

# 主題 1：PIA

## 認識個資稽核軌跡

<https://km.fubonlife.com.tw/confluence/pages/viewpage.action?pageId=805114115>

### 開始實作

步驟一：在資料庫建立資料表

請留意...下載的檔案內，內容末端會包含【預儲程序的 DDL 陳述式 SP\_PIA\_DEL\_HISTTRACK】，是可以不用建立的 !!

上游系統軌跡記錄資料表 DDL（人壽稽核軌跡建置專案）

步驟二：為資料表產生Entity、Repository、Service

資料表名稱	PIA_TRACK_TYPE (軌跡類別代碼檔)	PIA_EXEC_TYPE (動作類別代碼檔)	PIA_LOG_MAIN (個資稽核軌跡紀錄檔)	PIA_TRN_HIST (傳輸歷程紀錄檔)
需不需要	非必要	非必要	一定要	非必要
備註			因欄位 ACCOUNT_NAME 包含姓名，所以需要有遮罩	

步驟三：實作 AuthListener（介面）來讓資料表 PIA\_LOG\_MAIN 能留存使用者的登入、登出之行為

```
AuthListenerImpl.java

package com.fubonlife.mytest.api.security;

import com.fubonlife.boot.error.BadRequestException;
import com.fubonlife.boot.helper.IpHelper;
import com.fubonlife.boot.pia.model.PiaExecutionType;
import com.fubonlife.boot.pia.model.PiaTrackingType;
import com.fubonlife.boot.security.AuthListener;
import com.fubonlife.boot.security.model.AuthenticationResult;
import com.fubonlife.mytest.common.entity.primary.Account;
import com.fubonlife.mytest.common.entity.primary.PiaLogMain;
import com.fubonlife.mytest.common.service.AuthService;
import com.fubonlife.mytest.common.service.PiaLogMainService;
import com.fubonlife.mytest.common.util.CommonUtil;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Component;

import javax.persistence.Table;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.time.LocalDate;
import java.time.LocalDateTime;

@Component
@Slf4j
public class AuthListenerImpl implements AuthListener {
```

```

@Autowired
PiaLogMainService piaLogMainService;

@Autowired
AuthService authService;

@Autowired
IpHelper ipHelper;

@Override
public void onAuthenticationSuccess(HttpServletRequest request, HttpServletResponse response,
Authentication authentication, AuthenticationResult result) {
    if (result.isSuccess()) {
        SecurityContextHolder.getContext().setAuthentication(authentication);
        piaAccess(PiaTrackingType.LOGIN, PiaExecutionType.LOGIN_SUCCESS, request, "");
    } else {
        onAuthenticationFailure(request, response, new BadCredentialsException(authentication.getName() +
": Bad credentials"));
    }
}

@Override
public void onAuthenticationFailure(HttpServletRequest request, HttpServletResponse response,
AuthenticationException e) {
    piaAccess(PiaTrackingType.LOGIN, PiaExecutionType.LOGIN_FAILURE, request, "" + e.getMessage(), e);
}

@Override
public void onLogoutSuccess(HttpServletRequest request, HttpServletResponse response, Authentication
authentication) {
    SecurityContextHolder.getContext().setAuthentication(authentication);
    piaAccess(PiaTrackingType.LOGOUT, PiaExecutionType.LOGOUT, request, "");
}

private void piaAccess(PiaTrackingType trackingType, PiaExecutionType executionType, HttpServletRequest
request, String content, Exception... exceptions) {
    String domainId = "?";
    String accountName = "?";

    try {
        if (exceptions.length == 0) {
            Account account = authService.getCurrent();
            domainId = account.getDomainId();
            accountName = account.getEmployee() != null ? CommonUtil.getMaskName(account.getEmployee().
getName()) : account.getUsername();
        }
    } catch (BadRequestException ignored) {
        //
    }

    log.info("PIA [{)][{}]] Operation: {}", domainId, accountName, executionType.getDescription());

    try {
        piaLogMainService.add(
            PiaLogMain.builder()
                .trackingType(trackingType.getCode())
                .accessAccount(domainId)
                .accountName(accountName)
                .fromIp(ipHelper.getClientIp(request).toString() + ", " + ipHelper.getClientHostName
(request))
                .progfunName(this.getClass().getName())
                .accessobjName(Account.class.getAnnotation(Table.class).name())
                .executionType(executionType.getCode())
                .executionContent(content)
                .piaType("0000000000")
                .accessDate(LocalDate.now())
                .accessTime(LocalTime.now())
                .build()
        );
    }
}

