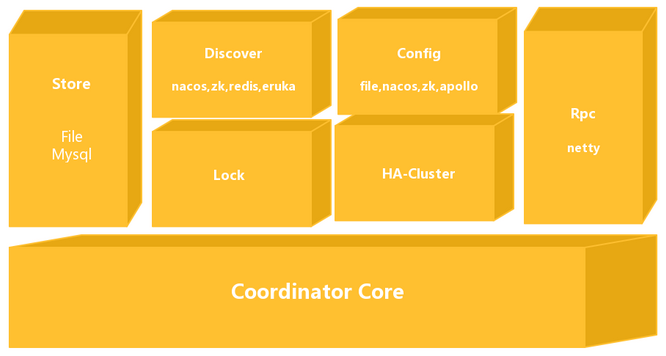
集成梳理

Seata-Server设计，是一个单独进程服务，与TM/RM的交互是RPC：



1.Coordinator Core：事务协调器核心代码，主要用来处理事务协调的逻辑，如是否Commit、Rollback等协调活动

2.Store：存储模块，用来将我们的数据持久化，防止重启或者宕机数据丢失

3.Discover：服务注册/发现模块，用于将Server地址暴露给Client

4.Config：用来存储和查找服务端的配置

5.Lock：锁模块，用于给 Seata 提供全局锁的功能

6.Rpc：用于和其他端通信

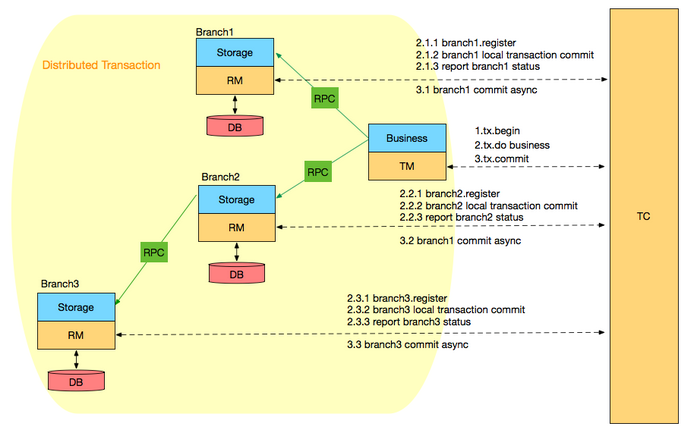
7.HA-Cluster：高可用集群，目前还没开源，为Seata提供可靠的高可用功能

核心组件：

TC：事务协调器，维护全局事务的运行状态，负责协调并驱动全局事务的提交或回滚（server端）

TM：控制全局事务的边界，负责开启一个全局事务，并最终发起全局提交或全局回滚的决议（client端）

RM：控制分支事务，负责分支注册、状态汇报，并接收事务协调器的指令，驱动分支（本地）事务的提交和回滚（client端）



开启全局事务：  
以@GlobalTransactional注解为入口，GlobalTransactionalInterceptor为切入点  
TM向TC发起一个请求（服务端使用netty）开启一个全局事务，生成全局事务的XID，通过服务调用链路传播

开启分支事务：  
1.由于seata对底层的DataSource，Connection等使用DataSourceProxy，ConnectionProxy代理

2.当进行数据库操作的时，ConnectionProxy会判断是否包含全局事务  
2.1.包含全局事务  
 2.1.1.RM向TC发起请求注册分支事务  
 2.1.2.插入回滚日志（undo\_log表，业务库必须新建这个表）  
 2.1.3.事务提交  
 2.1.4.向TC上报事务状态  
2.2.不包含全局事务  
 2.2.1.事务提交

全局事务提交：  
1.TM向TC发起全局事务提交请求  
2.TC收到之后，会向各个分支事务发起事务提交请求  
3.分支事务接收到请求，只需要删除全局事务的undo\_log记录即可

全局事物回滚：  
1.TM向TC发起全局事务回滚请求  
2.TC收到之后，会向各个分支事务发起事务回滚请求  
3.分支事务接收到请求，根据XID对应的undo\_log表记录进行回滚即可（记录执行前后的记录）

Seata-Server部署：

1.官网下载，地址<https://github.com/seata/seata/releases>，并解压

2.创建seata数据库，执行对应sql语句（https://github.com/seata/seata/blob/1.4.0/script/server/db/mysql.sql）

-- -------------------------------- The script used when storeMode is 'db' --------------------------------

-- the table to store GlobalSession data

CREATE TABLE IF NOT EXISTS `global\_table` (

`xid` VARCHAR(128) NOT NULL,

`transaction\_id` BIGINT,

`status` TINYINT NOT NULL,

`application\_id` VARCHAR(32),

`transaction\_service\_group` VARCHAR(32),

`transaction\_name` VARCHAR(128),

`timeout` INT,

`begin\_time` BIGINT,

`application\_data` VARCHAR(2000),

`gmt\_create` DATETIME,

`gmt\_modified` DATETIME,

PRIMARY KEY (`xid`),

KEY `idx\_gmt\_modified\_status` (`gmt\_modified`, `status`),

KEY `idx\_transaction\_id` (`transaction\_id`)

) ENGINE = InnoDB

DEFAULT CHARSET = utf8;

-- the table to store BranchSession data

CREATE TABLE IF NOT EXISTS `branch\_table` (

`branch\_id` BIGINT NOT NULL,

`xid` VARCHAR(128) NOT NULL,

`transaction\_id` BIGINT,

`resource\_group\_id` VARCHAR(32),

`resource\_id` VARCHAR(256),

`branch\_type` VARCHAR(8),

`status` TINYINT,

`client\_id` VARCHAR(64),

`application\_data` VARCHAR(2000),

`gmt\_create` DATETIME,

`gmt\_modified` DATETIME,

PRIMARY KEY (`branch\_id`),

KEY `idx\_xid` (`xid`)

) ENGINE = InnoDB

DEFAULT CHARSET = utf8;

-- the table to store lock data

CREATE TABLE IF NOT EXISTS `lock\_table` (

`row\_key` VARCHAR(128) NOT NULL,

`xid` VARCHAR(96),

`transaction\_id` BIGINT,

`branch\_id` BIGINT NOT NULL,

`resource\_id` VARCHAR(256),

`table\_name` VARCHAR(32),

`pk` VARCHAR(36),

`gmt\_create` DATETIME,

`gmt\_modified` DATETIME,

PRIMARY KEY (`row\_key`),

KEY `idx\_branch\_id` (`branch\_id`)

) ENGINE = InnoDB

DEFAULT CHARSET = utf8;

3.修改conf目录下file.conf配置文件



4.修改conf目录下registry.conf配置文件（前提集成注册中心）

5.启动seata-server，进入bin目录下

db存储方式：seata-server.bat -h 127.0.0.1 -p 8091 -m db

6.代码实战编写