HowToRunTapestry5OnJBoss6Dot1

Tapestry 5.x cannot find the core pages and components from the URLs provided from classloaders in JBoss 6.1. We will override Tapestry's ClasspathURLConverter with one customised for JBoss 6.1. This builds on the work done for JBoss 5 in https://issues.apache.org/jira/browse/TAP5-576 .

```
public class ClasspathURLConverterJBoss6Dot1 implements ClasspathURLConverter
{
    private static Logger log = Logger.getLogger(ClasspathURLConverterJBoss6Dot1.class);

    public URL convert(URL url)
    {
        // If the URL is a "vfs" URL (JBoss 6.1 uses a Virtual File System)...

        if (url != null && url.getProtocol().startsWith("vfs"))
        {
            // Ask the VFS what the physical URL is...

            try
            {
                String urlString = url.toString();

                // If the virtual URL involves a JAR file,
                // we have to figure out its physical URL ourselves because
                // in JBoss 6.1 the JAR files exploded into the VFS are empty
                // (see https://issues.jboss.org/browse/JBAS-8786).
                // Our workaround is that they are available, unexploded,
                // within the otherwise exploded WAR file.

                if (urlString.contains(".jar")) {

                    // An example URL: "vfs:/devel/jboss-6.1.0.Final/server/default/deploy/myapp.ear/myapp.war
/WEB-INF/lib/tapestry-core-5.2.6.jar/org/apache/tapestry5/corelib/components/"
                    // Break the URL into its WAR part, the JAR part,
                    // and the Java package part.

                    int warPartEnd = urlString.indexOf(".war") + 4;
                    String warPart = urlString.substring(0, warPartEnd);
                    int jarPartEnd = urlString.indexOf(".jar") + 4;
                    String jarPart = urlString.substring(warPartEnd, jarPartEnd);
                    String packagePart = urlString.substring(jarPartEnd);

                    // Ask the VFS where the exploded WAR is.

                    URL warURL = new URL(warPart);
                    URLConnection warConnection = warURL.openConnection();
                    Object jBossVirtualWarDir = warConnection.getContent();
                    // Use reflection so that we don't need JBoss in the classpath at compile time.
                    File physicalWarDir = (File) invoke(jBossVirtualWarDir, "getPhysicalFile");
                    String physicalWarDirStr = physicalWarDir.toURI().toString();

                    // Return a "jar:" URL constructed from the parts
                    // eg. "jar:file:/devel/jboss-6.1.0.Final/server/default/tmp/vfs/automount40a6ed1db5eabeab
/myapp.war-43e2c3dfa858f4d2//WEB-INF/lib/tapestry-core-5.2.6.jar!/org/apache/tapestry5/corelib/components/".

                    String actualJarPath = "jar:" + physicalWarDirStr + jarPart + "!" + packagePart;
                    return new URL(actualJarPath);
                }

                // Otherwise, ask the VFS what the physical URL is...

                else {

                    URLConnection connection = url.openConnection();
                    Object jBossVirtualFile = connection.getContent();
                    // Use reflection so that we don't need JBoss in the classpath at compile time.
                    File physicalFile = (File) invoke(jBossVirtualFile, "getPhysicalFile");
                    URL physicalFileURL = physicalFile.toURI().toURL();
                    return physicalFileURL;
                }
```

```
            }
            catch (Exception e)
            {
                logger.error(e.getCause().toString());
            }
        }

        return url;
    }

    private Object invokerGetter(Object target, String getter) throws NoSuchMethodException,
InvocationTargetException, IllegalAccessException
    {
        Class<?> type = target.getClass();
        Method method;
        try
        {
            method = type.getMethod(getter);
        }
        catch (NoSuchMethodException e)
        {
            method = type.getDeclaredMethod(getter);
            method.setAccessible(true);
        }
        return method.invoke(target);
    }

}
```

To override the default implementation of ClasspathURLConverter, just add the above ClasspathURLConverterJBoss6Dot1.java to your project and add the following piece of code to your AppModule.java.

```
public static void contributeServiceOverride(MappedConfiguration<Class,Object> configuration)
{
    configuration.add(ClasspathURLConverter.class, new ClasspathURLConverterJBoss6Dot1());
}
```

The above just override the default ClaspathURLConverter service. For more information on overriding a general service, please see http://tapestry.apache.org/ioc-cookbook-overriding-ioc-services.html.

The above fix has been tested with Tapestry 5.2 on JBoss 6.1.0.