JavaEE7 Tutorial

# はじめに

## このチュートリアルで学ぶこと

* Java EE7による基本的なアプリケーションの開発方法およびNetBeansプロジェクトの構築方法

## 検証環境

このチュートリアルは以下の環境で動作確認している。

|  |  |
| --- | --- |
| 種別 | 名前 |
| OS | OS X 10.8.4 |
| JVM | Java 1.7 |
| IDE | NetBeans 7.3.1 |
| AP Server | GlassFish Server Open Source Edition 4.0 |
| Web Browser | Google Chrome 27.0.1453.94 m |

上記環境に合わせる必要はない。

# 作成するアプリケーションの説明

## アプリケーションの概要

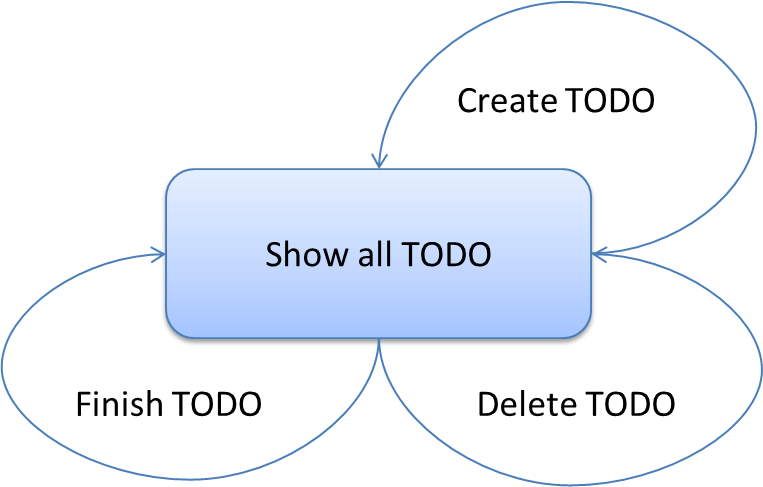
TODOを管理するアプリケーションを作成する。TODOの一覧表示、TODOの登録、TODOの完了、TODOの削除を行える。



## アプリケーションの業務要件

* B01 未完のTODOは5件までしか登録できない
* B02 完了済みのTODOは完了できない

## アプリケーションの画面遷移



|  |  |  |
| --- | --- | --- |
| Index | Process name |  |
| 1 | Show all TODO |  |
| 2 | Create TODO | 作成完了後1へリダイレクト |
| 3 | Finish TODO | 作成完了後1へリダイレクト |
| 4 | Delete TODO | 作成完了後1へリダイレクト |

### Show all TODO

* TODOを全件表示する
* 未完了のTODOに対しては”Done”と”Delete”用のボタンが付く
* 完了のTODOは打ち消し線で装飾する
* TODOの件名のみ

### Create TODO

* フォームから送信されたTODOを保存する
* TODOの件名は1文字以上30文字以下であること
* [2-2]のB01を満たさない場合はエラーコードE001でビジネス例外をスローする

### Finish TODO

* フォームから送信されたtodoIdに対応するTODOを完了済みにする
* [2-2]のB02を満たさない場合はエラーコードE002でビジネス例外をスローする
* 該当するTODOが存在しない場合はエラーコードE404でビジネス例外をスローする

### Delete TODO

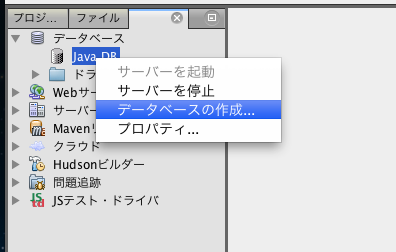
* フォームから送信されたtodoIdに対応するTODOを削除する
* 該当するTODOが存在しない場合はエラーコードE404でビジネス例外をスローする

## エラーメッセージ一覧

|  |  |  |
| --- | --- | --- |
| エラーコード | メッセージ | 置換パラメータ |
| E001 | The count of un-finished Todo must not be over {0}. | {0}… max unfinished count |
| E002 | The requested Todo is already finished. (id={0}) | {0}… todoId |
| E404 | The requested Todo is not found. (id={0}) | {0}… todoId |

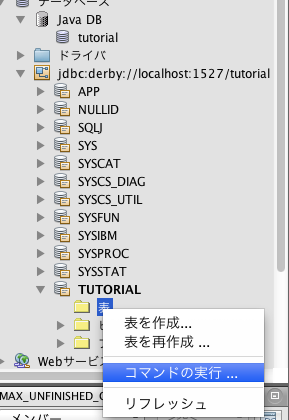
# 環境構築

## データベース作成









CREATE TABLE todo (

todo\_id INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY,

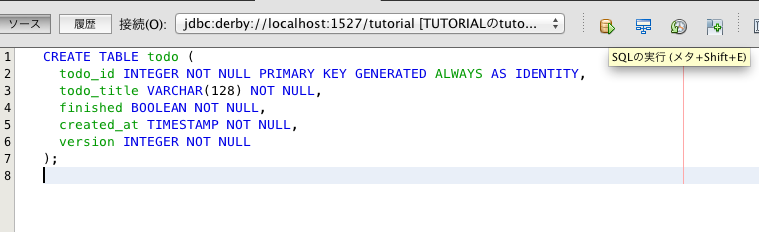
todo\_title VARCHAR(128) NOT NULL,

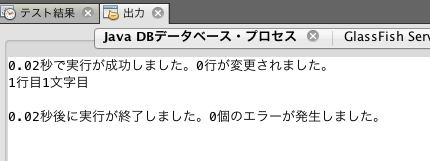
finished BOOLEAN NOT NULL,

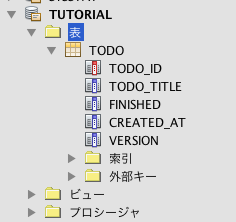
created\_at TIMESTAMP NOT NULL,

version INTEGER NOT NULL

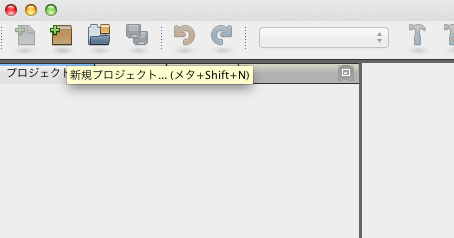
);

