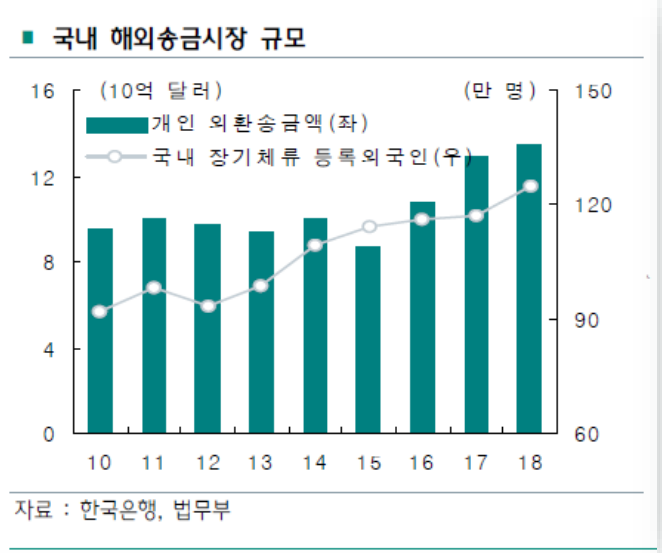
착오송금 걱정을 덜어주는

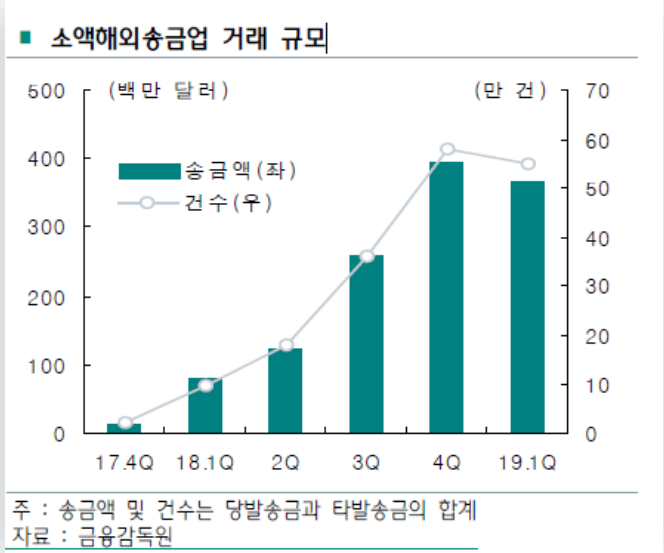
HANA SAFE
해외송금

하나금융티아이 교육생
광명융합기술교육원 데이터분석과 윤다영

목 차

01 기획 배경 및 서비스 소개





5년간 송금 실수 1兆..."예보, 100% 회수해도 손실 불가피"

입력 2019.10.20 17:45 | 수정 2019.10.21 01:51 | 지면 A14

3년 반새 착오송금 1조411억원...피해자 구제법 발의

입력 2020.06.11 17:38 | 수정 2020.06.11 17:38

✓ 빠르게 증가하는 해외송금시장 규모

✓ 간편해진 서비스, 증가한 착오 송금

해외송금 착오송금을 예방하고, 빠르게 해결할 서비스가 필요

01 기획 배경 및 서비스 소개

HANA SAFE 해외송금

"기존 하나은행 해외송금에서 착오송금 예방, 해결 기능을 추가한 서비스"

