Discussion: CDI

Discussion: Tasklets?

Discussion: Chunking

List for Next Meeting
Proposal: CDI should be available for developers but not an inherent part of the spec.

Developers can access CDI to instantiate Jobs, Steps, Readers/Writers
Discussion: CDI

```java
@Named
@Step(name="TestStep")
public class TestStep {

    @Inject DataCache cache;

    @CreateStep
    public static TestStep create() {
        ContainerLifecycle lifecycle = WebBeansContext.currentInstance().getService(ContainerLifecycle.class);
        lifecycle.startApplication(null);

        BeanManager beanManager = lifecycle.getBeanManager();
        Bean<?> bean = beanManager.getBeans("testStep").iterator().next();
        TestStep step = (TestStep) lifecycle.getBeanManager().getReference(bean, TestStep.class, beanManager.createCreationalContext(bean));

        return step;
    }
}
```

META-INF/beans.xml

```xml
<beans>
    <alternatives>
        <class>com..TestCache</class>
    </alternatives>
</beans>
```

Belongs in @BeginJob
What is difference between Step and Tasklet?

```xml
<step id="Step1">
  <tasklet>
    <chunk reader="itemReader" processor="itemProcessor" writer="itemWriter"/>
  </tasklet>
</step>
```

Can a step have multiple Tasklets?
Discussion: Chunking

- **Spring Model** *(ItemReader, ItemWriter, ItemStream)*
  - Container loops, reading item, processing item “commit-interval” times. Output is buffered.
  - Output from current interval written and committed.

- **WebSphere Model** *(BatchDataStream)*
  - Steps logic reads/writes/returns until done. Output written on each iteration.
  - Commit occurs after every “commit-interval” iterations.
Annotation-based version of WebSphere chunking model

```java
@RunStep
public StepDirective processNextRecord() {
    BatchRecord record = inputRecords.read();
    if (record != null) {
        // process record
        outputRecords.write(record);
        return StepDirective.CONTINUE;
    } else return StepDirective.END;
}
```

Could be viewed as “basic” structure that underlies specific patterns.

E.g. reader-processor-writer (see next slide)
@RunStep

```java
public StepDirective processNextChunk() {
    List<Object> items;
    for (int i = 0; i < commit-interval; i++)
        Object item = itemReader.read();
        if (item != null) {
            item = itemProcessor(item);
            if (item != null) {
                items.add(item);
            }
        } else return StepDirective.END;
    } else return StepDirective.END;
    itemWriter.write(items);
    return StepDirective.CHECKPOINT_AND_CONTINUE;
}
```
List for Next Meeting

- Job Initiation (submit command, launchers, etc)
- Concurrency
- Metrics
- What else?