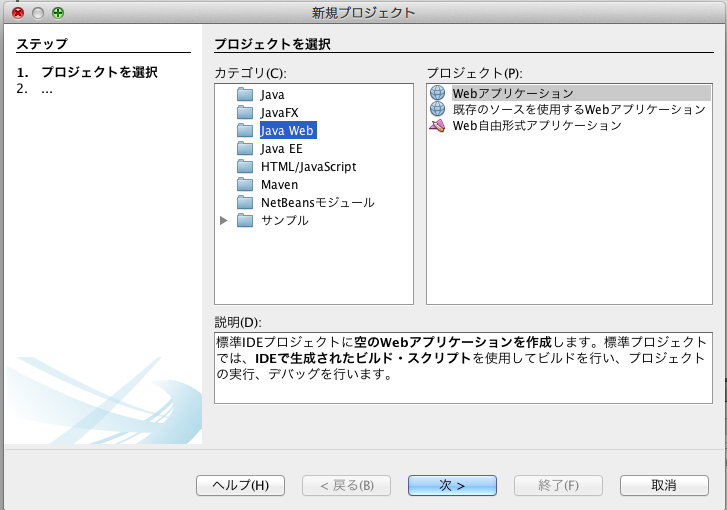


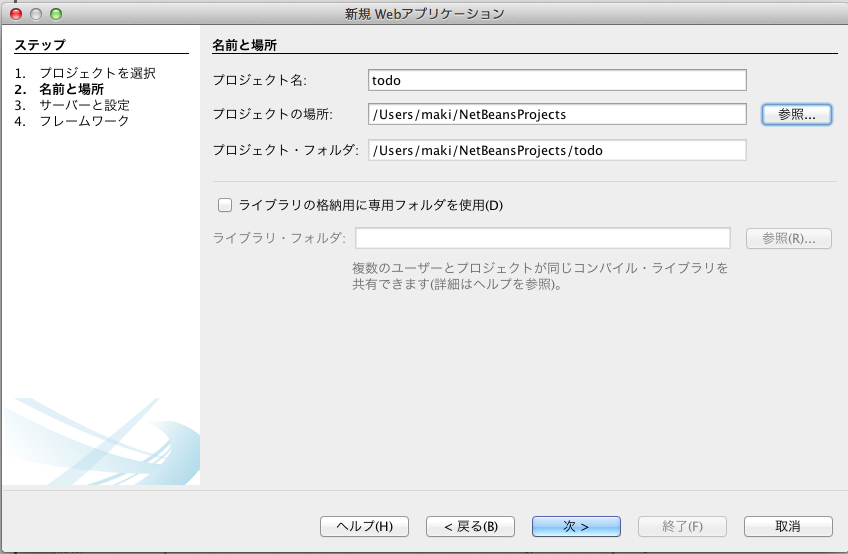


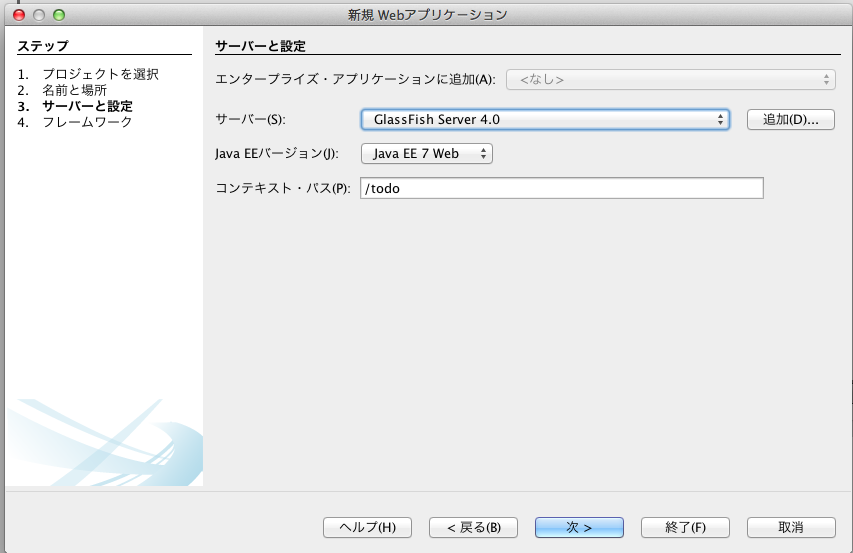


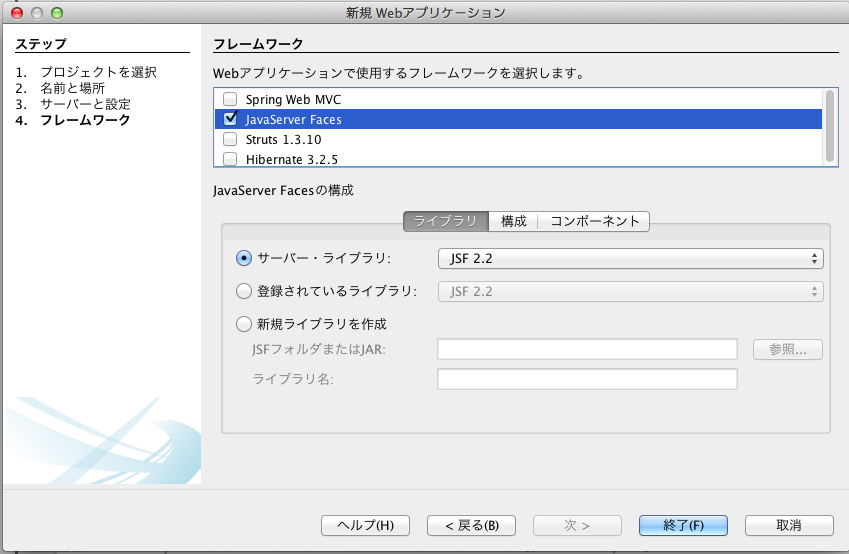
## プロジェクトの作成

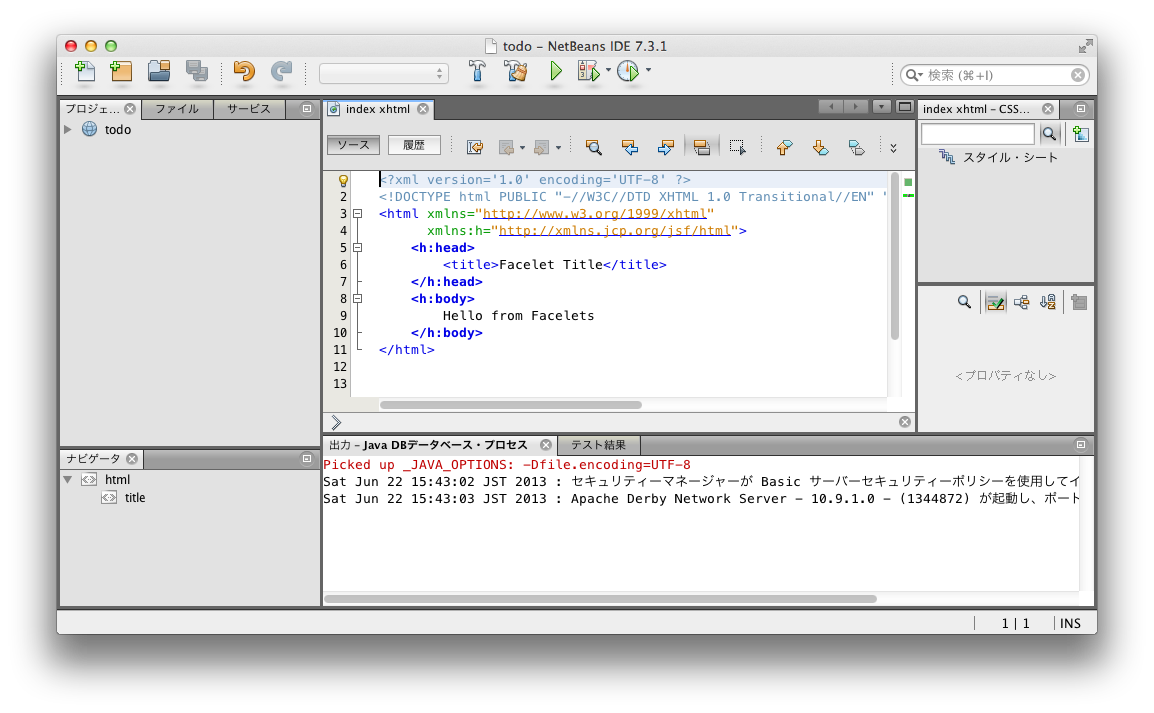




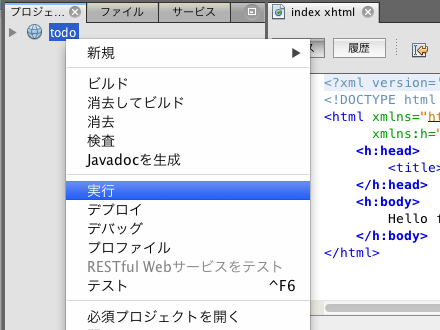


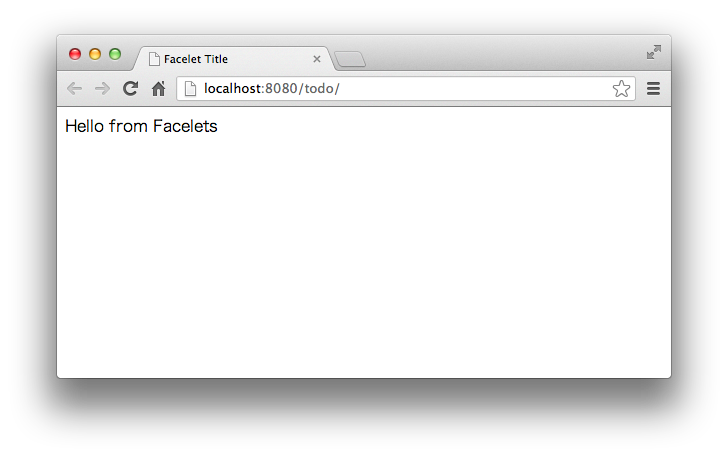






## 動作確認





## プロジェクト構成

以下のようなプロジェクト構成でアプリケーションを作成していく。

src/main/java

└todo ... TODOアプリケーションのベースパッケージ

├ app ... アプリケーション層を格納

│ └todo ... todo管理業務に関わるクラスを格納

└domain ... ドメイン層を格納

├model ... DomainObjectを格納

├repository ... Repositoryを格納

│ └todo ... Todo用Repository

└service ... Serviceを格納

└todo ... TODO業務Service

src/main/resources

├META-INF/spring ... spring関連の設定ファイルを格納

src/main/webapp

└WEB-INF

└views ... jspを格納

# Todoアプリケーションの作成

Todoアプリケーションを作成する。

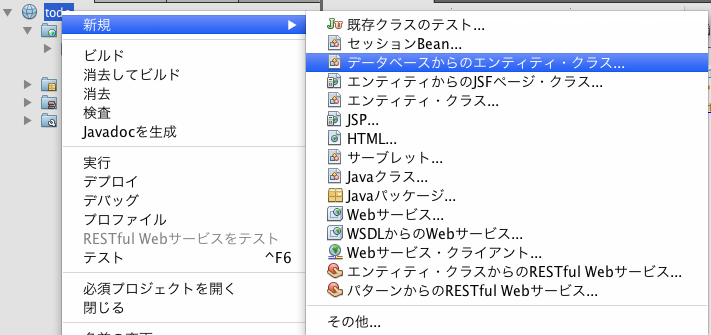
作成する順は以下の通りである。

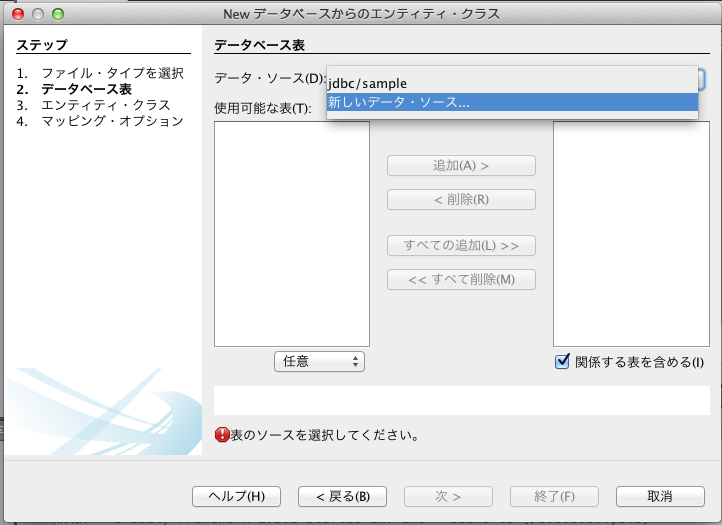
1. ドメイン層(+ インフラストラクチャ層)
   1. DomainObject作成
   2. Repository作成
   3. Service作成
2. アプリケーション層
   1. Controller作成
   2. Form作成
   3. View作成

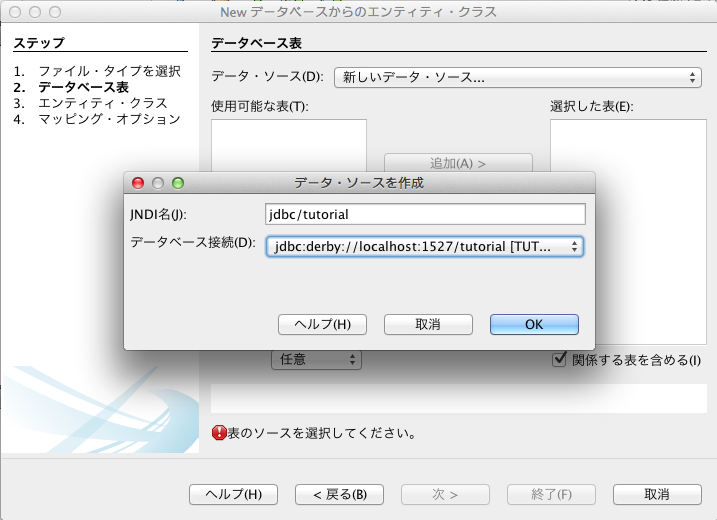
尚、本章ではTodoの保存にDBを使用しない。DBを使用するRepositoryの作成は第5章で行う。

## Entityの作成

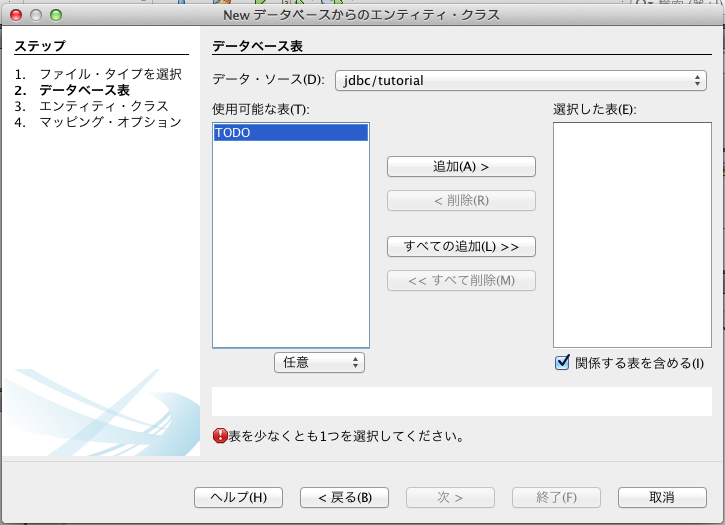
### リバースエンジニアリングでDBからエンティティ生成

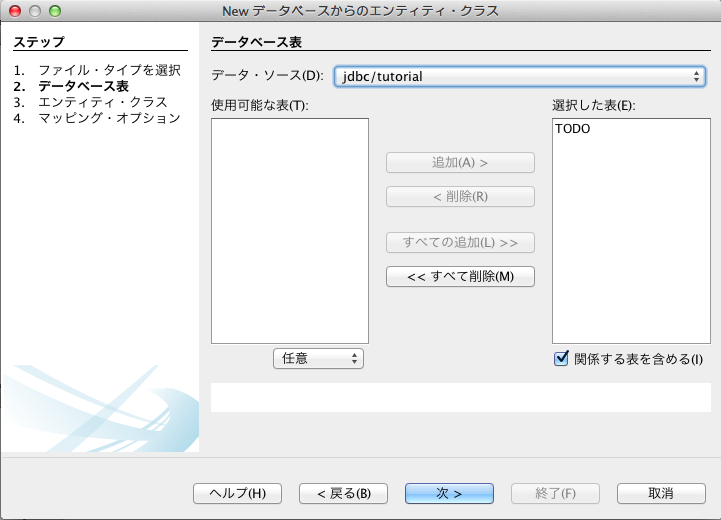


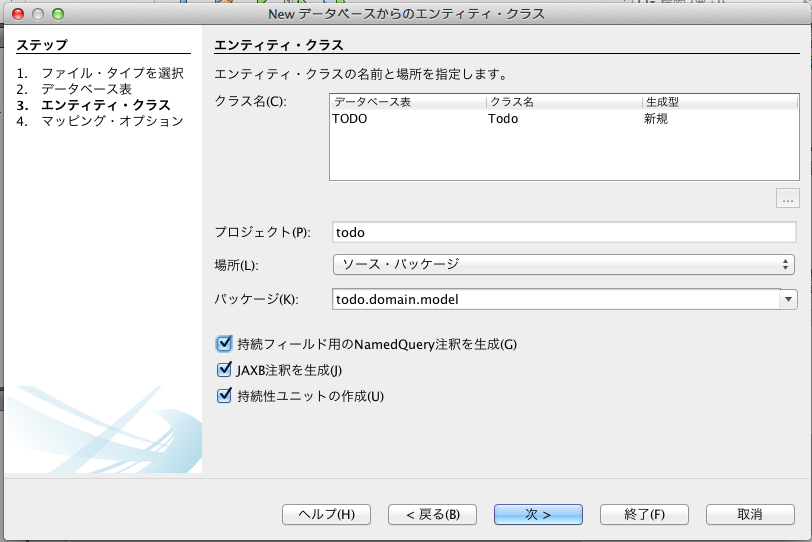


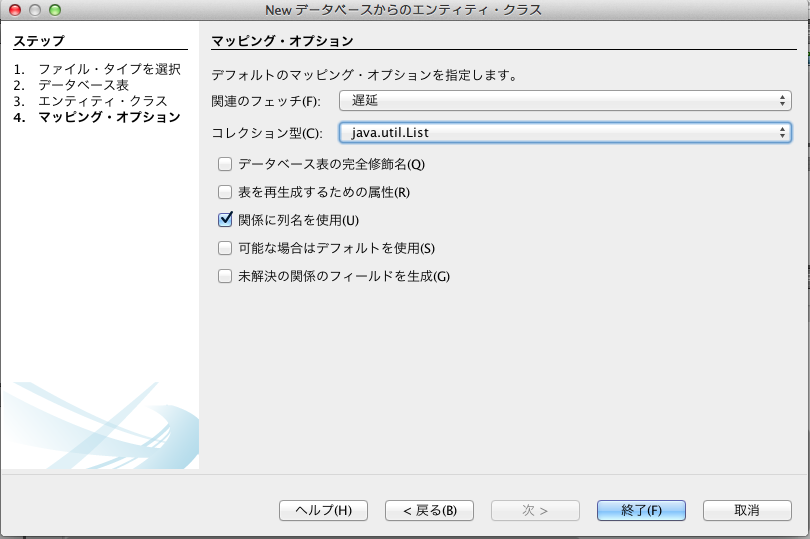










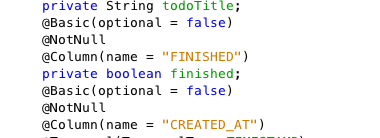


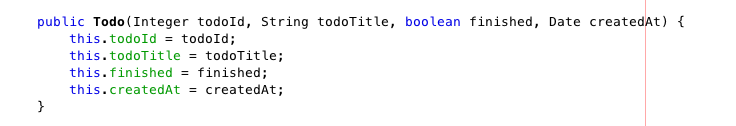


### 生成されたソースの修正

* フィールドとコンストラクタを修正

finishedの型をSerializableからbooleanに変更する。





* getFinishedとsetFinishedを削除して、getter/setter再生成

以下のコードを削除する。



ctrl+spaceで補完する







* @Versionアノテーションの追加

楽観ロック用のバージョンをJPA側で自動インクリメントさせるためにversionフィールドに@Versionアノテーションをつける。

package todo.domain.model;

import java.io.Serializable;

import java.util.Date;

import javax.persistence.Basic;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

package todo.domain.model;

import java.io.Serializable;

import java.util.Date;

import javax.persistence.Basic;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

import javax.persistence.Version;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import javax.xml.bind.annotation.XmlRootElement;

@Entity

@Table(name = "TODO")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Todo.findAll", query = "SELECT t FROM Todo t"),

@NamedQuery(name = "Todo.findByTodoId", query = "SELECT t FROM Todo t WHERE t.todoId = :todoId"),

@NamedQuery(name = "Todo.findByTodoTitle", query = "SELECT t FROM Todo t WHERE t.todoTitle = :todoTitle"),

@NamedQuery(name = "Todo.findByFinished", query = "SELECT t FROM Todo t WHERE t.finished = :finished"),

@NamedQuery(name = "Todo.findByCreatedAt", query = "SELECT t FROM Todo t WHERE t.createdAt = :createdAt"),

