CS544 EA

Applications

SH Web Apps: OpenEntityManagerInView

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ... version="3.0">

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/springconfig.xml</param-value>
  </context-param>
  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>

  <filter>
    <filter-name>SpringOpenEntityManagerInViewFilter</filter-name>
    <filter-class>
      org.springframework.orm.jpa.support.OpenEntityManagerInViewFilter
    </filter-class>
 </filter>

  <filter-mapping>
    <filter-name>SpringOpenEntityManagerInViewFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

Startup Spring

Create the Filter

Apply it everywhere

WebApplicationInitializer

```
package cs544.application05;

import javax.servlet.FilterRegistration;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;

import org.springframework.orm.jpa.support.OpenEntityManagerInViewFilter;
import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.ContextLoaderListener;
import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;

public class MyWebAppInitializer implements WebApplicationInitializer {

	@Override
	public void onStartup(ServletContext container) throws ServletException {
		AnnotationConfigWebApplicationContext rootContext =
				new AnnotationConfigWebApplicationContext();
		rootContext.register(Config.class);
		container.addListener(new ContextLoaderListener(rootContext));

		FilterRegistration.Dynamic openInView =
				container.addFilter("OpenInView", new OpenEntityManagerInViewFilter());
		openInView.addMappingForUrlPatterns(null, true, "/*");
	}
}
```

Or if you use a WebApplicationInitializer instead of web.xml you can register the filter like this

From DB to Web (with Filter)

```java
@Entity
public class Customer {
     @Id
     @GeneratedValue
     private Long id;
     private String name;
     @OneToOne(fetch = FetchType.LAZY)
     private Address address;

     ...
```

Added a LAZY association to demonstrate OpenEntityManagerInView working correctly

```java
@Entity
public class Address {
     @Id
     @GeneratedValue
     private Long id;
     private String place;

     ...
```

Import.sql

```sql
INSERT INTO Customer VALUES(NULL, "James Reagon", 1);
INSERT INTO Customer VALUES(NULL, "Lilly Johnson", 2);
INSERT INTO Customer VALUES(NULL, "George Tall", 3);
INSERT INTO Address VALUES(NULL, "New York");
INSERT INTO Address VALUES(NULL, "Los Angeles");
INSERT INTO Address VALUES(NULL, "Chicago");
```

DAO and Service

Using the more serious @Transactional

```
@Repository
@Transactional(propagation = Propagation.MANDATORY)
public class CustomerDao {
	@PersistenceContext
	private EntityManager em;

	public List<Customer> getAll() {
		return em.createQuery("from Customer", Customer.class).getResultList();
	}
}
```

```
@Service
@Transactional(propagation = Propagation.REQUIRES_NEW)
public class CustomerService {
	@Resource
	private CustomerDao customerDao;

	public List<Customer> getCustomers() {
		return customerDao.getAll();
	}
}
```

Controller

```
@WebServlet(name = "Customers", urlPatterns = { "/customers" })
public class Customers extends HttpServlet {
     private static final long serialVersionUID = 1L;

     @Override
     protected void doGet(HttpServletRequest request, HttpServletResponse response)
               throws ServletException, IOException {

         ServletContext context = getServletContext();
         WebApplicationContext applicationContext =
             WebApplicationContextUtils.getWebApplicationContext(context);
         CustomerService custServ = applicationContext.getBean(
             "customerService", CustomerService.class);


          request.setAttribute("customers", custServ.getCustomers());
          String jsp = "/Customers.jsp";
          RequestDispatcher dispatcher = context.getRequestDispatcher(jsp);
          dispatcher.forward(request, response);
     }
}
```

Same as before

JSP

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Customers</title>
</head>
<body>
     <h1>Customers:</h1>
     <ul>
          <c:forEach items="${customers}" var="customer">
               <li>${customer.name}: ${customer.address.place}</li>
          </c:forEach>
     </ul>
</body>
</html>
```

Lazy Loads Address