Prüfungscode

# Web.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="ISO-8859-1" standalone="no"***?>* <**web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="3.0"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"**>   <**display-name**>Basic Servlet Web Application</**display-name**>   <**servlet**>  <**servlet-name**>Basic Servlet</**servlet-name**>  <**servlet-class**>ch.fhnw.webfr.flashcard.web.BasicServlet</**servlet-class**>  </**servlet**>  <**servlet-mapping**>  <**servlet-name**>Basic Servlet</**servlet-name**>  <**url-pattern**>/app/\*</**url-pattern**>  </**servlet-mapping**>   <**context-param**>  <**param-name**>mode</**param-name**>  <**param-value**>test</**param-value**>  </**context-param**> </**web-app**> |

# BasicServlet.java

|  |
| --- |
| @SuppressWarnings(**"serial"**) @WebServlet(urlPatterns = {**"/\*"**}) **public class** BasicServlet **extends** HttpServlet {   **private** QuestionnaireRepository **questionnaireRepository**;   **protected void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** IOException {  response.setContentType(**"text/html; charset=utf-8"**);   String[] pathElements = request.getRequestURI().split(**"/"**);  **if** (isPathElementQuestionnare(pathElements)) {  handleQuestionnaireRequest(request, response, pathElements);  } **else if** (isLastPathElementQuestionnaires(pathElements)) {  handleQuestionnairesRequest(request, response);  } **else** {  handleIndexRequest(request, response);  }  }   **private boolean** isPathElementQuestionnare(String[] pathElements) {  **try** {  String questionnareString = pathElements[pathElements.**length** - 2];  Integer.*valueOf*(pathElements[pathElements.**length** - 1]);  **return** questionnareString.equals(**"questionnaires"**);  } **catch** (Exception exception) {  **return false**;  }  }   **private boolean** isLastPathElementQuestionnaires(String[] pathElements) {  **try** {  String last = pathElements[pathElements.**length** - 1];  **return** last.equals(**"questionnaires"**);  } **catch** (Exception exception) {  **return false**;  }  }   **private void** handleQuestionnaireRequest(HttpServletRequest request, HttpServletResponse response, String[] pathElements) **throws** IOException {  **try** {  String url = request.getContextPath() + request.getServletPath() + **"/questionnaires/"**;  Long questionnareIndex = Long.*valueOf*(pathElements[pathElements.**length** - 1]);  Questionnaire questionnaire = **questionnaireRepository**.findById(questionnareIndex);  PrintWriter writer = response.getWriter();  writer.append(**"<html><head><title>Example</title></head><body>"**);  writer.append(**"<h3>"** + questionnaire.getTitle() + **"</h3>"**);  writer.append(**"<p>"** + questionnaire.getDescription() + **"</p>"**);  writer.append(**"<p><a href='"** + response.encodeURL(url) + **"'>Back</a></p>"**);  writer.append(**"</body></html>"**);  } **catch** (Exception exception) {  String url = request.getContextPath() + request.getServletPath() + **"/questionnaires/"**;  PrintWriter writer = response.getWriter();  writer.append(**"<html><head><title>Example</title></head><body>"**);  writer.append(**"<h3>Error</h3>"**);  writer.append(**"<p>No questionnare found!</p>"**);  writer.append(**"<p><a href='"** + response.encodeURL(url) + **"'>Back</a></p>"**);  writer.append(**"</body></html>"**);  }  }   **private void** handleQuestionnairesRequest(HttpServletRequest request, HttpServletResponse response) **throws** IOException {  List<Questionnaire> questionnaires = **questionnaireRepository**.findAll();  PrintWriter writer = response.getWriter();  writer.append(**"<html><head><title>Example</title></head><body>"**);  writer.append(**"<h3>Fragebögen</h3>"**);  **for** (Questionnaire questionnaire : questionnaires) {  String url = request.getContextPath() + request.getServletPath();  url = url + **"/questionnaires/"** + questionnaire.getId().toString();  writer.append(**"<p><a href='"** + response.encodeURL(url) + **"'>"** + questionnaire.getTitle() + **"</a></p>"**);  }  writer.append(**"</body></html>"**);  }   **private void** handleIndexRequest(HttpServletRequest request, HttpServletResponse response) **throws** IOException {  PrintWriter writer = response.getWriter();  writer.append(**"<html><head><title>Example</title></head><body>"**);  writer.append(**"<h3>Welcome</h3>"**);  String url = request.getContextPath() + request.getServletPath();  writer.append(**"<p><a href='"** + response.encodeURL(url) + **"/questionnaires'>All Questionnaires</a></p>"**);  writer.append(**"</body></html>"**);  }   @Override  **public void** init(ServletConfig config) **throws** ServletException {  **super**.init(config);   ServletContext servletContext = config.getServletContext();  **questionnaireRepository** = (QuestionnaireRepository) servletContext.getAttribute(Globals.***KEY\_NAME\_REPOSITORY***);  } } |