@NamedQuery(name = "Todo.findByVersion", query = "SELECT t FROM Todo t WHERE t.version = :version")})

public class Todo implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "TODO\_ID")

private Integer todoId;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 128)

@Column(name = "TODO\_TITLE")

private String todoTitle;

@Basic(optional = false)

@NotNull

@Column(name = "FINISHED")

private boolean finished;

@Basic(optional = false)

@NotNull

@Column(name = "CREATED\_AT")

@Temporal(TemporalType.TIMESTAMP)

private Date createdAt;

@Basic(optional = false)

@NotNull

@Column(name = "VERSION")

@Version

private int version;

public Todo() {

}

public Todo(Integer todoId) {

this.todoId = todoId;

}

public Todo(Integer todoId, String todoTitle, boolean finished, Date createdAt, int version) {

this.todoId = todoId;

this.todoTitle = todoTitle;

this.finished = finished;

this.createdAt = createdAt;

this.version = version;

}

public Integer getTodoId() {

return todoId;

}

public void setTodoId(Integer todoId) {

this.todoId = todoId;

}

public String getTodoTitle() {

return todoTitle;

}

public void setTodoTitle(String todoTitle) {

this.todoTitle = todoTitle;

}

public boolean isFinished() {

return finished;

}

public void setFinished(boolean finished) {

this.finished = finished;

}

public Date getCreatedAt() {

return createdAt;

}

public void setCreatedAt(Date createdAt) {

this.createdAt = createdAt;

}

public int getVersion() {

return version;

}

public void setVersion(int version) {

this.version = version;

}

@Override

public int hashCode() {

int hash = 0;

hash += (todoId != null ? todoId.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

// TODO: Warning - this method won't work in the case the id fields are not set

if (!(object instanceof Todo)) {

return false;

}

Todo other = (Todo) object;

if ((this.todoId == null && other.todoId != null) || (this.todoId != null && !this.todoId.equals(other.todoId))) {

return false;

}

return true;

}

@Override

public String toString() {

return "todo.domain.model.Todo[ todoId=" + todoId + " ]";

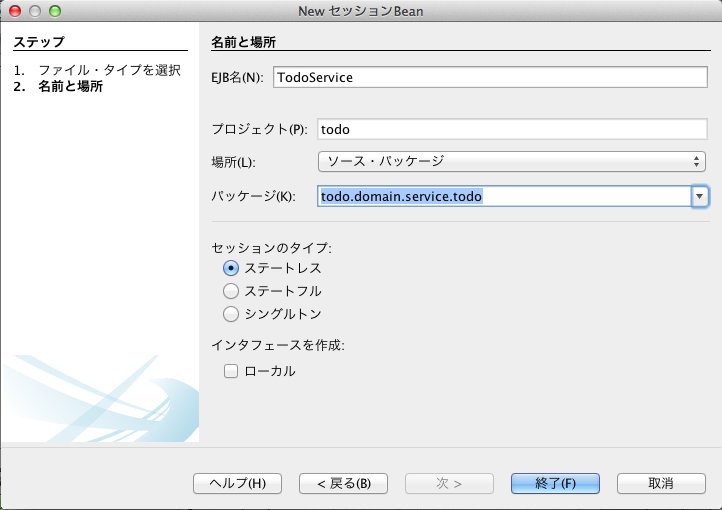
}

}

## EJBで業務処理を実装

### SessionBeanの作成





package todo.domain.service.todo;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.persistence.TypedQuery;

import todo.domain.model.Todo;

@Stateless

public class TodoService {

@PersistenceContext

protected EntityManager entityManager;

public List<Todo> findAll() {

TypedQuery q = entityManager.createNamedQuery("Todo.findAll", Todo.class);

return q.getResultList();

}

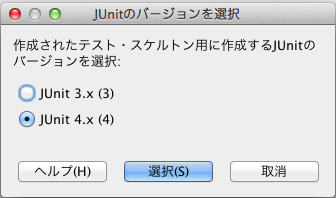
}

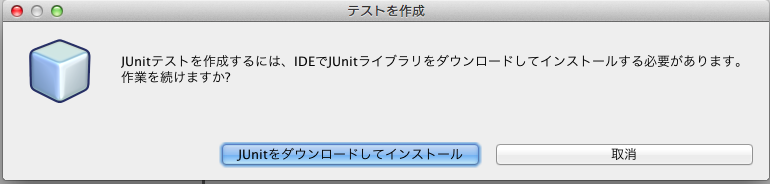
### JUnitの作成

動作確認を兼ねて予めJUnitのテストケースを作成する。

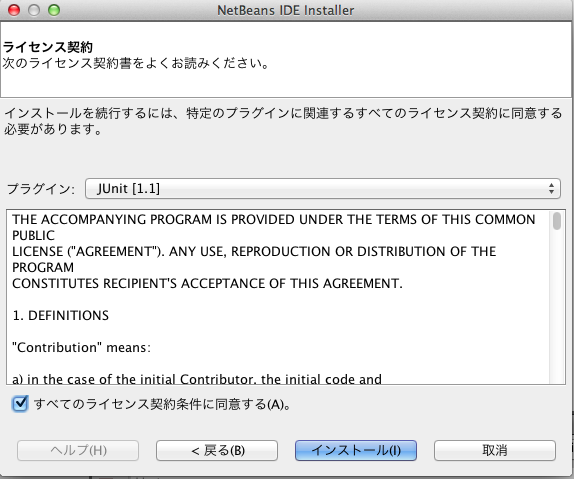
#### テストケースの作成

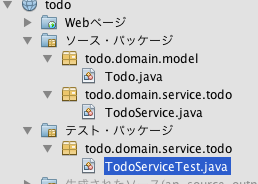












生成されたテストコードを修正

package todo.domain.service.todo;

import java.util.List;

import javax.ejb.embeddable.EJBContainer;

import javax.naming.Context;

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Test;

import static org.junit.Assert.\*;

import todo.domain.model.Todo;

public class TodoServiceTest {

private EJBContainer container;

private Context context;

public TodoServiceTest() {

}

@BeforeClass

public static void setUpClass() {

}

@AfterClass

public static void tearDownClass() {

}

@Before

public void setUp() {

container = javax.ejb.embeddable.EJBContainer.createEJBContainer();

context = container.getContext();

}

@After

public void tearDown() {

container.close();

}

/\*\*

\* Test of findAll method, of class TodoService.

\*/

@Test

public void testFindAll() throws Exception {

System.out.println("findAll");

TodoService instance = (TodoService)context.lookup("java:global/classes/TodoService");

List<Todo> result = instance.findAll();

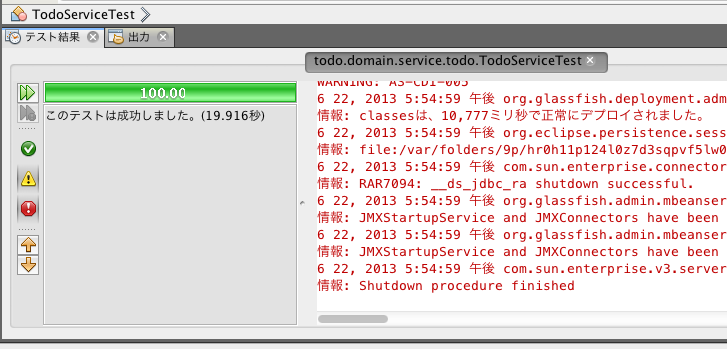
System.out.println(result);

assertNotNull(result);

}

}





### 業務処理の実装

業務処理を実装する。必要な処理は

* Todoの全件取得
* Todoの新規作成
* Todoの完了
* Todoの削除

である。

必要な処理と実装するメソッドの対応は以下の通り。

* Todoの全件取得→findAllメソッド
* Todoの新規作成→createメソッド
* Todoの完了→finishメソッド
* Todoの削除→deleteメソッド

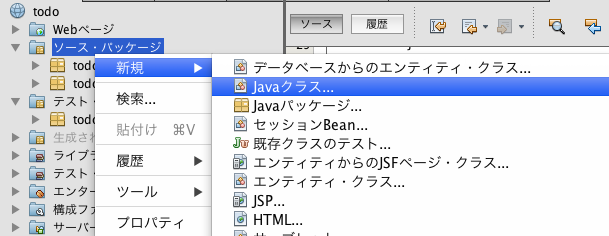
findAllは既に実装したので、残りの3処理を実装する

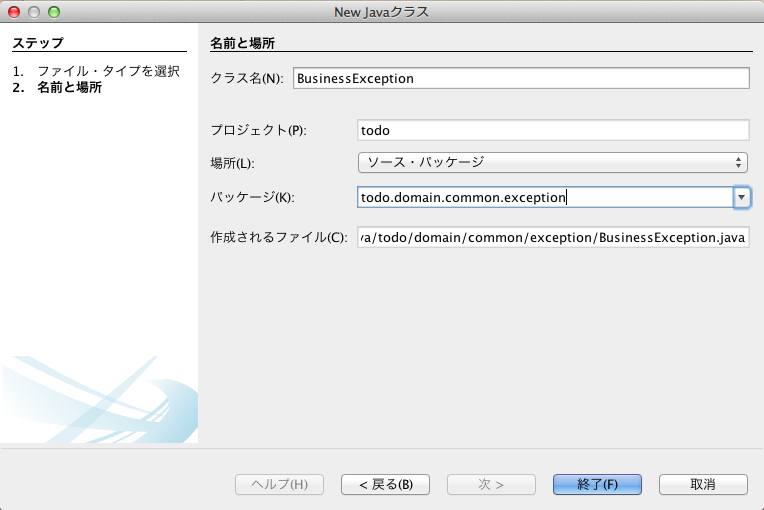
#### 例外の作成

業務処理を実装する前に、業務処理で使用する例外クラスを作成する。作成する例外は以下の2つ。

* BusinessException・・・業務エラーが発生した際にスローする例外
* ResourceNotFoundException・・・対象のリソース(エンティティ)が存在しない場合にスローする例外

例外ハンドリングやレスポンスステータスを変更できるようにこの2種類の例外は別にする。





package todo.domain.common.exception;

import javax.ejb.ApplicationException;

@ApplicationException

public class BusinessException extends RuntimeException {

public BusinessException(String message) {

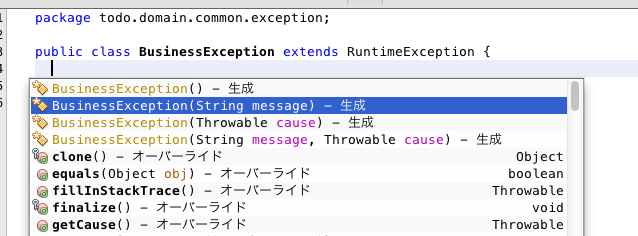
super(message);

}

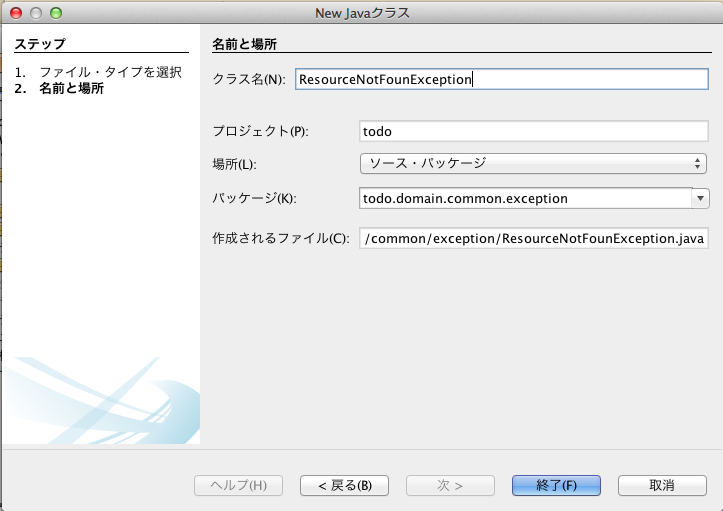
}

例外クラスには@ApplicationExceptionアノテーションをつける。このアノテーションがない場合はjavax.ejb.EJBExceptionにラップしてスローされる。

“extends RuntimeException”を書いた後はCtrl+Spaceでコンストラクタを自動生成できる。



同様にResourceNotFoundExceptionを作成



package todo.domain.common.exception;

import javax.ejb.ApplicationException;

@ApplicationException

public class ResourceNotFoundException extends RuntimeException{

public ResourceNotFoundException(String message) {

super(message);

}

}

#### 業務処理メソッドの実装

package todo.domain.service.todo;

import java.util.Date;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.persistence.TypedQuery;

import todo.domain.common.exception.BusinessException;

import todo.domain.common.exception.ResourceNotFoundException;

import todo.domain.model.Todo;

@Stateless

public class TodoService {

private static final long MAX\_UNFINISHED\_COUNT = 5;

@PersistenceContext

protected EntityManager entityManager;

public List<Todo> findAll() {

TypedQuery q = entityManager.createNamedQuery("Todo.findAll", Todo.class);

return q.getResultList();

}

public Todo findOne(Integer todoId) {

Todo todo = entityManager.find(Todo.class, todoId);

if (todo == null) {

throw new ResourceNotFoundException("[E404] The requested Todo is not found. (id=" + todoId + ")");

}

return todo;

}

public Todo create(Todo todo) {

TypedQuery<Long> q = entityManager.createQuery("SELECT COUNT(x) FROM Todo x WHERE x.finished = :finished", Long.class)

.setParameter("finished", false);

long unfinishedCount = q.getSingleResult();

if (unfinishedCount > MAX\_UNFINISHED\_COUNT) {

throw new BusinessException("[E001] The count of un-finished Todo must not be over "

+ MAX\_UNFINISHED\_COUNT + ".");

}

todo.setFinished(false);

todo.setCreatedAt(new Date());

entityManager.persist(todo);

return todo;

}

public Todo finish(Integer todoId) {

Todo todo = findOne(todoId);

if (todo.isFinished()) {

throw new BusinessException("[E002] The requested Todo is already finished. (id="

+ todoId + ")");

