

# Hibernate 小組報告: 會員登入及註冊

EEIT109 第六組

指導老師:陳奕兆

組長:陳冠宇

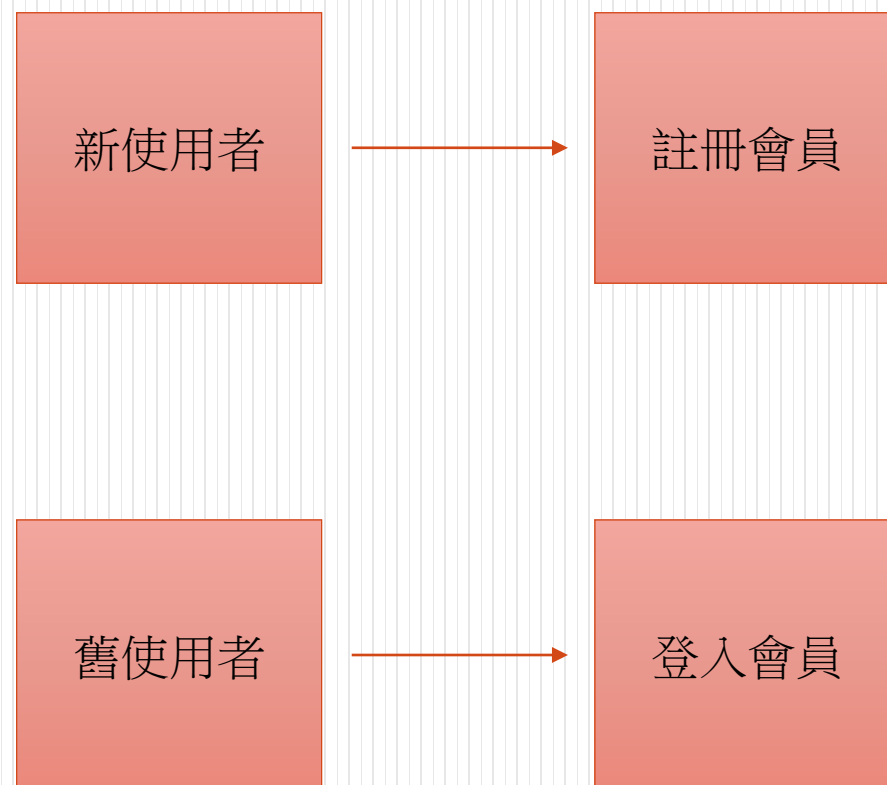
組員:

葉冠麟、廖于慧、徐達仁、葉家榮、卓振興

# 簡報大綱

- 使用者導覽
- 程式架構
- Hibernate程式片段

# 使用者導覽



# 使用者導覽—新使用者

[回首頁](#)[遊戲討論區](#)[遊戲商城](#)[電競新聞](#)[註冊會員](#)[登入會員](#)

申請帳號必填

帳號:  

密碼:  

(1.不可空白，2.至少六個字包含英文字母，數字，特殊字元(!@#\$%^&\*)

姓名:

(1.不可空白，2.至少兩個字以上，3.必須為中文字)

身分證字號:

地址:

性別: ☐ 男 ☐ 女

生日:

註冊

# 使用者導覽—舊使用者

回首頁

遊戲討論區

遊戲商城

電競新聞

註冊會員

登入會員



登入帳號

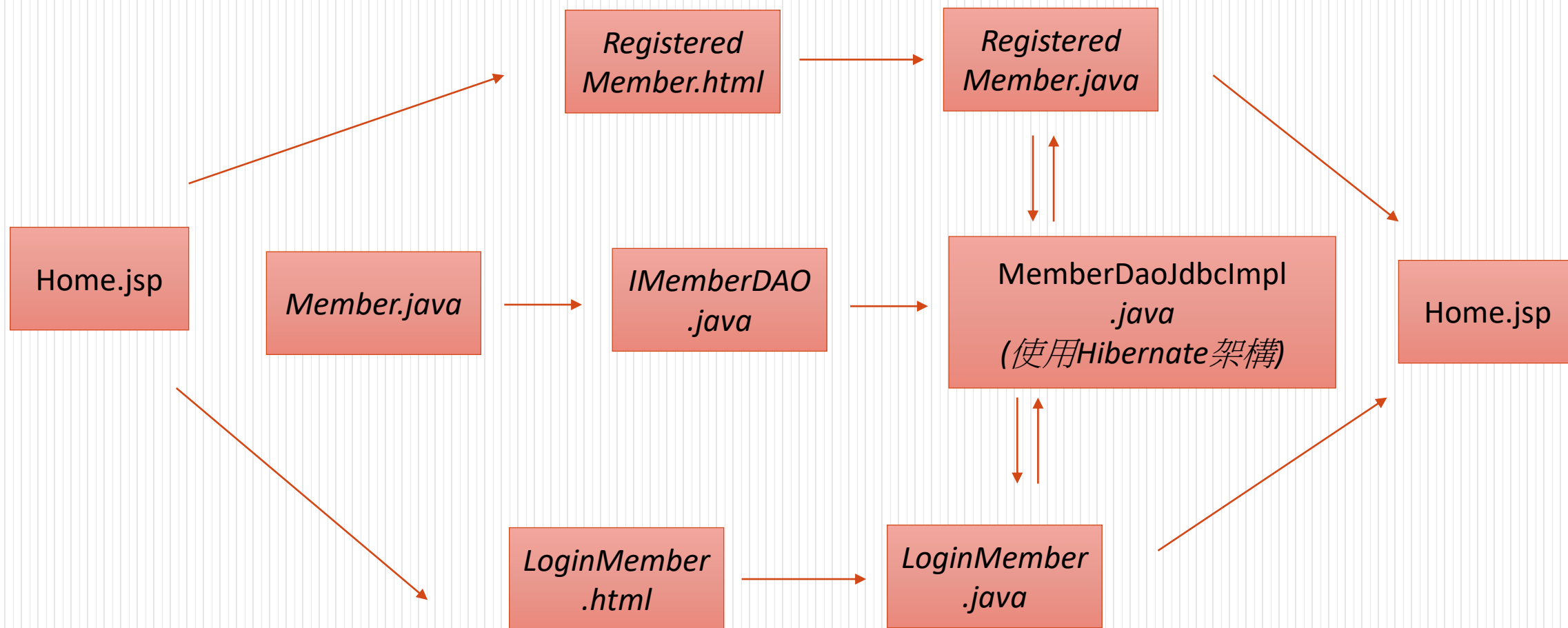
帳號:

密碼:

登入

忘記密碼

# 程式架構



# Hibernate程式片段—資料庫連線設定

- Context.xml

```
<Resource
    name="connectSqlServerJdbc/HibernateService"
    type="javax.sql.DataSource"
    auth="Container"
    username="sa"
    password="passw0rd"
    driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
    url="jdbc:sqlserver://localhost:1433;databaseName=team6project" />
```

# Hibernate程式片段—資料庫連線設定

- hibernate.cfg.xml

```
<property name="hibernate.connection.datasource">java:comp/env/connectSqlServerJdbc/HibernateService</property>
<property name="hibernate.current_session_context_class">thread</property>
<property name="hibernate.show_sql">true</property>
<property name="hibernate.format_sql">true</property>
<mapping class="com.eeit109team6.memberDao.Member" />
```



# Hibernate程式片段—SessionFactory建立

- HibernateUtil.java

```
package com.eeit109team6.memberDao;

import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory = createSessionFactory();

    private static SessionFactory createSessionFactory() {
        try {
            StandardServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder().configure().build();
            SessionFactory factory = new MetadataSources(serviceRegistry).buildMetadata().buildSessionFactory();
            return factory;
        } catch (Exception e) {
            e.printStackTrace();
            throw new ExceptionInInitializerError(e);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void closeFactory() {
        if (sessionFactory != null) {
            sessionFactory.close();
        }
    }
}
```

