

JPA

- ORM ໃຫຍ່ data ມີເປົ້າ obj (JPA ຕໍ່ ORM / JDBC ຕະຫຼອງຮານທຶນຈຳ)
- ສ່ວນ persistence ທີ່ນາ → EntityManagerFactory → EntityManager (ມີ Method C R U D [ສ່ວນ, ອຳນວຍ, update, ລູບ])
↳ query (ສ່ວນບໍລິໂພຂອງເຂົ້າ)

ADD ENTITY RELATIONSHIP (Office + Employee)

```
@OneToMany(mappedBy = "officeCode", fetch = FetchType.EAGER)
private List<Employee> EmployeeList;

public List<Employee> getEmployeeList() {
    return EmployeeList;
}

public void setEmployeeList(List<Employee> employeeList) {
    EmployeeList = employeeList;
}
```

```
public class EntityManagerService {
    // ໃຫຍ່ EntityManager
    private final static EntityManagerFactory emf = Persistence.createEntityManagerFactory("classic-model");

    public static EntityManager getEntityManager() {
        return emf.createEntityManager();
    }
}
```

FetchType.LAZY = (default) ກໍາໄມ່ getEmployeeList() ກໍ່ຈະໄດ້ເຫັນລົມ

FetchType.EAGER = ຈະດີ Employee ນາຟ້າເຂົ້າ (ສ່ວນຈຳກັດໄລ້ຊື່)

ENTITY MANAGER (CRUD)

```
public class OfficeRepository {
    private EntityManager getEntityManager() {
        return EntityManagerService.getEntityManager();
    }

    // R
    public Office find(String officeCode) {
        EntityManager entityManager = getEntityManager();
        Office office = entityManager.find(Office.class, officeCode);
        entityManager.close(); // ຖ້າອາຍາປີດຕົວກໍາທັນທີໃຫ້ Fetch ເປັນ EAGER
        return office;
    }

    // C persist() = ຄືອກຮັບທີ່ຂໍ້ມູນ ທີ່ຈະ officeCode ພ້າມຂ້າ
    public boolean save(Office office) {
        try {
            EntityManager entityManager = getEntityManager();
            entityManager.getTransaction().begin();
            entityManager.persist(office);
            entityManager.getTransaction().commit();
            entityManager.close();
        } catch (Exception e) {
            return false;
        }
        return true;
    }
}
```

```
// U merge() = update ເຊັ່ນ obj ດ້ວຍໄດ້ກີ່ກົດ return false
public boolean update(Office office) {
    try {
        EntityManager entityManager = getEntityManager();
        entityManager.getTransaction().begin();
        entityManager.merge(office);
        entityManager.getTransaction().commit();
        entityManager.close();
    } catch (Exception e) {
        return false;
    }
    return true;
}

// D ດັ່ງກ່າວ find ແລ້ວໄວ້ຕື່ອງເຈົ້າມານີ້ໃນ session ຊໍ່ນໍ້າ ດ້ວຍກີ່ກົດ merge ກ່ອນ ຂ້າມນເຄື່ອງກິດຂໍ້ມູນ office ເຊັ່ນ
public boolean delete(Office office) {
    return delete(office.getId());
}

public boolean delete(String officeCode) {
try {
    EntityManager entityManager = getEntityManager();
    entityManager.getTransaction().begin();
    Office office = find(officeCode);
    entityManager.remove(entityManager.contains(office) ? office : entityManager.merge(office));
    entityManager.getTransaction().commit();
    entityManager.close();
} catch (Exception e) {
    return false;
}
    return true;
}
```

R (READ) + U (UPDATE)

```
public class Main {
    public static void main(String[] args) {
        // ຢັງ office ທີ່ນີ້ id = 1
        OfficeRepository officeRepository = new OfficeRepository();
        Office office1 = officeRepository.find(officeCode: "1"); // R
        // equalsIgnoreCase ໄນສັນຕິເລັກຕົວໄທ້
        if(office1.getCity().equalsIgnoreCase(anotherString: "Bangkok")) {
            office1.setCity("Vientiane");
        } else {
            office1.setCity("Bangkok");
        }
        officeRepository.update(office1); // U
        System.out.println(office1);
    }
}
```

```
// R ຢັງ office ແລ້ວ id = 4 ມີຂໍ້ມູນ
Office office2 = officeRepository.find(officeCode: "4");
System.out.println("-----");
System.out.println("-----");
// ບັນລຸແນວໃນ forEach ມີລົງທຶນທີ່
office2.getEmployeeList().forEach(employee -> {
    // FORMAT : %5d ປິບຕື່ນີ້ 5 ດົກຕື່ນີ້ ແລ້ວໃນ decimal ມີນີ້ໄດ້ຕຽບກັນ
    // FORMAT : %-12s %-15s ທີ່ໄດ້ນຳໃຫ້ຂໍ້ມູນກີ່ນີ້ s = String
    // printf ຮອ່ມມູນຄົງ format
    System.out.printf("%5d %-12s %-15s %s\n",
        employee.getId(), employee.getFirstName(), employee.getLastName(), employee.getEmail());
});

