Google Summer of Code 2024 Ideas

JBoss has not been accepted to GSoC for the 2024 program. Thank you everybody for your interest.

The JBoss Community is planning to participate in Google Summer of Code in 2024.

All contributors & developers are welcome to participate in the https://summerofcode.withgoogle.com/ program with the JBoss Community.

If you are a contributor looking forward to participating in the GSoC 2024 with the JBoss Community:

- Feel free to browse the growing idea list below.
- Please don't hesitate to contact the mentor(s) indicated in the proposal for any related clarification and to discuss proposals.
- You can have a look at ideas list of previous years for inspiration.
- Please see our contributor guide.
- You may find a sample GSoC proposal document here which was for this idea.

Contributors: Please read the list above and also read our contributor guide.

A note to mentors

MENTORS: Red Hat employees can change this page directly to add ideas. Please be extra careful to not get other mentor's edits discarded.
Red Hatters should have linked their jboss.org account with Red Hat and can be checked on https://sso.jboss.org/login

Non-Red Hatters can add a comment to the page and admins will make sure the idea is added to the page.

Table of Contents

Administrators and Mentors

We will list the potential mentors in this place. For now, if you have any questions, please contact the GSoC administrators:

George Zaronikas (gzaronikas) and Sokratis Zappis (szappis AT redhat DOT com)

Communication channels

Gitter : **JBossOutreach/GSoC - Gitter**

Please take note - These channels are about generic doubts. For project-specific doubts you will need to contact project mentors and channels specified in the project description.

Idea template (for mentors)

Project title

Summary of idea:

-Idea

-Feature A

-Feature B

Knowledge prerequisite: Languages/Technologies goes here

Github repo:

Project size: medium (~175 hours) or large (~350 hours)

Skill level: Beginner/Intermediate/Advanced

Contact(s) / potential mentors(s): Mentor(s) name and contact details

Associated JBoss community project(s):

Idea Proposals

WildFly Elytron - Add support for Online Certificate Status Protocol (OCSP) stapling to WildFly Elytron, for use in the WildFly application server

Summary of idea:

If you want to learn about security, this is your chance to develop a new security feature for the WildFly Application Server! As a bonus, you'll get to work with a diverse team.

The WildFly Elytron project is a security framework for Java clients and application servers. WildFly is an open source application server. Elytron is used by the WildFly application server to secure applications that are deployed to the server and to secure management access to the server. Banks, retail stores, and governments are just some examples of end-users of the enterprise version of the WildFly application server.

The TLS protocol allows communication between a client and a server to be encrypted. WildFly Elytron allows users to configure policy information related to TLS. Currently, this includes things like key managers, trust managers, cipher suites, and protocols (see https://github.com/wildfly-security/wildfly-elytron/tree/1.x/ssl/src/main/java/org/wildfly/security/ssl).

The purpose of this project is to work on new OCSP feature for the WildFly server. In particular, the goal of this project is to add support for Online Certificate Status Protocol (OCSP) stapling to WildFly Elytron, for use in the WildFly application server.

OCSP stapling is a standard that's used to check the revocation status of an X.509 certificate. In particular, when presenting its certificate during a TLS handshake, the server first sends an OCSP request to an OCSP responder and the returned response is "stapled" to the server's certificate chain. Because the server is the one contacting the OCSP responder instead of the client, the advantage is that the server bears the resource cost and the OCSP response it receives can be cached and used multiple times for different clients.

Possible tasks for this project:

- Create a document that describes how you plan to approach the problem.
- Implement the ability for a WildFly server to use OCSP stapling when presenting its certificate. This will involve adding functionality to both the WildFly Elytron project as well as the WildFly Core project, where Elytron is actually integrated with the WildFly application server.
- Implement appropriate test cases.
- Write documentation.
- Create a blog post that gives an overview of your project.

The WildFly Elytron team is a diverse, distributed team that has a lot of experience working with interns and junior engineers.

