Dependency Injection in Action

Jan Västernäs

CALLISTA

Agenda (DI=Dependency Injection)

- Background
- What is DI ?
- What is the DI value promise ?
- Experiences
  - Using Spring DI container
- Does DI live up to the promise ?
- Future & Others

Copyright 2006, Callista Enterprise AB

Background

- Using spring framework for one year
- Limited by policy to some parts only
- Large industrial project
- My role : Application architect
- Architecture & method team
- Mix of very experienced developers and new-to-java developers

Copyright 2006, Callista Enterprise AB

CALLISTA

Spring usage

- Dependency Injection
  - 480 spring beans
  - 680 SQL statements as beans/properties
- JDBC support
  - JDBC Template
  - Exception Hierarchy
- JMS Support
  - JMS Template
- Using ”factory beans” to intercept calls for
  - Out-of-container transaction/connection support
  - Declarative transaction demarcation
- Test classes

Copyright 2006, Callista Enterprise AB

CALLISTA

What is Dependency Injection ?

- Dont call us - we’ll call you
- Referenced object are declared
- References are injected into an object
  - By calling setter method

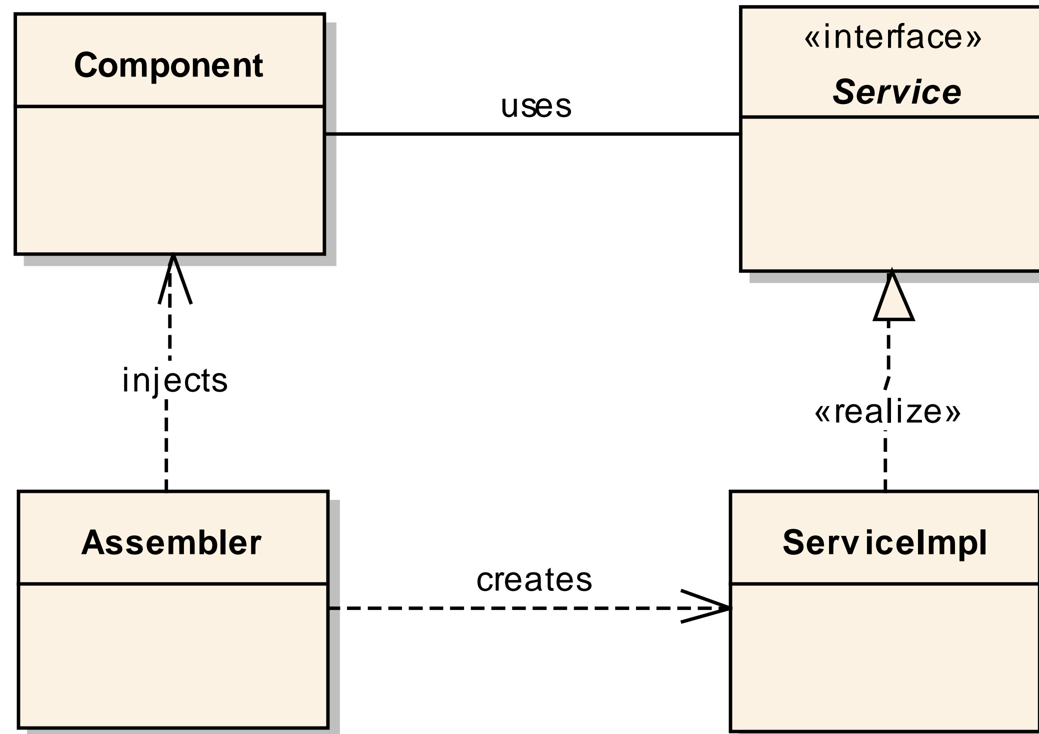
- Need some kind of runtime support (DI container)
- Enables call interceptors to be used

- Opposite of ”Dependency Lookup”

Copyright 2006, Callista Enterprise AB

CALLISTA

Dependency Injection




```
public void setDao(CustomerDao dao) {
   this.dao = dao;
 }
```


Copyright 2006, Callista Enterprise AB

CALLISTA

What is the DI value promise ?