One-Stop 송금정보 입력

- ✓ 수취인 직접 입력
- ✓ 송금 전 정보 심사

착오송금 신고

- ✓ 쉬운 착오송금 신고
- ✓ 처리 과정 확인

실시간 알림

관리자 승인

↓

승인 내역 알림

02 주요 기능

해외송금 보내기 / 예약

예약일자 9시에 자동송금

승인 내역 알림

해외송금/착오신고 업데이트 시 알림

송금정보
수취인 직접입력

착오송금 신고

해외송금 전용

실시간 환율 조회

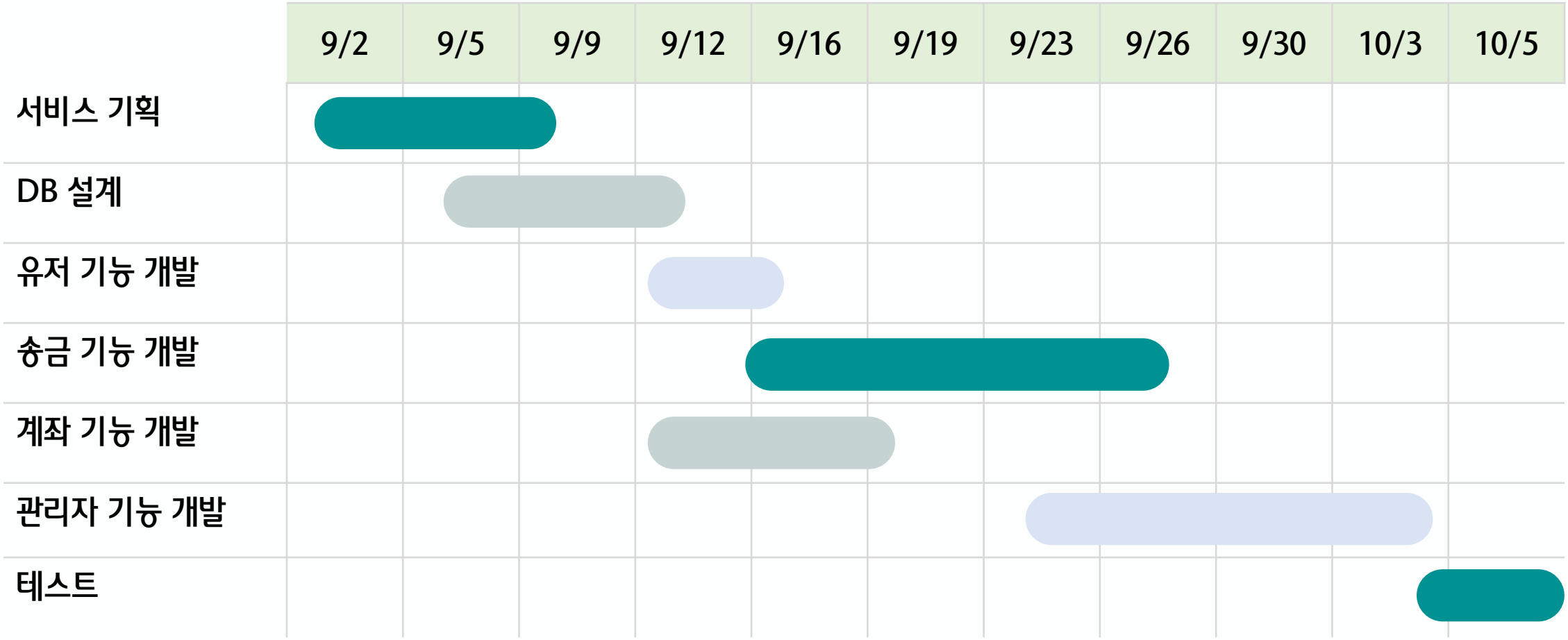
송금 시 실시간 송금 환율 계산

언어 선택

한국어 / 영어 / 중국어 3개국 지원

03 일정 상세

Hana Safe 해외송금
2020/09/02 ~ 2020/10/05 총 33일간 진행



04 아키텍쳐

FRONT-END

HTML

CSS

JAVASCRIPT

BOOTSTRAP

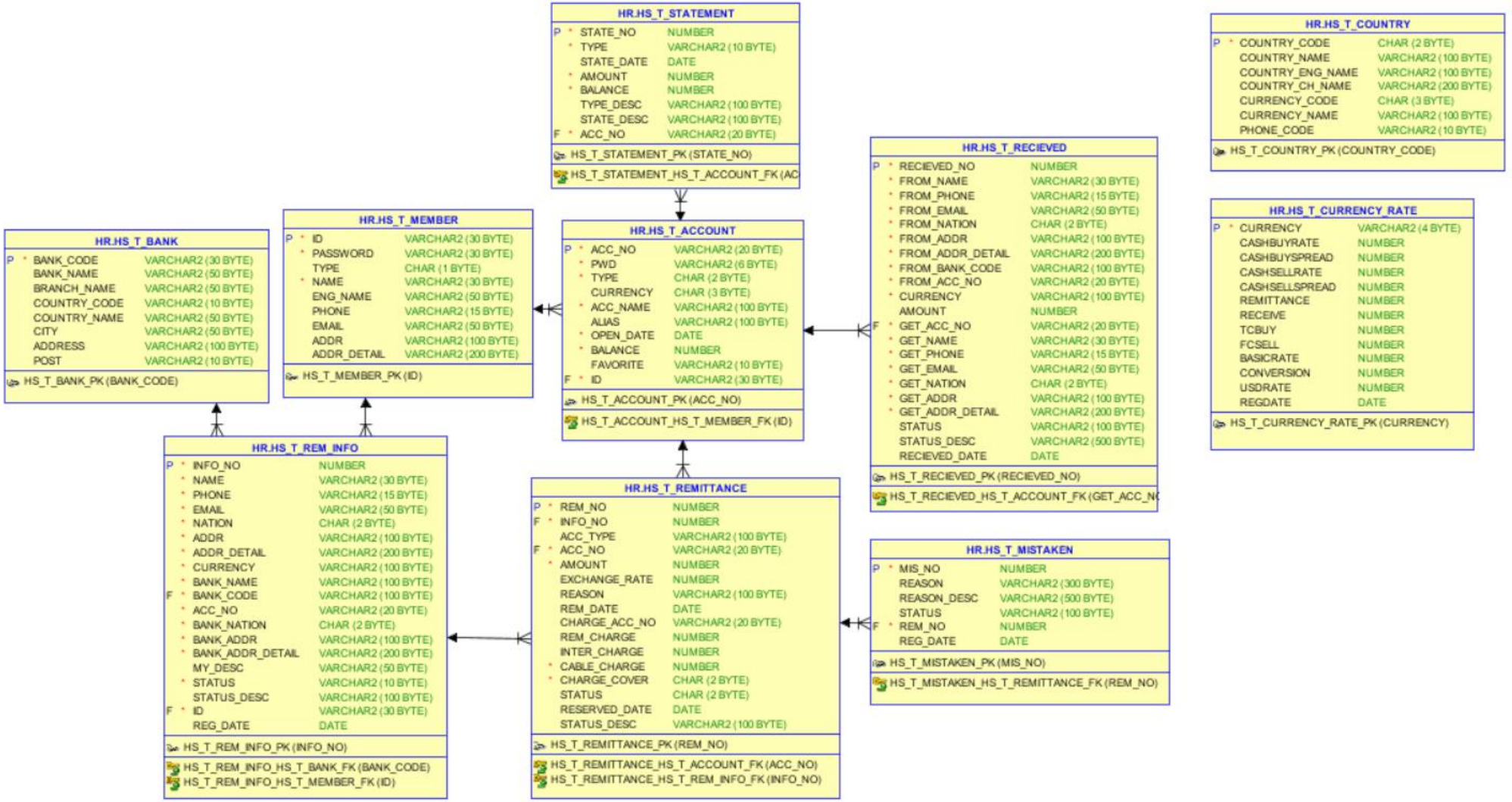
BACK-END

Spring Framework

MyBatis

Apache Tomcat

05 ERD


HR.HS_T_STATEMENT
P * STATE_NO NUMBER
* TYPE VARCHAR2 (10 BYTE)
STATE_DATE DATE
* AMOUNT NUMBER
* BALANCE NUMBER
TYPE_DESC VARCHAR2 (100 BYTE)
STATE_DESC VARCHAR2 (100 BYTE)
F * ACC_NO VARCHAR2 (20 BYTE)
HS_T_STATEMENT_PK (STATE_NO)
HS_T_STATEMENT_HS_T_ACCOUNT_FK (AC
HR.HS_T_COUNTRY
P * COUNTRY_CODE CHAR (2 BYTE)
COUNTRY_NAME VARCHAR2 (100 BYTE)
COUNTRY_ENG_NAME VARCHAR2 (100 BYTE)
COUNTRY_CH_NAME VARCHAR2 (200 BYTE)
CURRENCY_CODE CHAR (3 BYTE)
CURRENCY_NAME VARCHAR2 (100 BYTE)
PHONE_CODE VARCHAR2 (10 BYTE)
HS_T_COUNTRY_PK (COUNTRY_CODE)
HR.HS_T_BANK
P * BANK_CODE VARCHAR2 (30 BYTE)
BANK_NAME VARCHAR2 (50 BYTE)
BRANCH_NAME VARCHAR2 (50 BYTE)
COUNTRY_CODE VARCHAR2 (10 BYTE)
COUNTRY_NAME VARCHAR2 (50 BYTE)
CITY VARCHAR2 (50 BYTE)
ADDRESS VARCHAR2 (100 BYTE)
POST VARCHAR2 (10 BYTE)
HS_T_BANK_PK (BANK_CODE)
HR.HS_T_MEMBER
P * ID VARCHAR2 (30 BYTE)
* PASSWORD VARCHAR2 (30 BYTE)
TYPE CHAR (1 BYTE)
