LEARNING guice

#guice

Table of Contents

About 1

Chapter 1: Getting started with guice 2

Remarks 2

Examples 2

Setup of a 'Hello, world!' example 2

Sample classes 2

Run with Gradle 3

Run with Maven 4

Chapter 2: Guice method interceptor 6

Examples 6

Simple example 6

Chapter 3: Using Guice with Java WebSockets (JSR-356) 8

Examples 8

Configuration to get Guice injection on Websocket endpoints 8

Chapter 4: Using Guice with Jersey 2 (JAX-RS RI) 9

Examples 9

Getting Guice injections on JAX-RS resources 9

Credits 10

About

Chapter 1: Getting started with guice

Remarks

This section provides an overview of what guice is, and why a developer might want to use it.

It should also mention any large subjects within guice, and link out to the related topics. Since the Documentation for guice is new, you may need to create initial versions of those related topics.

Examples

Setup of a 'Hello, world!' example

Guice is a Java library. To use it you have to add a JAR file into the classpath of your Java project.

Sample classes

Below are several classes for a "Hello, world!" example.

An interface of a hello "service":

```
public interface HelloWorldService {
    public void sayHello();
}
```

The implementation of the service:

```
public class HelloWorldServiceImpl implements HelloWorldService {
    @Override
    public void sayHello() {
        System.out.println("Hello, world!");
    }
}
```

A Guice naming module. It is needed to instruct Guice that `HelloWorldServiceImpl` will be injected where a hello service is needed.

```
import com.google.inject.AbstractModule;

public class HelloWorldModule extends AbstractModule {
    protected void configure() {
        bind(HelloWorldService.class).to(HelloWorldServiceImpl.class);
    }
}
```

A main class where the actual injection of a hello service takes place:

```java
import javax.inject.Inject;

import com.google.inject.Guice;
import com.google.inject.Injector;
import com.google.inject.Module;

public class Main {

    @Inject
    private HelloWorldService service;//hello service

    public static void main(String[] args) {

        Main main = new Main();

        Module module = new HelloWorldModule();
        Injector injector = Guice.createInjector(module);
        injector.injectMembers(main);//injects the implementation of the service

        main.testGuice();
    }

    public void testGuice()
    {
        service.sayHello();//usage of the service
    }
}
```

Run with Gradle

To quicky setup and run with Gradle 2.2.+ and Java 8:

1. Install gradle if not already installed
2. Create an empty directory and navigate into it with a gradle enabled shell
3. Create an empty java project:

   ```
   gradle init --type java-library
   ```

4. In the automatically generated `build.gradle`:

- change `apply plugin: 'java'` to `apply plugin: 'application'`
- add the following line

  ```
  mainClassName = 'Main'
  ```

- in the dependencies section add a dependency to a version of guice, e.g.:

  ```
  dependencies {
    ...
    compile group: 'com.google.inject', name: 'guice', version: '4.1.0'
    ...
  }
  ```

5. Add the classes shown above into the default package in `src/main/java`, each in its own file

6. Run and enjoy

```
..> gradlew run
    :compileJava
    :processResources UP-TO-DATE
    :classes
    :run
    Hello, world!

    BUILD SUCCESSFUL

    Total time: 3.595 secs
```

Run with Maven

To quicky setup and run with Maven 3+ and Java 8:

1. Install maven if not already installed

2. Create an empty directory and navigate into it with a maven enabled shell

3. Create an empty java project:

```
mvn archetype:generate -DgroupId=com.example -DartifactId=guice -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

4. Switch to the `guice` subdirectory

5. In the automatically generated `pom.xml`:

- in the `dependencies` element add a dependency to guice:

```
<dependency>
   <groupId>com.google.inject</groupId>
   <artifactId>guice</artifactId>
   <version>4.1.0</version>
</dependency>
```

- add the following plugin to your project (allows an easy test run)

```
<project>
  .....
  <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.5.0</version>
                <configuration>
                    <mainClass>Main</mainClass>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

