URL-based dynamic loading of providers for artifact retrieval and deployment

URL-based dynamic loading of providers for artifact retrieval and deployment

With the addition of dynamic collections to the Plexus container it is now possible to dynamically load providers that are used for artifact retrieval and deployment. This would have the benefit of removing all the transports from the core distribution of Maven, and allow all transports to be used in the same consistent fashion. If we look at some of the patterns we use I think we can deterministically tell what provider we need to load for a given retrieval or deployment operation.

Common URLs used for deployment

1. file:///path/to/repository
2. scp://people.apache.org/www/people.apache.org/repo/m2-ibiblio-rsync-repository
3. dav:https://dav.codehaus.org/repository/plexus

Common URLs used for retrieval

1. http://people.apache.org/repo/m2-snapshot-repository
2. file:///path/to/repository
3. scp://people.apache.org/www/people.apache.org/repo/m2-ibiblio-rsync-repository
4. dav:https://dav.codehaus.org/repository/plexus

Now in the case in the first case we may have a problem if there are multiple implementations, say http, for one provider id. We may need more information as we are just assuming that there is only one provider in the classpath right now but there may be multiple implementations available. We may also want an extended notion to allow for the specification of the version of the provider. We might be able to do this in the distribution management element.

1. Look at the URL to determine the provider
2. Determine the version of the provider
3. Load the provider and its dependencies into a ClassLoader
4. Use the ClassLoader with the loaded provider for retrieval or deployment operation

Notes

- The operation of retrieval is completely orthogonal to the execution of the build lifecycle so it would be pretty easy to lay down the retrieved artifacts and then get a list of artifacts to work with.
- The key in the map for the provider needs to be keyed by version and provider ID if we are going to use a collection and cache the providers during the life of the embedder.
- For each key in the map we really need the per-component ClassLoader to execute the retrieval or deployment.
- We have the deployment configuration and requirements fragmented in two places. We have the distributionManagement element, and we additionally have the specification of deployment plugin. In 2.1 it may be wiser to acknowledge that installation and deployment are fundamental operations in Maven and we should configure that in one place. We can still allow the variation of version, but we move toward eliminating multiple places where deployment and installation information is kept.

```
<project>
  <distributionManagement>

    <repository>

      <id>apache.releases</id>

      <name>Apache Release Distribution
Repository</name>


<url>scp://people.apache.org/www/people.apac
he.org/repo/m2-ibiblio-rsync-repository</url
>
    </repository>

    <snapshotRepository>

      <id>apache.snapshots</id>

      <name>Apache Development Snapshot
Repository</name>


<url>scp://people.apache.org/www/people.apac
he.org/repo/m2-snapshot-repository</url>

    </snapshotRepository>

  </distributionManagement>
</project>
```

```
<project>
  <build>
    <plugins>
         <plugin>

            <inherited>true</inherited>


<artifactId>maven-deploy-plugin</artifactId>

            <version>2.3</version>

            <configuration>


<altDeploymentRepository>${deploy.altReposit
ory}</altDeploymentRepository>


<updateReleaseInfo>true</updateReleaseInfo>

            </configuration>

          </plugin>
  </plugins>
</project>
```

So this may be another option where we combine the information for deployment. This may not be ideal but it's a shot at what might be possible.

```
<project>
  <distributionManagement>

    <version>2.3</version>

    <configuration>


<altDeploymentRepository>${deploy.altReposit
ory}</altDeploymentRepository>


<updateReleaseInfo>true</updateReleaseInfo>

    </configuration>

    <repository>

      <id>apache.releases</id>

      <name>Apache Release Distribution
Repository</name>


<url>scp://people.apache.org/www/people.apac
he.org/repo/m2-ibiblio-rsync-repository</url
>
    </repository>

    <snapshotRepository>
```

```
      <id>apache.snapshots</id>

      <name>Apache Development Snapshot
Repository</name>


<url>scp://people.apache.org/www/people.apac
he.org/repo/m2-snapshot-repository</url>

    </snapshotRepository>
```

```
  </distributionManagement>
</project>
```

We may also want to consider official support for the many types of repositories that organizations need like the dev, test, qa, production in an extensible. These are typically dealt with in profiles but makes it hard to see the topology of the repositories in an organization.