- No energy waisted nor code needed to find other objects
  - No JNDI, singletons, factories etc
- Unit Test easier by mocking dependencies
  - May even make it possible without workarounds !
- Non-intrusive
  - In our project : Business code independant/unaware of technical infrastructure (including DI container).
- ”Lighter” than EJB

Copyright 2006, Callista Enterprise AB

CALLISTA

Experiences

- Typical scenario
- Setup
- Configuration
- Bootstrapping
- Namespaces
- Programming model
- Integration
- Test
- Interceptors

Copyright 2006, Callista Enterprise AB

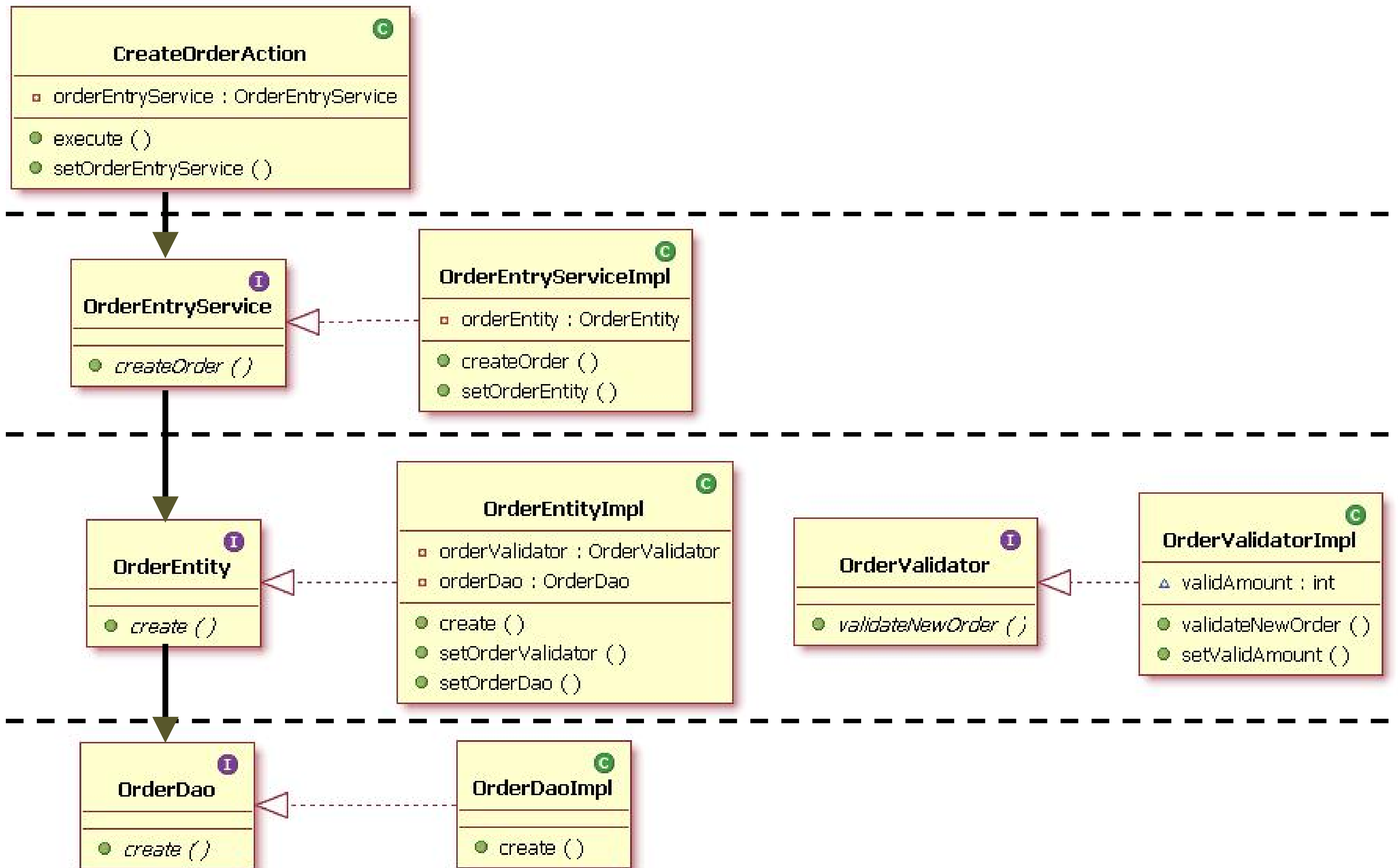
CALLISTA

Typical scenario

- Layered architecture
- Thin clients – request/response
- J2EE
- Web-apps
- EJB module only for MessageDriven EJB:s
- JDBC - generated and handwritten