}

todo.setFinished(true);

entityManager.persist(todo);

return todo;

}

public void delete(Integer todoId) {

Todo todo = findOne(todoId);

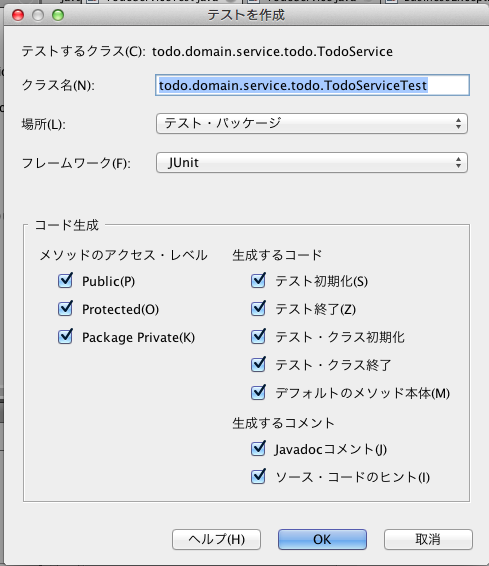
entityManager.remove(todo);

}

}

テストケースを追加する。



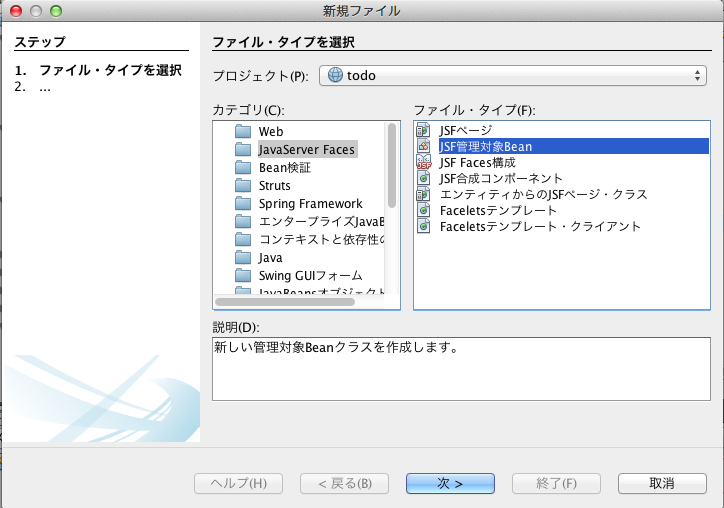


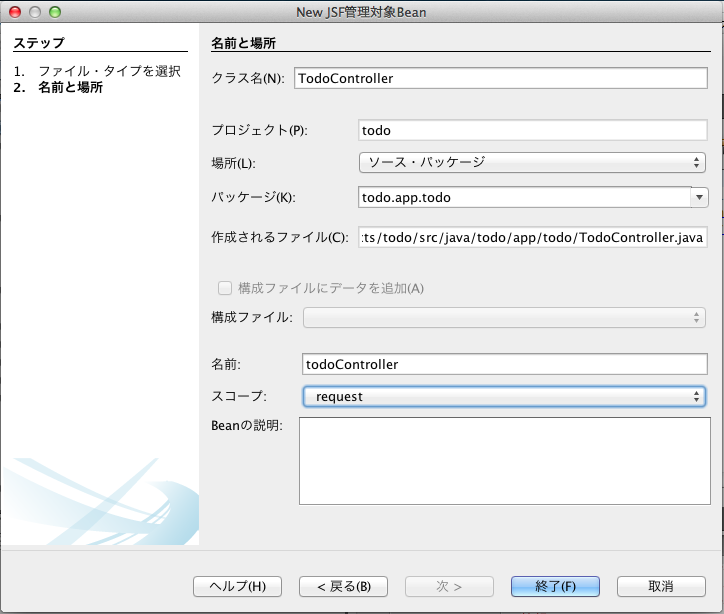
testFindOne, testCreate, testFinished, testDeleteが追加される。テストメソッドの修正は割愛する。

## JSFでUIを作成

### ManagedBeanの作成

Todo管理の画面遷移を制御するTodoControllerを作成する。





package todo.app.todo;

import javax.inject.Named;

import javax.enterprise.context.RequestScoped;

@Named(value = "todoController")

@RequestScoped

public class TodoController {

/\*\*

\* Creates a new instance of TodoController

\*/

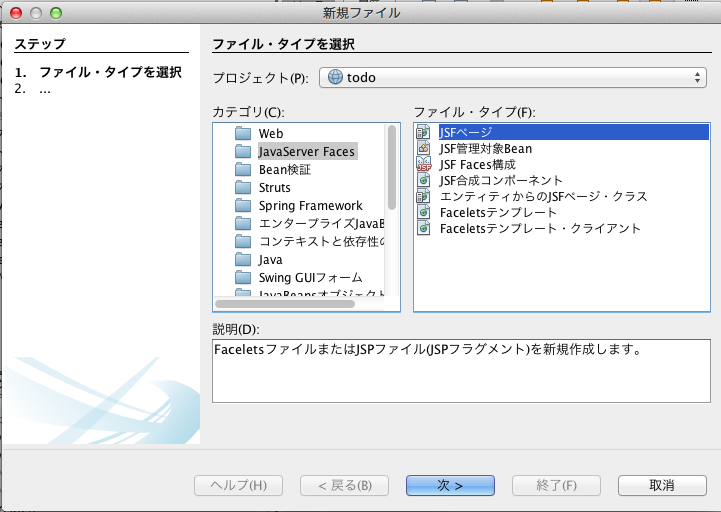
public TodoController() {

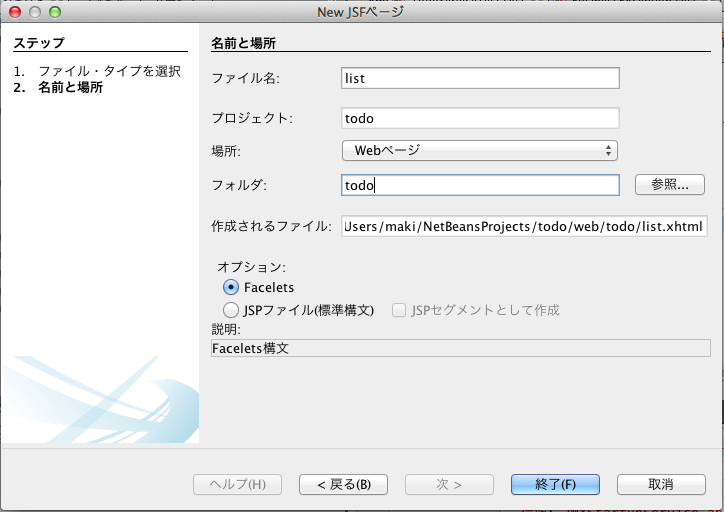
}

}

### Faceletsの作成

ViewとしてFacelets(list.xhtml)を作成する。





### Show all TODO

この画面ではTODOの全件表示を行う。

#### TodoControllerの修正

package todo.app.todo;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.inject.Named;

import javax.enterprise.context.RequestScoped;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Named(value = "todoController")

@RequestScoped

public class TodoController {

@EJB

protected TodoService todoService;

protected List<Todo> todoList;

/\*\*

\* Creates a new instance of TodoController

\*/

public TodoController() {

}

public List<Todo> getTodoList() {

return todoList;

}

@PostConstruct

public void init() {

todoList = todoService.findAll();

}

}

#### list.xhtmlの修正

<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>Todo List</title>

<style type="text/css">

.strike {

text-decoration: line-through;

}

</style>

</h:head>

<h:body>

<h2>Todo List</h2>

<h:dataTable value="#{todoController.todoList}" var="todo">

<h:column>

<f:facet name="header">

<h:outputText value="Titile" />

</f:facet>

<h:outputText value="#{todo.todoTitle}" rendered="#{!todo.finished}" />

<h:outputText value="#{todo.todoTitle}" rendered="#{todo.finished}" class="strike" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Created At" />

</f:facet>

<h:outputText value="#{todo.createdAt}" />

</h:column>

</h:dataTable>

</h:body>

</html>

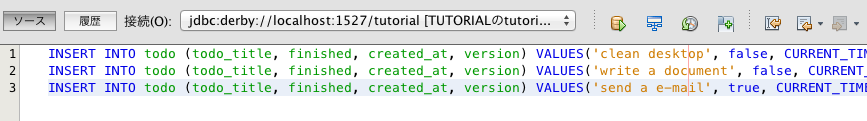
#### テストデータの投入

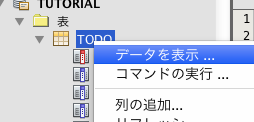


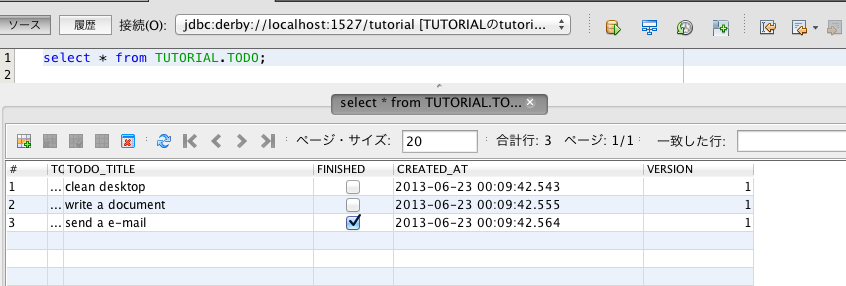
INSERT INTO todo (todo\_title, finished, created\_at, version) VALUES('clean desktop', false, CURRENT\_TIMESTAMP, 1);

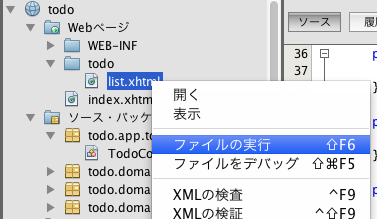
INSERT INTO todo (todo\_title, finished, created\_at, version) VALUES('write a document', false, CURRENT\_TIMESTAMP, 1);

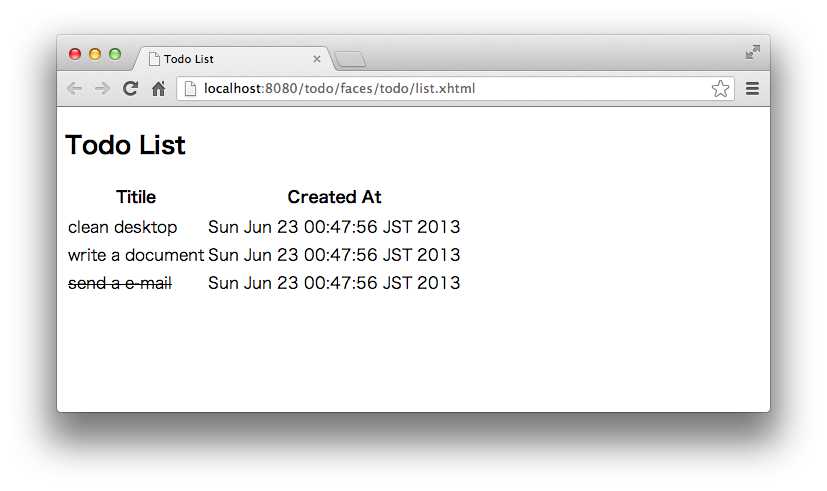
INSERT INTO todo (todo\_title, finished, created\_at, version) VALUES('send a e-mail', true, CURRENT\_TIMESTAMP, 1);





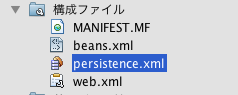


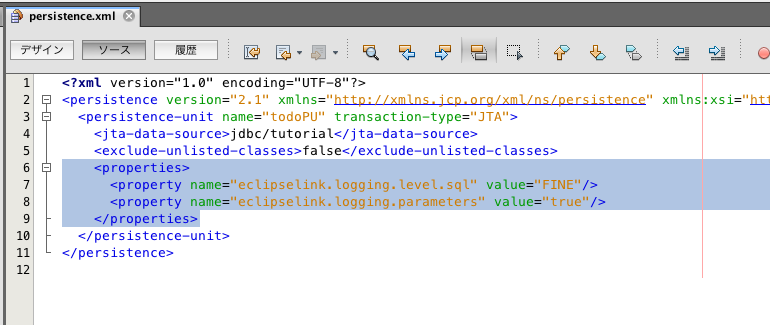




#### SQLログを出力する

persistence.xmlを開いてプロパティを変更する。





<?xml version="1.0" encoding="UTF-8"?>

<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">

<persistence-unit name="todoPU" transaction-type="JTA">

<jta-data-source>jdbc/tutorial</jta-data-source>

<exclude-unlisted-classes>false</exclude-unlisted-classes>

<properties>

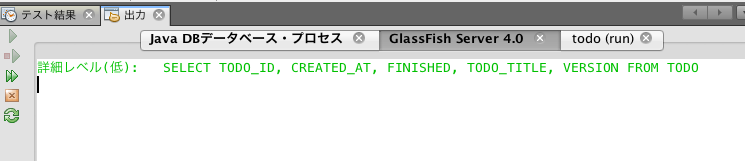
<property name="eclipselink.logging.level.sql" value="FINE"/>

<property name="eclipselink.logging.parameters" value="true"/>

</properties>

</persistence-unit>

</persistence>



### Create TODO

新規作成フォームを作成する。

#### TodoControllerの修正

package todo.app.todo;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.inject.Named;

import javax.enterprise.context.RequestScoped;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Named(value = "todoController")

@RequestScoped

public class TodoController {

@EJB

protected TodoService todoService;

protected Todo todo = new Todo();

protected List<Todo> todoList;

/\*\*

\* Creates a new instance of TodoController

\*/

public TodoController() {

}

public Todo getTodo() {

return todo;

}

public List<Todo> getTodoList() {

return todoList;

}

@PostConstruct

public void init() {

todoList = todoService.findAll();

}

public String create() {

todoService.create(todo);

return "list?faces-redirect=true";

