border):
 - Attributes: networkingID (PK), topic, time.
 - Relationships: about (with FEEDBACK), during (with SESSION).

Mermaid Diagram
Enlarge Diagram

Tell us how you want to change the diagram (e.g., 'Add customer address') and press Enter

← → + - ✖

Adaptive Generator for Jakarta EE

Conference Management System

Search

Sign out

- [!\[\]\(05885265dcf07af39e36cff295433c51_img.jpg\) Dashboard](#)
- [!\[\]\(26633b49a6a049dbcdb72c211022769c_img.jpg\) Conference](#)
- [!\[\]\(c1eef8597e8624b27c40bc77c24c1933_img.jpg\) Attendee](#)
- [!\[\]\(1b7c2a07f96414bd52fd74aa4674183e_img.jpg\) Registration](#)
- [!\[\]\(c87cd6f776d9cb3983bb1a982eab3057_img.jpg\) Feedback](#)
- [Speaker](#)
- [!\[\]\(23b5671960ccffff4fa693b6b1d6b495_img.jpg\) Session](#)
- [Networking](#)
- [OTHERS !\[\]\(5a2bf27b958b744af2f6bb8676805b0f_img.jpg\)](#)
- [!\[\]\(54708819a52629073d2c887abae0172e_img.jpg\) About US](#)

A Comprehensive Conference Management Application

Welcome to our Conference Management System. Here you can find detailed information about upcoming conferences, register as an attendee, and provide feedback on sessions. Our platform is designed to streamline the process of organizing and attending conferences, making it easier for everyone involved.

Generated Application with Payara Starter

Conference Management System

Search

Sign out

- [Dashboard](#)
- [Conference](#)
- [Attendee](#)
- [Registration](#)
- [Feedback](#)
- [Speaker](#)
- [Session](#)
- [Networking](#)
- [OTHERS](#)
- [About US](#)

Conference

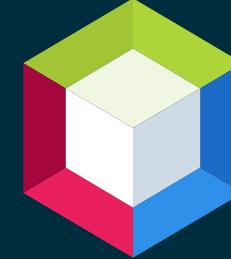
Represents a conference event.

Add Conference

Conference ID	Name	Date	Location	Actions
1	SFSCON 2024	2024-11-08	Bozen Italy	Edit Delete



Jeddict AI Assistant for Apache NetBeans IDE



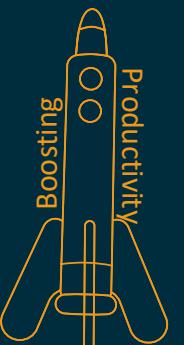
- Completely Free AI Assistant
- Project / Package context-aware Chat
- AI-Driven inline Code Completion
- SQL inline completion
- Context-aware database interactions
- Create REST Endpoints
- Support for multiple LLM Provider

Jeddict AI Assistant

Intelligent Code Completion

The screenshot shows a Java IDE interface with the following details:

- Projects View:** Shows a project named "sfscon" with several source packages:
 - Source Packages:** fish.payara.domain (containing Attendee.java, Conference.java, Feedback.java, Networking.java, Registration.java, Session.java, Speaker.java), fish.payara.resource, fish.payara.service (containing AbstractService.java, AttendeeService.java, ConferenceService.java, FeedbackService.java, NetworkingService.java, RegistrationService.java, SessionService.java, SpeakerService.java).
 - SpeakerService.java** is selected in the list.
- Code Editor:** The file "SpeakerService.java" is open. The cursor is at line 23, after the comment // send email to speaker. A tooltip provides a detailed description of the method: "Method to send email to a speaker. Uses the JavaMail API to send an email, which is essential for notifying speakers." Below the tooltip, a list of code completion suggestions is shown:
 - private static final String DEFAULT_EMAIL SUBJECT = "Importa
 - public List<Speaker> getAllSpeakers() { return em.createC
 - public void sendEmailToSpeaker(Speaker speaker, String subje
 - public void updateSpeaker(Speaker speaker) { em.merge(spe
 - SpeakerService(EntityManager em, Class<E> entityClass) - gen
 - clone() - override
 - count() - override
 - create(Speaker entity) - override
 - edit(Speaker entity) - override