* NAME VARCHAR2 (30 BYTE)
ENG_NAME VARCHAR2 (50 BYTE)
PHONE VARCHAR2 (15 BYTE)
EMAIL VARCHAR2 (50 BYTE)
ADDR VARCHAR2 (100 BYTE)
ADDR_DETAIL VARCHAR2 (200 BYTE)
HS_T_MEMBER_PK (ID)
HR.HS_T_ACCOUNT
P * ACC_NO VARCHAR2 (20 BYTE)
* PWD VARCHAR2 (6 BYTE)
* TYPE CHAR (2 BYTE)
CURRENCY CHAR (3 BYTE)
* ACC_NAME VARCHAR2 (100 BYTE)
ALIAS VARCHAR2 (100 BYTE)
* OPEN_DATE DATE
* BALANCE NUMBER
FAVORITE VARCHAR2 (10 BYTE)
F * ID VARCHAR2 (30 BYTE)
HS_T_ACCOUNT_PK (ACC_NO)
HS_T_ACCOUNT_HS_T_MEMBER_FK (ID)
HR.HS_T_RECIEVED
P * RECIEVED_NO NUMBER
* FROM_NAME VARCHAR2 (30 BYTE)
* FROM_PHONE VARCHAR2 (15 BYTE)
* FROM_EMAIL VARCHAR2 (50 BYTE)
* FROM_NATION CHAR (2 BYTE)
* FROM_ADDR VARCHAR2 (100 BYTE)
* FROM_ADDR_DETAIL VARCHAR2 (200 BYTE)
* FROM_BANK_CODE VARCHAR2 (100 BYTE)
* FROM_ACC_NO VARCHAR2 (20 BYTE)
* CURRENCY VARCHAR2 (100 BYTE)
AMOUNT NUMBER
F * GET_ACC_NO VARCHAR2 (20 BYTE)
* GET_NAME VARCHAR2 (30 BYTE)
* GET_PHONE VARCHAR2 (15 BYTE)
* GET_EMAIL VARCHAR2 (50 BYTE)
* GET_NATION CHAR (2 BYTE)
* GET_ADDR VARCHAR2 (100 BYTE)
* GET_ADDR_DETAIL VARCHAR2 (200 BYTE)
STATUS VARCHAR2 (100 BYTE)
STATUS_DESC VARCHAR2 (500 BYTE)
RECIEVED_DATE DATE
HS_T_RECIEVED_PK (RECIEVED_NO)
HS_T_RECIEVED_HS_T_ACCOUNT_FK (GET_ACC_NO
HR.HS_T_CURRENCY_RATE
P * CURRENCY VARCHAR2 (4 BYTE)
CASHBUYRATE NUMBER
CASHBUYSPREAD NUMBER
CASHSELLRATE NUMBER
CASHSELLSPREAD NUMBER
REMITTANCE NUMBER
RECEIVE NUMBER
TCBUY NUMBER
FCSELL NUMBER
BASICRATE NUMBER