Knowledge pre-requisites:

- Experience with Java
- Git
- Maven

GitHub repo: https://github.com/wildfly-security/wildfly-elytron

Other useful links:

- Contribution Guide
- Elytron Website
- Getting Started Guide

Project size: Medium (~175 hours)

Skill level: Intermediate

Project chat: https://wildfly.zulipchat.com/#narrow/stream/173102-wildfly-elytron

Contact(s) / potential mentors(s): Farah Juma <fjuma@redhat.com> and Diana Krepinska <dvilkola@redhat.com>

Associated JBoss community project(s): Elytron, WildFly

EAT - Testing Infinite Software Project Versions

Summary of idea:

The innovative part of EAT is creating the test once and testing with any version of the tested software. It may be firstly applied for the JBoss Servers, but, in general, a similar structure, can be used for creating tests about any software with multiple versions or for multiple software programs that have a part of the testsuite in common. EAT is a project under the statement.

Possible tasks for this project :

- extend the existing AT testsuites with latest JBOSS Community server snapshot
- extend the android related functionality (e.g. automatic creation of test apks, automatic test with multiple devices, etc)
- create an android mobile application tested with EAT
- the contributors are also welcome to make their proposals
- proposals should be also added at the EAT-PROPOSALS mobile app (https://play.google.com/store/apps/details?id=edu.eatproposals.eatapp&hl=en&gl=US)

Project size: large (~350 hours)

Github repo: https://github.com/EAT-JBCOMMUNITY/EAT

Contact / potential mentors: Panagiotis Sotiropoulos (psotirop@redhat.com)

Associated JBoss community project: EAT

Setting up a reverse proxy to a Tomcat server with the use of Ansible

Summary of idea:
Create an Ansible playbook which builds the source code and installs Apache Tomcat and Apache Httpd server to a selected machine.
Integrate testing of the aforementioned packages with the Ansible playbook.
Set up a reverse proxy for the Apache Tomcat using the Apache HTTPD server with the Ansible playbook.The purpose of this project is to familiarize candidates with Apache Tomcat and Apache Httpd as well as provide them experience working with Ansible.
Project size: medium (~175 hours)
Skill level: Beginner
Contact(s) / potential mentors(s): Dimitris Soumis <dsoumis@redhat.com>
Associated JBoss community project(s): Ansible, Tomcat, Httpd

Debezium - Debezium UI

Summary of idea

Debezium is an open-source platform for change data capture (CDC) and lets you stream data change events out of various databases such as MySQL, Postgres, SQL Server, MongoDB, and others. This proposal aims to work on the improvements to the Debezium UI offering in providing a streamlined process of configuring any given Debezium connector via the web interface and promoting the Debezium UI among the community user.

Possible tasks for this project include:

- Build upon the existing GUI to work the overall UX improvement to the current GUI.
- Adding the E2E test cases.

- Create demos (video examples) and blog posts showcasing the different use cases of GUI.
- Promoting the Debezium UI among various channels e.g., X, youtube.

Contributor Benefits

You will learn how to implement a UI with React and TypeScript with proper E2E test cases and setting up a delivery pipeline. You will also gain experience with Debezium and a basic understanding of how Debezium works with Apache Kafka Connect.

Knowledge Pre-requisite

JavaScript, Git, any frontend library/framework ideally React.

Project Details

Github repo:
https://github.com/debezium/debezium,
https://github.com/debezium/debezium-ui

Project size: medium (~175 hours)

Skill level: Intermediate

Project chat: https://debezium.zulipchat.com/
UI Channel: https://debezium.zulipchat.com/#narrow/stream/392459-community-ui

Contact(s) Indra Raj Shukla (ishukla AT redhat DOT com), Rene Kerner (rkerner AT redhat DOT com), Chris Cranford (ccranfor AT redhat DOT com)

Associated JBoss community project(s): Debezium

SIMD support in a Java WebAssembly runtime

Summary of idea: Chicory is a JVM native WebAssembly runtime. It allows you to run WebAssembly programs with zero native dependencies or JNI.
Chicory can run Wasm anywhere that the JVM can go. It is designed with simplicity and safety in mind.
The WebAssembly specification defines a full section of Vector Instructions and there is a preview JEP(438) in Java for supporting a Vector API.
In this project, you will be asked to define and implement the best possible support for SIMD operations in Chicory, understanding the tradeoffs and offering viable solutions for different JVM versions and supported APIs.

Outcome:

At the end of the project, we expect to be able to run on top of Chicory libraries that make use of SIMD instructions(like Libsodium).
Stretch: Full support for all of the SIMD instructions defined by the WebAssembly specification, backed by the Vector API but with programmatic fallbacks when not available.

Skills:

Some prior experience with the Java programming language is highly desirable.

