# circuit\_breaker\_20190516

OK

## Technologies

* Spring cloud / 客户端弹性 / circuit breaker mode + fallback mode
* Servlet filter
* Spring web interceptor

# jdbc\_20190208

## ok

## Technologies

* jdbc(DataSource) + JNDI + Servlet
* jdbc(establish connection by DriverManager)
* mysql
* Jboss EAP 7.2

## Details

# jpa\_applicationmanaged\_entitymanager\_20190217

ok / mysql + jpa(application-managed EntityManager)(Hibernate)

# mvcjpa\_20140415 / 20180415

ok / Spring mvc + spring framework data access - JPA(hibernate) + thymeleaf LocalContainerEntityManagerFactoryBean org.apache.commons.dbcp2.BasicDataSource /src/main/resources/META-INF/persistence.xml

# mybatis\_20190204

ok  
mybatis + Java SE

# mybatis\_spring\_20190317

ok

Spring webmvc REST

## Problem

Type interface jack.helloworld.mybatis.spring.mapper.ProductMapper is not known to the MapperRegistry.

sqlSessionFactoryBean.**setMapperLocations**(resolver.getResources(**"classpath:mybatis/mappers/\*Mapper.xml"**))

@Bean

**public** SqlSessionFactory sqlSessionFactory() **throws** Exception {  
 SqlSessionFactoryBean sqlSessionFactoryBean = **new** SqlSessionFactoryBean();  
 sqlSessionFactoryBean.setDataSource(mmalDs());  
 PathMatchingResourcePatternResolver resolver = **new** PathMatchingResourcePatternResolver();  
 sqlSessionFactoryBean.setMapperLocations(resolver.getResources(**"classpath:mybatis/mappers/\*Mapper.xml"**));  
 **return** (SqlSessionFactory) sqlSessionFactoryBean.getObject();  
}

Caused by: java.lang.ClassNotFoundException: org.springframework.jdbc.datasource.TransactionAwareDataSourceProxy

<**dependency**>

<**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-jdbc</**artifactId**>  
 <**version**>5.1.5.RELEASE</**version**>  
</**dependency**>

Caused by: org.apache.ibatis.type.TypeException: Could not resolve type alias 'EProduct'. Cause: java.lang.ClassNotFoundException: Cannot find class: Eproduct

sqlSessionFactoryBean.setConfigLocation(**new** ClassPathResource(**"mybatis/mybatis-config.xml"**));

<**configuration**>  
 <**typeAliases**>  
 <**package name="jack.helloworld.mybatis.spring.pojos"**/>  
 </**typeAliases**>  
</**configuration**>

# mybatis\_springboot\_20190423

OK

mybatis - spring boot

log4j - spring boot

# servlet\_jta\_20190223

ok  
Servlet + JTA + Jboss EAP 7.2  
JTA\_20190222.vsdx /51.7 Transactions in Web Components

* EAP-7.0-Configuration\_Guide-en-US.pdf / CHAPTER 14. CONFIGURING TRANSACTIONS  
  XA datasources are already JTA capable by default
* JNDI 获取 UserTransaction InitialContext ctx = new InitialContext(); userTx = (UserTransaction) ctx.lookup("java:comp/UserTransaction");
* @Resource 注入 Resource injection enables you to inject any resource available in the JNDI namespace into any container-managed object, such as a servlet, an enterprise bean, or a managed bean.

@Resource private UserTransaction userTx;

* EAP-7.0-Developing\_EJB\_Applications-en-US.pdf 11.7. TRANSACTIONS USAGE IN PRACTICE

?? How to use @Resource

# springmvc\_20180414

ok  
Spring webmvc +

[http://localhost:8080/springmvc\_20180414-1.0-SNAPSHOT/hello/greeting](http://localhost:8080/springmvc_20180414-1.0-SNAPSHOT/hello/greeting" \o "http://localhost:8080/springmvc_20180414-1.0-SNAPSHOT/hello/greeting)

* Servlet root url  
  {domain}/{project.finalname}/c
* MyWebApplnitializer extends AbstractAnnotationConfigDispatcherServletInitializer

# spring\_orm\_jpa\_20190220

( ok servlet + jpa + mysql + Spring(Spring framework data access /ORM-JAP)

## LocalContainerEntityManagerFactoryBean

links to an existing JDBC DataSource instead of DataSource JNDI lookup

keeping the JDBC configuration in {@code persistence.xml}

)

# spring\_orm\_jpa\_20190224

ok

servlet + jpa + mysql + Spring(Spring framework data access /ORM-JPA /Obtaining an EntityManagerFactory from JNDI)

* Obtaining an EntityManagerFactory from JNDI
* Jboss EAP binds EntityManagerFactory with JNDI name
* IoC /Java-based container configuration /@ImportResource("classpath:spring\_beans.xml")
* how to use Spring IoC in Servlet, use @Autowired in Servelt class
* web applicaton launch procedures /Servlet EventListener /AnnotationConfigWebApplicationContext /web.xml
* Spring DAOs with @PersistenceUnit and @PersistenceContext
* JPA Criteria API
* Spring transaction

/JtaTransactionManager / use JTA in a Java EE container, then you use a container DataSource, obtained through JNDI

/declarative transaction /@EnableTransactionManagement /@Transactional(readOnly = true, timeout = 60)

# spring\_webmvc\_rest\_20190327

ok

Spring webmvc REST / version WebApplicationInitializer

* Content negotiation

@GetMapping(value = **"/hello"**, produces = **"text/plain;charset=utf-8"**)

* MIME Type

text/plain , text/xml , application/json