}

}

#### list.xhtmlの修正

<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>Todo List</title>

<style type="text/css">

.strike {

text-decoration: line-through;

}

</style>

</h:head>

<h:body>

<h2>Create Todo</h2>

<h:form>

<h:panelGrid columns="2">

<h:outputLabel value="Title: " />

<h:inputText value="#{todoController.todo.todoTitle}" />

<h:commandButton value="Create" action="#{todoController.create}" />

</h:panelGrid>

</h:form>

<h2>Todo List</h2>

<h:dataTable value="#{todoController.todoList}" var="todo">

<h:column>

<f:facet name="header">

<h:outputText value="Titile" />

</f:facet>

<h:outputText value="#{todo.todoTitle}" rendered="#{!todo.finished}" />

<h:outputText value="#{todo.todoTitle}" rendered="#{todo.finished}" class="strike" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Created At" />

</f:facet>

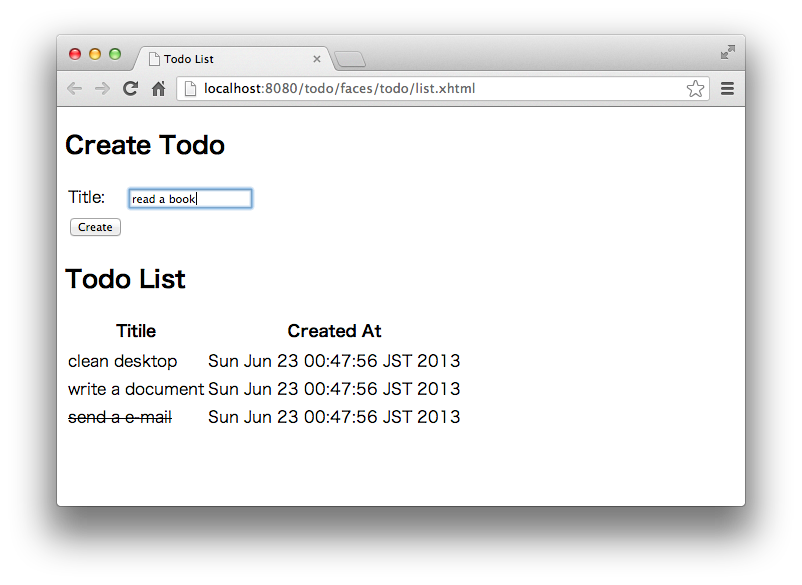
<h:outputText value="#{todo.createdAt}" />

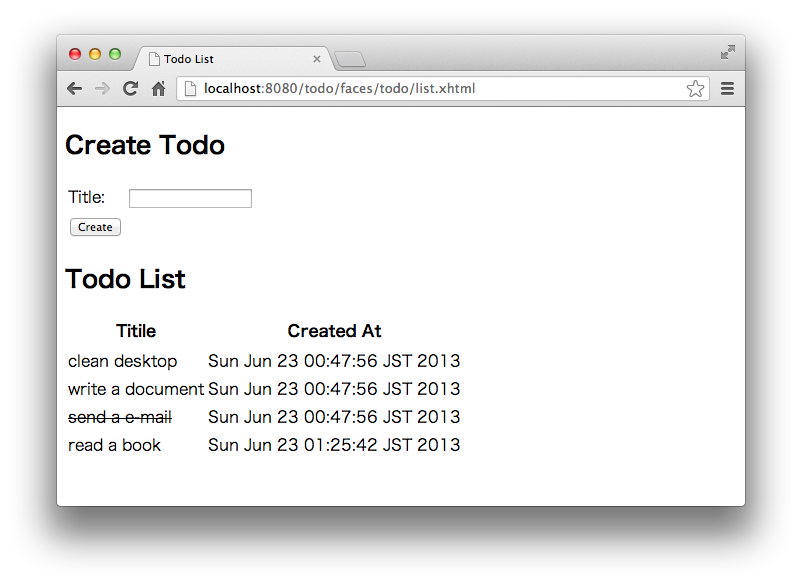
</h:column>

</h:dataTable>

</h:body>

</html>





#### 入力エラーメッセージの表示

<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>Todo List</title>

<style type="text/css">

.strike {

text-decoration: line-through;

}

.alert {

border: 1px solid;

padding: 3px;

width: 80%;

}

.alert-error {

background-color: #c60f13;

border-color: #970b0e;

color: white;

}

</style>

</h:head>

<h:body>

<h:messages layout="table" styleClass="alert alert-error" />

<h2>Create Todo</h2>

<h:form>

<h:panelGrid columns="2">

<h:outputLabel value="Title: " />

<h:inputText value="#{todoController.todo.todoTitle}" />

<h:commandButton value="Create" action="#{todoController.create}" />

</h:panelGrid>

</h:form>

<h2>Todo List</h2>

<h:dataTable value="#{todoController.todoList}" var="todo">

<h:column>

<f:facet name="header">

<h:outputText value="Titile" />

</f:facet>

<h:outputText value="#{todo.todoTitle}" rendered="#{!todo.finished}" />

<h:outputText value="#{todo.todoTitle}" rendered="#{todo.finished}" class="strike" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Created At" />

</f:facet>

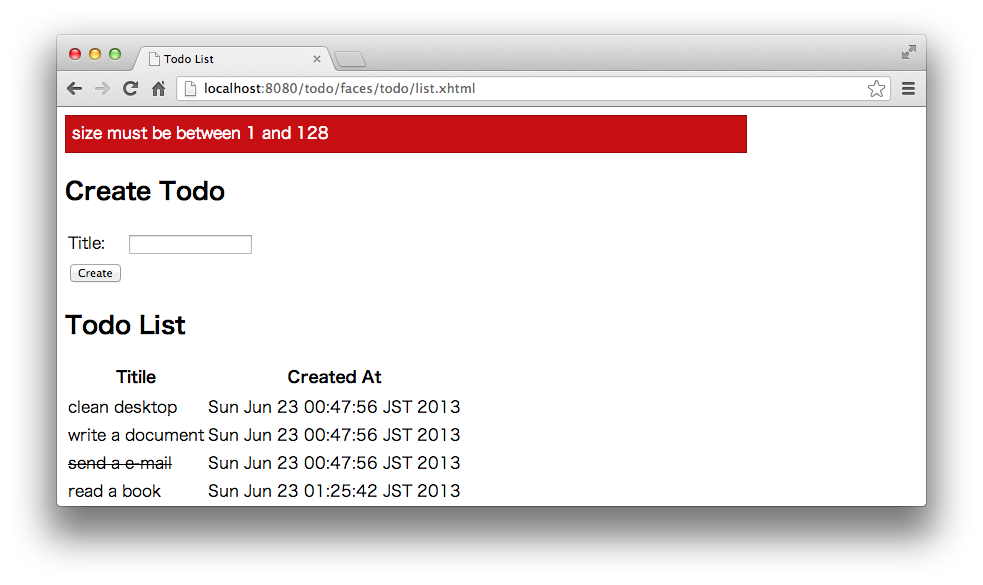
<h:outputText value="#{todo.createdAt}" />

</h:column>

</h:dataTable>

</h:body>

</html>



Memo:テキストフィールドの横にエラーメッセージを出力したい場合は、以下のように変更すればよい。

<h:form>

<h:panelGrid columns="3">

<h:outputLabel value="Title: " />

<h:inputText id=”title” value="#{todoController.todo.todoTitle}" />

<h:message for=”title” errorClass=”text-error” />

<h:commandButton value="Create" action="#{todoController.create}" />

</h:panelGrid>

</h:form>

#### 業務例外をハンドリングする

package todo.app.todo;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.inject.Named;

import javax.enterprise.context.RequestScoped;

import javax.faces.application.FacesMessage;

import javax.faces.context.FacesContext;

import todo.domain.common.exception.BusinessException;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Named(value = "todoController")

@RequestScoped

public class TodoController {

@EJB

protected TodoService todoService;

protected Todo todo = new Todo();

protected List<Todo> todoList;

/\*\*

\* Creates a new instance of TodoController

\*/

public TodoController() {

}

public Todo getTodo() {

return todo;

}

public List<Todo> getTodoList() {

return todoList;

}

@PostConstruct

public void init() {

todoList = todoService.findAll();

}

public String create() {

try {

todoService.create(todo);

} catch (BusinessException e) {

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_ERROR, e.getMessage(), null));

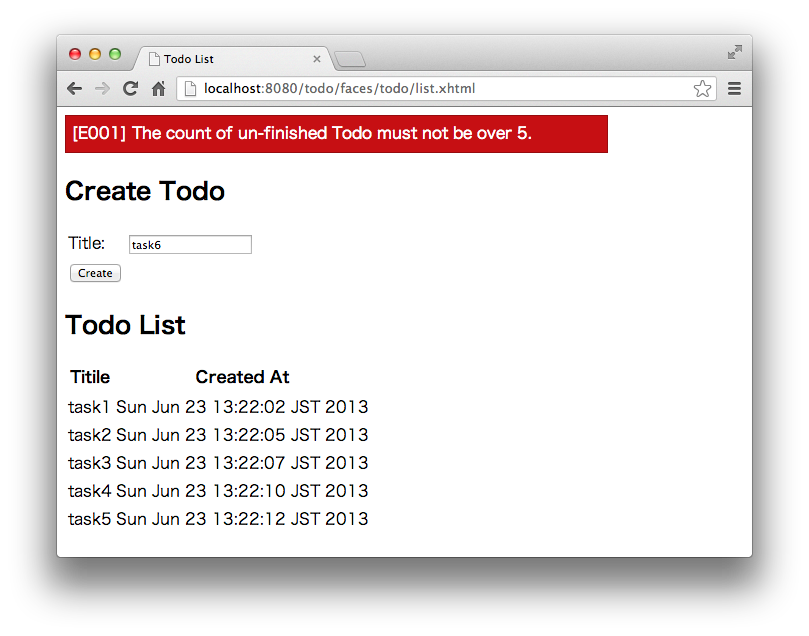
return "list.xhtml";

}

return "list.xhtml?faces-redirect=true";

}

}



#### 正常終了メッセージを出力する

リダイレクト先にもメッセージが残るようにFacesContextのFlashスコープを使用する。

package todo.app.todo;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.inject.Named;

import javax.enterprise.context.RequestScoped;

import javax.faces.application.FacesMessage;

import javax.faces.context.FacesContext;

import todo.domain.common.exception.BusinessException;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Named(value = "todoController")

@RequestScoped

public class TodoController {

@EJB

protected TodoService todoService;

protected Todo todo = new Todo();

protected List<Todo> todoList;

/\*\*

\* Creates a new instance of TodoController

\*/

public TodoController() {

}

public Todo getTodo() {

return todo;

}

public List<Todo> getTodoList() {

return todoList;

}

@PostConstruct

public void init() {

todoList = todoService.findAll();

}

public String create() {

try {

todoService.create(todo);

} catch (BusinessException e) {

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_ERROR, e.getMessage(), null));

return "list.xhtml";

}

FacesContext.getCurrentInstance().getExternalContext().getFlash().setKeepMessages(true);

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_INFO, "Created successfully!", null));

return "list.xhtml?faces-redirect=true";

}

}

FacesContextに設定したメッセージの種別で出力するメッセージのスタイルが変わるようにする。

<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>Todo List</title>

<style type="text/css">

.strike {

text-decoration: line-through;

}

.alert {

border: 1px solid;

padding: 3px;

width: 80%;

}

.alert-error {

background-color: #c60f13;

border-color: #970b0e;

color: white;

}

.alert-success {

background-color: #5da423;

border-color: #457a1a;

color: white;

}

</style>

</h:head>

<h:body>

<h:messages layout="table" styleClass="alert alert-success" rendered="#{facesContext.maximumSeverity.ordinal == 0}" />

<h:messages layout="table" styleClass="alert alert-error" rendered="#{facesContext.maximumSeverity.ordinal > 0}"/>

<h2>Create Todo</h2>

<h:form>

<h:panelGrid columns="2">

<h:outputLabel value="Title: " />

<h:inputText value="#{todoController.todo.todoTitle}" />

<h:commandButton value="Create" action="#{todoController.create}" />

</h:panelGrid>

</h:form>

<h2>Todo List</h2>

<h:dataTable value="#{todoController.todoList}" var="todo">

<h:column>

<f:facet name="header">

<h:outputText value="Titile" />

</f:facet>

<h:outputText value="#{todo.todoTitle}" rendered="#{!todo.finished}" />

<h:outputText value="#{todo.todoTitle}" rendered="#{todo.finished}" class="strike" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Created At" />

</f:facet>

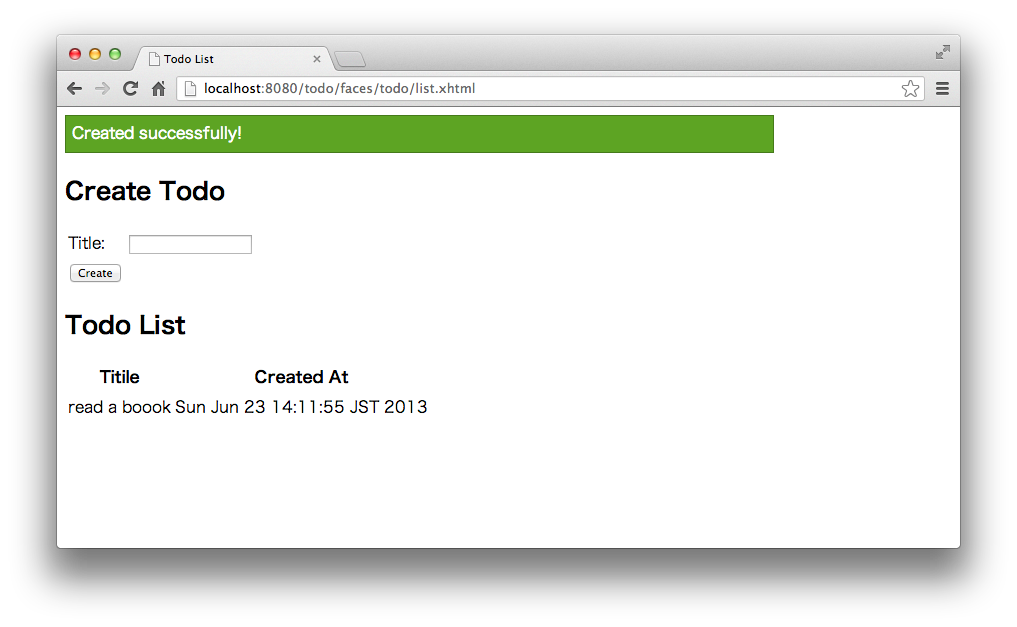
<h:outputText value="#{todo.createdAt}" />

