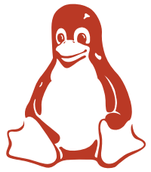
Apache DeltaSpike - the CDI toolbox

Antoine Sabot-Durand · Rafael Benevides

Rafael Benevides

</> DeltaSpike P.M.C member

 Red Hat, Inc.

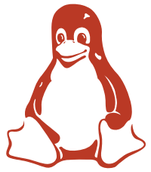
 @rafabene

 rafabene.com

 github.com/rafabene

Antoine Sabot-Durand

 CDI spec lead

 Red Hat, Inc.

 @antoine_sd

 next-presso.com

 github.com/antoinesd

 A talk about CDI eco-system

 Don’t need to be a CDI guru

 But you need to know CDI

Should I stay or should I go ?

If you know the most of these you can stay

`@Inject`

`@Produces`

`Event<T>`

`@Observes`

`@Qualifier`

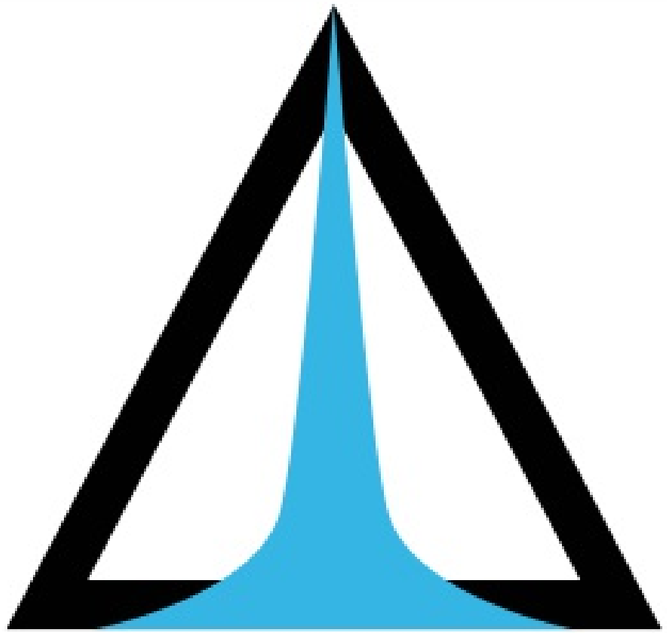
`InjectionPoint`

Agenda

- What is DeltaSpike ?
- Core Module
- Other DeltaSpike Modules
- Question & Answers

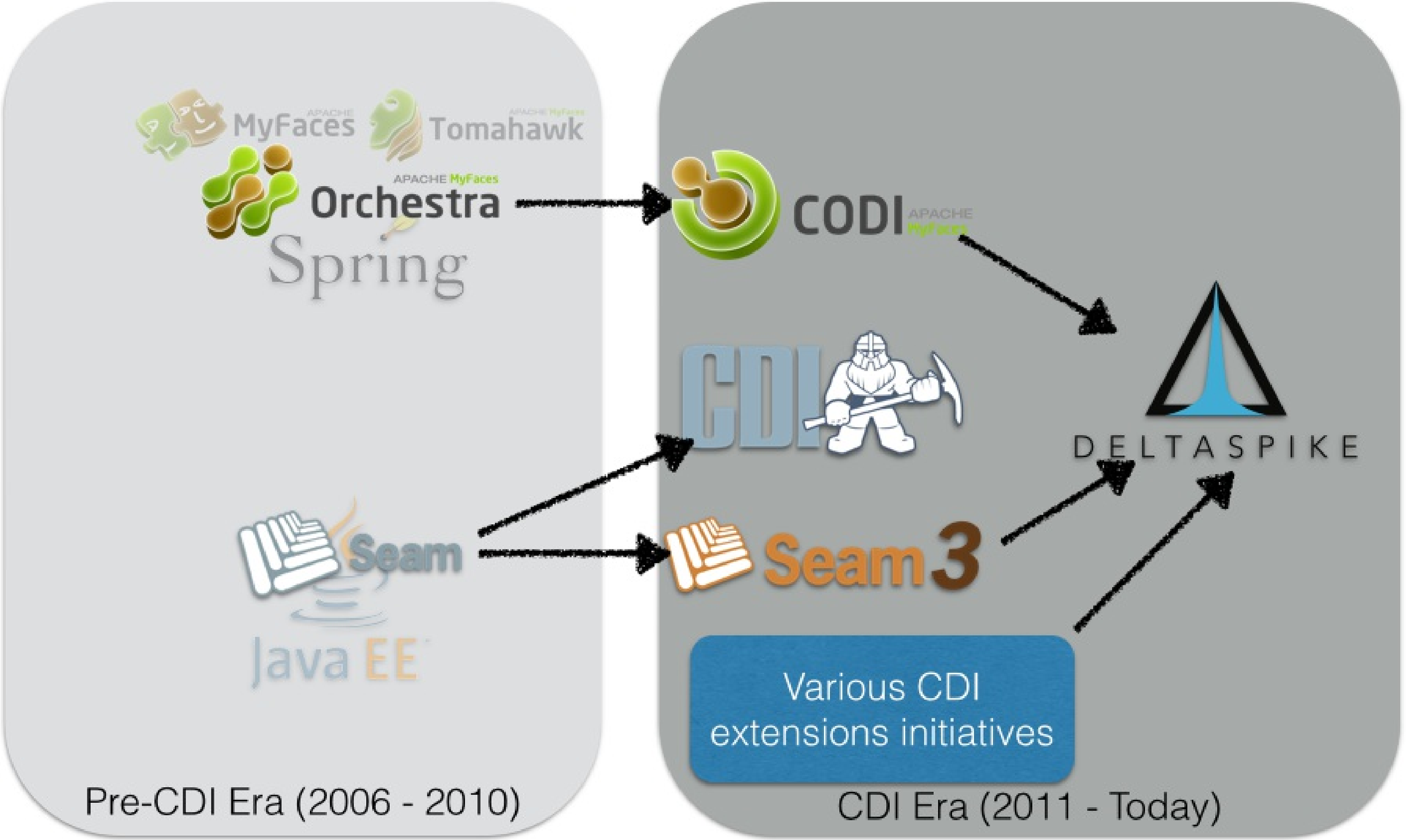
Slides available at rafabene.github.io/deltaspike-cdi-toolbox/