# Hibernate程式片段—當專案開啟連線資料庫

- SessionFactoryListner.java

```
package com.eeit109team6.servletlistener;

import javax.servlet.ServletContextEvent;

public class SessionFactoryListner implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        HibernateUtil.getSessionfactory();
        System.out.println("Session Factory is Created");
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        HibernateUtil.closeFactory();
        System.out.println("Session Factory is Closed");
    }
}
```

# Hibernate程式片段—處理Transaction

- OpenSessionViewFilter.java

```
package com.eeit109team6.servletfilter;

import java.io.IOException;

@WebFilter("/OpenSessionViewFactory")
public class OpenSessionViewFilter implements Filter {

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        SessionFactory sessionFactory = HibernateUtil.getSessionfactory();
        try {
            sessionFactory.getCurrentSession().beginTransaction();
            System.out.println("Transaction Begin");
            chain.doFilter(request, response);
            sessionFactory.getCurrentSession().getTransaction().commit();
            System.out.println("Transaction Commit");
        } catch (Exception e) {
            sessionFactory.getCurrentSession().getTransaction().rollback();
            System.out.println("Transaction Rollback");
            e.printStackTrace();
        } finally {
            sessionFactory.getCurrentSession().close();
            System.out.println("Transaction Closed");
        }
    }
}
```

# Hibernate程式片段—JavaBean

- Member.java

```
package com.eeit109team6.memberDao;

import javax.persistence.Column;

@Entity
@Table(name = "member")
public class Member {
    private int member_id;
    private String account ;
    private String password ;
    private String username ;
    private String idnumber ;
    private String sex ;
    private String birth ;
    private String registeredtime ;
    private int isactive;
    private String token;
    private String address ;
    @Column(name = "ACCOUNT")
    public String getAccount() {
        return account;
    }
    public void setAccount(String account) {
        this.account = account;
    }
}
```

# Hibernate程式片段—DAO

- IMemberDao.java

```
package com.eeit109team6.memberDao;

import java.sql.Connection;

public interface IMemberDao {

    public int add(Member m) throws SQLException;

    public void update(Member m) throws SQLException;

    public void delete(Member m) throws SQLException;

    public ArrayList<Member> fintAll() throws SQLException;

    public Member fintById(Member m) throws SQLException;

    public Member login(Member m) throws SQLException;

    public boolean openActive(Member m) throws SQLException;

    public void forgetPwd(Member m) throws SQLException;

    public void changePwd(Member m) throws SQLException;

    public boolean checkAccount(Member m) throws SQLException;

}
```

# Hibernate程式片段—DAOImpl

- MemberDaoJdbcImpl.java(節錄)

```
package com.eeit109team6.memberDao;

import java.sql.Connection;

public class MemberDaoJdbcImpl implements IMemberDao {

    private Connection conn;
    private Session session;
    private SessionFactory sessionFactory;

    public MemberDaoJdbcImpl(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    public SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    @Override
    public int add(Member m) throws SQLException {
        getSessionFactory().getCurrentSession().save(m);

        return m.getMember_id();
    }
}
```

# Hibernate程式片段—DAOImpl

- MemberDaoJdbcImpl.java(節錄)

```
@Override
public Member login(Member m) throws SQLException {
    System.out.println("account" + m.getAccount());
    System.out.println(m.getPassword());
    List<Member> memList = null;
    Query query = getSessionFactory().getCurrentSession()
        .createQuery("from Member where account = ?1 and password = ?2 and isactive = 1");
    query.setParameter(1, m.getAccount());
    query.setParameter(2, m.getPassword());
    memList = (List<Member>) query.getResultList();
    System.out.println("memList.size()" + memList.size());
    if (memList.size() != 0) {
        return memList.get(0);
    } else {
        return null;
    }
}
```

# Hibernate程式片段—DAOImpl

- MemberDaoJdbcImpl.java(節錄)

```
@Override
public Member findById(Member m) throws SQLException {

    List<Member> memList = null;
    Query query = getSessionFactory().getCurrentSession()
        .createQuery("from Member where member_id = ?1 and token = ?2");
    query.setParameter(1, m.getMember_id());
    query.setParameter(2, m.getToken());
    memList = (List<Member>) query.getResultList();

    if (memList.size() != 0) {
        memList.get(0).setIsActive(1);
        return memList.get(0);
    } else {
        return null;
    }
}
```



