

# Fhlintstone User Manual

## Table of Contents

1. Introduction .....	2
2. Prerequisites .....	3
3. Usage .....	4
3.1. General Principles .....	4
3.2. StructureDefinition Settings .....	5
3.3. Maven Integration .....	6
3.4. Command Line Interface .....	7
4. Limitations .....	8
4.1. General Limitations .....	8
4.2. StructureDefinition Class Generation .....	8
4.3. ValueSet Enum Generation .....	8
5. Further Information .....	9
6. Glossary .....	10

---

# 1. Introduction

Fhlintstone is a Java code generator to support [HL7 FHIR](#) developers using the popular [HAPI FHIR](#) framework. Based on one or multiple [NPM Packages](#) containing the various definitions, it can be used to

- generate Java enums for [ValueSets](#)
- generate Java classes to implement [custom resource and extension classes](#) for [StructureDefinitions](#) and
- facilitate [handling profiles and extensions](#) by providing builders tailored to the specific FHIR adaptations described by the package contents.

If you are a Java developer tasked with developing a [HL7 FHIR](#) application that is based on [HAPI FHIR](#) (or you are considering [HAPI FHIR](#) as a base of your implementation), Fhlintstone might be able to save you a lot of time by automating the rather tedious task of generating Java enums and classes that correspond to the [Implementation Guide](#) you probably have to adhere to and the [profiles](#) contained within.

## 2. Prerequisites

In order to use Fhlintstone, you need one or more [FHIR NPM Packages](#) that contain the definitions of the objects you want to generate code for. Not only that, but you also need to **understand** the contents of the package and possibly the relations to other packages. Fhlintstone is a sophisticated tool in some ways, but it is not intelligent - it requires clear directions in the form of a configuration that you have to provide. Fhlintstone is not designed to support the interactive exploration of FHIR NPM package contents - there are other tools out there that are much more suitable for this.

Note that you currently have to provide NPM packages that contain [Snapshot statements](#). Fhlintstone is currently not able to work with packages that only contain [Differential statements](#) (see also [Limitations](#)).

You also will need to provide all relevant dependencies of the NPM packages, up to and including the HL7 base package. Fhlintstone will check the NPM package manifests and warn you about missing packages, but it currently will not attempt to download these packages automatically. Also note that not all dependencies are stated explicitly: It is not uncommon **not** to add a dependency to a package that contains a [CodeSystem](#) that is being referred to. You will have to resolve these dependencies manually in order for Fhlintstone to be able to work with the package contents, although Fhlintstone will at least notify you about the missing resources.

You can either use Fhlintstone from the command line or integrate it into your Maven build process. You will need a Java Runtime Environment version 21 or higher to run Fhlintstone. The Maven integration has been tested with version 3.9.

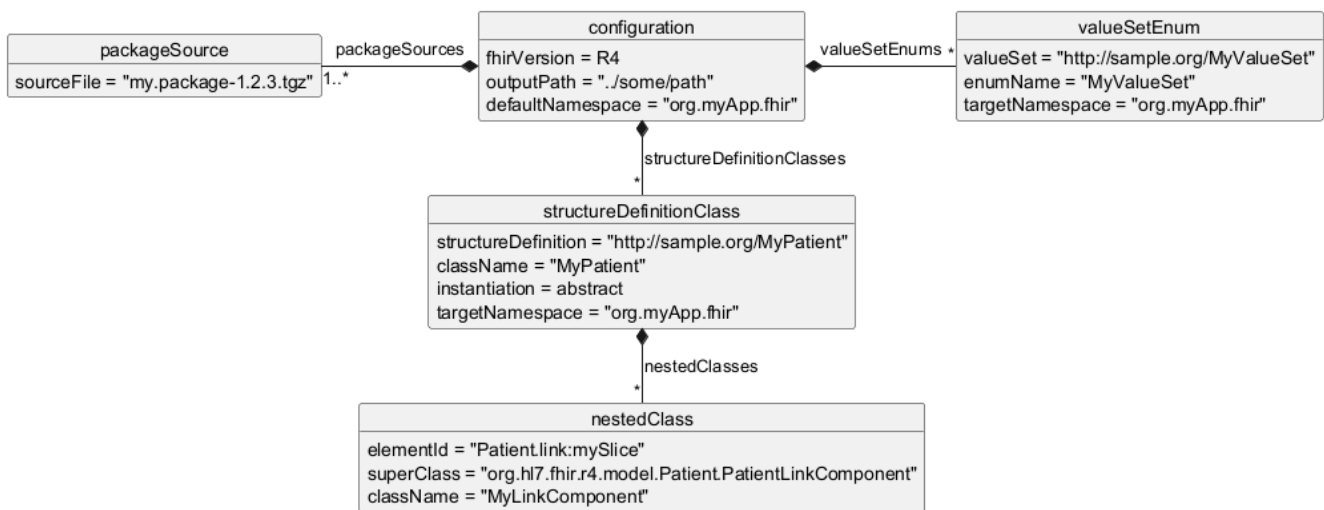
Please check the list of [limitations](#) to ensure that you won't be surprised by one of Fhlintstone's current (or maybe even permanent) limitations.