What is DeltaSpike ?



DELTASPIKE

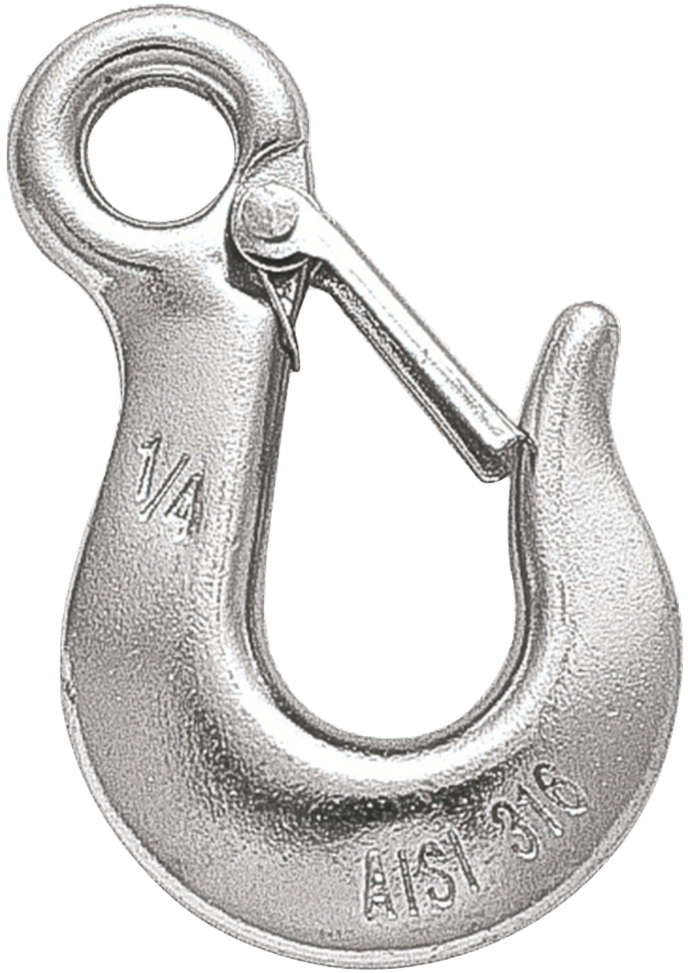
Where does it come from ?



A bit of history

- Dec 2011: project launch
- Feb 2012: version 0.1
- May 2013: version 0.4 (out of incubator)
- June 2014: version 1.0
- August 2015: version 1.5

CDI & DeltaSpike



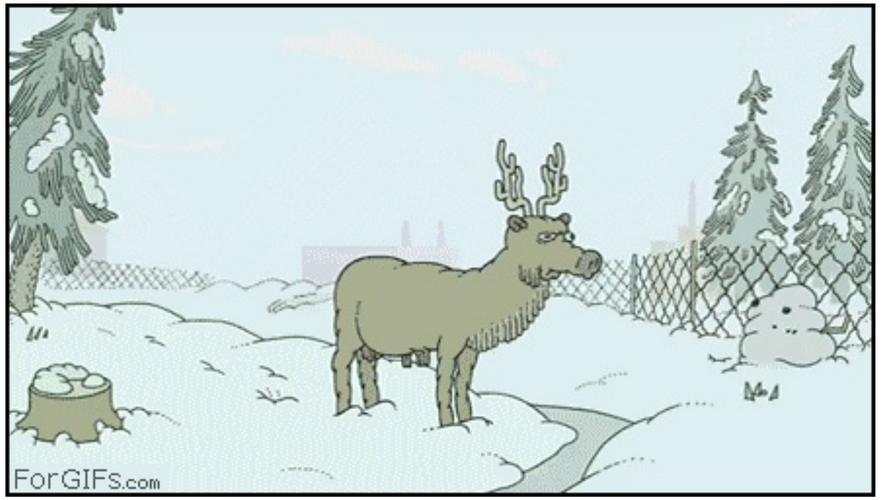
- CDI is a specification. It doesn’t provide business features
- but it includes a powerful hook to add these business features
- The "Portable extensions" feature is this hook
- Thanks to it, CDI can be easily enhanced with new high level features