Copyright 2006, Callista Enterprise AB

CALLISTA



Copyright 2006, Callista Enterprise AB

CALLISTA

Spring setup

- 1 hour to get started with basic DI container functionality

Copyright 2006, Callista Enterprise AB

CALLISTA

Spring DI Configuration

- Overhead writing Xml config is acceptable
  - Use classnames as bean id and reference name
  - Copy-paste - nobrainer

```
<bean id="orderEntryService"
   class="se.cadec.impl.OrderEntryServiceImpl">
   <property name="orderEntity" ref="orderEntity"/>
</bean>
<bean id="orderEntity" class="se.cadec.impl.OrderEntityImpl">
   <property name="orderDao" ref="orderDao"/>
   <property name="orderValidator" ref="orderValidator"/>
</bean>

   private OrderEntity orderEntity;
   public void setOrderEntity(OrderEntity orderEntity)   {
         this.orderEntity = orderEntity;
```


Copyright 2006, Callista Enterprise AB

CALLISTA

Spring DI Configuration

- Debugging configuration errors is OK
- Plugin helps

org.springframework.beans.NotWritablePropertyException: Invalid property 'dao' of bean class [se.cadec.impl.OrderEntityImpl]: Bean property 'dao' is not writable or has an invalid setter method: Does the parameter type of the setter match the return type of the getter?

Copyright 2006, Callista Enterprise AB

CALLISTA

Bootstrapping – modularization

- Ear
  - Web module
    - Bootstrapping servlet
    - spring-config.xml
  - Xxx.jar
    - spring-config.xml
  - Yyy.jar
    - spring-config.xml
      - spring-config1.xml
      - spring-config2.xml
  - Zzz.jar
    - spring-config.xml

Copyright 2006, Callista Enterprise AB

CALLISTA

Solution – Custom Bootstrapping

- A help class that finds all spring configuration files on the classpath following a naming convention
- The bootstrapping servlet calls this help class
- Each file has references to the actual files used in that jar/war with the import function

Copyright 2006, Callista Enterprise AB

CALLISTA

No namespace support

- All configuration in one file – no problem
- Each module has its own configuration file – may cause collisions
- Solution
  - componentname:beanname

  <bean id="***ordercomponent:***orderEntryService"

Copyright 2006, Callista Enterprise AB

CALLISTA

HiveMind solution is better

Unlike Spring, HiveMind has built-in namespace support, and fundamentally assumes that the graph of related services will be built from multiple locations (that is, each JAR on the classpath will be packaged with its own XML descriptor).

Howard Lewis Ship, the creator of HiveMind and Tapestry explains why tapestry does not use Spring.

Copyright 2006, Callista Enterprise AB

Programming model

- Which objects should be injected ?
- What are the alternatives ?
  - new
  - static
  - Factory
  - JNDI
- “Stateless Session” Bean obvious, Works very well with stateless singleton-style service-oriented beans.
- Data Transfer Objects:s - never
- In the middle – try to singletonize
- One consistent way makes it easier

Copyright 2006, Callista Enterprise AB

CALLISTA

Programming model 2

- Static methods
  - Can not be injected with bean
  - Can not be mocked
- new
  - Can not be injected with beans
  - Can not be mocked

Copyright 2006, Callista Enterprise AB

CALLISTA

Programming model 3

```
private void doIt() {

   MyStatefulObject myobj = getANewMyStatefulObject();

   myobj.init(thisandthat, somethingelse);
   myobj.doSomething();
   myobj.saveResult();
}


private MyStatefulObject getANewMyStatefulObject() {
   return null;
}
```


Copyright 2006, Callista Enterprise AB

CALLISTA

Get new instance on each call

- Method Injection (Lookup method).
- Uses bytecode modification
- We did not use this
- Solution was to design service-style instead and use singleton beans

- Non-singleton beans has been used rarely

Copyright 2006, Callista Enterprise AB

CALLISTA

Integration

- The DI container must create all beans
- Other objects can not have injected references