```

```

        } catch (Exception e) {
            log.error("domainId: {}, accountName: {}", domainId, accountName, e);
        }
    }
}

```

### CommonUtil.java

```

package com.fubonlife.mytest.common.util;

import org.apache.commons.lang3.StringUtils;
import org.apache.commons.lang3.math.NumberUtils;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

public class CommonUtil {

    /**
     * UUID
     */
    public static String getUUID() {
        return UUID.randomUUID().toString();
    }

    /**
     *
     */
    public static String trim2SpaceChar(String inputString) {
        if (StringUtils.isNotBlank(inputString)) {
            char[] chars = inputString.toCharArray();
            int inputStringLength = chars.length;

            int startIndex = (Character.isSpaceChar(chars[0])) ? 1 : 0;
            int endIndex = (Character.isSpaceChar(chars[inputStringLength - 1])) ? inputStringLength - 1 :
inputStringLength;

            return inputString.substring(startIndex, endIndex);
        } else {
            return "";
        }
    }

    /**
     *
     */
    public static String combineWords(String newWord, String oldWord) {
        if (StringUtils.isBlank(oldWord)) {
            return newWord;
        } else {
            return String.format("%s(%s)", newWord, oldWord);
        }
    }

    /**
     *
     */
    public static String convertToChineseNumber(Object number) {
        int result = 0;

        if (number instanceof String) {
            String s = String.valueOf(number);
            if (NumberUtils.isDigits(s)) {
                result = Integer.parseInt(s);
            }
        }

        switch (result) {

```

```

        case 1:
            return "";
        case 2:
            return "";
        case 3:
            return "";
        case 4:
            return "";
        case 5:
            return "";
        case 6:
            return "";
        case 7:
            return "";
        case 8:
            return "";
        case 9:
            return "";
        case 10:
            return "";
        case 11:
            return "";
        case 12:
            return "";
        default:
            return "";
    }
}

/**
 * HTML-
 */
public static String getHtmlCodeForTable(List<List<String>> data, boolean hasHeader, String...
overrideTableStyle) {
    StringBuilder htmlCode = new StringBuilder(data.size() * 100);

    //
    if (overrideTableStyle != null && overrideTableStyle.length > 0) {
        htmlCode.append("<table border='1' cellspacing='0' cellpadding='5' style='");
        for (String tableStyle : overrideTableStyle) {
            htmlCode.append(tableStyle).append(";");
        }
        htmlCode.append(">");
    } else {
        htmlCode.append("<table border='1' cellspacing='0' cellpadding='5' style='text-align:center;margin-
top:20px;'>");
    }

    for (List<String> oneRow : data) {
        htmlCode.append("<tr>");
        if (hasHeader) {
            for (String contentForTH : oneRow) {
                htmlCode.append("<th>").append(contentForTH).append("</th>");
            }
            hasHeader = false;
        } else {
            for (String contentForTD : oneRow) {
                htmlCode.append("<td>").append(contentForTD).append("</td>");
            }
        }
        htmlCode.append("</tr>");
    }

    return htmlCode.append("</table>").toString();
}

/**
 *
 */
public static String getMaskName(String username) {
    if (StringUtils.isNotBlank(username)) {