CDI Portable extensions

- One of the most powerful feature of the CDI specification

- Not really popularized, partly due to:

1. Their high level of abstraction
2. The good knowledge on Basic CDI and SPI
3. Lack of information (CDI is often reduced to a basic DI solution)

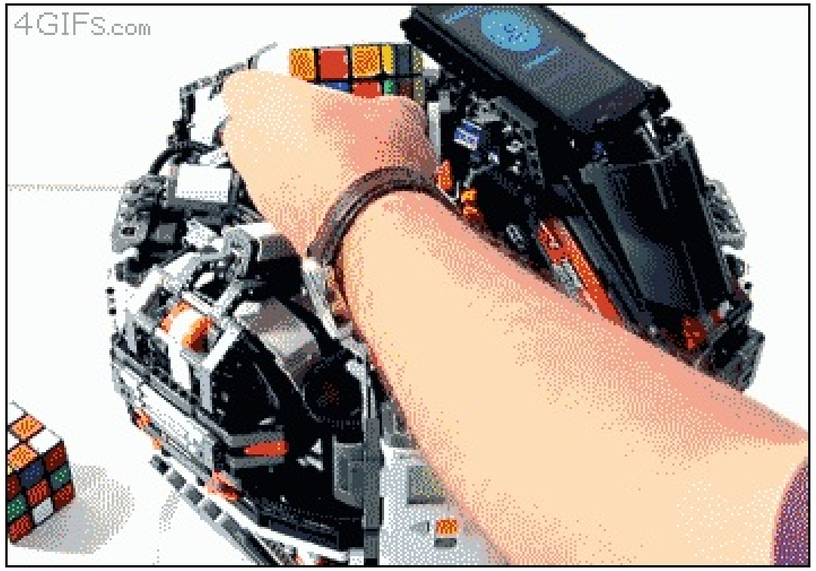


Extensions, what for?

- To integrate 3rd party libraries, frameworks or legacy components
- To change existing configuration or behavior
- To extend CDI and Java EE
- Thanks to them, Java EE can evolve between major releases

Extensions, how?

- Observing SPI events at boot time related to the bean manager lifecycle
- Checking what meta-data are being created
- Modifying these meta-data or creating new ones