</h:column>

</h:dataTable>

</h:body>

</html>



### Finish TODO

#### TodoControllerの修正

“Create TODO”をほぼ同じようにTodoControllerにfinishメソッドを追加する。

package todo.app.todo;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.inject.Named;

import javax.enterprise.context.RequestScoped;

import javax.faces.application.FacesMessage;

import javax.faces.context.FacesContext;

import todo.domain.common.exception.BusinessException;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Named(value = "todoController")

@RequestScoped

public class TodoController {

@EJB

protected TodoService todoService;

protected Todo todo = new Todo();

protected List<Todo> todoList;

/\*\*

\* Creates a new instance of TodoController

\*/

public TodoController() {

}

public Todo getTodo() {

return todo;

}

public List<Todo> getTodoList() {

return todoList;

}

@PostConstruct

public void init() {

todoList = todoService.findAll();

}

public String create() {

try {

todoService.create(todo);

} catch (BusinessException e) {

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_ERROR, e.getMessage(), null));

return "list.xhtml";

}

FacesContext.getCurrentInstance().getExternalContext().getFlash().setKeepMessages(true);

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_INFO, "Created successfully!", null));

return "list.xhtml?faces-redirect=true";

}

public String finish(Integer todoId) {

try {

todoService.finish(todoId);

} catch (BusinessException e) {

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_ERROR, e.getMessage(), null));

return "list.xhtml";

}

FacesContext.getCurrentInstance().getExternalContext().getFlash().setKeepMessages(true);

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_INFO, "Finished successfully!", null));

return "list.xhtml?faces-redirect=true";

}

}

#### list.xhtmlの修正

<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>Todo List</title>

<style type="text/css">

.strike {

text-decoration: line-through;

}

.alert {

border: 1px solid;

padding: 3px;

width: 80%;

}

.alert-error {

background-color: #c60f13;

border-color: #970b0e;

color: white;

}

.alert-success {

background-color: #5da423;

border-color: #457a1a;

color: white;

}

</style>

</h:head>

<h:body>

<h:messages layout="table" styleClass="alert alert-success" rendered="#{facesContext.maximumSeverity.ordinal == 0}" />

<h:messages layout="table" styleClass="alert alert-error" rendered="#{facesContext.maximumSeverity.ordinal > 0}"/>

<h2>Create Todo</h2>

<h:form>

<h:panelGrid columns="2">

<h:outputLabel value="Title: " />

<h:inputText value="#{todoController.todo.todoTitle}" />

<h:commandButton value="Create" action="#{todoController.create}" />

</h:panelGrid>

</h:form>

<h2>Todo List</h2>

<h:dataTable value="#{todoController.todoList}" var="todo">

<h:column>

<f:facet name="header">

<h:outputText value="Titile" />

</f:facet>

<h:outputText value="#{todo.todoTitle}" rendered="#{!todo.finished}" />

<h:outputText value="#{todo.todoTitle}" rendered="#{todo.finished}" class="strike" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Created At" />

</f:facet>

<h:outputText value="#{todo.createdAt}" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Actions" />

</f:facet>

<h:form>

<h:commandButton value="Done" action="#{todoController.finish(todo.todoId)}" />

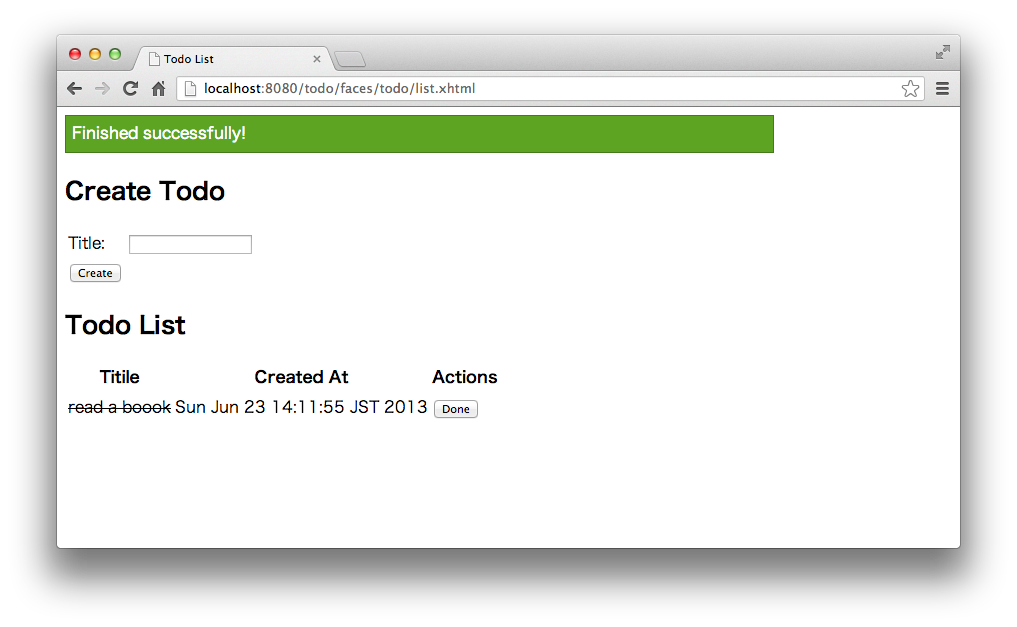
</h:form>

</h:column>

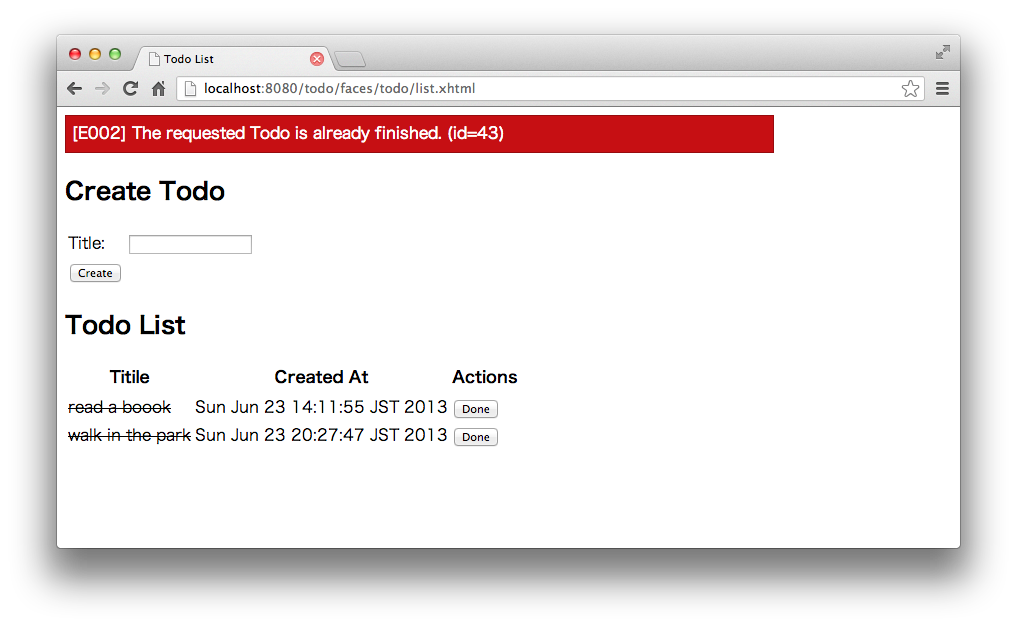
</h:dataTable>

</h:body>

</html>



Todoを一件作成し、「Done」をクリックして完了させた後、もう一回で「Done」をクリックするとE002のエラーが表示される。



完了済みの場合に”Done”ボタンを表示しないように修正する

<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>Todo List</title>

<style type="text/css">

.strike {

text-decoration: line-through;

}

.alert {

border: 1px solid;

padding: 3px;

width: 80%;

}

.alert-error {

background-color: #c60f13;

border-color: #970b0e;

color: white;

}

.alert-success {

background-color: #5da423;

border-color: #457a1a;

color: white;

}

</style>

</h:head>

<h:body>

<h:messages layout="table" styleClass="alert alert-success" rendered="#{facesContext.maximumSeverity.ordinal == 0}" />

<h:messages layout="table" styleClass="alert alert-error" rendered="#{facesContext.maximumSeverity.ordinal > 0}"/>

<h2>Create Todo</h2>

<h:form>

<h:panelGrid columns="2">

<h:outputLabel value="Title: " />

<h:inputText value="#{todoController.todo.todoTitle}" />

<h:commandButton value="Create" action="#{todoController.create}" />

</h:panelGrid>

</h:form>

<h2>Todo List</h2>

<h:dataTable value="#{todoController.todoList}" var="todo">

<h:column>

<f:facet name="header">

<h:outputText value="Titile" />

</f:facet>

<h:outputText value="#{todo.todoTitle}" rendered="#{!todo.finished}" />

<h:outputText value="#{todo.todoTitle}" rendered="#{todo.finished}" class="strike" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Created At" />

</f:facet>

<h:outputText value="#{todo.createdAt}" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Actions" />

</f:facet>

<h:form>

<h:commandButton value="Done" action="#{todoController.finish(todo.todoId)}" rendered="#{!todo.finished}" />

</h:form>

</h:column>

</h:dataTable>

</h:body>

</html>

### Delete TODO

#### TodoControllerの修正

package todo.app.todo;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.inject.Named;

import javax.enterprise.context.RequestScoped;

import javax.faces.application.FacesMessage;

import javax.faces.context.FacesContext;

import todo.domain.common.exception.BusinessException;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Named(value = "todoController")

@RequestScoped

public class TodoController {

@EJB

protected TodoService todoService;

protected Todo todo = new Todo();

protected List<Todo> todoList;

/\*\*

\* Creates a new instance of TodoController

\*/

public TodoController() {

}

public Todo getTodo() {

return todo;

}

public List<Todo> getTodoList() {

return todoList;

}

@PostConstruct

public void init() {

todoList = todoService.findAll();

}

public String create() {

try {

todoService.create(todo);

} catch (BusinessException e) {

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_ERROR, e.getMessage(), null));

return "list.xhtml";

}

FacesContext.getCurrentInstance().getExternalContext().getFlash().setKeepMessages(true);

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_INFO, "Created successfully!", null));

return "list.xhtml?faces-redirect=true";

}

public String finish(Integer todoId) {

try {

todoService.finish(todoId);

} catch (BusinessException e) {

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_ERROR, e.getMessage(), null));

return "list.xhtml";

}

FacesContext.getCurrentInstance().getExternalContext().getFlash().setKeepMessages(true);

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_INFO, "Finished successfully!", null));

return "list.xhtml?faces-redirect=true";

}

public String delete(Integer todoId) {

todoService.delete(todoId);

FacesContext.getCurrentInstance().getExternalContext().getFlash().setKeepMessages(true);

FacesContext.getCurrentInstance()

.addMessage(null, new FacesMessage(FacesMessage.SEVERITY\_INFO, "Deleted successfully!", null));

return "list.xhtml?faces-redirect=true";

}

}

#### list.xhtmlの修正

<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:h="http://xmlns.jcp.org/jsf/html"

xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:head>

<title>Todo List</title>

<style type="text/css">

.strike {

text-decoration: line-through;

}

.alert {

border: 1px solid;

padding: 3px;

width: 80%;

}

.alert-error {

background-color: #c60f13;

border-color: #970b0e;

color: white;

}

.alert-success {

background-color: #5da423;

border-color: #457a1a;

color: white;

}

</style>

</h:head>

<h:body>

<h:messages layout="table" styleClass="alert alert-success" rendered="#{facesContext.maximumSeverity.ordinal == 0}" />

<h:messages layout="table" styleClass="alert alert-error" rendered="#{facesContext.maximumSeverity.ordinal > 0}"/>

<h2>Create Todo</h2>

<h:form>

<h:panelGrid columns="2">

<h:outputLabel value="Title: " />

<h:inputText value="#{todoController.todo.todoTitle}" />

<h:commandButton value="Create" action="#{todoController.create}" />

</h:panelGrid>

</h:form>

<h2>Todo List</h2>

<h:dataTable value="#{todoController.todoList}" var="todo">

<h:column>

<f:facet name="header">

<h:outputText value="Titile" />

</f:facet>

<h:outputText value="#{todo.todoTitle}" rendered="#{!todo.finished}" />

<h:outputText value="#{todo.todoTitle}" rendered="#{todo.finished}" class="strike" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Created At" />

</f:facet>

<h:outputText value="#{todo.createdAt}" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="Actions" />

</f:facet>

<h:form>

<h:commandButton value="Done" action="#{todoController.finish(todo.todoId)}" rendered="#{!todo.finished}" />

<h:commandButton value="Delete" action="#{todoController.delete(todo.todoId)}" />