CONVERSION NUMBER
USDRATE NUMBER
REGDATE DATE
HS_T_CURRENCY_RATE_PK (CURRENCY)
HR.HS_T_REM_INFO
P * INFO_NO NUMBER
* NAME VARCHAR2 (30 BYTE)
* PHONE VARCHAR2 (15 BYTE)
* EMAIL VARCHAR2 (50 BYTE)
* NATION CHAR (2 BYTE)
* ADDR VARCHAR2 (100 BYTE)
* ADDR_DETAIL VARCHAR2 (200 BYTE)
* CURRENCY VARCHAR2 (100 BYTE)
* BANK_NAME VARCHAR2 (100 BYTE)
F * BANK_CODE VARCHAR2 (100 BYTE)
* ACC_NO VARCHAR2 (20 BYTE)
* BANK_NATION CHAR (2 BYTE)
* BANK_ADDR VARCHAR2 (100 BYTE)
* BANK_ADDR_DETAIL VARCHAR2 (200 BYTE)
MY_DESC VARCHAR2 (50 BYTE)
* STATUS VARCHAR2 (10 BYTE)
STATUS_DESC VARCHAR2 (100 BYTE)
F * ID VARCHAR2 (30 BYTE)
REG_DATE DATE
HS_T_REM_INFO_PK (INFO_NO)
HS_T_REM_INFO_HS_T_BANK_FK (BANK_CODE)
HS_T_REM_INFO_HS_T_MEMBER_FK (ID)
HR.HS_T_REMITTANCE
P * REM_NO NUMBER
F * INFO_NO NUMBER
ACC_TYPE VARCHAR2 (100 BYTE)
F * ACC_NO VARCHAR2 (20 BYTE)
* AMOUNT NUMBER
EXCHANGE_RATE NUMBER
REASON VARCHAR2 (100 BYTE)
REM_DATE DATE
CHARGE_ACC_NO VARCHAR2 (20 BYTE)
REM_CHARGE NUMBER
INTER_CHARGE NUMBER
* CABLE_CHARGE NUMBER
* CHARGE_COVER CHAR (2 BYTE)
STATUS CHAR (2 BYTE)
RESERVED_DATE DATE
STATUS_DESC VARCHAR2 (100 BYTE)
HS_T_REMITTANCE_PK (REM_NO)
HS_T_REMITTANCE_HS_T_ACCOUNT_FK (ACC_NO)
HS_T_REMITTANCE_HS_T_REM_INFO_FK (INFO_NO)
HR.HS_T_MISTAKEN
P * MIS_NO NUMBER
REASON VARCHAR2 (300 BYTE)
REASON_DESC VARCHAR2 (500 BYTE)
STATUS VARCHAR2 (100 BYTE)
F * REM_NO NUMBER
REG_DATE DATE
HS_T_MISTAKEN_PK (MIS_NO)
HS_T_MISTAKEN_HS_T_REMITTANCE_FK (REM_NO)