```

4 : Paris
1102 Gerard Bondur gbondur@classicmodelcars.com
1337 Loui Bondur lbondur@classicmodelcars.com
1378 Gerard Hernandez ghernande@classicmodelcars.com
1401 Pamela Castillo pcastillo@classicmodelcars.com
1702 Martin Gerard mgerard@classicmodelcars.com

C (CREATE)

```
addNewOffice(officeRepository); // C
```

```
private static void addNewOffice(OfficeRepository officeRepository) {
    Scanner sc = new Scanner(System.in); // ຂັ້ນຂ່າຍການເປັນພົນ
    System.out.println("Enter Office Code : ");
    String officeCode = sc.nextLine();
    System.out.println("Enter city : ");
    String city = sc.nextLine();

    Office office = new Office();
    office.setId(officeCode);
    office.setCity(city);
    office.setAddressLine1("124 Pracha-utit");
    office.setPhone("0954796673");
    office.setPostalCode("10140");
    office.setTerritory("NA");
    office.setCountry("TH");
    officeRepository.save(office);
}
```

```
Enter Office Code :
10
Enter city :
Bangkok
```

OfficeCode	city	phone	addressLine1
1	Vientiane	+65 219 4782	100 Market Street
2	Bangkok	0954796673	124 Pracha-utit
3	Boston	+1 215 837 8825	1550 Court Place
4	NYC	+1 212 655 3888	523 East 3rd Street
5	Tokyo	+81 33 526 5888	4-1 Kita-cho
6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue
7	London	+44 20 7877 2841	25 Old Broad Street

D (DELETE)

```
officeRepository.delete(officeCode: "10"); // D
```

OfficeCode	city	phone	addressLine1
1	Vientiane	+65 219 4782	100 Market Street
2	Boston	0954796673	124 Pracha-utit
3	Boston	+1 215 837 8825	1550 Court Place
4	NYC	+1 212 655 3888	523 East 3rd Street
5	Tokyo	+81 33 526 5888	4-1 Kita-cho
6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue
7	London	+44 20 7877 2841	25 Old Broad Street

JPQL (QUERY)

- flexible / object-oriented នូវការ SQL
- persistence engine ॥ផ្សាយពីរំពូល String
- ឧ: Query តួងនៃ EntityManager នៅ

parameter dynamic សំគាល់ក្នុង (Array)

e.g. findAll(int... pageInfos)

QUERY INSTANCES ARE OBTAINED USING

1. EntityManager.createNamedQuery (static) = បែបណ៍ក្រុវេយ្យទៅនា
2. EntityManager.createQuery (dynamic) = ក្រសួង query ថ្មីបានដោល
3. EntityManager.createNativeQuery (native) = query ភាសា sql នូវខ្លួន

QUERY API (METHOD)

- getResultList() ទៅបង្ហរក query នៃវានៃបាន list
- getSingleResult() ទៅបង្ហរកតុលាប័ណ្ណ (បុណ្យទេ)
- executeUpdate() query ប្រអប់ insert, update
- setFirstResult() ពិនិត្យការរំលែកពី record ទី n } (ឡើង - ឡាតាំង) Paging
- setMaxResult() លោកស្រាវក្រុម record
- setParameter() ទិន្នន័យ / query តាម parameter ជាលិខ្លួន
- setFlushMode()

STATIC (NAMED) QUERY

- បែបណ៍ក្រុវេយ្យ Entity @ NamedQuery
- ក្រសួងក្នុង multiple @ Named Queries

```
@NamedQueries({
    @NamedQuery(name = "Product.FindAll", query = "SELECT p FROM Product p"),
    @NamedQuery(name = "Product.count", query = "SELECT count(p) FROM Product p")
})

public class ProductRepository {
    private static final int PAGE_SIZE = 10;

    private EntityManager entityManager() { return EntityManagerService.getEntityManager(); }

    public List<Product> findAll(int... pageInfos) {
        EntityManager entityManager = entityManager();
        Query query = entityManager.createNamedQuery("Product.FindAll");
        query.setFirstResult(pageInfos.length > 0 ? pageInfos[0] : 0);
        query.setMaxResults(pageInfos.length > 1 ? pageInfos[1] : PAGE_SIZE);
        List<Product> productList = query.getResultList();
        entityManager.close();
        return productList;
    }

    public int countAll() {
        EntityManager entityManager = entityManager();
        int number = ((Number) entityManager.createNamedQuery("Product.count").getSingleResult()).intValue();
        entityManager.close();
        return number;
    }

    public Product find(String productCode) {
        EntityManager entityManager = entityManager();
        Product product = entityManager.find(Product.class, productCode);
        entityManager.close();
        return product;
    }
}
```