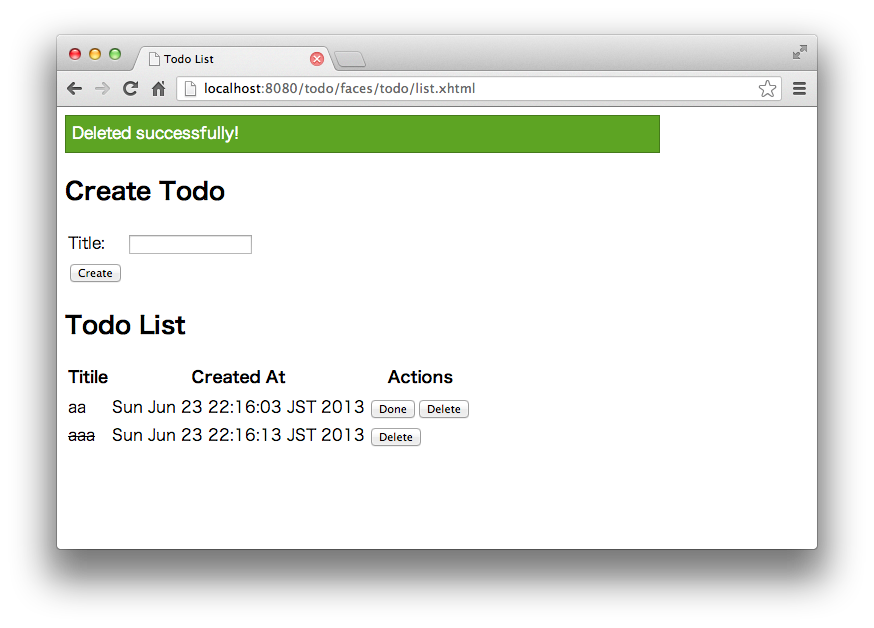
</h:form>

</h:column>

</h:dataTable>

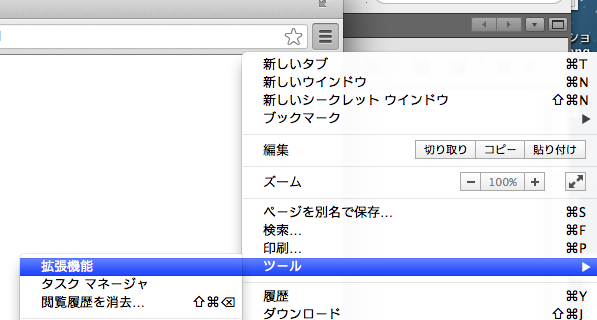
</h:body>

</html>

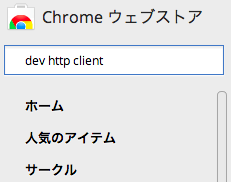


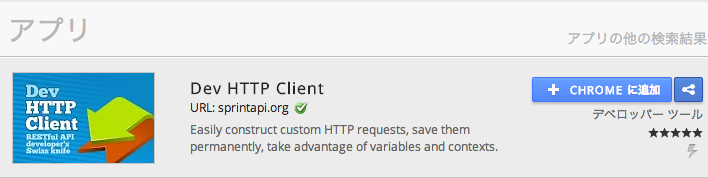
## JAX-RSでREST APIを作成

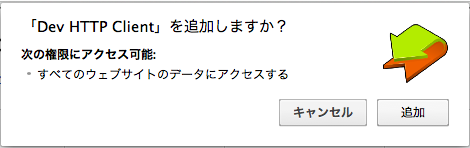
### Dev HTTP Clientのインストール

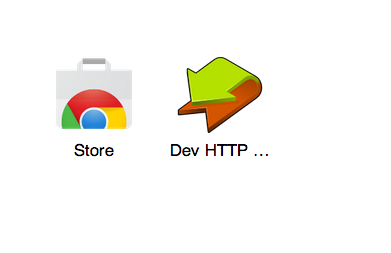




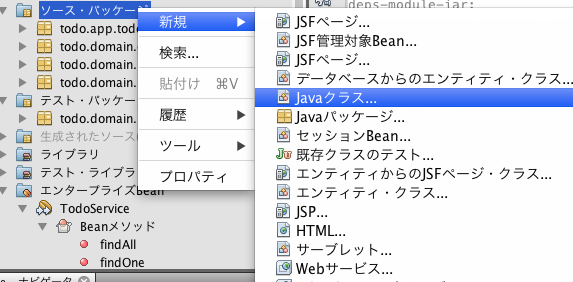


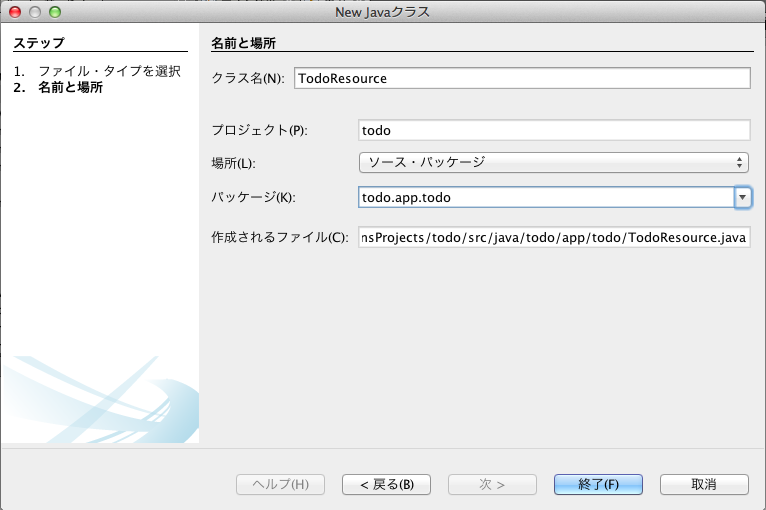


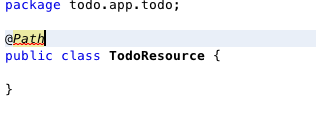




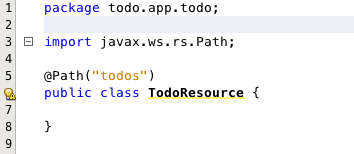
### リソースクラスの作成



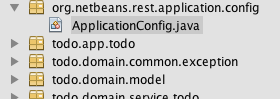


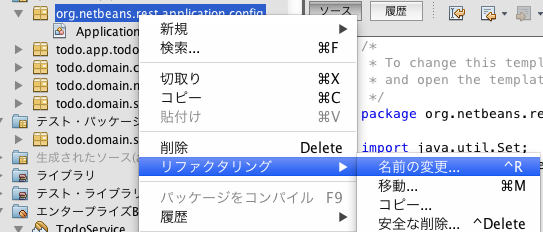


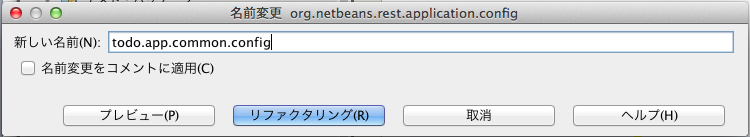




黄色のランプをクリックして「Java EE6仕様を使用してRESTを構成します」をクリック。







以下のクラスが自動生成される。

package todo.app.common.config;

import java.util.Set;

import javax.ws.rs.core.Application;

@javax.ws.rs.ApplicationPath("webresources")

public class ApplicationConfig extends Application {

@Override

public Set<Class<?>> getClasses() {

Set<Class<?>> resources = new java.util.HashSet<Class<?>>();

// following code can be used to customize Jersey 2.0 JSON provider:

try {

Class jsonProvider = Class.forName("org.glassfish.jersey.jackson.JacksonFeature");

// Class jsonProvider = Class.forName("org.glassfish.jersey.moxy.json.MoxyJsonFeature");

// Class jsonProvider = Class.forName("org.glassfish.jersey.jettison.JettisonFeature");

resources.add(jsonProvider);

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(getClass().getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

addRestResourceClasses(resources);

return resources;

}

/\*\*

\* Do not modify addRestResourceClasses() method.

\* It is automatically re-generated by NetBeans REST support to populate

\* given list with all resources defined in the project.

\*/

private void addRestResourceClasses(Set<Class<?>> resources) {

resources.add(todo.app.todo.TodoResource.class);

}

}

### GET Todos

#### TodoResourceの修正

package todo.app.todo;

import java.util.List;

import javax.ejb.EJB;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Path("todos")

public class TodoResource {

@EJB

protected TodoService todoService;

@GET

@Produces(MediaType.APPLICATION\_JSON)

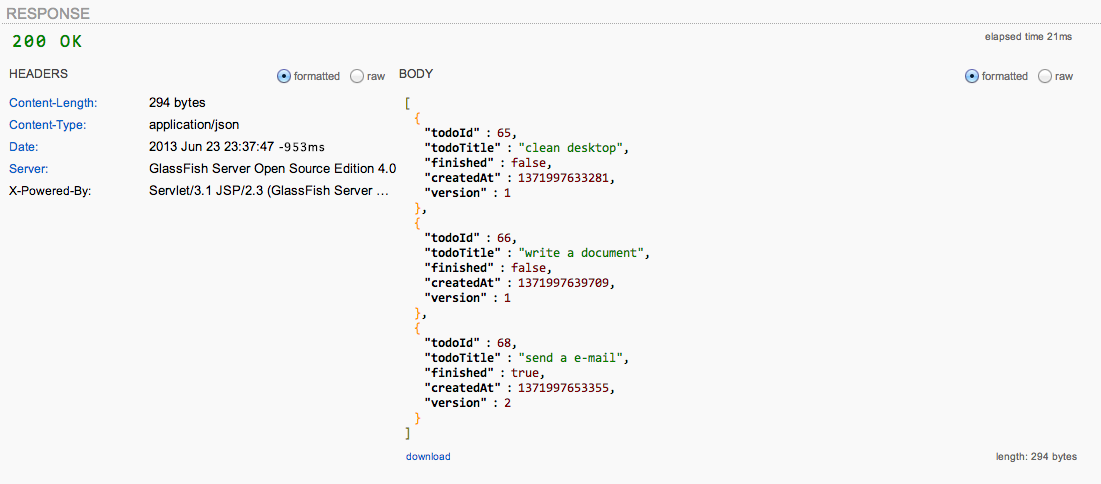
public List<Todo> getTodos() {

return todoService.findAll();

}

}





### GET Todo

#### TodoResourceの修正

package todo.app.todo;

import java.util.List;

import javax.ejb.EJB;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Path("todos")

public class TodoResource {

@EJB

protected TodoService todoService;

@GET

@Produces(MediaType.APPLICATION\_JSON)

public List<Todo> getTodos() {

return todoService.findAll();

}

@GET

@Path("{todoId}")

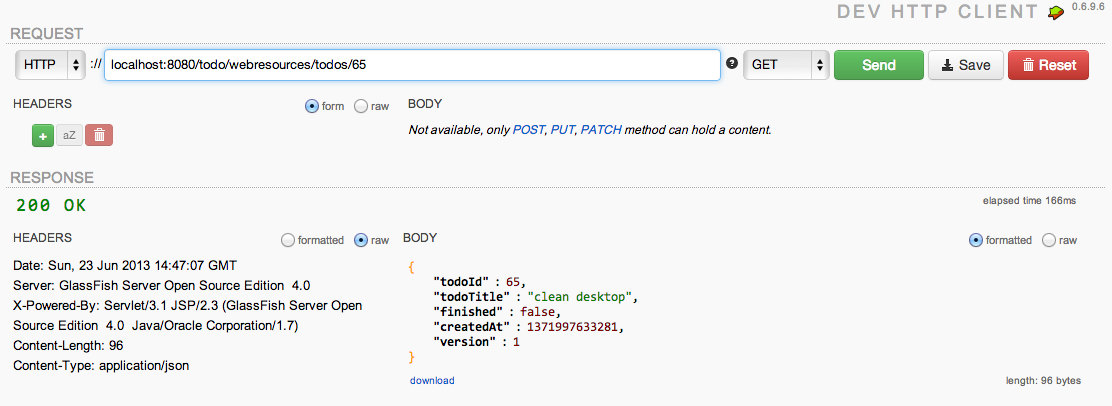
@Produces(MediaType.APPLICATION\_JSON)

public Todo getTodo(@PathParam("todoId") Integer todoId) {

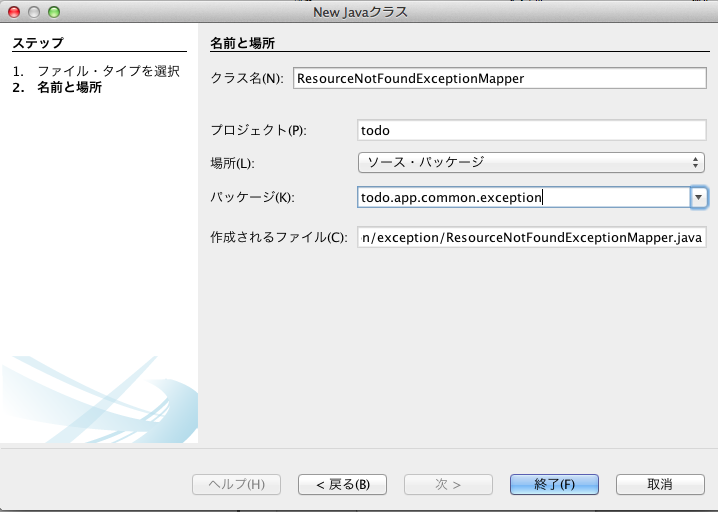
return todoService.findOne(todoId);

}

}



#### 例外ハンドラの作成



package todo.app.common.exception;

import javax.ws.rs.core.Response;

import javax.ws.rs.ext.ExceptionMapper;

import javax.ws.rs.ext.Provider;

import todo.domain.common.exception.ResourceNotFoundException;

@Provider

public class ResourceNotFoundExceptionMapper implements ExceptionMapper<ResourceNotFoundException> {