## 3. Usage

### 3.1. General Principles

Fhlintstone uses a common configuration structure for both the Maven integration and the command line interface. The configuration consists of the following sections:

- general settings,
- a list of one or multiple [FHIR NPM Package](#) files,
- a list of [ValueSet](#) mappings and
- a list of [StructureDefinition](#) mappings.



Either (or both) of the mapping lists may be empty, i.e. it is possible to generate only ValueSet enums or only StructureDefinition classes. Each mapping entry specifies both the source object (i.e. the ValueSet or the StructureDefinition) and the name of the target type (enum or class). Fhlintstone will not attempt to automatically generate Java type names from the FHIR definitions since that will usually lead to some near-illegible type names.

#### 3.1.1. General Settings

The general section provides the following options:

Option	Description	Required	Values
<b>fhirVersion</b>	<i>The FHIR version (or release) to use.</i>	yes	<i>R4, R4B, R5 (see <a href="#">Limitations</a>)</i>
<b>outputPath</b>	<i>The path to write the generated sources to.</i>	yes	
<b>defaultNamespace</b>	<i>The Java namespace (package name) that will be used for all generated objects that do not have an individual namespace assigned.</i>	no	

### 3.1.2. NPM Package Files

The NPM package files are listed by name only; currently there are no further package-level configuration options.

Option	Description	Required	Values
<code>sourceFile</code>	<i>The name (and optional path) to the NPM package file.</i>	yes	

### 3.1.3. ValueSet Settings

To generate a Java enum type for a [ValueSet](#), an entry with the following options must be added to the configuration:

Option	Description	Required	Values
<code>valueSet</code>	<i>The URI identifying the ValueSet.</i>	yes	
<code>targetNamespace</code>	<i>The Java namespace (package name) that will be used for the enum. If not specified, the <code>defaultNamespace</code> will be used.</i>	no	
<code>enumName</code>	<i>The name of the Java type to generate.</i>	yes	

## 3.2. StructureDefinition Settings

To generate a Java class type for a [StructureDefinition](#), an entry with the following options must be added to the configuration:

Option	Description	Required	Values
<code>structureDefinition</code>	<i>The URI identifying the StructureDefinition.</i>	yes	
<code>targetNamespace</code>	<i>The Java namespace (package name) that will be used for the class. If not specified, the <code>defaultNamespace</code> will be used.</i>	no	
<code>className</code>	<i>The name of the Java type to generate.</i>	yes	
<code>instantiation</code>	<i>The way in which the generated class can be instantiated.</i>	yes	<i>abstract, default, builder</i>

Additionally, you can specify a mapping of nested classes for each StructureDefinition. These are commonly used when a FHIR StructureDefinition contains an anonymous type that is mapped to a nested class in the HAPI structures.

Option	Description	Required	Values
<code>elementId</code>	The full ID of the element for which the nested class is generated (e.g. <code>Patient.name</code> ).	yes	
<code>className</code>	The name of the nested class to generate.	yes	
<code>superClass</code>	The name of the superclass the nested class is derived from.	yes	
<code>instantiation</code>	The way in which the generated class can be instantiated.	yes	<i>abstract, default, builder</i>

The instantiation mode controls how the class or nested class is generated:

- **abstract** will generate an abstract class that cannot be instantiated. This is useful if you have to include intermediary StructureDefinitions that you're not planning on using yourself.
- **default** will generate a standard implementation class as described by the [HAPI FHIR](#) documentation. Note that these classes contain the structural components defined by the package contents, but do not contain provisions to handle slices or fixed values.
- **builder** will (eventually) supply a builder for each class that will provide additional measures to ensure that the generated resource complies with the package contents. It will, for example, set fixed values and ensure that cardinality constraints are met. Note that this option is not yet implemented (see [Limitations](#)).