Jeddict AI Assistant

Context Aware Variable / Method Naming

FOOBAR

data|x2

data | X

foobar'

foo 22'

temp123'

Projects X

- sfscn
 - Web Pages
 - Remote Files
 - RESTful Web Services
 - Source Packages
 - fish.payara.domain
 - Attendee.java
 - Conference.java
 - Feedback.java
 - Networking.java
 - Registration.java
 - Session.java
 - Speaker.java
 - fish.payara.resource
 - AttendeeResource.java
 - ConferenceResource.java
 - FeedbackResource.java
 - HeaderUtil.java
 - HelloWorldResource.java
 - NetworkingResource.java
 - RegistrationResource.java
 - RestConfiguration.java
 - SessionResource.java
 - SpeakerResource.java
 - fish.payara.service
 - fish.payara.service.producer
 - Other Sources
 - Dependencies
 - Non-classpath Dependencies
 - Java Dependencies
 - Project Files

SpeakerService.java X SpeakerResource.java X

```

28     @Path("/api/speaker")
29     public class SpeakerResource {
30
31         private static final Logger LOG = Logger.getLogger(SpeakerResource.class.get
32
33         @Inject
34         private SpeakerService speakerService;
35
36         @Inject
37         private ConferenceService conferenceService;
38
39         private static final String ENTITY_NAME = "speaker";
40
41         /**
42          * POST : Create a new speaker ...9 lines */
43         @POST
44         @Consumes(MediaType.APPLICATION_JSON)
45         @Produces(MediaType.APPLICATION_JSON)
46         public Response createSpeaker(Speaker speaker) throws URISyntaxException {
47             LOG.log(Level.INFO, "addNewSpeaker");
48             if (speaker == null) {
49                 speaker = createNewSpeaker();
50                 speaker.setConference(enrollSpeaker(speaker));
51             } else {
52                 speaker = initiateSpeakerCreation(speaker);
53                 insertSpeaker(speaker);
54                 persistSpeaker(speaker);
55                 processSpeakerCreation(speaker);
56             }
57             speakerService.registerSpeaker(speaker);
58             return HeadResponseBuilder.create()
59                     .entity(speaker)
60                     .status(201)
61                     .build();
62         }
63     }
  
```

fish.payara.resource.SpeakerResource > createSpeaker >

data | 22

data | x?

foobar x2

temp|22

temp122

temp123

Jeddict AI Assistant Log Message / JavaDocs

The screenshot shows an IDE interface with the following elements:

- Projects X** panel on the left displaying the project structure of `sfscon`, including `Source Packages` like `fish.payara.domain` and `fish.payara.resource`, and `SpeakerResource.java`.
- Code Editor** showing `SpeakerResource.java` code:

```
28 @Path("/api/speaker")
29 public class SpeakerResource {
30
31     private static final Logger LOG = Logger.getLogger(SpeakerResource.class.get
32
33     @Inject
34     private SpeakerService speakerService;
35
36     @Inject
37     private ConferenceService conferenceService;
38
39     private static final String ENTITY_NAME = "speaker";
40
41     /** POST : Create a new speaker ...9 lines */
42     @POST
43     @Consumes(MediaType.APPLICATION_JSON)
44     @Produces(MediaType.APPLICATION_JSON)
45     public Response enrollSpeaker(Speaker speaker) throws URISyntaxException {
46         LOG.log(Level.FINE, "REST request to save Speaker : {}", speaker);
47         if (speaker.getConfer
48             speaker.setConfer
49         } else {
50             speaker.setConfer
51         }
52         speakerService.createSpeaker(speaker);
53         return HeaderUtil.createResponseHeader("Location", URI.create(
54             UriComponentsBuilder.fromPath("/{entityName}/{id}").build()
55         ).entity(spea
56
57     }
58
59 }
```