getResultSet()

```
// QUERY -----
private static final String FIND_ALL = "SELECT o FROM Office o";
private static final String FIND_BY_COUNTRY = "SELECT o FROM Office o WHERE o.country like :country";

// query.setParameter តាមគឺចិត្តអមិដ % និងតាមតួរការ country ដើម្បីបង្កើតគឺតាមតួរការ
public List<Office> findByCountry(String findCountry) {
    EntityManager entityManager = getEntityManager();
    Query query = entityManager.createQuery(FIND_BY_COUNTRY);
    query.setParameter("country", " " + findCountry + "%");
    List<Office> officeList = query.getResultList();
    entityManager.close();
    return officeList;
}

// createQuery(FIND_ALL).getResultList() ត្រូវតាម office ដើម្បីបង្កើតគឺតាមតួរការ
public List<Office> findAll() {
    EntityManager entityManager = getEntityManager();
    List<Office> officeList = entityManager.createQuery(FIND_ALL).getResultList();
    entityManager.close();
    return officeList;
}
```

```
// QUERY : បង្ហរកតុលាប័ណ្ណ
for(Office office3 : officeRepository.findAll()) {
    System.out.println(office3.getId() + " : " + office3.getCity());
    System.out.println("-----");
    office3.getEmployeeList().forEach(employee -> {
        System.out.printf("%d %12s %15s %s\n",
            employee.getId(), employee.getFirstName(), employee.getLastName(), employee.getEmail());
    });
}

// QUERY : បង្ហរក country ដើម្បីបង្កើតគឺតាមតួរការ U
System.out.println("----- COUNTRY START WITH U -----");
for(Office office4 : officeRepository.findByCountry("U")) {
    System.out.println(office4.getId() + " : " + office4.getCity() + "-" + office4.getCountry());
}
```

1 : Vientiane	Murphy	dmurphy@classicmodelcars.com
1056 Mary	Patterson	mpatterso@classicmodelcars.com
1076 Jeff	Firrelli	jfirrelli@classicmodelcars.com
1143 Anthony	Bow	abow@classicmodelcars.com
1165 Leslie	Jennings	ljennings@classicmodelcars.com
1166 Leslie	Thompson	lthompson@classicmodelcars.com
2 : Boston		
1188 Julie	Firrelli	jfirrelli@classicmodelcars.com
1216 Steve	Patterson	spatterson@classicmodelcars.com
3 : NYC		
1286 Foon Yue	Tseng	ftseng@classicmodelcars.com
1323 George	Vanau	gvanau@classicmodelcars.com
4 : Paris		
1102 Gerard	Bondur	gbondur@classicmodelcars.com
1337 Louis	Bondur	lbondur@classicmodelcars.com
1370 Hernandez	Hernandez	ghernandez@classicmodelcars.com
1401 Pamela	Castillo	pcastillo@classicmodelcars.com
1702 Martin	Gerard	mgrard@classicmodelcars.com
5 : Tokyo		
1621 Mami	Nishi	mnishi@classicmodelcars.com
1625 Yoshimi	Kato	ykato@classicmodelcars.com

----- COUNTRY START WITH U -----
1 : Vientiane-USA
2 : Boston-USA
3 : NYC-USA
7 : London-UK

MVC USING JPA WITH AJAX

AJAX = Asynchronous JavaScript (เรบกแบบปาง syn กัน ไม่ต้อง request) ทำใช้ XML (object: XMLHttpRequest)

มีการ send + receive

/ อุปกรณ์ server

FEATURES : ส่ง requests แบบไม่ต้อง reload / receive + work with data from server

XMLHttpRequest

- สร้าง XMLHttpRequest

```
let xhr = new XMLHttpRequest();
```

- config ว่าจะเรียกด้วย method อะไร (GET, POST) / URL request ที่ต้อง / optional)

```
xhr.open( method, url, [async, user, password] )
```

L default = true ถ้า set ให้เป็น false ก็ได้ but ให้ร้องขอ API และท้าว่าใจไม่ได้ต้องส่งข้อมูลกลับมาเก็บ

- 보내ท่วน ต่อส่วน body ไปด้วย

```
xhr.send([body])
```

- Listen ลักษณะ response มาซึ่ง

Load = เมื่อ request ไป server สำเร็จ (HTTP status 400 or 500)

Error = ผิดพลาด

Progress = กำลังท่วงงาน (state)

AJAX (Asynchronous JavaScript And XML) เป็นเพียงวิธีการใช้ JavaScript เพื่อดึงข้อมูลจากเซิร์ฟเวอร์โดยไม่ต้องโหลดเริ่มใช้ XML เป็นรูปแบบข้อมูลที่คล้ายกับ HTML เมื่อใช้ AJAX เราสามารถขอข้อมูลบางส่วนในพื้นหลัง แบบอะซิงโคครันส์ด้วย JavaScript และแสดงต่อผู้ใช้เมื่อเราได้รับ เราไม่จำเป็นต้องโหลดเข้าหน้าอีกต่อไป

อ่านแล้ว
13:40

<script>

```
// XMLHttpRequest
function loadOffice(officeCode) {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("body-content").innerHTML = xhttp.responseText;
  }
  xhttp.open("GET", "office-list?officeCode="+officeCode);
  xhttp.send();
}

function loadProduct(page, pageSize=document.getElementById("itemsPage").value) {
  //alert("page: "+ page + ", size: "+ pageSize)
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("body-content").innerHTML = xhttp.responseText;
  }
  xhttp.open("GET", "product-list?page="+page+"&pageSize="+pageSize);
  xhttp.send();
}
```