# Hibernate程式片段—DAOImpl

- MemberDaoJdbcImpl.java(節錄)

```
@Override
public boolean openActive(Member m) throws SQLException {

    List<Member> memList = null;
    Query query = getSessionFactory().getCurrentSession()
        .createQuery("from Member where member_id = ?1 and token = ?2");
    query.setParameter(1, m.getMember_id());
    query.setParameter(2, m.getToken());
    memList = (List<Member>) query.getResultList();

    if (memList.size() != 0) {
        memList.get(0).setIsActive(1);
        return true;
    } else {
        return false;
    }
}
```

# Hibernate程式片段—DAOImpl

- MemberDaoJdbcImpl.java(節錄)

```
@Override
public void forgetPwd(Member m) throws SQLException {

    List<Member> memList = null;
    Query query = getSessionFactory().getCurrentSession().createQuery("from Member where account = ?1");
    query.setParameter(1, m.getAccount());

    memList = (List<Member>) query.getResultList();

    if (memList.size() != 0) {
        memList.get(0).setToken(m.getToken());
    } else {
        System.out.println("找不到該帳號");
    }
}
```

# Hibernate程式片段—DAOImpl

- MemberDaoJdbcImpl.java(節錄)

```
@Override
public void changePwd(Member m) throws SQLException {
    List<Member> memList = null;
    Query query = getSessionFactory().getCurrentSession()
        .createQuery("from Member where account =?1 and token = ?2");
    query.setParameter(1, m.getAccount());
    query.setParameter(2, m.getToken());

    memList = (List<Member>) query.getResultList();

    if (memList.size() != 0) {
        memList.get(0).setPassword(m.getPassword());
    } else {
        System.out.println("找不到該帳號");
    }
}
```

# Hibernate程式片段—DAOImpl

- MemberDaoJdbcImpl.java(節錄)

```
@Override
public boolean checkAccount(Member m) throws SQLException {
    System.out.println("checkAccount");
    List<Member> memList = null;
    Query query = getSessionFactory().getCurrentSession().createQuery("from Member where account =?1");
//    System.out.println("m.getAccount()" + m.getAccount());
    query.setParameter(1, m.getAccount());
//    System.out.println("query.getSingleResult() ==="+query.getSingleResult());
    memList = (List<Member>) query.getResultList();

    if (memList.size() != 0) {
        return false;
    } else {
        return true;
    }
}
```

# Hibernate程式片段—會員註冊

- RegisteredMember.java

```
package com.eeit109team6.servletmember;

import java.io.IOException;

@WebServlet("/RegisteredMember")
public class RegisteredMember extends HttpServlet {
    private static final long serialVersionUID = 1L;
    @Resource(name = "jdbc/team6project")
    private DataSource ds;
    Connection conn;
    private SessionFactory sessionFactory;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        // =====取值=====
        String account = request.getParameter("account");
        String password = request.getParameter("password");
        String alladdress = request.getParameter("alladdress");
        String username = request.getParameter("username");
        String idnumber = request.getParameter("idnumber");
        String sex = request.getParameter("sex");
        String birth = request.getParameter("birth");
        // =====/取值=====

        System.out.println("sex = " + sex); // true
        System.out.println("(sex == null) = " + (sex == null)); // true
    }
}
```

# Hibernate程式片段—會員註冊

- RegisteredMember.java

```
// =====處理資料=====
if (account != "" && password != "" && alladdress != "" && username != "" && idnumber != "" && sex != null
    && birth != "") {

    // =====設定創建帳號時間=====
    Calendar rightNow = Calendar.getInstance();
    String registeredtime = rightNow.get(Calendar.YEAR) + "-" + (rightNow.get(Calendar.MONTH) + 1) + "-"
        + rightNow.get(Calendar.DATE) + " " + rightNow.get(Calendar.HOUR) + ":"
        + rightNow.get(Calendar.MINUTE) + ":" + rightNow.get(Calendar.SECOND);
    // =====/設定創建帳號時間=====

    // =====密碼加密=====
    int isactive = 0;
    String key = "MickeyKittyLouis";
    String password_AES = CipherUtils.encryptString(key, password).replaceAll("[\\pP\\p{Punct}]", "")
        .replace(" ", "");
    // =====/密碼加密=====
```

