Type Migration

The Type Migration refactoring allows you to automatically change a member type (e.g. from integer to string), and data flow dependent type entries, like method return types, local variables, parameters etc. across the entire project. It also lets automatically convert variable or method return type between arrays and collections. If any conflicts are found IntelliJ IDEA warns you about them.

> This refactoring is also available from UML Class diagram.

Example

`f: int -> String`

<table>
<tr><th>Before</th><th>After</th></tr>
<tr><td>

```
               int
               f;
void bar(int i) {}
void foo() {
    bar(f);
}
```

</td><td>

```
String f;
void bar(String i) {}
void foo() {
    bar(f);
}
```

</td></tr>
</table>

`I<String> -> I<Integer>`

<table>
<tr><th>Before</th><th>After</th></tr>
<tr><td>

```
               interface I<T> {
    T foo(T t);
}

class A implements I<String> {
    String myString;
    public String foo(final String s) {
        if (s == null) {
            return
               myString;
        }
        return s;
    }
}
```

</td><td>

```
               interface I<T> {
    T foo(T t);
}

class A implements I<Integer> {
    Integer myString;
    public Integer foo(final Integer s) {
        if (s == null) {
            return
               myString;
        }
        return s;
    }
}
```

</td></tr>
</table>

`myResult: ArrayList<String> -> String[]`

<table>
<tr><th>Before</th><th>After</th></tr>
<tr><td>

```
               public
               class ResultContainer {
    private ArrayList<String> myResult;
    public String[] getResult() {
        return
               myResult.toArray(new String[myResult.size()]);
    }
}
```

</td><td>

```
               public
               class ResultContai
    private String[] myResult;
    public String[] getResult() {
        return
               myResult;
    }
}
```

</td></tr>
</table>

To change a type

1. Place the caret on the type to be refactored in the editor.
2. On main menu, choose **Refactor | Type Migration**, or press `Ctrl+Shift+F6`.
3. In the **Type Migration** dialog box, specify the new type and scope where to look for the usages.
4. Click **Preview** and review the items that will be affected. If needed, exclude usages from refactoring. To do that, right-click usage in the Type Migration Preview and select **Exclude**. When done, click **Migrate**.

See Also

Reference:

- Type Migration Dialog
- Type Migration Preview

Web Resources:

- Developer Community