(navbar)

```
<%-- สำคัญเป็นการเรียกตัว Function ใน JavaScript มาทำงาน--%>
<li class="nav-item">
  <a class="nav-link" href="javascript:loadOffice()>Office</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="javascript:loadProduct(1,15)">Product</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="javascript:void(0)">Order History</a>
</li>
```

สำหรับแสดงผล office

```
<div class="row">
  <c:forEach items="${offices}" var="office">
    <div class="col-2 border border-secondary p-2 m-2 div-link"
        ${office.id == selectedOffice.id ? 'bg-warning' : ''}>
      <div>
        ${office.city}, ${office.country}
      </div>
      <div> ${office.phone}</div>
    </div>
  </c:forEach>
</div>
<br>
```

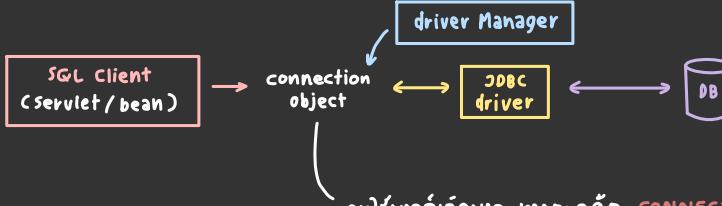
> Bg แหล่งที่มาล้วนๆ

Classic Model Offices ::

| | | | | |
|--------------------------------------|--------------------------------|-----------------------------|----------------------------------|--------------|
| Vientiane, USA
+1 650 219 4782 | Boston, USA
+1 216 837 0825 | NYC, USA
+33 14 723 4404 | Paris, France
+81 33 224 5000 | Tokyo, Japan |
| Sydney, Australia
+61 2 9264 2451 | London, UK
+44 20 7877 2041 | | | |

CONNECTION POOLING

DB CONNECTION VIA JDBC CALLS

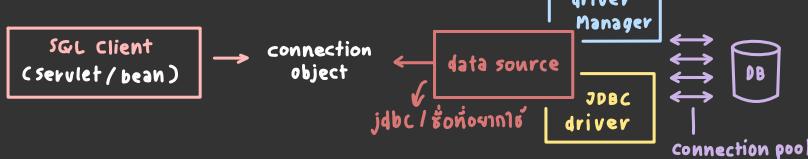


1. load JDBC driver
2. connection obj
3. ส่ง SQL command หรือ statement และ query (Queries / Update)
4. close

อนิจฉากรรมก็จะมี ทางบันได CONNECTION POOLING

- connection ใช้แล้วจะไม่ถูก reuse มากับคนอื่นที่
- performance + scalability
 - └ กําหนดขนาด (min-max)

DB CONNECTION USING DATASOURCE



- * **data source** จะสร้าง connection ตามที่ต้องการ ไม่ว่า ส่วนต่อไปนี้จะเป็น connection ใดก็ตามที่ต้องการสร้าง connection ในที่นั้น
- But!** ถ้าเกิด connection บ่อกับ data source จะสร้างใหม่ ไม่ใช่เรื่องของ connection ที่อยู่ใน connection pool
- ต้องก่อตัว dataSource ต่างๆ ผ่าน JNDI

SETTING UP A DATASOURCE

- Web app server + JDBC will create and manage connection pool ส่วนที่ต้อง set คือ
 - dataSource name
 - Actual driver + db name
 - Min + Max size of the pool
 - Various timeouts
- class ที่สามารถทำงานกับ data source ได้
 - └ ConnectionPoolDataSource
 - └ PooledConnection
 - └ ConnectionEventListener
 - └ ConnectionEvent

SESSION STRATEGIES

- Client: cookies, Hidden Fields, URL Rewriting
- Server: HTTP session, content Based Routing, Store state in a db, JWT (JSON WEB TOKEN)

COOKIES (name + value)

- Browser ต้อง cookies เป็นกับในรูปแบบ text (1 ค่าต่อคัน)
- server สร้างแล้วใส่ response นั้น เมื่อ Browser รับมาจะไปเซ็ต Header ว่ามี cookies ซึ่งก็จะถูกเก็บในคัน client (น่องๆ)
 - └ ถ้า request ไป server ที่มีคัน cookies นั้น add ให้ request ที่ส่งกลับ server
- server ใส่ cookies ไว้ใน Header response → browser ใจไปเก็บไว้แล้วส่งกลับมา (request)
- ข้อดีของการใช้ cookie / ห้ามเก็บข้อมูลสำคัญ!

transient เก็บเฉพาะหน้าเว็บชั่วคราว
 persistence ถ้าหน้าเว็บหายไป
 |
 |เก็บบนไฟล์