06 주요 기술

01
WebSocket
관리자 승인 시 실시간 알림

02
Javax mail
Thymeleaf
템플릿 형태 이메일 전송

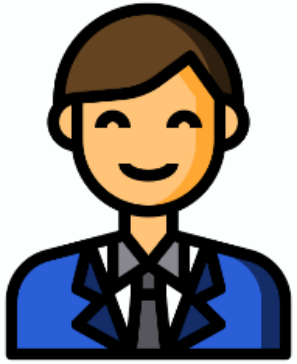
03
Spring Locale
한/영/중 3개 언어 지원

04
Python Web Scraping
10분마다 환율 정보 업데이트

05
Spring Scheduler
영업일 아침 9시마다
예약일 도래할 경우 자동송금

06
Transaction
송금, 착오송금 반환 시
계좌, 거래내역 입력 트랜잭션

07 시나리오



홍길동 손님

싱가포르에 살고 있는 **딸(홍다영)**에게
해외송금을 보내야 한다.
큰 금액인만큼 **착오송금이 걱정**되는데…

착오송금을 예방하고 빠르게 해결해주는
HANASAFE 해외송금에서 송금하자!

07 시나리오

이메일 전송

모든 과정 알림

홍다영 (홍길동 딸)

홍길동 손님

관리자 은행원

아빠(홍길동)
대신 송금정보 입력

중국어가 편한 다영,
중국어로 언어 변환

딸에게 송금 정보 요청

송금 보내기

문제 발생
착오송금 신고

딸에게 다시
송금 예약

홍다영(홍길동 딸)
정보 승인

착오송금 반환 처리

해외송금
중계은행으로 접수

착오송금 걱정을 덜어주는

HANA SAFE 해외송금

- 시연영상 -

하나금융티아이 교육생 윤다영

08 보완사항

BankCodesAPI의 은행코드
은행 정보 API 접근 불가,
DB에 은행 데이터 삽입

GCP, AWS 등 클라우드 환경
deploy 계획

외환 업무에 대한 정확한
프로세스 파악 후
관리자단 개선 필요

Q&A

감사합니다

Appendix

부록

#01_WebSocket

```
<websocket:handlers>
    <websocket:mapping handler="echoHandler" path="/echo" />
    <!-- handshake -> handler에 접근하기 전에 httpSession에 접근하여 저장된 값을 읽어 들여 사용할 수 있도록 처리 -->
        <websocket:handshake-interceptors>
            <bean class="org.springframework.web.socket.server.support.HttpSessionHandshakeInterceptor"/>
        </websocket:handshake-interceptors>

 </websocket:handlers>
```

```
//소켓에 메세지 보냈을 때
@Override
protected void handleTextMessage(WebSocketSession session, TextMessage message) throws Exception {
    System.out.println("handleTextMessage : " + session + " : " + message);

    String senderId = getId(session);

    //Protocol : cmd(remittance, recieve, reserve, mistaken), 해당 id, 해당 이름, status(승인 과정)
    String msg = message.getPayload();
    if(!msg.equals("")) {
        String[] strs = msg.split(",");

        if(strs != null && strs.length == 4) {
            String cmd = strs[0];
            String recieverId = strs[1];
            String recieverName = strs[2];
            String status = strs[3];

            WebSocketSession recieverSession = userSessionsMap.get(recieverId);

            if(recieverSession != null) {
                TextMessage sendMsg = null;
                if(cmd.equals("remittance")) {
                    sendMsg = new TextMessage(recieverName + "님이 보낸 해외송금 내역이 업데이트되었습니다 (" + status + ")");
                }else if(cmd.equals("recieved")) {
                    sendMsg = new TextMessage(recieverName + "님께 새로운 해외송금이 도착했습니다");
                }else if(cmd.equals("remInfo")) {
                    sendMsg = new TextMessage(recieverName + "님이 신청한 해외송금정보가 업데이트되었습니다 (" + status + ")");
                }else if(cmd.equals("mistaken")) {
                    sendMsg = new TextMessage(recieverName + "님이 신고한 착오송금 처리가 업데이트되었습니다 (" + status + ")");
                }
                recieverSession.sendMessage(sendMsg);
            }
```

```
//서버에 접속 성공 - client가 server에 접속 성공
@Override
public void afterConnectionEstablished(WebSocketSession session) throws Exception {
    System.out.println("afterConnectionExtablished" + session);

    //접속한 유저는 다 여기 안에 들어감
    sessions.add(session);

    //session에 현재 등록되어있는 id 조회 - 로그인 되어있으면 id, 아니면 session id
    String senderId = getId(session);
    userSessionsMap.put(senderId, session);
}
```

```
//연결 해제될때 - connection closed
@Override
public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws Exception {
    System.out.println("afterConnectionEstablished : " + session + " : " + status);


    userSessionsMap.remove(session.getId());
