1. Move the 'dspace-pom' module from the 'dspace-pom' directory to the root directory of the project.

2. Remove the profile activations from the 'dspace' that are based on the presence of a pom.xml file all together, and instead force the modules to be "always-on" by putting them in <module>/<modules> elements in the 'dspace-pom' module (now in the root directory).

This still allows for modularity, and the Eclipse M2 plugin will recognize each sub-project.

3. Move the remaining profiles from 'dspace' and put them in the 'dspace-pom' module (e.g. the oracle and postgres related profiles).

This is so that users can activate profiles from the root of the project, and removes non-assembly related roles from the 'dspace' module.

4. Rename the 'dspace' module to 'dspace-assembly'.

Now the role of the 'dspace-assembly' project is clear.

5. Create a new profile, named 'assemble' to activate the 'dspace-assembly' module in the 'dspace-pom' module.

So that assembly needs to be specifically invoked, keep it out of the regular build cycle. I suppose this really isn't needed, but.

6. My suggestion would be to update module ancestry to follow the new directory structure:

```
- dspace-pom
  - dspace-jspui
  - dspace-api
  - dspace-assembly
  - dspace-oai
  - language-packs
  - dspace-xmLui
    - dspace-xmLui-api
    - dspace-xmLui-webapp
  - dspace-xmlui
    - dspace-xmlui-wing
  - dspace-lni
    - dspace-lni-core
    - dspace-lni-client
    - dspace-lni-webapp
```

Maven 2 and its plugins just seem to deal nicer when the directory structure of the modules mimics the ancestry of a module.

7. Don't use '<relativePath>', and update subproject dependencies as required.

This results in a source layout like:
<big>7. You'll need to tweak the paths of the assembly descriptors. Assembly would be invoked from the root of the project.</big>

</html>