Copyright 2006, Callista Enterprise AB

CALLISTA

Struts Integration

- DelegatingTilesRequestProcessor
- One tag in struts config
- A spring configuration file listing all actions
  - Action-mapping to bean
- Works well

Copyright 2006, Callista Enterprise AB

CALLISTA

Message Driven Beans

- EJB:s that is activated by incoming JMS messages
- Generic MDB class that
  - Gets the bean id from JNDI
  - Gets the bean instance by calling the spring applicationContext
  - Return to Dependency Lookup
- MDB bean can be injected with references

Copyright 2006, Callista Enterprise AB

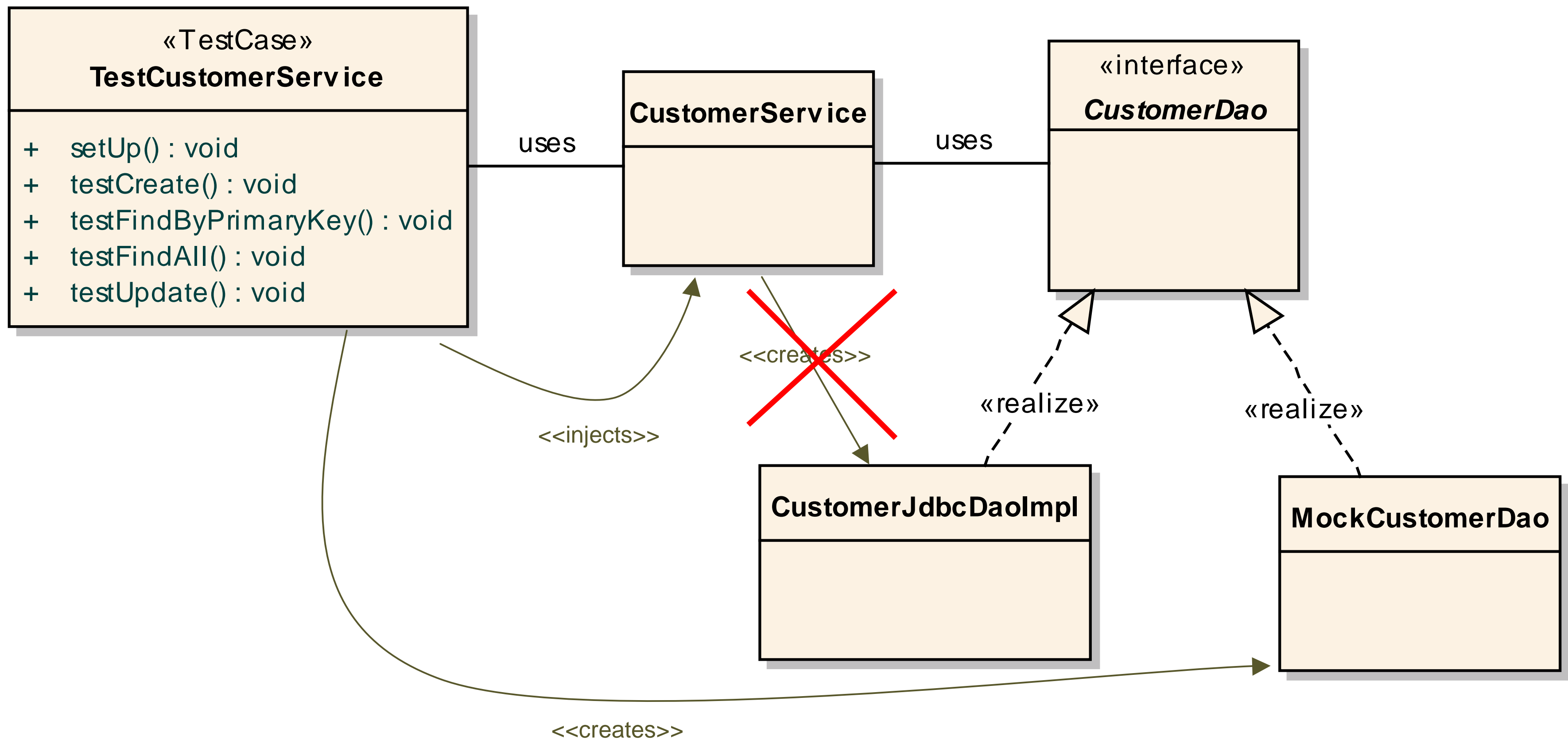
CALLISTA

Succesfull usage

- Action, service, entity, dao, helpers, utilities
  - Everything is injected
- Properties like
  - SQL
  - Config parameters
- Current time
- Map
- Inheritance
- JNDI to find datasource
- Reuse action class with parameters