6. Add the classes shown above into the default package in `src/main/java`, each in its own file
7. Run and enjoy

```
...\guice>mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building guice 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.5.0:java (default-cli) @ guice ---
Hello, world!
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.800 s
[INFO] Finished at: 2016-10-09T11:44:41+03:00
[INFO] Final Memory: 10M/309M
[INFO] ------------------------------------------------------------------------
```

Read Getting started with guice online: https://riptutorial.com/guice/topic/3449/getting-started-with-guice

Chapter 2: Guice method interceptor

Examples

Simple example

Intercepted method:

```
public class ExampleService implements Example {
    @MyAnnotation
    public doHomework() {
        System.out.println("working");
    }
}
```

Annotation:

```
    @Retention(RetentionPolicy.RUNTIME) @Target(ElementType.METHOD)
    public @interface MyAnnotation {}
```

Interceptor:

```
public class MyInterceptor implements MethodInterceptor {

    @Override
    public Object invoke(MethodInvocation arg0) throws Throwable {
        System.out.println("***** intercepted *****");
        return arg0.proceed();
    }
}
```

Module:

```
public class MyModule extends AbstractModule {


    @Override
    protected void configure() {
        bind(CallerService.class).to(Caller.class);
        bind(ExampleService.class).to(Example.class);

        bindInterceptor(Matchers.any(), Matchers.annotatedWith(MyAnnotation.class), new
MyInterceptor());
    }
}
```

Caller:

```
public class CallerService implements Caller {
    @Inject
    private Client client;
    public void call() {
        client.doHomework();
    }
}
```

Output:

```
***** intercepted *****
working
```

Read Guice method interceptor online: https://riptutorial.com/guice/topic/7907/guice-method-interceptor

Chapter 3: Using Guice with Java WebSockets (JSR-356)

Examples

Configuration to get Guice injection on Websocket endpoints

First, we need a custom endpoint builder.

```
public class WSConfigurator extends ServerEndpointConfig.Configurator {
  @Inject
  private static Injector injector;

  @Override
  public <T> T getEndpointInstance(Class<T> endpointClass)
          throws InstantiationException
  {
    return injector.getInstance(endpointClass);
  }
}
```

We need to bootstrap the injector in the above configurator from one of our Guice modules.

```
public class WebSocketModule extends AbstractModule {
  @Override
  protected void configure() {
    requestStaticInjection(WSConfigurator.class);
  }
}
```

Finally, we can use `@Inject` on the endpoints' constructor.

```
@ServerEndpoint(
        value = "/ws/sync",

       configurator = WSConfigurator.class)
public class WSSync extends AsyncWebSocketServer {
  @Inject
  public WSSync(EventBus eventBus) {
    ...
  }
}
```

Read Using Guice with Java WebSockets (JSR-356) online:
https://riptutorial.com/guice/topic/6462/using-guice-with-java-websockets--jsr-356-

Chapter 4: Using Guice with Jersey 2 (JAX-RS RI)

Examples

Getting Guice injections on JAX-RS resources

You will need the guice-bridge in your project.

```
@ApplicationPath("api")
public class ApiRest extends ResourceConfig {
    @Inject
    public ApiRest(ServiceLocator serviceLocator, ServletContext servletContext) {
        packages("net.sargue.app.api");

        GuiceBridge.getGuiceBridge().initializeGuiceBridge(serviceLocator);
        GuiceIntoHK2Bridge guiceBridge = serviceLocator.getService(GuiceIntoHK2Bridge.class);
        Injector injector = (Injector) servletContext.getAttribute(Injector.class.getName());
        if (injector == null)
            throw new RuntimeException("Guice Injector not found");
        guiceBridge.bridgeGuiceInjector(injector);

        register(RolesAllowedDynamicFeature.class);
    }
}
```

Read Using Guice with Jersey 2 (JAX-RS RI) online: https://riptutorial.com/guice/topic/6376/using-guice-with-jersey-2--jax-rs-ri-

Credits

S. No	Chapters	Contributors
1	Getting started with guice	Community, Lachezar Balev
2	Guice method interceptor	David
3	Using Guice with Java WebSockets (JSR-356)	sargue
4	Using Guice with Jersey 2 (JAX-RS RI)	sargue