z2-Environment - Improvement #2059

Enhance standard module layout for better reload performance due to impl updates.

15.03.2020 18:07 - Henning Blohm

Status:	In Progress	**Start date:**	15.03.2020
Priority:	Normal	**Due date:**	
Assignee:	Henning Blohm	**% Done:**	0%
Category:	z2-core	**Estimated time:**	0.00 hour
Target version:	2.10		
origin:			

Description

More efficient handling of API and Impl in Rebuilds (#2059)

Currently, in the dependency map used by the ComponentsBuilder, we analyze the necessity of a rebuild by comparing build timestamps maintained in a dependency map in dependent components with a latest built timestamp kept in the build lock file of the dependency component.

If there is a rebuild of an implementation part of a Java component, the build timestamp is updated and so a rebuild of all dependency components forced - which is unnecessary, unless there was a change in the API as well.

We would like that a rebuild of dependency Java components is not forced, if there is no change of the API of a dependency component

With the current model

At the level of the dependency component:

- Separate build timestamps for API and implementation as maintained in the build lock file.

At the level of the dependent component:

- When checking for rebuild in dependency components only check for matches with the API build timestamp.
- Keep a cache of build results and copy from there, if no deemed necessary - because the sources are not younger than the last build-timestamp

A new component split

The solution would be altogether completely natural, if api and impl as well as test would be different "Java" components.

We could have a simple layout:

```
<module>/api
<module>/impl
```

For compatibility, we could consider

```
<module>/impl -(private)-> <module>/java/impl
<module>/java --(public)-> <module>/java/api
```

So overall we have

iuggAIKKKCAAgoooEAhBQySCjksuTploZyLyZMqKuD8qOjAe9sKKKCAAgoooIACCiiQVsAgKa1vytaHhRD4n4cCCnxbwPnhU6GAAgoooIACCiiggAIKJBAwSEqAapMKKKCAAgoooIACCiiggAIKKKBAVxQwSOqKo+o9KaCAAgoooIACCiiggAIKKKCAAgkEDJISoNqkAgoooIACCiiggAIKKKCAAgoo0BUFDJK64qh6TwoooIACCiiggAIKKKCAAgoooEACAYOkBKg2qYACCiiggAIKKKCAAgoooIACCnRFAYOkrjiq3pMCCiiggAIKKKCAAgooIACCiiQQMAgKQGqTSqggAIKKKCAAgoooIACCiiggAJdUcAgqSuOqvekgAIKKKCAAgoooIACCiiggAIKJBD4f9hUOcvfF6jeAAAAAElFTkSuQmCC

Problems to Expect

This approach has a few downsides:

Missing Dependencies

In many places provisioning of implementation instances via API provided lookups has been implemented. So, effectively an API dependent component holds on to an impl instance without being subject to a dependency that would invalidate the consuming component, if only the implemtation component is invalidated.

Approach: Upon lookup supply an object or a resource handle representing the dependant. Upon an object, its classloader will be inspected and the related Java component made a dependant to the providing implementation. Upon a resource handle, the identified resource will be made a dependant.

Acceptance Criterias

- There is a new component type **com.zfabrik.impl**
  - The impl component is a reduced Java Component that only supports a private loader.
  - All sources are found in **<component>/src**, binaries in **<component>/bin/{lib|classes}**
  - The component type **com.zfabrik.impl** supports **impl.references** and **impl.includes**, etc.

- There is a new component type **com.zfabrik.api**
  - The impl component is a reduced Java Component that only supports a public loader.
  - All sources are found in**<component>/src**, binaries in **<component>/{lib|classes}**
  - The component type **com.zfabrik.api** supports **api.references** and **api.includes**, etc.
  - The component type **com.zfabrik.java** by default has a public reference to <module>/api

- There is a new component type **com.zfabrik.test**
  - The test component is a reduced Java Component that only supports a private loader.
  - All sources are found in **<component>/src**, binaries in **<component>/bin/{lib|classes}**
  - The component type **com.zfabrik.test** supports **test.references** and **test.includes**, etc.
  - For test components, **testing.references** can access the private loader of the target
  - A test component has a default **testing.reference** to **<module>/impl**

- **JavaComponentUtil.getJavaComponent** is deprecated and replaced by **JavaComponentUtil.getImplComponent** and **JavaComponentUtil.getApiComponent**
  - All usages of **JavaComponentUtil.getJavaComponent** when looking for component implementations are replaced by **JavaComponentUtil.getImplComponent**
  - **JavaComponentUtil.getImplComponent** checks for **<module>/impl** and if that cannot be found falls back to **<module>/java**
  - **JavaComponentUtil.getApiComponent** checks for **<module>/java** and if that cannot be found defaults to **<module>/api**

- Eclipsoid resolves for **<module>/java** and **<module>/api**
- Eclipsoid supports two module templates:
  - One with /java (legacy, pre 2.9)
  - One with /api and /impl

Related issues:

Related to z2-Environment - Improvement #2081: Support a kotlin compiler addon	**New**	**28.09.2020**
Related to z2-Environment - Improvement #2082: Support a clojure compiler addon	**New**	**28.09.2020**

History

#1 - 05.04.2020 23:06 - Henning Blohm

- Description updated

#2 - 05.04.2020 23:12 - Henning Blohm

- Description updated

#3 - 28.09.2020 11:42 - Henning Blohm

- Related to Improvement #2081: Support a kotlin compiler addon added

#4 - 02.10.2020 16:17 - Henning Blohm

- Related to Improvement #2082: Support a clojure compiler addon added

#5 - 11.03.2021 21:35 - Henning Blohm

- Status changed from New to In Progress

#6 - 11.07.2021 16:27 - Henning Blohm

- Target version changed from 2.9 to 2.10