# Hibernate程式片段—會員註冊

- RegisteredMember.java

```
// =====設定member物件=====
Member mem = new Member();
mem.setAccount(account);
mem.setPassword(password_AES);
mem.setUsername(username);
mem.setIdnumber(idnumber);
mem.setAddress(alladdress);
mem.setSex(sex);
mem.setBirth(birth);
mem.setRegisteredtime(registeredtime);
mem.setIsactive(isactive);
// =====/設定member物件=====

// =====設定token=====
KeyGenerator keyGen;
try {
    keyGen = KeyGenerator.getInstance("AES");
    keyGen.init(256, new SecureRandom());
    SecretKey secretKey = keyGen.generateKey();
    byte[] iv = new byte[16];
    SecureRandom prng = new SecureRandom();
    prng.nextBytes(iv);
    Long math = Long.valueOf((long) (Math.random() * 999999999));
    String token_notformat = AES_CBC_PKCS5PADDING.Encrypt(secretKey, iv, math.toString());
    String token = token_notformat.replaceAll("[\\pP\\p{Punct}]", "").replace(" ", "");
    mem.setToken(token);
} catch (Exception e) {

    e.printStackTrace();
}
// =====/設定token=====
```

# Hibernate程式片段—會員註冊

- RegisteredMember.java

```
// =====寫進資料庫=====
sessionFactory = HibernateUtil.getSessionfactory();
// session = sessionFactory.openSession();
int memberId = 0;
try {
    IMemberDao MemDao = MemberDaoFactoery.createMember(sessionFactory);
    memberId = MemDao.add(mem);
    System.out.println("id:" + memberId);
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
// //=====寫進資料庫=====
```



# Hibernate程式片段—會員註冊

- RegisteredMember.java

```
String host = "smtp.gmail.com";
int port = 587;
final String Email = "tytw2000@gmail.com";
final String EmailPwd = "jtes5505"; // your password

Properties props = new Properties();
props.put("mail.smtp.host", host);
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.starttls.enable", "true");
props.put("mail.smtp.port", port);
Session session = Session.getInstance(props, new Authenticator() {
    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(Email, EmailPwd);
    }
});

try {

    Message message = new MimeMessage(session);
    message.setFrom(new InternetAddress(""));
    message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(account));
    message.setSubject("驗證信");
    message.setText("Wellcome To FootBook \n http://localhost:8090/EEIT109_35/CheckMember.do?id=" + memberId
        + "&token=" + mem.getToken());

    Transport transport = session.getTransport("smtp");
    transport.connect(host, port, Email, EmailPwd);

    Transport.send(message);

    System.out.println("寄送email結束.");
    request.setAttribute("msg", "請至您輸入的信箱收取驗證信開通帳號");
    RequestDispatcher rd = request.getRequestDispatcher("member/jump.jsp");
```

# Hibernate程式片段—會員註冊

- RegisteredMember.java

```
        rd.forward(request, response);
    } catch (MessagingException e) {
        throw new RuntimeException(e);
    }

} else {
    System.out.println("註冊 else");
    PrintWriter out = response.getWriter();
    out.write("<script>alert('你是不是想找麻煩!');history.go(-1);</script>");
    out.close();
} // =====/處理資料=====

}

}
```

# 會員註冊

- 紅色記得點選有無重複
- 之後照者欄位輸入資料
- 黑色部分註冊才會打開

申請帳號必填

帳號:  ☒ 格式符合

密碼:  ☒ 格式符合

(1.不可空白, 2.至少六個字包含英文字母, 數字, 特殊字元(!@#\$%^&\*))

姓名:  ☒ 格式符合

(1.不可空白, 2.至少兩個字以上, 3.必須為中文字)