    sessions.remove(session);


}
```

#01_WebSocket

```
function connect(){
    var ws = new WebSocket("ws://" + location.host + "/Hana-Safe/echo");

    socket = ws;

    ws.onopen = function(){
        console.log('Info:connection opened.');


        //connection이 되어야 message를 받으니까
        ws.onmessage = function(event){
            console.log("Receive Message : " + event.data + '/n')
        }
    }



    ws.onclose = function(event){
        console.log('info : connection closed ');
        //setTimeout(function(){    connect();  }, 1000) //retry connection
    };

    ws.onerror = function(err){ console.log('error :', err); };
}


connect();
```

```
var socket = null;


function connect(){
    var ws = new WebSocket("ws://" + location.host + "/Hana-Safe/echo");
    socket = ws;

    ws.onopen = function(){
        console.log('Info:connection opened.');


        //connection이 되어야 message를 받으니까
        ws.onmessage = function(event){
            console.log("Receive Message : " + event.data + '/n')

            toastr.info(event.data, '새로운 해외송금 소식').css({
            "width": "500px",
            "background-color" : "#008485"

                })
                }
            }



            ws.onclose = function(event){
                console.log('info : connection closed ');
    };

    ws.onerror = function(err){ console.log('error :', err); };
}


connect();
```

#02_Javax mail, Thymeleaf

```xml
<!-- Gmail -->
<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="${email.id.gmail}" />
    <property name="password" value="${email.password.gmail}" />
    <property name="javaMailProperties">
        <props>
            <prop key="mail.smtp.auth">true</prop>
            <prop key="mail.smtp.starttls.enable">true</prop>
        </props>
    </property>
</bean>
```

```xml
<!-- 이메일 format thymeleaf 설정 -->
<bean id="templateResolver"
      class="org.thymeleaf.spring4.templateresolver.SpringResourceTemplateResolver">
    <property name="prefix" value="classpath:/templates/" />
    <property name="suffix" value=".html" />
    <property name="templateMode" value="HTML5" />
    <property name="characterEncoding" value="UTF-8" />
    <property name="cacheable" value="false" />
</bean>
<bean id="templateEngine" class="org.thymeleaf.spring4.SpringTemplateEngine">
    <property name="templateResolver" ref="templateResolver" />
    <property name="enableSpringELCompiler" value="true"></property>
</bean>
<bean id="viewResolver" class="org.thymeleaf.spring4.view.ThymeleafViewResolver">
    <property name="characterEncoding" value="UTF-8" />
    <property name="templateEngine" ref="templateEngine" />
</bean>
```

```java
@Override
public void sendEmail(String emailAddr, String fromName) throws MessagingException{

    Context context = new Context();
    context.setVariable("homepageUrl", "http://localhost:9999/Hana-Safe/remInfo/ask/auth");


    String from = "hanasafetransfer@gmail.com";
    MimeMessage message = mailSender.createMimeMessage();
    MimeMessageHelper helper = new MimeMessageHelper(message, true, "utf-8");

    helper.setFrom(from);
    helper.setTo(emailAddr);
    helper.setSubject("[HanaSafe] " + fromName + "님께서 해외송금 정보입력을 요청했습니다");

    String htmlContent = springTemplateEngine.process("remInfoEmail", context);
    helper.setText(htmlContent, true);


    mailSender.send(message);

}
```