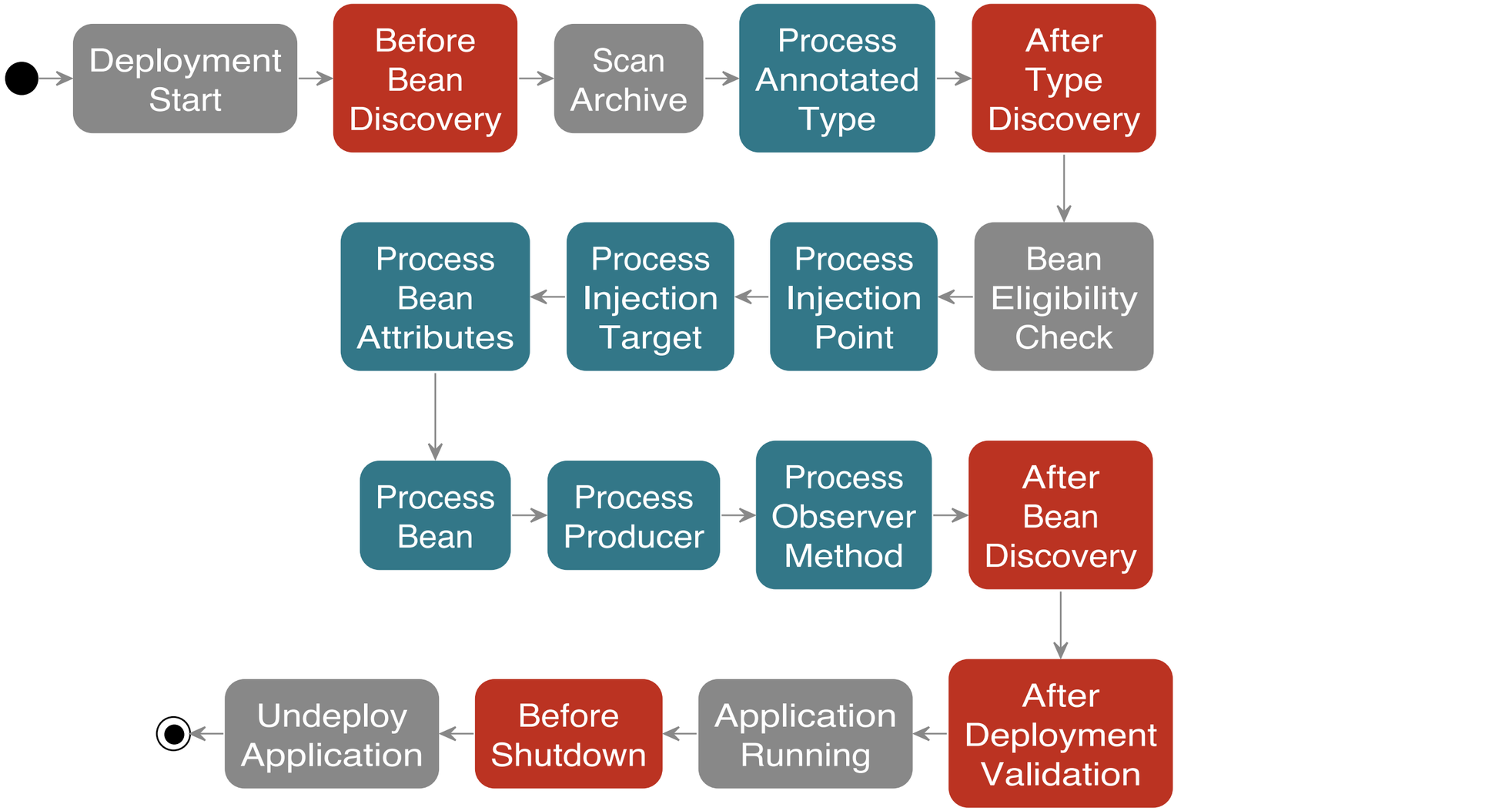
More concretely

Service provider of the service `javax.enterprise.inject.spi.Extension` declared in `META-INF/services`

Just put the fully qualified name of your extension class in this file

```
import javax.enterprise.event.Observes;
import javax.enterprise.inject.spi.Extension;

public class CdiExtension implements Extension {

    void beforeBeanDiscovery(@Observes BeforeBeanDiscovery bbd) {
    }
    //...

    void afterDeploymentValidation(@Observes AfterDeploymentValidation adv) {
    }
}
```

Bean manager lifecycle



Example: Ignoring JPA entities

The following extension prevents CDI to manage entities

This is a commonly admitted good practice

```
public class VetoEntity implements Extension {

    void vetoEntity(@Observes @WithAnnotations(Entity.class)
                    ProcessAnnotatedType<?> pat) {
        pat.veto();
    }
}
```

⚠ Extensions are launched during bootstrap and are based on CDI events

⚠ Once the application is bootstrapped, the Bean Manager is in read-only mode (no runtime bean registration)

⚠ You only have to `@Observes` built-in CDI events to create your extensions

Apache DeltaSpike is…

- A collection of ready to use extensions to help you in your projects
- A toolbox to help you develop new CDI portable extensions
- A great way to learn how to develop your own extension by browsing the source code
- The most obvious entry point to CDI eco-system

DeltaSpike is tested with

- CDI 1.0, 1.1, 1.2 and 2.0
- JBoss Weld and Apache OpenWebBeans
- JBoss AS 7.x, WildFly 8.x - 10.x
- JBoss EAP 6.x - 7.x
- Apache TomEE 1.0.x - 1.7.x
- Oracle GlassFish 3.x, 4.x
- Oracle Weblogic 12c
- IBM Websphere 8.x

