JSR 352 Expert Group

Working Session

17 February 2012

Agenda

- Discussion: CDI
- Discussion: Tasklets?
- Discussion: Chunking
- List for Next Meeting

Discussion: CDI

- 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

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

Belongs in @BeginJob

Discussion: Tasklets

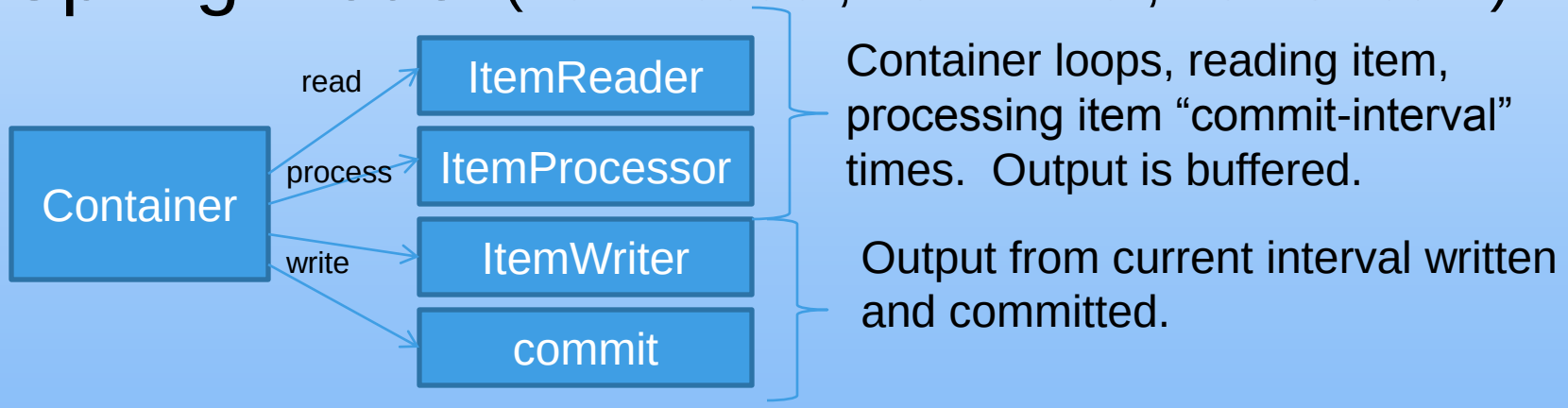
- What is difference between Step and Tasklet?

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

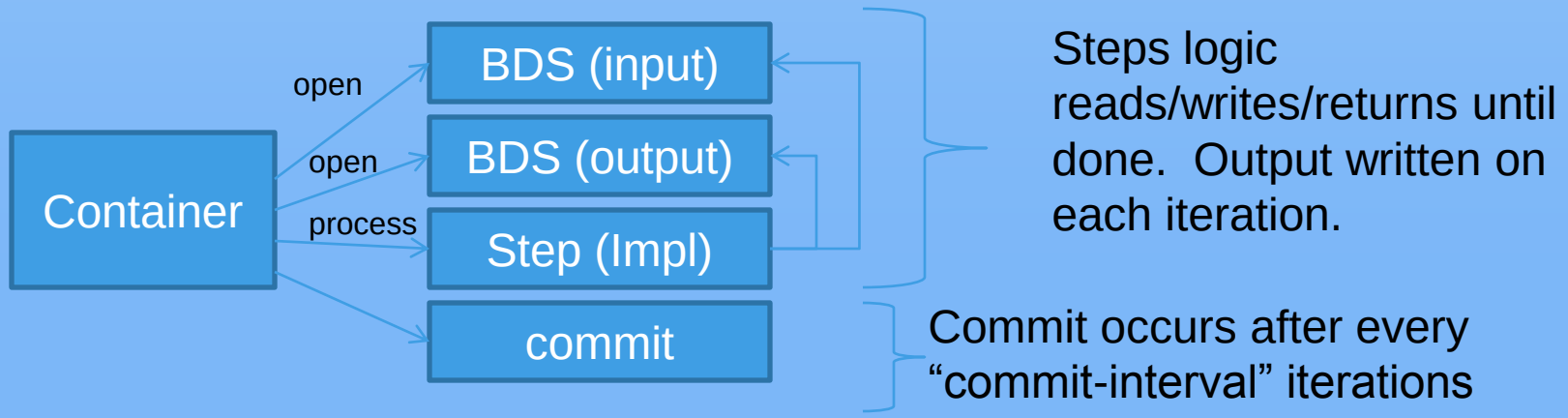
- Can a step have multiple Tasklets?

Discussion: Chunking

- Spring Model (ItemReader, ItemWriter, ItemStream)



- WebSphere Model (BatchDataStream)



Discussion: Chunking

- Annotation-based version of WebSphere chunking model

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

```
public interface BatchReader  <T> {
    public  T read();
}
public interface BatchWriter  <T> {
     public  void write(T record);
}
```

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

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

Discussion: Chunking

```
@RunStep
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;
}
```

Annotations?

@ItemReader
@ItemProcessor
@ItemWriter

List for Next Meeting

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