#03_Spring Locale

```
<!-- 언어 message file 설정 -->
<bean id="messageSource" class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
    <property name="basename" value="classpath:/locale/messages" />
    <property name="defaultEncoding" value="UTF-8" />
    <property name="fallbackToSystemLocale" value="false" />
</bean>

<!-- LocaleResolver 설정 -->
<bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver">
    <property name="defaultLocale" value="ko"></property>
</bean>

<!-- change interceptor -->
   <mvc:interceptors>
    <bean id="localeChangeInterceptor" class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
       <property name="paramName" value="locale" />
    </be
    </mv
```

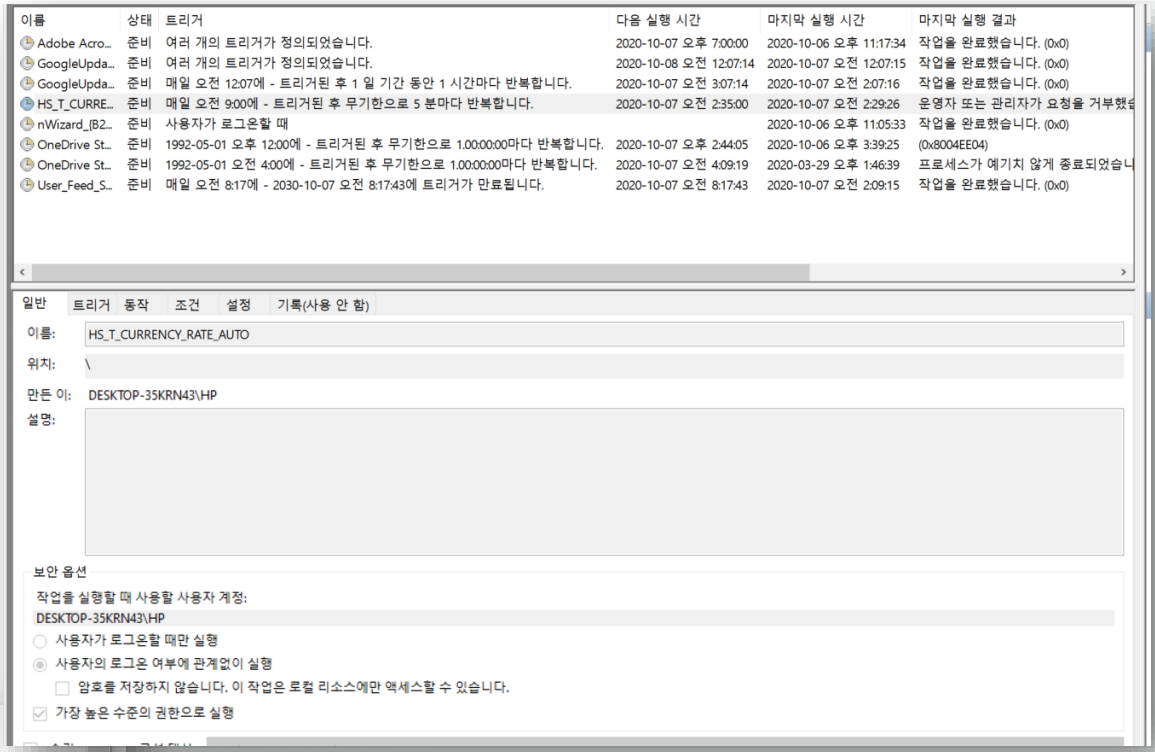
```
config
locale
    messages_en.properties
    messages_ko.properties
    messages_zh.properties
```

```
function localeChange(locale){
    var url = getContextPath() + '/change?locale=' + locale;

    $.ajax({
        url : url,
        type : 'get',
        success : function(){
            location.reload();
        },
        error : function(){
            alert('언어 변환 과정에 에러 발생')
        }

    })

}
```

```
@GetMapping("/change")
public String changeLocale(Locale locale, HttpSession session) {
    session.setAttribute("langauge", locale.getLanguage());
    return "redirect:/";
}
```

