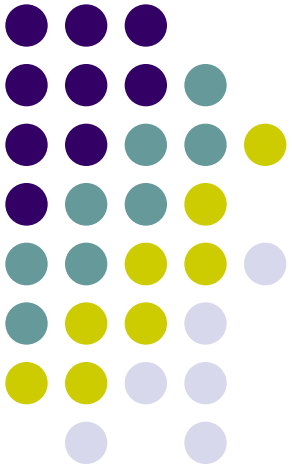
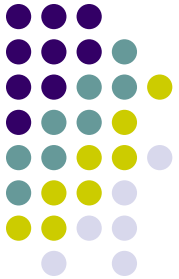
Spring

Otros



Contenidos



- Task Execution and Scheduling
- Cache
- WebService
- Testing
- Mail

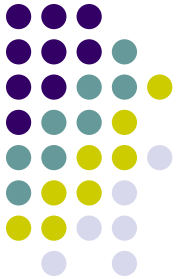
@Scheduled

```
@Scheduled(fixedRate=5000)
public void doSomething() {
……..
}

@Scheduled(cron="*/5 * * * * MON-FRI")
public void doSomething() {
……
}
```

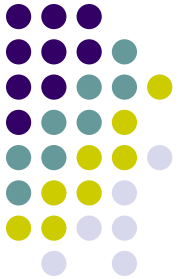
```
<task:annotation-driven executor="myExecutor" scheduler="myScheduler"/>
<task:scheduler id="myScheduler" pool-size="10"/>
```

@Async

```
@Async
public void doSomething() {
.....
}

@Async
void doSomething(String s) {
.....
}

@Async
Future<String> returnSomething(int i) {
.....
}
```



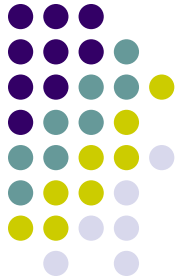
@Async



```
<task:annotation-driven executor="myExecutor" scheduler="myScheduler"/>
<task:executor id="myExecutor" pool-size="5"/>
```

@Cacheable

```
@Cacheable(cacheName = “pedido")
public Pedido buscarPedido(long id) {
…….
}
```

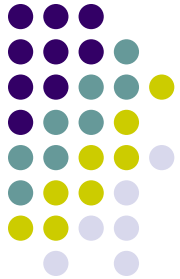


@WebService

```
@WebService
public interface IxxxWS {
  public void hacerX();
}

@WebService(endpointInterface = "IxxxWS")
public class XXXWSImpl implements IxxxWS {
  public void hacerX() {
    ........
  }
}
```

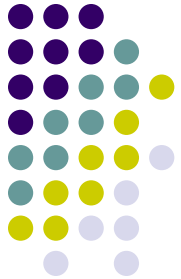
```
<bean id=" xxxWSImpl " scope="prototype" class=“xxx.xxx.XXXWSImpl " />
<jaxws:endpoint id=“urlWS" implementor="# xxxWSImpl" address="/urlWS" />
```

Testing



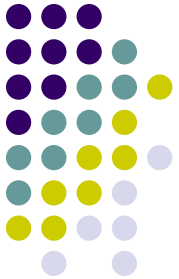
- Mock objects para SpringMVC, Servlets, JNDI, entre otros
- Gestión del AppContext
- Gestión de transacciones
- DI dentro de los casos de prueba

Testing



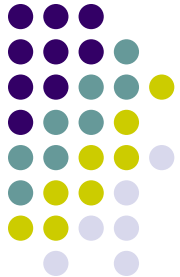
- @ContextConfiguration
- @TransactionConfiguration
- @Rollback
- @BeforeTransaction
- @AfterTransaction
- @NotTransactional
- @IfProfileValue
- @ProfileValueSourceConfiguration
- @ExpectedException
- @Timed
- @Repeat

Testing



```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={"classpath:/applicationContext.xml"})
@TransactionConfiguration(transactionManager="transManX",defaultRollback=false)
public class TestXXXX extends AbstractTransactionalJUnit4SpringContextTests  {

        @Autowired
        private XXXService xxxService;


        @Test
        public void test……..() {
```

Testing

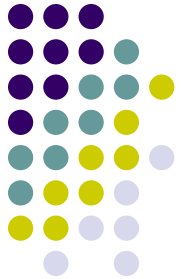


```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
@TransactionConfiguration(transactionManager="txMgr", defaultRollback=false)
@Transactional
public class FictitiousTransactionalTest {
    @BeforeTransaction
    public void verifyInitialDatabaseState() {
    }
    @Before
    public void setUpTestDataWithinTransaction() {
    }

    @Test
    @Rollback(true)
    public void modifyDatabaseWithinTransaction() {
    }

    @After
    public void tearDownWithinTransaction() {
     }

    @AfterTransaction
    public void verifyFinalDatabaseState() {
        // logic to verify the final state after transaction has rolled back
    }

}
```

Mail

```
<bean id="mailSender"
class="org.springframework.mail.javamail.JavaMailSenderImpl">
        <property name="host" value="smtp.gmail.com" />
        <property name="port" value="587" />
        <property name="username" value="xxxxxxxxxxxx" />
        <property name="password" value="yyyyyyyyyyyy" />
        <property name="javaMailProperties">
          <props><prop key="mail.smtp.auth">true</prop>
                  <prop key="mail.smtp.starttls.enable">true</prop>
        </props>
        </property>
</bean>
```

Mail



```
@Autowired
private MailSender mailSender;

public void sendMail(String from, String to, String subject, String msg) {
        SimpleMailMessage message = new SimpleMailMessage();
        message.setFrom(from);
        message.setTo(to);
        message.setSubject(subject);
        message.setText(msg);
        mailSender.send(message);
}
```

Ejercicio

1. Calcular el saldo total de dinero que dispone el banco cada 15 minutos y guardarlo en una tabla de históricos de saldos (Enviar un mail de notificación - utilizar Async)
2. Permitir consultar el estado de un cuenta bancaria mediante un WebService
3. Implementar una política de cache para la consulta de la información de la cuenta bancaria (Usar un objeto de servicio)