COOKIES API

- สร้าง cookie(String name, String value) [อยู่ในตัว server เป็น obj]
- 送去 Browser HttpServletResponse.addCookie(Cookie aCookie) [จัดเก็บลง Browser]
- Retrieving cookies HttpServletRequest.getCookies() [request ไป server เป็น Array วนลูป]
- Retrieving value cookies aCookie.getName()
- Retrieving value cookies aCookie.getValue()
- changing cookie value aCookie.setValue(String) [set จับ obj เมื่อเปลี่ยนแล้วต้องลง response ใหม่]

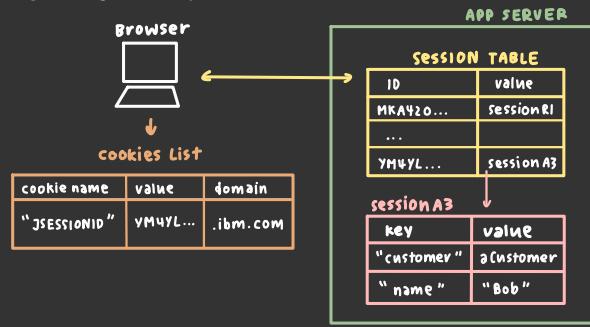
COOKIE APPLICABILITY

- กําหนดวันเดือนปี `setMaxAge(int expiryInSeconds)` [default วันเดือนปี -1 (transient)]
- สามารถ `setDomain(String)`, `setPath(String)`

អរ COOKIES ក្នុង HTTP SESSION

- `SESSIONID` — (String 32 គឺ) ដែលបានរំពោងនៃ Browser 1 នៅ ទីតាំងទាំងអស់ ក្នុងឱ្យបាន `transient` (ប៉ុណ្ណោះ browser = 1 នៅ)
 - នៅពេល cookies នៃ tracking នៃ session នឹងបានរំពោង

SESSIONS AT RUN TIME



```
// ทำส่วนของ COOKIES
Cookie ck = new Cookie( name: "lastpage", value: "office-list");
ck.setMaxAge(7*24*60*60);
response.addCookie(ck);

// ทำส่วนของ COOKIES
Cookie ck = new Cookie( name: "lastpage", value: "product-list");
ck.setMaxAge(7*24*60*60);
response.addCookie(ck);
```

* จากภาพพองซึ่งเป็นมีกุชช่าตัวที่น้ำ product-list

| Request Cookies | ■ show filtered out request cookies | | | | | | | | |
|------------------|-------------------------------------|-------|------|--------------|------|--------|-------|-------|------|
| Name | Value | D... | P... | Expires /... | S... | Secure | Sa... | Sa... | P... |
| JSESSIONID | aee2d43802b2e08d19996... | ... | P... | Session | 3... | ✓ | | | |
| lastpage | product.list | lo... | / | 2021-12-0... | 2... | | | | M... |
| | product.list | lo... | / | 2021-12-3... | 2... | | | | M... |
| cart | %25B1%7B%25product%22,%... | lo... | / | 2021-12-3... | 1... | | | | M... |
| JSESSIONID | aee4704845a0cd07831781... | lo... | / | Session | 3... | ✓ | | | M... |
| treeForm-tree-hi | treeForm-tree:applications | lo... | / | Session | 4... | | | | M... |

| Name | Value | D... | P... | Expires / ... | S... | H... | Secure | Sa... | Sa... | P... |
|------|-------|------|------|---------------|------|------|--------|-------|-------|------|
|------|-------|------|------|---------------|------|------|--------|-------|-------|------|

SERVLET (WEB) FILTERING (រាយករ config ទីក្រុងការបង្កាន់អំពីការបង្កាន់ទៅមិនមែន filter)

FILTERS : - เป็น java class ที่ reuse ได้

- filter HTTP requests, responses, header
 - configured ໃນរូបិយ្យ chains (នៅដែល filter នាមួយទេ)
 - ធ្វើការជារូបិយ្យ web resource នៃការពិនិត្យពីរុញពីរុច filter នឹង (ក្នុងមុន និងក្នុងក្រោម)

unction - process request (សំណងទិន្នន័យ) ក្នុងទិន្នន័យ resource
 - process response (សំណងទិន្នន័យ) អវត្ដនាករទិន្នន័យ resource
 - Modify response / request
 - សំគាល់ការពិនិត្យ (chain)
 - Block (បន្ទាត់ពិនិត្យ)

TYPICAL USE (ឧបតម្យ) – Authentication សិក្សាអនុញ្ញាត

- Logging and auditing เช่น log เอกซ์เพรสชันไปที่ audit
 - data compression ที่มาที่ -oon
 - Encryption การเข้ารหัส
 - XML ไปจัดองค์กร化 XML

FILTER CHAIN

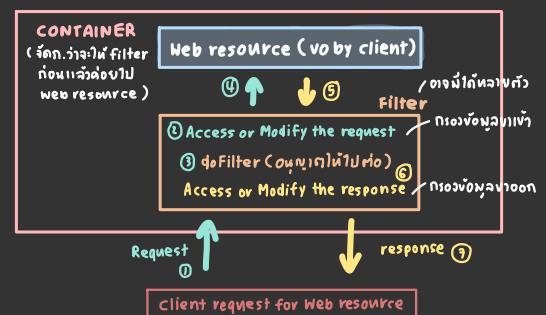
- FilterChain ลักษณะของ filter ที่ต้องผ่าน + ผู้ resource 0: ไรบ้าง (เอก鞍ใช้หัวงานได้)