Mentors:

Andrea Peruffo <aperuffo@redhat.com>

Size:
We aim for a full 350 hour project to end up with a pretty complete support of SIMD.
Although tuning the expected outcome and scoping for a limited number of instructions support is totally possible for candidates for high interest but less time.

Github repo:

https://github.com/dylibso/chicory

Skill level:

intermediate

Text format support for a Java WebAssembly runtime

Summary of idea: Chicory is a JVM native WebAssembly runtime. It allows you to run WebAssembly programs with zero native dependencies or JNI.
Chicory can run Wasm anywhere that the JVM can go. It is designed with simplicity and safety in mind.
The WebAssembly specification defines two different representation formats: binary and text.
Chicory lacks, at the moment, support for the Text Format of WebAssembly (.wat files).
This is a pain point as the project is forced to use external tooling, with possibly tricky integrations, to be able to make use of textual WebAssembly modules.

Outcome:

At the end of the project, we expect to be able to use the text parser to feed .wat WebAssembly programs into the interpreter.

Stretch: The WebAssembly testsuite is defined using an extension to the Text Format called Wast. Having support for it means that we can dog-food and avoid using external tools for generating JUnit specs from the testsuite.

Skills:

Some prior experience with the Java programming language is highly desirable.
Some exposure to writing parsers it's not required but welcome.

Mentors:

Andrea Peruffo <aperuffo@redhat.com>

Size:
Depending on the prior experience of the candidate in writing parsers it can be a 175 or 350 hours project.

Github repo:

https://github.com/dylibso/chicory

Skill level:

intermediate

WASI support in a Java WebAssembly runtime

Summary of idea: Chicory is a JVM native WebAssembly runtime. It allows you to run WebAssembly programs with zero native dependencies or JNI.
Chicory can run Wasm anywhere that the JVM can go. It is designed with simplicity and safety in mind.
WebAssembly programs are completely sandboxed, this means that they do not have built in capabilities to access host functionalities of any kind.
This makes Wasm programs hardly usable/useful outside of a plugin system or the browser nowaday.
To overcome this limitation there is a preview specification of WASI, which defines a set of primitives portable across systems.
Please note that some programming languages are unable to compile to pure Wasm instructions and they require some level of support of WASI even to run simple functions.
In Chicory we have a very immature support for WASI, in this project we aim to complete, or at least increase, it's support.

Outcome:

At the end of the project, we expect to be able to run programs leveraging the WASI preview 1 on top of Chicory.

Stretch: Full support for the WASI specification fully portable across operating systems.

Skills:

Some prior experience with the Java programming language is highly desirable.

Mentors:

Andrea Peruffo <aperuffo@redhat.com>

Size:
We are pretty flexible and it's easy to scope the work to fit any of the available size 90/175/350 hours depending on the complexity and number of supported functions.

Github repo:

https://github.com/dylibso/chicory

Skill level:

intermediate

Compile and run Kiota on top of a Wasm runtime outside of the browser

Summary of idea: Kiota is a command line tool for generating an API client to call any OpenAPI described API you are interested in. The goal is to eliminate the need to take a dependency on a different API SDK for every API that you need to call. Kiota API clients provide a strongly typed experience with all the features you expect from a high quality API SDK, but without having to learn a new library for every HTTP API.

The dotnet toolchain is already capable of compiling standard .NET code to WebAssembly and we are successfully shipping Kiota compiled to Wasm for the browser.
The support for running standard dotnet applications on Wasi enabled Wasm runtimes is not yet fully ironed but is catching up, the results of the last evaluation spike shows gaps in 4 main areas:

- Virtual FileSystem(VFS)
- Parallel computation

- Crypto
- Http client

Outcome:

At the end of the project, we expect to be able to run Kiota on a standard Wasi enabled Wasm runtime like Wasmtime.
Building a sustainable process to be able to compile and ship for every new release of Kiota is also in scope.

Stretch: Run Kiota compiled to Wasm embedded in a Maven plugin leveraging the Chicory Wasm interpreter.

Skills:

Some prior experience with the C# programming language and the dotnet toolchain is desirable but not required.

Mentors:

Andrea Peruffo <aperuffo@redhat.com>

Size:
This is a full 350 hours project, we can accept 175 hours applications from people already comfortable with the dotnet Wasm toolchain.

Github repo:

https://github.com/microsoft/kiota

Skill level:

intermediate