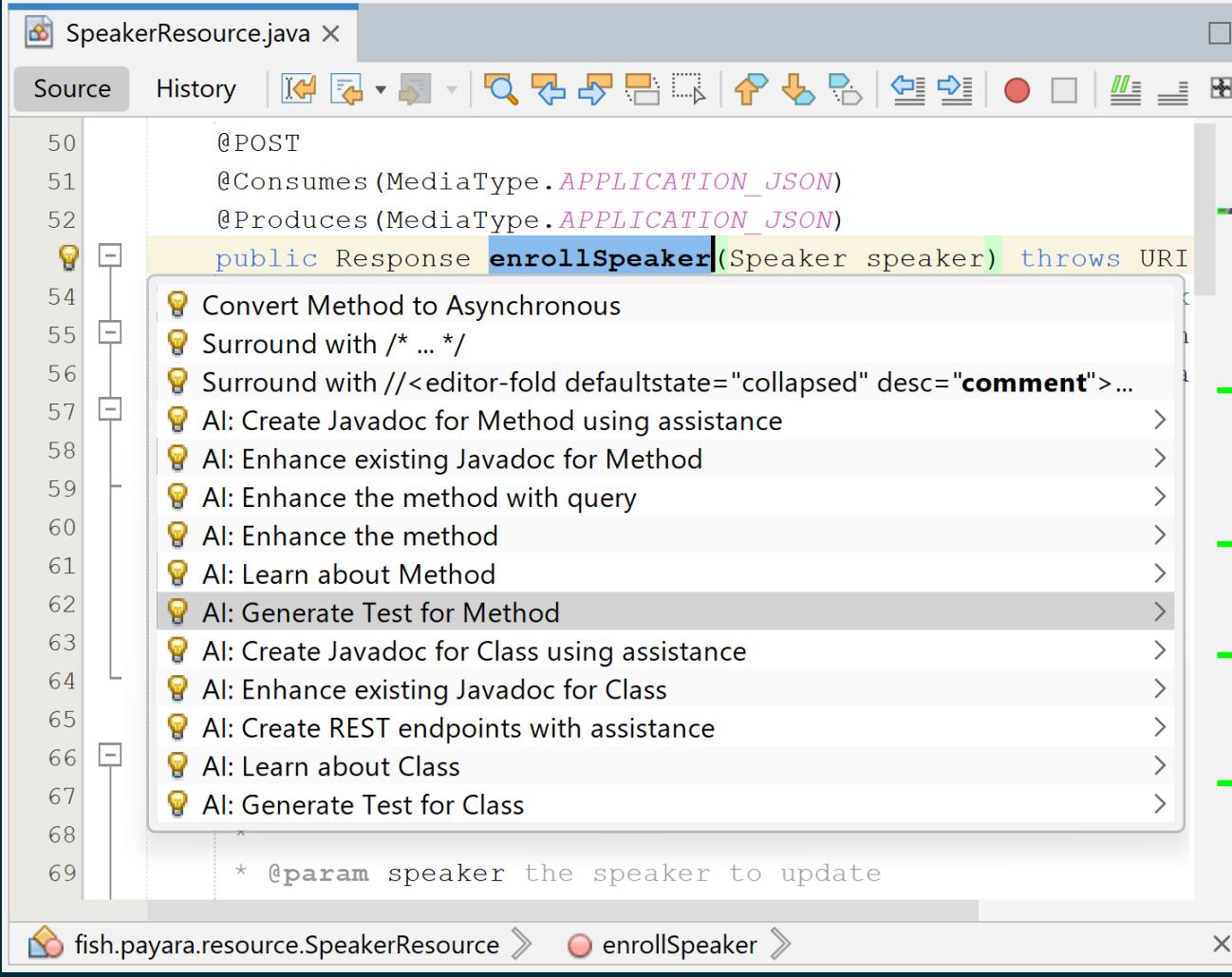
- Tooltips** and **IntelliSense** are visible, showing suggestions for `LOG.log` parameters.
- Logs** panel on the right showing a log message from the AI assistant:

```
Attempting to save Speaker data: {}
Initiating REST request to save Speaker: {}
Logging Speaker creation request: {}
Processing request to create Speaker: {}
Request received to enroll Speaker: {}
Request to save Speaker entity: {}
Saving Speaker information: {}
Storing new Speaker details: {}
```

- Java Docs** icon on the left side of the IDE.

