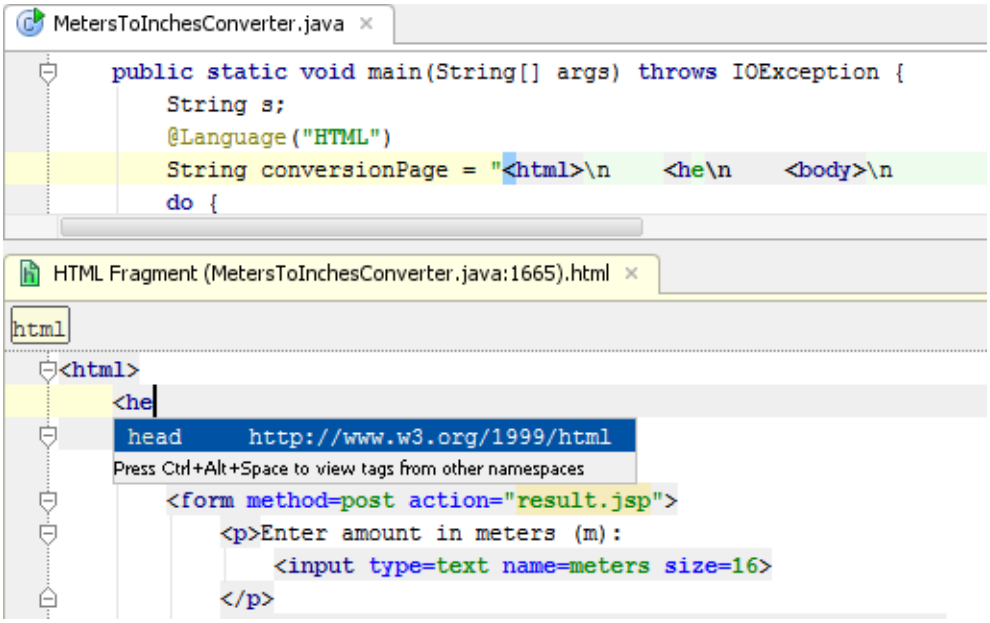
Overview

IntelliLang is a combination of three basic kinds of functionality that are meant to support the developer in dealing with certain tasks that relate to (custom) languages in IntelliJ IDEA:

- **Language Injection**: Editing the code that is embedded into String literals and XML fragments **natively**.

  You can open and modify an injected language code fragment in the editor. In this way you get all the necessary coding assistance, as if you were working with the source code in the corresponding language.



  To open an injected language code fragment in the editor, use the *Edit <Language> Fragment* intention action.

- **Pattern Validation**: Provides assistance in making sure that Strings being passed to and from methods match a particular regular expression
- **Regular Expression Support**: A custom language implementation for regular expressions

Language injection

This makes use of the new possibilities of IntelliJ IDEA to treat String literals, XML text and attributes as fragments of an arbitrary language (called *Language Injection*). The plugin makes this newly introduced API readily available to everybody for their daily use through two very simple means: Either by using some Java annotations to mark String fields, local variables, method parameters and methods returning Strings as containing a certain language, or by just using a simple UI configuration. There is a set of annotations provided by the plugin, but the actual annotations are freely configurable to avoid any unwanted dependencies.

This enables the developer to get the benefit of a wide range of edit-time features, such as syntax error highlighting, completion, inspections and a lot more while editing fragments of e.g. JavaScript inside regular Java code or in XML files of custom schemas that IntelliJ IDEA usually doesn't know about.

Pattern validation

Additionally, the plugin allows you to annotate Java elements of type String to have them checked for compliance with certain Regular Expressions. This can be useful for very simple *languages* where the developer needs to make sure that an expression conforms to a certain syntax, e.g. that a String is a legal Java identifier or a valid printf-like pattern used by `java.util.Formatter`.

This can both be validated on-the-fly while editing the code as well as during runtime (method parameters and return value only, like the @NotNull instrumentation of IntelliJ IDEA core) by instrumenting the compiled classes with assertions that match the value against the supplied pattern.

Regular expression support

This part of the plugin implements language-support for `java.util.regex.Pattern` and has been mainly created to support the IntelliLang plugin by adding support for the micro-language that is probably one of the most often used one inside Strings. It features complete support for the syntax of the SDK's regular expression implementation and adds some further features, such as

- Completion and validation for character property names (e.g. `\p{javaJavaIdentifierStart}`) which nobody can usually remember anyway
- Validation and navigation for the use of back-references (e.g. `\1`), e.g ctrl-b navigates to the capturing group the backref refers to.
- Intention Actions to simplify usages of repeated character occurrences, e.g. `a{0,1}` is offered to be converted to `a?`
- "Surround With" capturing/non-capturing group
- and more