Deltaspike is for all CDI developers

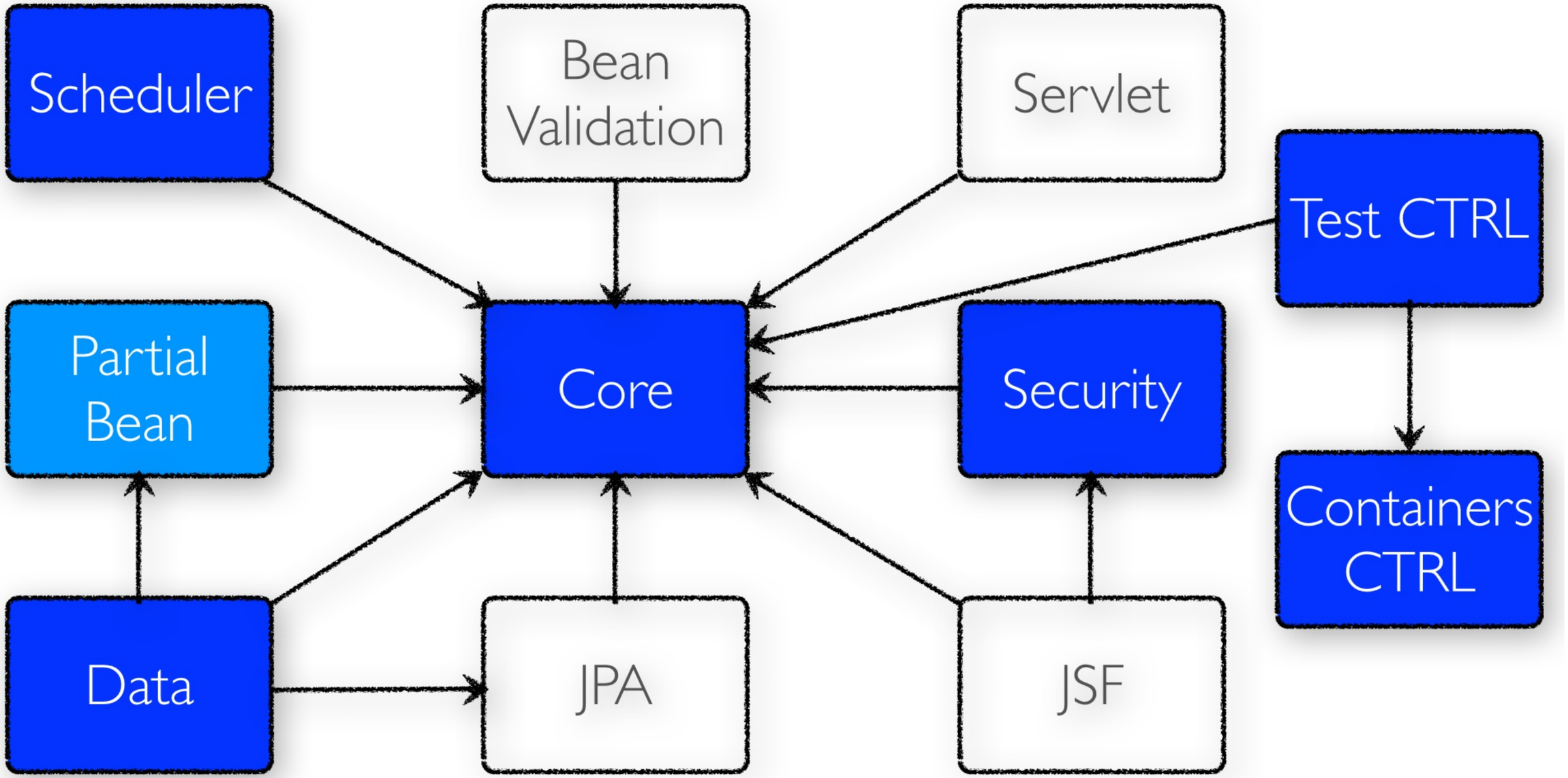
- While this talk is focused on classical project developers…
- …advanced developers may find interesting helpers in Deltaspike
- For instance metadata builder are useful for portable extension developers

```
public void registerGenericBeans(@Observes AfterBeanDiscovery abd) {
    BeanBuilder<User> ubb = new BeanBuilder<User>(beanManager).readFromType(User.class)
            .passivationCapable(true)
            .addTypes(otherTypes);
    if (weAreOnWeb)
        ubb.scope(SessionScoped.class);
    else
        ubb.scope(ApplicationScoped.class);
    abd.addBean(ubb.create());
}
```

More on DeltaSpike

- Apache DeltaSpike received **Duke's choice award 2014**
- This talk shows only a small part of the framework
- More info on: deltaspike.apache.org/

Modules and dependencies



Core Module

Core - Exception Handler

```
public class InventoryActions {
    @PersistenceContext private EntityManager em;
    @Inject private Event<ExceptionToCatchEvent> catchEvent; (1)

    public Integer queryForItem(Item item) {
        try {
          Query q = em.createQuery("SELECT i from Item i where i.id = :id");
          q.setParameter("id", item.getId());
          return q.getSingleResult();
        } catch (PersistenceException e) {
          catchEvent.fire(new ExceptionToCatchEvent(e)); (2)
        }
    }
}
```

1. The Event of generic type ExceptionToCatchEvent is injected into your class for use later within a try/catch block.
2. The event is fired with a new instance of ExceptionToCatchEvent constructed with the exception to be handled.

Core - Exception Handler

Exceptions are handled asynchronously.

```
@ExceptionHandler (1)
public class MyHandlers {
    void printExceptions(@Handles ExceptionEvent<Throwable> evt) { (2)
        System.out.println("Something bad happened:" +
        evt.getException().getMessage());
        evt.handleAndContinue(); (3)
    }
}
```

1. Exception handler methods are registered on beans annotated with @ExceptionHandler
2. The @Handles annotation on the first parameter designates this method as an exception handler.
3. This handler does not modify the invocation of subsequent handlers, as designated by invoking handleAndContinue().