CALLISTA

Testability – Unit test



Copyright 2006, Callista Enterprise AB

CALLISTA

Unit Test

- new to create implemention class
- ”inject” mock objects for dependencies
  - By calling setters
- Dont neew the full environment
- Creating exceptional conditions easy
- Unit Testing and mocking is not trivial
  - Which test to write ?
  - Create a lot of dto:s
  - Some classes are ”perfect” for testing – some not
- DI makes it less painful

Copyright 2006, Callista Enterprise AB

CALLISTA

Integration Test

- Including
  - DI container
  - Config xml files
  - All dependencies as real classes
  - Database etc
- Get Class to test from DI container
  - beanFactory.getBean(”beanName”)
  - Introduces a binding to Spring
  - Return to Dependency Lookup

Copyright 2006, Callista Enterprise AB

CALLISTA

Interceptor usage

- DI enables interceptors out of the box
  - Requires own ”bean”-factory in non-DI setup
- Typically NOT application development
  - More complex requirements
  - Insert a proxy betwen the calling and called object
- Sample
  - Out-of-container transaction/connection support
  - Declarative transaction demarcation
- Implemented by using spring Factory Bean concept & explicit proxy class definition

Copyright 2006, Callista Enterprise AB

CALLISTA

What about the promises ?

- No energy waisted to find objects
  - xml format is simple
- Unit Test easier by mocking dependencies
  - YES !
- Non-intrusive
  - DI invisible in beans
  - struts/junit etc integration not easy to replace
- ”Lighter” than EJB
  - Absolutely

CALLISTA

Future & Others

CALLISTA

Spring vs EJB3

- EJB3 has dependency injection
- EJB:s are POJOs
- Non-EJB classes can NOT be injected
  - Except resurces and other JNDI items
  - Many of our beans must be made EJB:s
- Injection into struts action classes ?
- In our case – 250 local EJB:s
- How does a appserver handle 250 EJB:s ?
  - Websphere ? JBoss/Geronimo/Glassfish etc ?
  - A appserver must be able to deliver services for an EJB, can it ever be as light-weight ?

Copyright 2006, Callista Enterprise AB

CALLISTA

Migration between Spring – EJB3

- Least common denominator
  - No annotations
  - Setter injection
  - Everything in xml
  - Production code platform independent

Copyright 2006, Callista Enterprise AB

CALLISTA

EJB2 -> Spring/EJB3

```java
public class MyBean {

    public void doIt(String arg1, List arg2) {

        MyService service = (MyService) MyFactory.getComponent("myservice");
        service.doIt(arg2);
```

```java
public class MyBean {
    private SessionContext ctx;
    private MyService service;

    public void setSessionContext(SessionContext sc) {
        ctx = sc;
        MyService service = (MyService) MyFactory.getComponent("myservice");
        setService(service);
    }
    public void doIt(String arg1, List arg2) {
        service.doIt(arg2);
    }
    public void setService(MyService service) {
        this.service = service;
    }
}
```


Copyright 2006, Callista Enterprise AB

CALLISTA

Spring 2.0

- Message-driven POJO:s
- Enhanced xml configuration
- AOP, AspectJ integration
- Others ....

- Are we moving away from ”simple” ??

Copyright 2006, Callista Enterprise AB

CALLISTA

DI future

- Dependency Injection as a concept is here to stay
- John Rodson "J2EE without DI"
  - A title we will probably not see
- Why bother about simplicity ? We are all professionals right ?

Copyright 2006, Callista Enterprise AB

CALLISTA

Is Java too complicated ?

Copyright 2006, Callista Enterprise AB

CALLISTA

Ruby sample syntax

```
# Check for pre-formatted string given

check=Regexp.new('^'+Regexp.escape
(format).gsub(/\\#/,'.')+'$')

return string if string =~ check
```


Copyright 2006, Callista Enterprise AB

CALLISTA

Time for Questions!

Copyright 2006, Callista Enterprise AB

CALLISTA