### 3.3. Maven Integration

To use Fhlintstone in your Maven build, add a plugin to your POM:

```
<plugin>
  <groupId>de.fhlintstone</groupId>
  <artifactId>fhlintstone-maven-plugin</artifactId>
  <version>x.y.z</version>
  <configuration>
    <fhirVersion>R4</fhirVersion>
    <outputPath>${project.basedir}/src/gen/java</outputPath>
    <defaultNamespace>org.myApp.fhir</defaultNamespace>
    <packageSources>
      <packageSource>
        <sourceFile>src/main/resources/hl7.fhir.r4.core-4.0.1.tgz</sourceFile>
      </packageSource>
      <packageSource>
        <sourceFile>src/main/resources/my-profile.1.2.3.tgz</sourceFile>
      </packageSource>
    </packageSources>
    <valueSetEnums>
      <valueSetEnum>
        <valueSet>http://sample.org/demo/ValueSet/MyValueSet</valueSet>
      </valueSetEnum>
    </valueSetEnums>
  </configuration>
</plugin>
```

```

        <enumName>MyValueSet</enumName>
    </valueSetEnum>
</valueSetEnums>
<structureDefinitionClasses>
    <structureDefinitionClass>

<structureDefinition>http://sample.org/demo/StructureDefinition/GeneralPatient</structureDefinition>
        <className>GeneralPatient</className>
        <instantiation>abstract</instantiation>
    </structureDefinitionClass>
<structureDefinitionClass>

<structureDefinition>http://sample.org/demo/StructureDefinition/SpecializedPatient</structureDefinition>
        <className>SpecializedPatient</className>
        <instantiation>builder</instantiation>
        <nestedClasses>
            <nestedClass>
                <elementId>Patient.link:mySlice</elementId>
                <superClass>

org.hl7.fhir.r4.model.Patient.PatientLinkComponent</superClass>
                    <className>MyLinkComponent</className>
            </nestedClass>
        </nestedClasses>
    </structureDefinitionClass>
</structureDefinitionClasses>
</configuration>
<executions>
    <execution>
        <goals>
            <goal>generate-code</goal>
        </goals>
    </execution>
</executions>
</plugin>

```

## 3.4. Command Line Interface

TODO: document the command line interface

## 4. Limitations

[HL7 FHIR](#) is a very powerful framework that allows for the definition of complex structures. Not all of the options provided by FHIR are currently available when using Fhlintstone. You may want to check the [list of known issues](#). You should be especially aware of the following limitations:

### 4.1. General Limitations

- Currently only FHIR release R4 is fully supported. Support for Releases R4B and R5 is mostly implemented, but incomplete. ([issue #15](#))

### 4.2. StructureDefinition Class Generation

- [StructureDefinitions](#) that only contain [Differential statements](#) are not supported at the moment. The package has to contain a [Snapshot statement](#). ([issue #17](#))
- Generation of Builders is not yet implemented. ([issue #36](#))
- While it is possible to generate Java enums for [ValueSets](#), the enums are currently not used in the generated classes. ([issue #73](#))
- Typed references like `Reference(Organization|Practitioner|PractitionerRole)` are not yet fully supported. ([issue #50](#), [issue #72](#))

### 4.3. ValueSet Enum Generation

- Support for filter specifications in [ValueSets](#) is limited:
- Currently only String and Integer are supported as data types for value set filter. ([issue #44](#))
- The semantics for the filter "=" with values of type integer, dateTime and decimal are not implemented. ([issue #34](#))
- Designations for the filters "=" and "regex" are not implemented. ([issue #33](#))
- [ValueSets](#) that use exclusions are currently not supported. ([issue #18](#))
- For [StructureDefinitions](#), not all attributes of the `@Child` annotation are generated. ([issue #39](#))
- Currently, the enum values can only be generated based on the code. ([issue #9](#))
- The enum values are missing Javadoc comments. This is a limitation of the underlying code generator. ([issue #5](#))