Core - Type-safe ProjectStage

- The current ProjectStage can be injected.

```
@Inject
private ProjectStage projectStage;

//...

boolean isDevelopment = ProjectStage.Development.equals(this.projectStage);
```

- You can also use the ProjectStage at XHTML files.

```
<h:panelGroup layout="block"rendered="#{applicationConfig.projectStage == 'Development'}" >
    <!-- HTML Snippet is shown only in Development stage -->
</h:panelGroup>
```

Core - Type-safe ProjectStage

DeltaSpike comes with the following pre-defined ProjectStages:

1. UnitTest
2. Development
3. SystemTest
4. IntegrationTest
5. Staging
6. Production

But you can create your own custom ProjectStages.

Core - Project Stage Configuration

It can be set using DeltaSpike Configuration Mechanism

```
-D org.apache.deltaspike.ProjectStage=Development
```

How to provide these Key/Values to DeltaSpike?

1. System properties
2. Environment properties
3. JNDI values - the base name is "java:comp/env/deltaspike/"
4. Properties file values - default filename is "META-INF/apache-deltaspike.properties"

Core - DeltaSpike Configuration Mechanism

Configuration API

```
String userName = ConfigResolver.getPropertyValue("user.name");  (1)
String dbUserName = ConfigResolver.getProjectStageAwarePropertyValue("db.username"); (2)
Integer dbPort = ConfigResolver
    .resolve("db.port")  (3)
    .as(Integer.class)
    .withProjectStage(true)
    .withDefault(3306)
    .getValue();
Date deadline = ConfigResolver.resolve("project.deadline")  (4)
  .as(Date.class, new CustomDateConverter()).getValue());
```

Properties (Key / Value)

```
user.name = "Rafael"  (1)
db.username.Production = "Antoine"  (2)
db.username.Development = "Benevides"  (2)
db.port = 1234  (3)
project.deadline = 2017-04-01  (4)
```

Core - DeltaSpike Configuration Mechanism

Injection of configured values into beans using @ConfigProperty

```
@ApplicationScoped
public class SomeRandomService
{
    @Inject
    @ConfigProperty(name = "endpoint.poll.interval")
    private Integer pollInterval;

    @Inject
    @ConfigProperty(name = "endpoint.poll.servername")
    private String pollUrl;

    ...
 }
```

Type-safe messages - Bean creation

```
@Named("msg")
@MessageBundle
public interface MyMessages {

    public String welcome();

    public String welcomeTo(String username);  (1)

    @MessageTemplate("{custom_message}")  (2)
    public String message();
}
```

(1) in the message bundle: welcometo=Welcome to %s

(2) in the message bundle: custom_message=DeltaSpike is awesome!

Core - Messages and i18n

Now the messages bean is ready to be used in Java Classes

```
@Inject
private MyMessages messages;
//
new FacesMessage(messages.welcomeTo("Rafael"));
log.info(messages.message());
```

…or even inside JSP/JSF because it uses a @Named annotation.

```
<h1>#{msg.welcome}</h1>
```

It uses the “partial bean” module to dynamically create implementation at runtime.

Core - @Exclude

It's like @Vetoed from CDI 1.1 but better!

Excluding a Bean in any Case

```
@Exclude
public class NoBean{  }
```

Excluding a Bean in Case of ProjectStageDevelopment

```
@Exclude(ifProjectStage = ProjectStage.Development.class)
public class MyBean{  }
```

Excluding a Bean if the ProjectStage is different from Development

```
@Exclude(exceptIfProjectStage = ProjectStage.Development.class)
public class MyDevBean{ }
```

Excluding a Bean based on an Expression which Evaluates to True

```
@Exclude(onExpression = "db==prodDB")
public class DevDbBean {  }
```

Core - Injecting Resources

DeltaSpike has simple APIs for performing basic resource loading and property file reading.

