Type Extension Methods

The language jetbrains.mps.extensionMethods provides a way to extend any valid MPS type with newly defined or overriden methods, akin to Java static methods.

```
"This is an ordinary string with a surpricing behavior!".myNewMethod()
"Develop with pleasure!".isAPleasureMessage()
int num = ...
num.isPrime()
```

Whereas static methods never become an internal part of the extended class and one has to always specify the "extended" object to operate on as one of the parameters to the extended method, with an extension method the new method gets added directly to the list of operations available on the target type.

So, provided we wanted to add a reverse method to the string type, instead of the good old "static method" way:

```
public static string reverse(string target) {
    //reverse the target

}
```

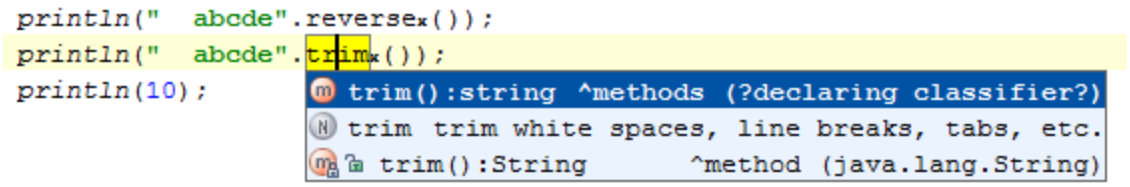
we would create new Extension Methods through New -> j.m.baseLanguage.extensionMethods/type extension, define the new method and tie it to the string class:

```
public extension methods MySampleMethods for string {
  <<static fields>>
  public string reverse() {
    //reverse the string, refering to it through 'this'
  }
}
```

The very same mechanism can be used to override existing methods. And when in need to call the original method, just call it on this:

```
 public extension methods MySampleMethods for string {
  <<static fields>>
  public string trim() {
    this.trim() + " trimmed"  //calls the original trim() implementation
  }
}
```

Since MPS does a good job to visually distinguish the original and overriden methods through the extension methods mechanism, you can't make a mistake picking the right one from the drop-down list.



Obviously this mechanism can be used to implement orthgonal concepts on your own domain objects as well:

```
public extension methods Sample for my_type {
  <<static fields>>
  public int foo() {
    return this.bar();
  }

}
```

With the declaration as above, one could write an operation on type my_type:

```
my_type var = ...;
var.foo();
```

Root Nodes

There are two equally good ways to extend types with methods. Type Extension allows to you to add methods to a single type in one place, while Simple Extension Method Container comes in handy, when you need one place to implement an orthogonal concept for multiple different types.

Type Extension

This root contains declarations of extension methods for a single type.

```
[public|protected|private] extension methods containerName for extendedType {
  <<static fields>>
    <<extension methods>>
}
```

Extension method declaration.

```
public ret_Type methodName() {
  <no statements>
}
```

Simple Extension Method Container

```
[public|protected|private] extension methods containerName {
  <<static fields>>
    <<extension methods>>
}
```

Extension method declaration. The target type is specified per method.

```
public ret_type methodName() for extendedType {
  <no statements>
}
```

declaration part	allowed contents
containerName	any valid identifier
extendedType	any valid MPS type

Both roots may contain one or more static fields. These are available to all methods in the container.

—

Previous Next