身分證字號:  ☒ 格式符合

地址:

性別: ☒ 男 ☐ 女

生日:  ☒ 格式符合

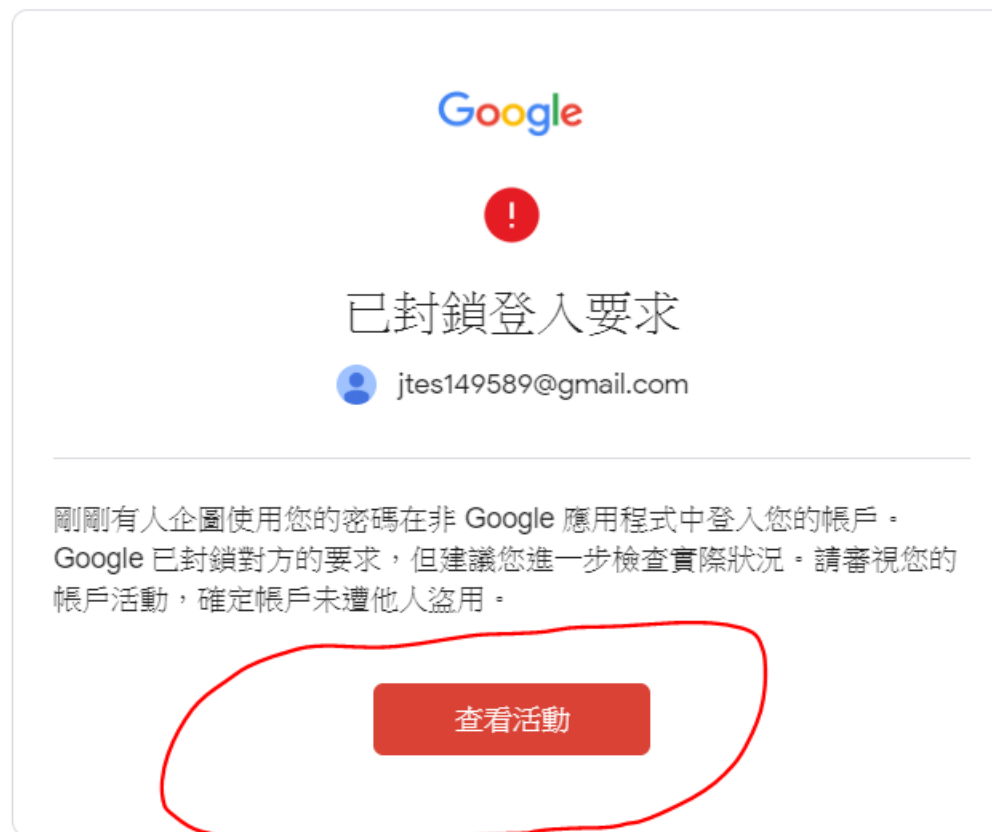
# Gmail 寄信失敗問題

Gmail 第一次使用應用程式寄信，要設定安全信，你會收到一封信。

**重大安全性警示 - 已封鎖登入要求** jtes149589@gmail.com 剛剛有人企圖使用您的密碼在非 Google

# Gmail 寄信失敗問題

信內容請點選[查看活動](#)



您的 Google 帳戶和服務有重大異動，系統特此發送這封電子郵件通知您。

© 2019 Google LLC, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

# Gmail 寄信失敗問題

請點選有印象


點選瞭解詳情


## ! 重大安全性快訊

系統已阻止一個可疑的應用程式存取您的帳戶

 jtes149589@gmail.com

Google 已拒絕某位使用者在非 Google 應用程式中登入您的帳戶。如果進行這項操作的不是您本人，表示有其他人知道您的密碼，建議您立即變更密碼。

 無法辨識的裝置

 1 分鐘前

 台灣附近  
36.228.132.235 (IP 位址) 