#04_Python Web Scraping

```
# virable

url = "https://www.kebhana.com/cms/rate/wpfxd651_01i_01.do"
data={"ajax": "true", "curCd": "", "tmpInqStrDt": nowDateFormat1, "pbldDvCd": 3, "pbldSqn":"", "hid_key_data":"",
      "inqStrDt":nowDateFormat2,"inqKindCd":1 ,"hid_enc_data":"", "requestTarget":"searchContentDiv" }
#data에 tmpInqStrDt 랑 inqStrDt는 조회를 원하는 날짜로 수정하셔야합니다.


# request(POST)
res = requests.post(url, data)

# soup
soup = BeautifulSoup(res.content, "html.parser")

#print(soup)


title_data=soup.select('.txtAr')

array = [[0 for col in range(12)] for row in range(50)] # 49행12열 선언

myList=[]
moneyList=['USD','JPY','EUR','CNY','HKD','THB','TWD','PHP','SGD','AUD','VND','GBP','CAD',
      'MYR','RUB','ZAR','NOK','NZD','DKK','MXN','MNT','BHD','BDT','BRL','BND','SAR',
      'LKR','SEK','CHF','AED','DZD','OMR','JOD','ILS','EGP','INR','IDR','CZK','CLP',
      'KZT','QAR','KES','COP','KWD','TZS','TRY','PKR','PLN','HUF','dum']
```




```
for i in range(50):
    kk = array[i]
    print(kk)
    sql_query = 'update hs_t_currency_rate set CASHBUYRATE=:b, CASHBUYSPREAD=:c, CASHSELLRATE=:d, CASHSELLSPREAD=:e, REMITT
    cursor.execute(sql_query,  b=kk[1], c=kk[2] ,  d=kk[3], e=kk[4],f=kk[5],g=kk[6],h=kk[7],i=kk[8],j=kk[9],k=kk[10],l=kk[11], m=now , a=kk

db.commit()
```

#05_Spring Scheduler

```
 <!--  task scheduler 예약 -->
<task:scheduler id="jobScheduler" pool-size="10"/>
<task:annotation-driven scheduler="jobScheduler"/>
```

```
@Scheduled(cron = "0 0 9 * * MON-FRI")
@Override
@Transactional
public void insertRemittanceFromReservation() {
    System.out.println("스케줄러 실행중~~");

    //오늘 = 예약날짜 일 경우 모든 remittanceList를 가져옴
    List<RemittanceVO> remList = remittanceDAO.selectReservedTodayDateEqual();
    //각 remList를 돌며 모든 remittance 돈 나가게하고, 거래내역 넣기
    if(remList != null) {
        for(RemittanceVO remittanceVO : remList) {
            insertRemittance(remittanceVO);
        }
    }
}
```

#06_Transaction

```
<!-- 트랜잭션 처리 -->
<bean class="org.springframework.jdbc.datasource.DataSourceTransactionManager" id="transactionManager" >
    <property name="dataSource" ref="dataSource" />
</bean>
<tx:annotation-driven transaction-manager="transactionManager"/>
```

```
@Override
@Transactional
public void insertRemittance(RemittanceVO remittanceVO) {
    //거래내역에 필요한 정보 불러옴
    Integer chargeTotal = (remittanceVO.getCableCharge() + remittanceVO.getInterCharge() + remittanceVO.getRemCharge());
    BigDecimal chargeTotalBig = new BigDecimal(String.valueOf(chargeTotal));
    RemInfoVO remInfo = remInfoDAO.selectRemInfoDetail(remittanceVO.getInfoNo());
    BigDecimal mainAccBalance = accountDAO.selectAccountByAccNo(remittanceVO.getAccNo()).getBalance();
    BigDecimal chargeAccBalance = accountDAO.selectAccountByAccNo(remittanceVO.getChargeAccNo()).getBalance();

    //송금내역 추가
    remittanceDAO.insertRemittance(remittanceVO);
    accountDAO.updateRemittanceAmount(remittanceVO);
    accountDAO.updateRemittanceAmount(new RemittanceVO(remittanceVO.getChargeAccNo(), chargeTotalBig));


    //거래내역 생성, 거래내역 insert
    StatementVO mainStatement = new StatementVO(remittanceVO.getAmount(), mainAccBalance.subtract(remittanceVO.getAmount()),
                                                 remInfo.getName(), remittanceVO.getAccNo(), "출금");
    StatementVO chargeStatement = new StatementVO(chargeTotalBig, chargeAccBalance.subtract(chargeTotalBig),
                                                  "해외송금 수수료", remittanceVO.getChargeAccNo(), "출금");
    statementDAO.insertRemittanceStatement(mainStatement);
    statementDAO.insertRemittanceStatement(chargeStatement);

}
```