- នានា web container ឲ្យលើ doFilter() ទាំង ៣ រូបមាន parameters ៣ នូយៗ - Request of type ServletRequest

doFilter() nested calls



- resource FILTER PROCESSING FLOW



- Request of type `ServletRequest`
 - Request of type `ServletResponse`

IMPLEMENTING (තුරා) FILTER

- สร้าง class
 - implement methods `init()` เริ่มต้นของ component เดิมที่ web container new filter
`doFilter()` (ทำงานหลัก)
 - `destroy()` เริ่มต้นที่จะ unload
 - ยังคงโดยใช้ annotation > ที่ filter's chaining configuration
④ Webfilter

Oscar glassfish 10

```
_ThreadName=http-listener-1(2) [timeMillis:  
?f;add-to-cart executed duration is 9 ms]  
    ↴  
    ↴  
_ThreadName=http-listener-1(3) [timeMillis:  
?f;add-to-cart executed duration is 3 ms]  
  
_ThreadName=http-listener-1(1) [timeMillis:  
?f;add-to-cart executed duration is 4 ms]  
  
_ThreadName=http-listener-1(4) [timeMillis:  
?f;add-to-cart executed duration is 5 ms]  
  
_ThreadName=http-listener-1(5) [timeMillis:  
?f;ViewCart.jsp executed duration is 358 ms]
```

```
    // AuthenticationFilter.java x
    ...
    // servletNames დანართულია filter #შემდეგ
    @WebFilter(filterName = "AuthenticationFilter",
        servletNames = {"OfficeEmployeeListServlet"})
    }

    public class Authenticationfilter implements Filter {
        private FilterConfig config;

        public void init(FilterConfig config) throws ServletException {
            this.config = config;
        }

        @Override
        public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws ServletException, IOException {
            HttpSession session = ((HttpServletRequest) request).getSession();
            if (session == null || session.getAttribute("user") == null) {
                ((HttpServletResponse) response).sendError(HttpServletResponse.SC_UNAUTHORIZED, "Please login.");
                return;
            } else {
                chain.doFilter(request, response);
            }
        }

        public void destroy() {
        }
    }
}
```

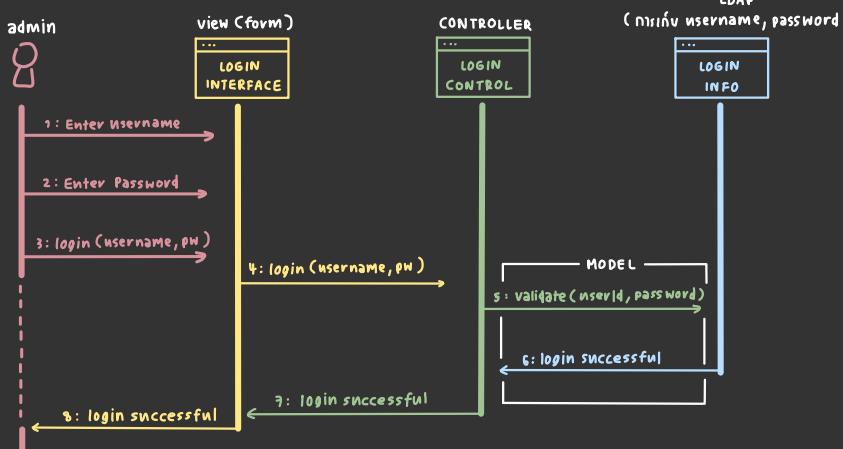
```
  @WebFilter(filterName = "LoggerFilter", urlPatterns = "/*")
public class LoggerFilter implements Filter {
    private FilterConfig config;

    public void init(FilterConfig config) throws ServletException {
        this.config = config;
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain
        long before = System.currentTimeMillis();
        chain.doFilter(request, response);
        long after = System.currentTimeMillis();
        String msg = ((HttpServletRequest) request).getServletPath() + " executed durat
        config.getServletContext().log(msg);
    }

    public void destroy() {
    }
}
```

AUTHENTICATION



bcrypt-generator.com

Encrypt

Encrypt some text. The result shown will be a Bcrypt encrypted hash.

```
$2a$10$QQotthd3WFernlPkYfITkckmnG9vElIdLM4k0msmPn2H/Grlfj6
```

| | |
|--------|---------|
| 1234 | Encrypt |
| Rounds | |
| - 10 + | |

การเก็บ PASSWORD (use Bcrypt algorithm) / ไม่เกิน 100 ตัวอักษร

`\$2a$10$N9qo8uLOickgx2ZMRZoMyeljZAsgcfI7p92ldGxad68LJZdL17lhWy
Alg Cost Salt Hash`