系統最近是否曾拒絕您登入 Google 帳戶？

☒ 有印象，是我本人

☐ 沒有，採取帳戶安全措施

## 已封鎖低安全性應用程式

您想使用的應用程式不符合 Google 的安全性標準，因此我們已封鎖該應用程式。

某些應用程式和裝置採用的登入技術安全性較低，將導致您的帳戶出現安全漏洞。建議您停用這類應用程式的存取權；當然，您也可以選擇啟用存取權，但請瞭解相關風險。如果您並未使用這項設定，Google 會自動將其關閉。

[瞭解詳情](#)

[更改回覆](#)

# Gmail 寄信失敗問題

## 點選低安全性應用程式存取權

### 低安全性應用程式和您的 Google 帳戶

如果有人使用不符合 Google 安全標準的應用程式或網站嘗試登入帳戶，我們可能會予以拒絕。低安全性應用程式可能會提高駭客入侵帳戶的機率，因此封鎖這類應用程式的登入活動可協助維護帳戶安全。

如果您的帳戶開啟了「低安全性應用程式存取權」



低安全性應用程式可能會導致帳戶出現更多安全漏洞，因此如果您並未使用這項設定，Google 會自動將其關閉。

如果您帳戶中的「低安全性應用程式存取權」設定仍然是開啟的狀態，我們建議您立即關閉該設定，並改用高安全性應用程式。

### 關閉「低安全性應用程式存取權」

為確保您的帳戶安全，我們建議您將這項設定一律設為關閉，並使用高安全性應用程式。

1. 前往 Google 帳戶中的「低安全性應用程式存取權」部分。您可能需要登入帳戶。
2. 關閉「允許低安全性應用程式」設定。

## 打開低安全性存取權

### ← 低安全性應用程式存取權

某些應用程式和裝置採用的登入技術安全性較低，將導致您的帳戶出現安全漏洞。建議您停用這類應用程式的存取權；當然，您也可以選擇啟用存取權，但請瞭解相關風險。如果您並未使用這項設定，Google 會自動關閉該權限。[瞭解詳情](#)

允許低安全性應用程式：已開啟



# Hibernate程式片段—會員登入

- LoginMember.java

```
package com.eeit109team6.servletmember;

import java.io.IOException;

@WebServlet("/LoginMember")
public class LoginMember extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private SessionFactory sessionFactory;
    private Session hbSession;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        System.out.println("login" + request.getParameter("login"));
        System.out.println("fergetpwd" + request.getParameter("fergetpwd"));
        if (request.getParameter("fergetpwd") != null) {
            RequestDispatcher rd = request.getRequestDispatcher("/member/forgetPWD.html");
            rd.forward(request, response);
        } else {

            request.setCharacterEncoding("UTF-8");
            response.setContentType("text/html;charset=UTF-8");
            String account = request.getParameter("account");
            String password = request.getParameter("password");

            String key = "MickeyKittyLouis";
            String password_AES = CipherUtils.encryptString(key, password).replaceAll("[\\pP\\p{Punct}]", "")
                .replace(" ", "");
```



# Hibernate程式片段—會員登入

- LoginMember.java

```
Member mem = new Member();
mem.setAccount(account);
mem.setPassword(password_AES);
HttpSession session = request.getSession();
IMemberDao MEMDao = null;

sessionFactory=HibernateUtil.getSessionfactory();

try {
    MEMDao = MemberDaoFactoery.createMember(sessionFactory);

    Member member = MEMDao.login(mem);
    // System.out.println("member.getAccount() = "+member.getAccount());
    // System.out.println(member.getAccount() != "");
    System.out.println("member != null = " + (member != null));
    if (member != null) {
        session.setAttribute("username", member.getUsername());
        session.setAttribute("token", member.getToken());
        session.setAttribute("account", member.getAccount());
        session.setAttribute("member_id", member.getMember_id());
        response.getWriter().write("<script>alert('歡迎光臨');</script>");

        request.setAttribute("msg", "歡迎光臨");
        RequestDispatcher rd = request.getRequestDispatcher("member/jump.jsp");
        rd.forward(request, response);
    } else {

        response.getWriter().write("<script>alert('帳號或密碼錯誤，或者未開通');history.go(-1);</script>");
    }
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

簡報結束，謝謝觀賞！