Lab 5.2 – Container Dependency Injection (CDI)

- Model View Controller (MVC) may have an Issue with our Servlet also being a JMS Producer
- CDI does not only work with Provided Resources (classes). It can also be used on your own Java classes
- Create a class AppMessageProducer in package com.student.ejb
- MOVE your method sendMessage(Student student) to this new class (make the method public this time though)
- Inject into this class our JNDI Administered Objects (ConnectionFactory and Destination)

```
@Resource
private ConnectionFactory foo Queue;
@Resource(name = "FooQueue")
private Queue fooQueue;
```

- REMOVE these Resources from your Servlet

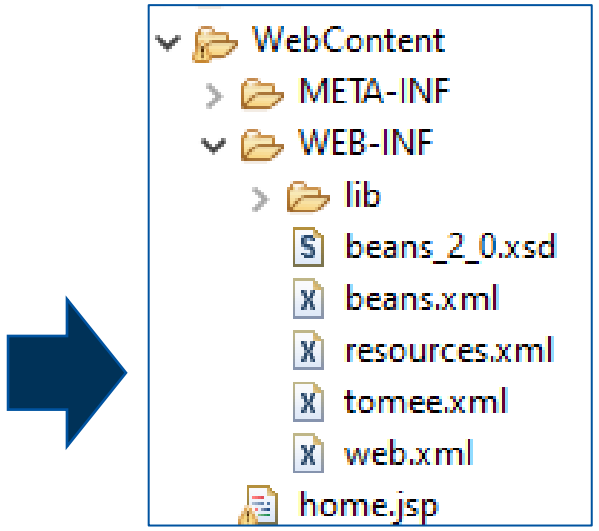
© LearnQuest 2020

A new Producer

- Inject into your StudentServlet this new class

```
@Inject
private AppMessageProducer appMessageProducer;
```

- Where you were previously calling the now deleted sendMessage(Student student) method, delegate to the injected bean i.e. appMessageProducer.sendMessage(student);
- Nearly there. Using CDI 1.0 for custom beans requires a beans.xml under your WEB-INF directory. Even if it is empty.
- We have one for you in the lab setup directory for this Lab. Copy AND the file beans_2.0.xsd (its Schema) to be under your WEB-INF directory in your project



© LearnQuest 2020

CDI 1.0 for Custom Beans

- Restart your server and launch your application by selecting the root node of your project->right click->Run on Server
- Select a single student to retrieve in the web page.
- This should trigger the JMS Producer method in the injected AppMessageProducer and the Listener should pick it up via the configured destination (Queue)
- CDI works for your own defined bean as well!

© LearnQuest 2020