- Alg hash algorithm identifier
- cost ตั้ง hash ซ้ำกี่รอบ (ช่วงเบื้องต้น)
- salt random (ลักษณะของ password แรกทัน)
- Hash

```
@NamedQueries({
    @NamedQuery(name = "FIND_USER",
        query = "SELECT c FROM Customer c WHERE concat(trim(c.contactFirstName), ' ', trim(c.contactLastName)) = :user_account")
})
```

```
AuthenticationServlet.java ×

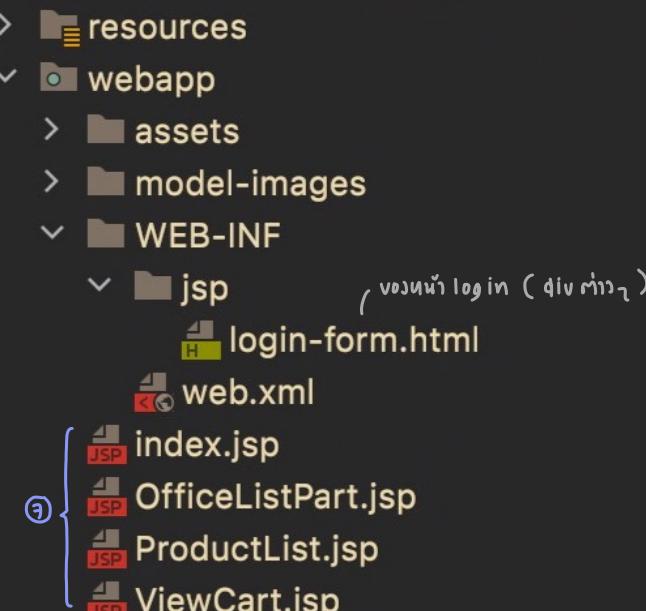
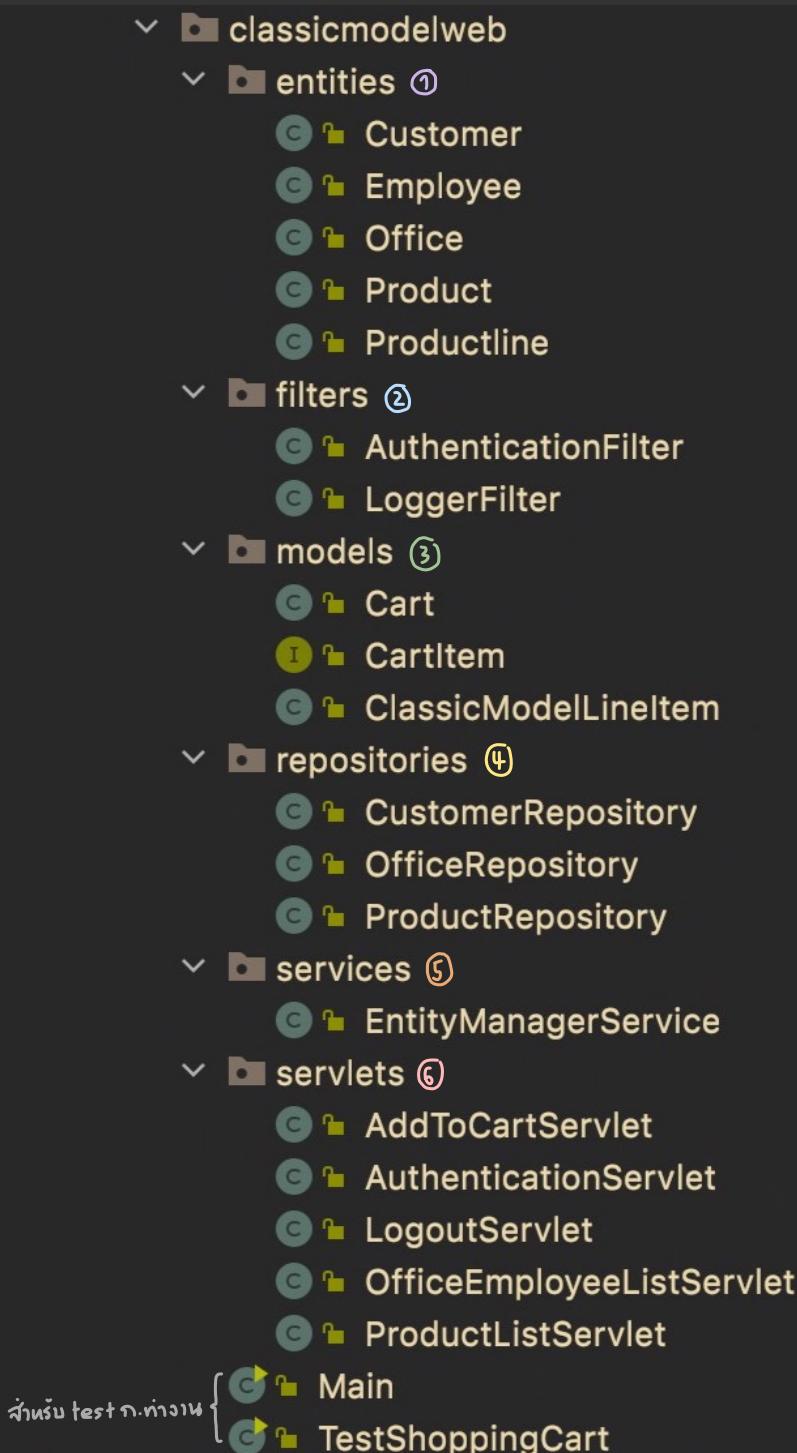
@WebServlet(name = "AuthenticationServlet", value = "/login")
public class AuthenticationServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doPost(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String password = request.getParameter("password");
        String userName = request.getParameter("userName");
        if (userName == null || userName.trim().length() == 0) {
            // မည်မှတ်မှု User သို့ ERROR 400
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }
        System.out.println(password);
        System.out.println(userName);
        CustomerRepository customerRepository = new CustomerRepository();
        Customer customer = customerRepository.findByName(userName);
        if (customer == null) {
            // ဘုံး User မှတ်လွှာသွား သို့ ERROR 403
            response.sendError(HttpServletResponse.SC_FORBIDDEN);
        } else {
            BCrypt.Result result = BCrypt.verifyer().verify(password.toCharArray(), customer.getPassword());
            if (!result.verified || !result.validFormat()) {
                // ဘုံး password မှတ်နေသွား သို့ ERROR 401
                response.sendError(HttpServletResponse.SC_UNAUTHORIZED);
            } else {
                // အောင်မြတ်
                request.getSession().setAttribute("user", customer);
            }
        }
    }
}
```