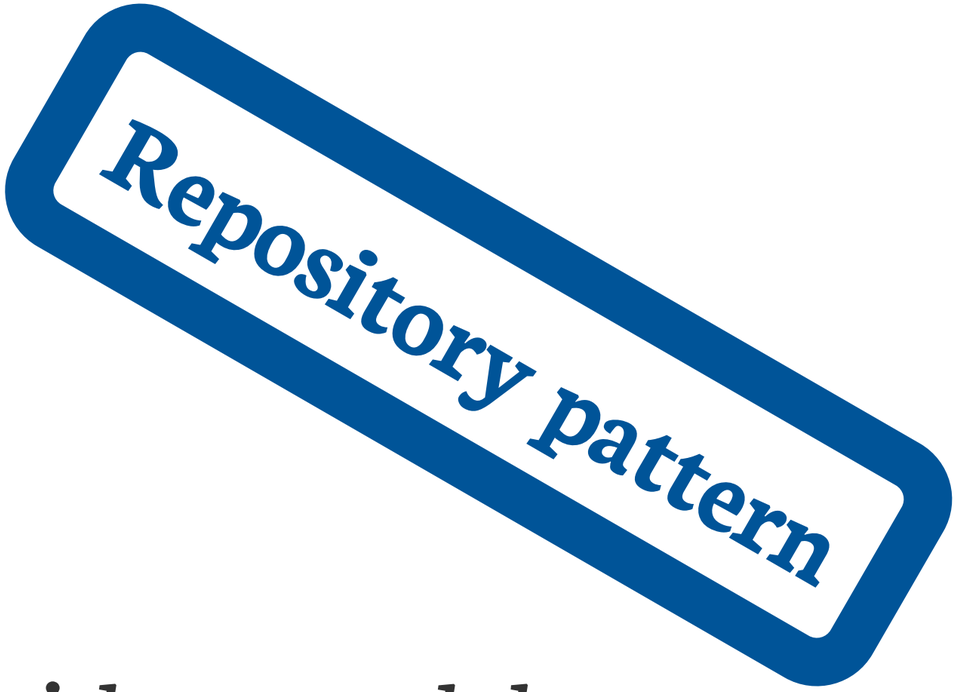
```
@Inject
@InjectableResource("myfile.properties")
private InputStream is;

public String getVersion() throws IOException {
    try (BufferedReader br = new BufferedReader(new InputStreamReader(is))) {
      return br.readLine();
    }
}
```

The InjectableResourceProvider interface can be implemented to allow reading from alternate sources if needed.

Data Module

Data Module

- Data module is an implementation of the repository pattern.
- At the moment it only support RDBMS thru JPA.
- But it could be extended to support other data services.



> *"A Repository represents all objects of a certain type as a conceptual set.*
>
> *It acts like a collection, except with more elaborate querying capability."*
>
> Domain Driven Design
>
> — Eric Evans

Data Module - Creating a Repository

```
@Repository
public interface UserRepository extends EntityRepository<User, Long> {
/* DeltaSpike creates a proxy which implements:

Long count();
List<E> findAll();
E findBy(PK);
void flush();
void refresh();
void remove(E);
E save(E);
E saveAndFlush(E);

...and many others */
}
```

It uses the “partial bean” module to dynamically create implementation at runtime.

Data Module - Making queries

```
@Repository
public interface UserRepository extends EntityRepository<User, Long> {

  public User findByUsernameAndPassword(String username, char[] password); (1)

  @Query("SELECT u FROM User AS u WHERE u.role in (?1)") (2)
  public List<Role> findByRoles(List<Role> roles);

}
```

(1) The name of the method automatically creates the query. Example:
"SELECT u FROM User u WHERE u.username = ?1 AND u.password = ?2 "

(2) The query is defined inside the @Query annotation.

Data Module - Pagination

```
@Repository
public interface UserRepository extends EntityRepository<User, Long> {

  @Query("select p from Person p where p.age between ?1 and ?2")
  QueryResult<Person> findAllByAge(int minAge, int maxAge);

}

QueryResult<Person> paged = personRepository.findByAge(age)

// Query API style
paged.maxResults(10).firstResult(50).getResultList();

// or paging style
paged.withPageSize(10).toPage(5).getResultList();

int totalPages = paged.countPages();
```

Security Module

Security Module - Simple interceptor-style authorization

Type-safe authorization

```
@Retention(value = RetentionPolicy.RUNTIME)
@Target({ ElementType.TYPE, ElementType.METHOD })
@SecurityBindingType
public @interface AdminOnly {
}

@ApplicationScoped
public class SecuredBean {

    @AdminOnly
    public void doSomething() {
        //...
    }
}
```

Security Module - Simple interceptor-style authorization

An interceptor-style class is used to define the access

```
@ApplicationScoped
public class ApplicationAuthorizer {

    @Secures
    @AdminOnly
    public boolean verifyPermission(InvocationContext invocationContext,
BeanManager manager, @Loggged User user) throws Exception {
      return user.getRole().equalsIgnoreCase("Admin");
    }
}
```

JSF Module

JSF Module - JSF Messages

```
@MessageBundle
public interface Messages {

    @MessageTemplate("Welcome to DeltaSpike")
    String welcomeToDeltaSpike();

}

@Model
public class MyJSFBean {

    @Inject
    private JsfMessage<Messages> messages;

    //...
    messages.addInfo().welcomeToDeltaSpike();
}
```

JSF Module - @WindowScoped

"The window-scope is like a session per window"

```
@WindowScoped
public class PreferencesBean implements Serializable {
     //...
}
```

"There isn’t a lot of use-cases which need shared data between windows"

JSF Module - Double-Submit Prevention

"To avoid that the same content of a form gets submitted and therefore processed multiple times"