```

```

        int length = username.length();

        if ((username.getBytes().length != length) && (username.getBytes().length % length == 0)) {
            // 2*1
            if (length == 2) {
                username = String.format("%s", username.charAt(0));
            }
            // 2*2
            else if (length > 2) {
                username = String.format("%s%s", username.substring(0, length - 2), username.substring
(length - 1));
            }
            } else {
                if (length == 1) {
                    username = "";
                }
                // 7*2
                else if (length <= 7) {
                    username = String.format("%s%s", username.substring(0, length - 2), username.substring
(length - 1));
                }
                // 7*1~6
                else {
                    username = String.format("%s", username.substring(6));
                }
            }
        } else {
            username = "";
        }

        return username;
    }

    /**
     * EMail address
     */
    public static List<String> getMaskEmailAddress(List<String> mailAddresses) {
        return mailAddresses.stream().map(CommonUtil::getMaskEmailAddress).collect(Collectors.toList());
    }

    /**
     * EMail address
     */
    public static String getMaskEmailAddress(String emailAddress) {
        if (StringUtils.isNotBlank(emailAddress)) {
            String [] mailArr = emailAddress.split("@");
            if (mailArr.length > 0) {
                return String.format("%s%s", getMaskString(mailArr[0], 4, 0), getMaskString(mailArr[1], 4, 8));
            }
        }
        return "";
    }

    /**
     *
     *
     * @param originalString
     * @param startIndex index
     * @param endIndex index
     * @return String
     */
    private static String getMaskString(String originalString, int startIndex, int endIndex) {
        if (StringUtils.isNotBlank(originalString)) {
            // indexindex-1
            if (startIndex > originalString.length()) {
                return getMaskString(originalString, startIndex - 1, endIndex);
            } else {
                startIndex = startIndex <= 1 ? 1 : startIndex - 1;
                if (endIndex == 0) { // index
                    return String.format("%s", originalString.substring(0, startIndex));
                }
            }
        }
    }

```

```

        } else {
            endIndex = endIndex > originalString.length() ? originalString.length() - 1 : endIndex;
            return String.format("%s%s", originalString.substring(0, startIndex), originalString.
substring(endIndex));
        }
    }
}
return "";
}
}
}

```

步驟四：實作 `PiaOperationHandler`（介面）來讓資料表 `PIA_LOG_MAIN` 能留存使用者存取個資之行為

請留意...因個資包含姓名，所以需要有遮罩 !!

#### PiaOperationHandlerImpl.java

```

package com.fubonlife.mytest.common.pia;

import com.fubonlife.boot.error.BadRequestException;
import com.fubonlife.boot.pia.PiaOperationHandler;
import com.fubonlife.boot.pia.model.PiaOperation;
import com.fubonlife.mytest.common.entity.primary.Account;
import com.fubonlife.mytest.common.entity.primary.PiaLogMain;
import com.fubonlife.mytest.common.service.AuthService;
import com.fubonlife.mytest.common.service.PiaLogMainService;
import com.fubonlife.mytest.common.util.CommonUtil;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.time.LocalDate;
import java.time.LocalDateTime;

@Component
@Slf4j
public class PiaOperationHandlerImpl implements PiaOperationHandler {

    @Autowired
    PiaLogMainService piaLogMainService;

    @Autowired
    AuthService authService;

    @Override
    public void handle(PiaOperation operation) {
        String domainId = "?";
        String accountName = "?";

        try {
            Account account = authService.getCurrent();
            domainId = account.getDomainId();
            accountName = account.getEmployee() != null ? CommonUtil.getMaskName(account.getEmployee().
getName()) : account.getUsername();
        } catch (BadRequestException ignored) {
            //
        }

        log.info("PIA [{}][{}] Operation: {}", domainId, accountName, operation);

        String accessObjName = String.join("|", operation.getStatement().getObjects());

        String executionContent = String.format("%s\n%s",