## 5. Further Information

TODO: assemble further information

## 6. Glossary

Term	Definition	Links
<i>CodeSystem</i>	<i>The CodeSystem resource is used to declare the existence of and describe a code system or code system supplement and its key properties, and optionally define a part or all of its content. Code systems define which codes (symbols and/or expressions) exist, and how they are understood.</i>	<a href="#">Resource CodeSystem - Content</a>
<i>Differential statement</i>	<i>Differential statements are one of two ways to describe the inner structure of a <a href="#">StructureDefinition</a>. Differential statements describe only the differences that they make relative to the base structure definition. In order to properly understand a differential structure, it must be applied to the structure definition on which it is based.</i>	<a href="#">Base Resource Definitions</a>
<i>Extension</i>	<i>Every element in a <a href="#">Resource</a> can have extension child elements to represent additional information that is not part of the basic definition of the resource. The use of extensions is what allows the FHIR specification to retain a core simplicity for everyone. To make the use of extensions safe and manageable, there is strict governance applied to the definition and use of extensions. Although any implementer can define and use extensions, there is a set of requirements that must be met as part of their use and definition.</i>	<a href="#">Extensibility</a>
<i>FHIR NPM Package</i>	<i>A set of <a href="#">Resources</a> bundled as a machine-readable archive file. Not to be confused with <a href="#">FHIR Package</a>.</i>	<a href="#">FHIR NPM Packages</a>
<i>FHIR Package</i>	<i>In the context of <a href="#">Profiling</a>, a group of related adaptations that are published as a group within an Implementation Guide. Not to be confused with <a href="#">FHIR NPM Package</a>.</i>	<a href="#">Profiling FHIR</a>
<i>FHIR Profile</i>	<i>A set of constraints on a <a href="#">Resource</a> represented as a <a href="#">StructureDefinition</a>.</i>	<a href="#">Profiling FHIR</a>
<i>HAPI FHIR</i>	<i>The HAPI FHIR library is an implementation of the <a href="#">HL7 FHIR</a> specification for Java.</i>	<a href="#">HAPI FHIR</a>
<i>HL7 FHIR</i>	<i>Fast Healthcare Interoperability Resources (FHIR) are a standard for health care data exchange, published by HL7®.</i>	<a href="#">HL7 FHIR</a>

Term	Definition	Links
<i>Implementation Guide</i>	<i>A coherent and bounded set of adaptations that are published as a single unit. Validation occurs within the context of the Implementation Guide.</i>	<a href="#">Profiling FHIR</a>
<i>JavaPoet</i>	<i>JavaPoet is a Java API for generating <code>.java</code> source files.</i>	<a href="#">JavaPoet (Github)</a>
<i>Mockito</i>	<i>Mockito is a mocking framework that provides a clean &amp; simple API, resulting in readable tests and clean verification errors.</i>	<a href="#">Mockito</a>
<i>Resource</i>	<i>A resource is an entity that has a known identity by which it can be addressed, identifies itself as one of the types of resource defined in the <a href="#">HL7 FHIR</a> specification, contains a set of structured data items as described by the definition of the resource type and has an identified version that changes if the contents of the resource change.</i>	<a href="#">Base Resource Definitions</a>
<i>Snapshot statement</i>	<i>Snapshot statements are one of two ways to describe the inner structure of a <a href="#">StructureDefinition</a>. Snapshot statements are a fully calculated form of the structure that is not dependent on any other structure.</i>	<a href="#">Base Resource Definitions</a>
<i>StructureDefinition</i>	<i>A definition of a FHIR structure. This resource is used to describe the underlying resources, data types defined in FHIR, and also for describing extensions and constraints on resources and data types.</i>	<a href="#">Resource StructureDefinition - Content</a>
<i>ValueSet</i>	<i>A ValueSet resource instance specifies a set of codes drawn from one or more code systems, intended for use in a particular context. ValueSets link between <a href="#">CodeSystems</a> definitions and their use in coded elements.</i>	<a href="#">Resource ValueSet - Content</a>