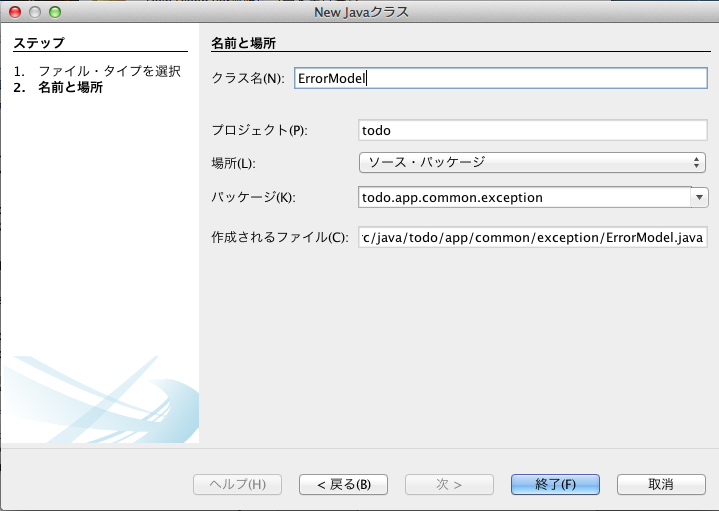
@Override

public Response toResponse(ResourceNotFoundException exception) {

throw new UnsupportedOperationException("Not supported yet.");

}

}



package todo.app.common.exception;

import java.util.List;

public class ErrorModel {

private final List<String> errorMessages;

public ErrorModel(List<String> errorMessages) {

this.errorMessages = errorMessages;

}

public List<String> getErrorMessages() {

return errorMessages;

}

}

package todo.app.common.exception;

import java.util.Arrays;

import javax.ws.rs.core.Response;

import javax.ws.rs.ext.ExceptionMapper;

import javax.ws.rs.ext.Provider;

import todo.domain.common.exception.ResourceNotFoundException;

@Provider

public class ResourceNotFoundExceptionMapper implements ExceptionMapper<ResourceNotFoundException> {

@Override

public Response toResponse(ResourceNotFoundException exception) {

ErrorModel errorModel = new ErrorModel(Arrays.asList(exception.getMessage()));

return Response.status(Response.Status.NOT\_FOUND)

.entity(errorModel).build();

}

}



### POST Todos

#### TodoResourceの修正

package todo.app.todo;

import java.net.URI;

import java.util.List;

import javax.ejb.EJB;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import javax.ws.rs.core.UriInfo;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Path("todos")

public class TodoResource {

@EJB

protected TodoService todoService;

@GET

@Produces(MediaType.APPLICATION\_JSON)

public List<Todo> getTodos() {

return todoService.findAll();

}

@GET

@Path("{todoId}")

@Produces(MediaType.APPLICATION\_JSON)

public Todo getTodo(@PathParam("todoId") Integer todoId) {

return todoService.findOne(todoId);

}

@POST

@Produces(MediaType.APPLICATION\_JSON)

public Response postTodos(Todo todo, @Context UriInfo uriInfo) {

Todo createdTodo = todoService.create(todo);

Integer todoId = createdTodo.getTodoId();

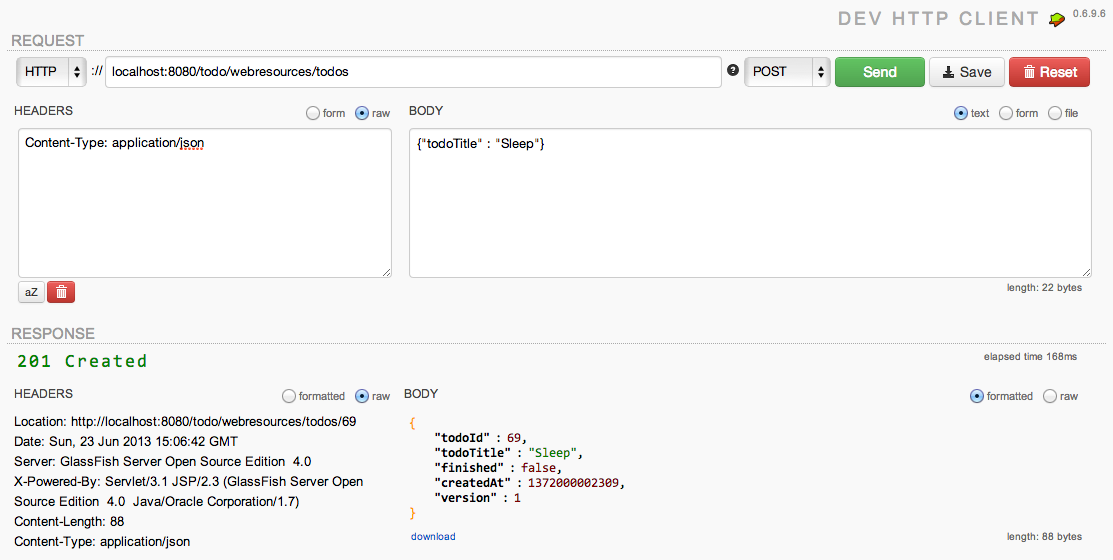
URI newUri = uriInfo.getRequestUriBuilder()

.path(todoId.toString()).build();

return Response.created(newUri).entity(createdTodo).build();

}

}



#### 例外ハンドラの追加

package todo.app.common.exception;

import java.util.Arrays;

import javax.ws.rs.core.Response;

import javax.ws.rs.ext.ExceptionMapper;

import javax.ws.rs.ext.Provider;

import todo.domain.common.exception.BusinessException;

@Provider

public class BusinessExceptionMapper implements ExceptionMapper<BusinessException> {

@Override

public Response toResponse(BusinessException exception) {

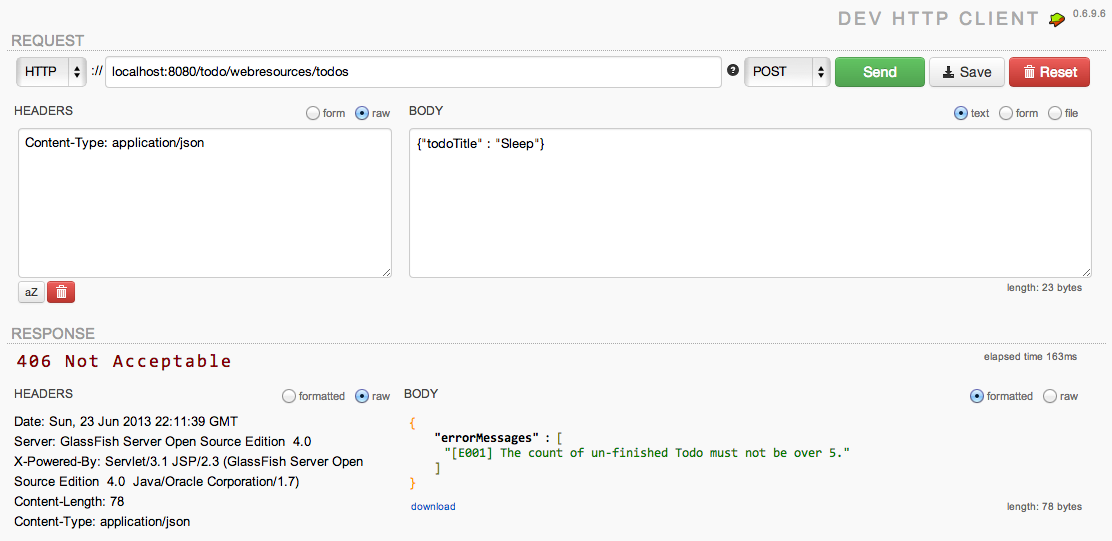
ErrorModel errorModel = new ErrorModel(Arrays.asList(exception.getMessage()));

return Response.status(Response.Status.NOT\_ACCEPTABLE)

.entity(errorModel).build();

}

}



#### 入力チェックの追加

### PUT Todos

#### TodoResourceの修正

package todo.app.todo;

import java.net.URI;

import java.util.List;

import javax.ejb.EJB;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.PUT;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import javax.ws.rs.core.UriInfo;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Path("todos")

public class TodoResource {

@EJB

protected TodoService todoService;

@GET

@Produces(MediaType.APPLICATION\_JSON)

public List<Todo> getTodos() {

return todoService.findAll();

}

@GET

@Path("{todoId}")

@Produces(MediaType.APPLICATION\_JSON)

public Todo getTodo(@PathParam("todoId") Integer todoId) {

return todoService.findOne(todoId);

}

@POST

@Produces(MediaType.APPLICATION\_JSON)

public Response postTodos(Todo todo, @Context UriInfo uriInfo) {

Todo createdTodo = todoService.create(todo);

Integer todoId = createdTodo.getTodoId();

URI newUri = uriInfo.getRequestUriBuilder()

.path(todoId.toString()).build();

return Response.created(newUri).entity(createdTodo).build();

}

@PUT

@Path("{todoId}")

@Produces(MediaType.APPLICATION\_JSON)

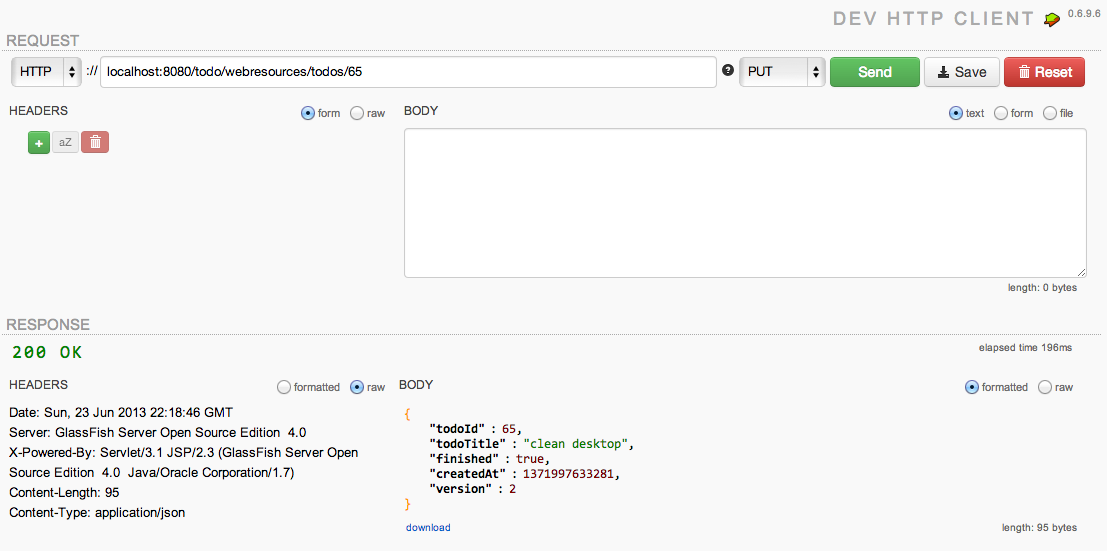
public Todo putTodos(@PathParam("todoId") Integer todoId) {

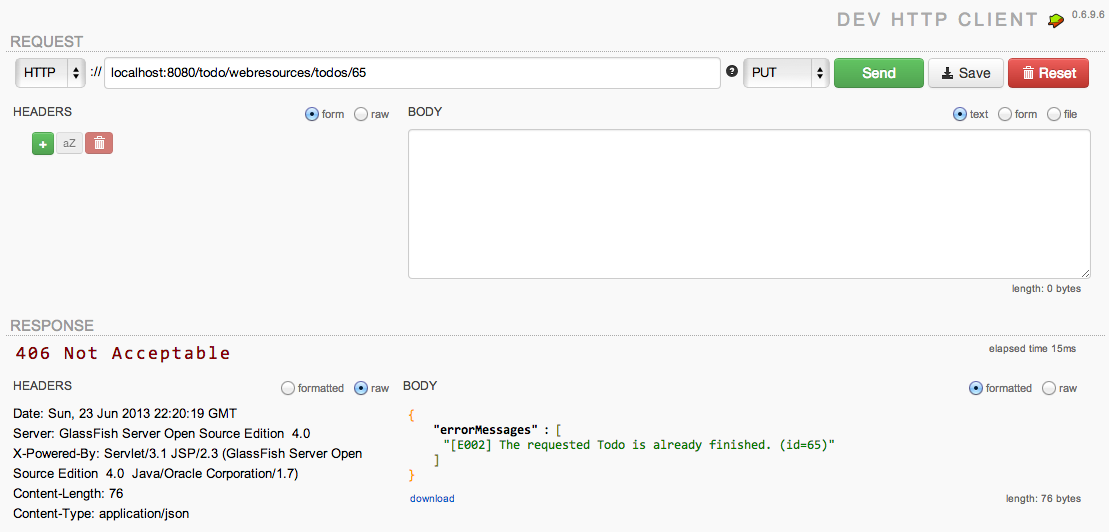
Todo todo = todoService.finish(todoId);

return todo;

}

}





### DELETE Todos

#### TodoResourceの修正

package todo.app.todo;

import java.net.URI;

import java.util.List;

import javax.ejb.EJB;

import javax.ws.rs.DELETE;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.PUT;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import javax.ws.rs.core.UriInfo;

import todo.domain.model.Todo;

import todo.domain.service.todo.TodoService;

@Path("todos")

public class TodoResource {

@EJB

protected TodoService todoService;

@GET

@Produces(MediaType.APPLICATION\_JSON)

public List<Todo> getTodos() {

return todoService.findAll();

}

@GET

@Path("{todoId}")

@Produces(MediaType.APPLICATION\_JSON)

public Todo getTodo(@PathParam("todoId") Integer todoId) {

return todoService.findOne(todoId);

}

@POST

@Produces(MediaType.APPLICATION\_JSON)

public Response postTodos(Todo todo, @Context UriInfo uriInfo) {

Todo createdTodo = todoService.create(todo);

Integer todoId = createdTodo.getTodoId();

URI newUri = uriInfo.getRequestUriBuilder()

.path(todoId.toString()).build();

return Response.created(newUri).entity(createdTodo).build();

}

@PUT

@Path("{todoId}")

@Produces(MediaType.APPLICATION\_JSON)

public Todo putTodos(@PathParam("todoId") Integer todoId) {

Todo todo = todoService.finish(todoId);

return todo;

}

@DELETE

@Path("{todoId}")

@Produces(MediaType.APPLICATION\_JSON)

public void deleteTodos(@PathParam("todoId") Integer todoId) {

todoService.delete(todoId);

}

}