Jeddict AI Assistant

Test case generation



The screenshot shows an IDE interface with a Java file named `SpeakerResource.java`. The code defines a REST endpoint:

```
50     @POST  
51     @Consumes(MediaType.APPLICATION_JSON)  
52     @Produces(MediaType.APPLICATION_JSON)  
53     public Response enrollSpeaker(Speaker speaker) throws URI  
54  
55     Convert Method to Asynchronous  
56     Surround with /* ... */  
57     Surround with //<editor-fold defaultstate="collapsed" desc="comment">...</editor-fold>  
58     AI: Create Javadoc for Method using assistance  
59     AI: Enhance existing Javadoc for Method  
60     AI: Enhance the method with query  
61     AI: Enhance the method  
62     AI: Learn about Method  
63     AI: Generate Test for Method  
64     AI: Create Javadoc for Class using assistance  
65     AI: Enhance existing Javadoc for Class  
66     AI: Create REST endpoints with assistance  
67     AI: Learn about Class  
68     AI: Generate Test for Class  
69     * @param speaker the speaker to update
```

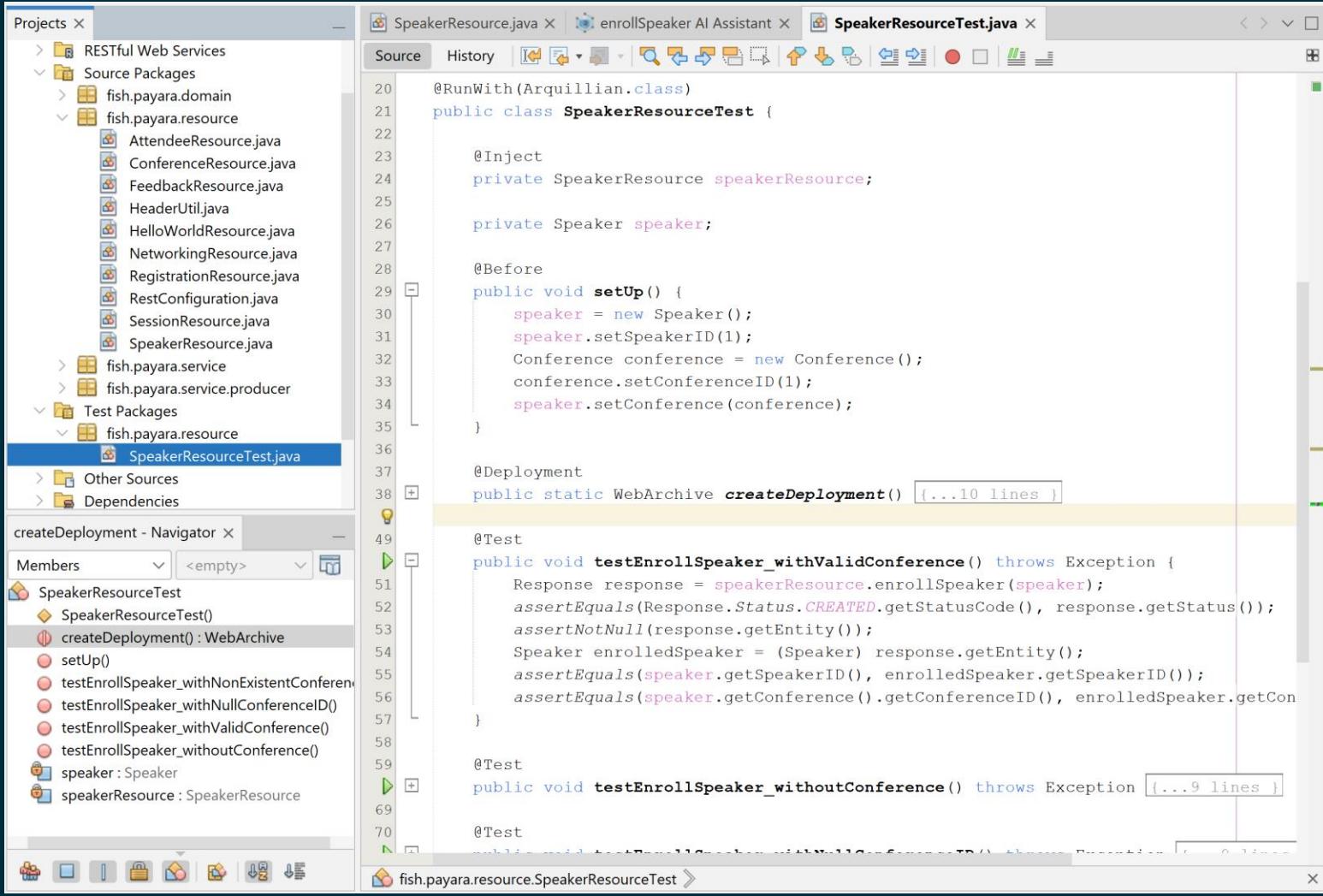
A tooltip is displayed over the `enrollSpeaker` method, listing various AI-assisted actions:

- Convert Method to Asynchronous
- Surround with /* ... */
- Surround with //<editor-fold defaultstate="collapsed" desc="comment">...</editor-fold>
- AI: Create Javadoc for Method using assistance
- AI: Enhance existing Javadoc for Method
- AI: Enhance the method with query
- AI: Enhance the method
- AI: Learn about Method
- AI: Generate Test for Method
- AI: Create Javadoc for Class using assistance
- AI: Enhance existing Javadoc for Class
- AI: Create REST endpoints with assistance
- AI: Learn about Class
- AI: Generate Test for Class



Jeddict AI Assistant

Test case generation

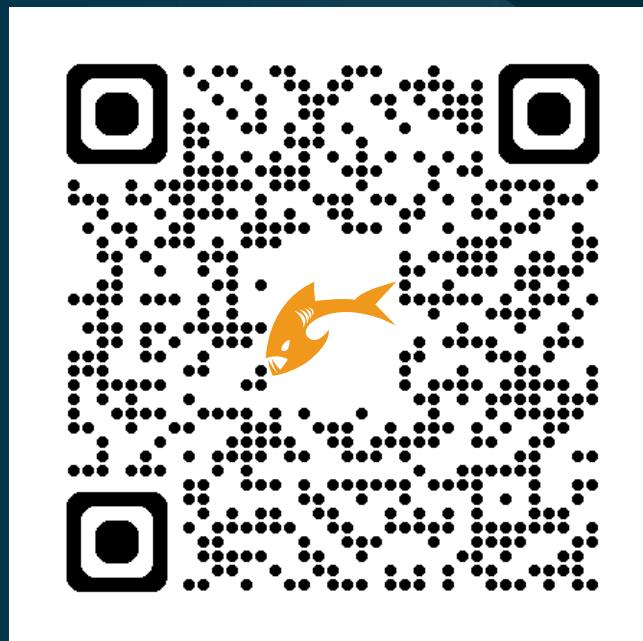


The screenshot shows an IDE interface with the following components:

- Projects View:** Shows a tree structure of projects and source packages. Under "Source Packages", there are "fish.payara.domain" and "fish.payara.resource". "fish.payara.resource" contains files like AttendeeResource.java, ConferenceResource.java, FeedbackResource.java, HeaderUtil.java, HelloWorldResource.java, NetworkingResource.java, RegistrationResource.java, RestConfiguration.java, SessionResource.java, and SpeakerResource.java. It also contains "fish.payara.service" and "fish.payara.service.producer" packages, and a "Test Packages" folder containing "SpeakerResourceTest.java".
- Code Editor:** Displays the content of "SpeakerResourceTest.java". The code includes annotations like @RunWith, @Inject, @Before, and @Deployment, and methods for testing speaker enrollment.
- Navigator View:** Shows a list of members for "SpeakerResourceTest", including "createDeployment()", "setUp()", and several test methods: "testEnrollSpeaker_withValidConference()", "testEnrollSpeaker_withNonExistentConference()", "testEnrollSpeaker_withNullConferenceID()", "testEnrollSpeaker_withValidConference()", and "testEnrollSpeaker_withoutConference()". It also lists variables "speaker" and "speakerResource".
- Code Coverage:** A circular progress bar indicates 85% coverage.

Thank You

<https://start.payara.fish>



<https://jeddict.github.io>