```

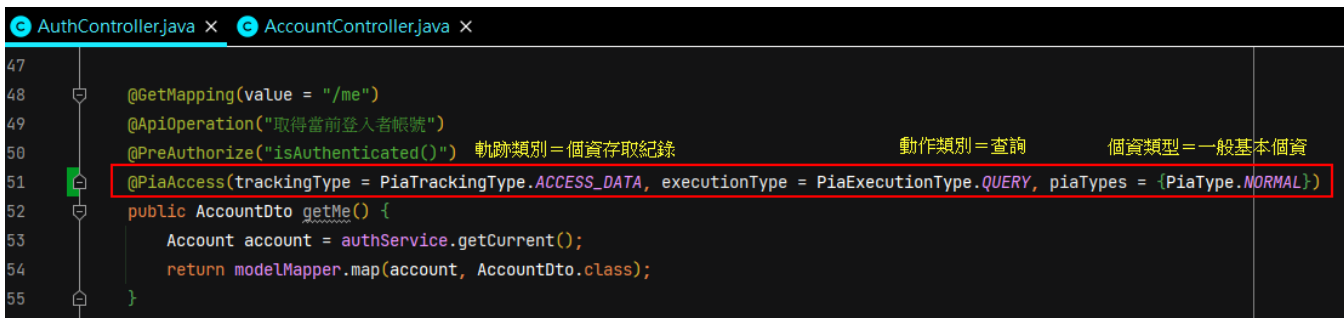
```

        operation.getStatement().getQuery(),
        String.join("\n", operation.getStatement().getParams()));

    try {
        piaLogMainService.add(
            PiaLogMain.builder()
                .trackingType(operation.getTrackingType())
                .accessAccount(domainId)
                .accountName(accountName)
                .fromIp(operation.getClientIp() + "," + operation.getClientHostName())
                .progfunName(operation.getMethod())
                .accessobjName(accessObjName)
                .executionType(operation.getStatement().getExecutionType().getCode())
                .executionContent(executionContent)
                .piaType(operation.getPiaTypeCode())
                .affectRows(operation.getStatement().getLoaded() + operation.getStatement().getAffected())
                .accessDate(LocalDate.now())
                .accessTime(LocalTime.now())
                .piaOwner1("")
                .piaOwner2("")
                .build()
        );
    } catch (Exception e) {
        log.error("", e);
    }
}
}

```

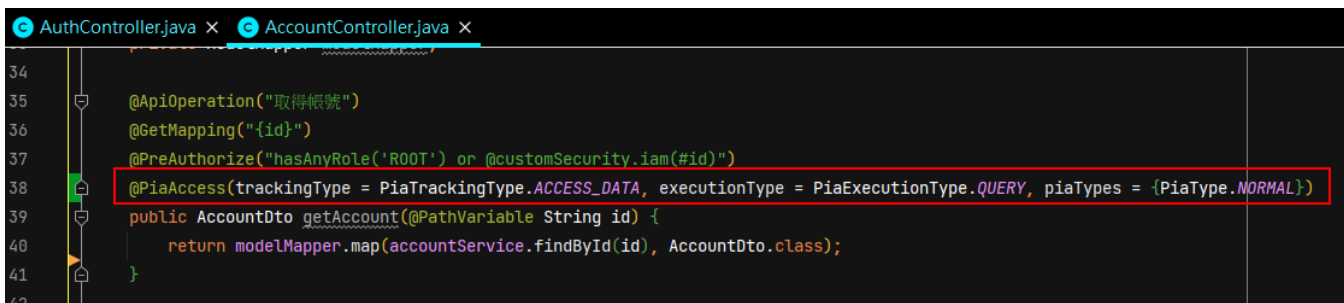
步驟五：立即為 `AuthController`、`AccountController` 導入PIA



```

47
48 @GetMapping(value = "/me")
49 @ApiOperation("取得當前登入者帳號")
50 @PreAuthorize("isAuthenticated()") 軌跡類別=個資存取紀錄 動作類別=查詢 個資類型=一般基本個資
51 @PiaAccess(trackingType = PiaTrackingType.ACCESS_DATA, executionType = PiaExecutionType.QUERY, piaTypes = {PiaType.NORMAL})
52 public AccountDto getMe() {
53     Account account = authService.getCurrent();
54     return modelMapper.map(account, AccountDto.class);
55 }

```



```

34
35 @ApiOperation("取得帳號")
36 @GetMapping("/{id}")
37 @PreAuthorize("hasAnyRole('ROOT') or @customSecurity.iam(#id)")
38 @PiaAccess(trackingType = PiaTrackingType.ACCESS_DATA, executionType = PiaExecutionType.QUERY, piaTypes = {PiaType.NORMAL})
39 public AccountDto getAccount(@PathVariable String id) {
40     return modelMapper.map(accountService.findById(id), AccountDto.class);
41 }
42

```

步驟六：啟用 PIA