201 test

```
CustomerRepository customerRepository = new CustomerRepository();
Customer customer = customerRepository.findByLastName("Jean King");
System.out.println(customer);
String password = "1234";
BCrypt.Result result = BCrypt.verifyer().verify(password.toCharArray(), customer.getPassword());
System.out.println("Password is " + (result.verified?"Match":"NOT Match"));
Customer customer123, contactLastName="King", contactFirstName="Jean", password="$2a$18$00tthdMevLkxyfIkxcm6jy9Ejd.LT4KksmsPn2hGrfn6";
passwordHashed123Match(result);
```

โครงสร้างการทำงานของ classicmodelweb (ผู้อ่านต้น หน้าไปบล็อก :-;)



- ① entity = 101 ไฟล์ผูกันเป็น class
 - Customer.java มี @NamedQueries "FIND_USER"
 - Office.java มีการถึง Employee มาก
 - Product.java มี @NamedQueries "Product.FindAll"
 - └ "Product.count"
- ② filters = 10 ไฟล์ซึ่งก่อตัวใน user ไปใช้ web resource 1 ตัว
 - AuthenticationFilter.java ส่วนรับกันหน้า
 - officeEmployeeListServlet ว่าต้อง log in ก่อนถึงเข้าได้
 - LoggerFilter.java จัดเวลาในการที่ execute ของทุกอย่าง
- ③ models = 3 วิธี method จัดการ item
 - cart.java 101 ↘ ไม่ต่อ interface จัดการทั้งหมด
 - cartItem.java interface
 - classModelLineItem.java จัดการการเพิ่มสินค้า, ด้านวง
- ④ Repository = เก็บงำนพูดการที่ข้อมูลเก็บงำนฐานข้อมูลผ่าน CRUD, Query
 - CustomerRepository.java มี findBy Name ตาม query FIND_USER จาก Customer.java
 - OfficeRepository.java มีการใช้ CRUD (save, find, update, delete) + findAll
 - ProductRepository.java มี findAll และ CountAll ที่ใช้ Product.FindAll และ Product.count ตามมาด้วยจาก Product.java และ 1 ตัว R (find)
- ⑤ EntityManagerService.java = สร้าง EntityManager นำไปใน repository
- ⑥ Servlets = ควบคุมการรับส่งข้อมูลกับ server
 - AddToCartServlet.java value เป็น add-to-cart จัดการการเพิ่มของเข้าตะกร้า
 - AuthenticationServlet.java value เป็น login มีการรับ parameter username, password จากฟอร์มและกรณีผิด error หรือไม่
 - LogoutServlet.java value เป็น logout ส่านรับ session ที่มี log in อยู่
 - OfficeEmployeeListServlet.java value เป็น office-list จัดการหน้าเว็บชื่อ office มีการเก็บ cookies
 - ProductListServlet.java value เป็น product-list จัดการหน้าเว็บชื่อ product มีการเก็บ cookies
- ⑦ ไฟล์ ต่างๆ = ทุกหน้าที่ในส่วนการแสดงผล
 - index.jsp และหน้าแรก 1 ตัว XMLHTTP เรียกว่า jsp, servlet จัดการรูป navigation กดเพื่อให้ function ต่างๆ
 - OfficeListPart.jsp จัดการก.แสดงผล office
 - ProductList.jsp จัดการ.แสดงผล product / ณ. ห้องต่อหน้า
 - ViewCart.jsp จัดการการแสดงผล cart