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ds="http://deltaspike.apache.org/jsf">
    <h:head>
        <!-- head content -->
    </h:head>
    <h:body>
        <h:form>
            <!-- form content -->
            <ds:preventDoubleSubmit/>
        </h:form>
    </h:body>
</html>
```

Scheduler Module

Scheduler Module

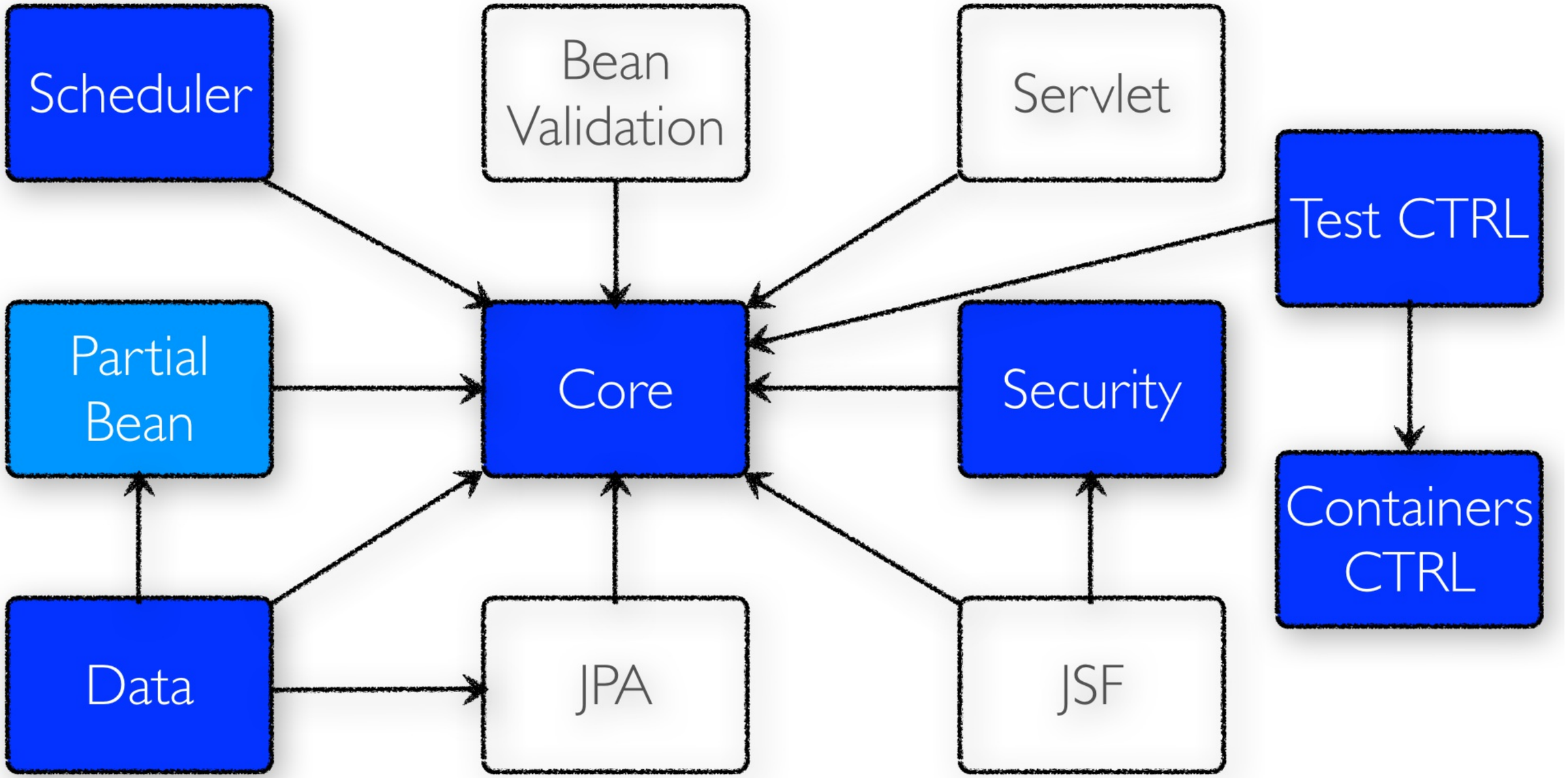
Provides integration with Quartz.

```
// Job will execute each minute
@Scheduled(cronExpression = "0 0/1 * * * ?", onStartup = false)
public class CdiAwareQuartzJob implements org.quartz.Job {

    // And it can receive CDI injections
    @Inject
    private AdminServices service;

    @Override
    public void execute(JobExecutionContext context) throws JobExecutionException {
        service.executeAdministrativeTask();
    }
}
```

```
@Inject
private Scheduler<Job> jobScheduler;

//...
jobScheduler.registerNewJob(CdiAwareQuartzJob.class);
```

Other Modules

Other Modules



Want to go farther on CDI?

 **TUT2376:** Advanced CDI in live coding

 Tuesday 27th at 8:30

 Cyril Magnin II / III

 Antonin Stefanutti & Antoine SD

Questions ???

deltaspike.apache.org/