# BasicListener.java

|  |
| --- |
| @WebListener() **public class** BasicListener **implements** ServletContextListener, HttpSessionListener, HttpSessionAttributeListener {   **private static** Logger *logger* = Logger.*getLogger*(BasicListener.**class**);   **public void** contextInitialized(ServletContextEvent servletContextEvent) {  ServletContext servletContext = servletContextEvent.getServletContext();  String mode = servletContext.getInitParameter(Globals.***KEY\_NAME\_MODE***);  QuestionnaireRepository questionnaireRepository = !mode.equalsIgnoreCase(Globals.***VALUE\_NAME\_MODE***) ? **new** QuestionnaireRepository() : **new** QuestionnaireInitializer().initRepoWithTestData();  servletContext.setAttribute(Globals.***KEY\_NAME\_REPOSITORY***, questionnaireRepository);  *logger*.debug(**"Repository initialized and set with mode: "** + mode);  *// Context is initialized* }   **public void** contextDestroyed(ServletContextEvent sce) {  *// Context is created* }   **public void** sessionCreated(HttpSessionEvent httpSessionEvent) {  *// Session is created.* }   **public void** sessionDestroyed(HttpSessionEvent httpSessionEvent) {  *// Session is destroyed.* }   **public void** attributeAdded(HttpSessionBindingEvent httpSessionBindingEvent) {  *//This method is called when an attribute is added to a session.* }   **public void** attributeRemoved(HttpSessionBindingEvent httpSessionBindingEvent) {  *// This method is called when an attribute is removed from a session.* }   **public void** attributeReplaced(HttpSessionBindingEvent httpSessionBindingEvent) {  *// This method is invoked when an attribute is replaced in a session.* } } |

# I18NFilter.java

|  |
| --- |
| @WebFilter(urlPatterns = {**"/\*"**}) **public class** I18NFilter **extends** HttpFilter {   **private static final** Logger ***logger*** = Logger.*getLogger*(I18NFilter.**class**);   **private** Properties **properties**;   @Override  **public void** init(FilterConfig filterConfig) **throws** ServletException {  ServletContext servletContext = filterConfig.getServletContext();   String fileName = servletContext.getInitParameter(Globals.***KEY\_MESSAGE\_FILENAME***);  **if** (fileName == **null** || fileName.isEmpty()) {  ***logger***.debug(**"No message filename found. Using fallback message filename "** + Globals.***VALUE\_MESSAGE\_FILENAME***);  fileName = Globals.***VALUE\_MESSAGE\_FILENAME***;  }   **try** (Reader reader = **new** ReaderUTF8(Thread.*currentThread*().getContextClassLoader().getResourceAsStream(fileName))) {  **properties** = **new** Properties();  **properties**.load(reader);  ***logger***.debug(**"I18N filter created"**);  } **catch** (IOException exception) {  **throw new** ServletException(exception.getMessage());  }  }   @Override  **public void** doFilter(HttpServletRequest servletRequest, HttpServletResponse servletResponse, FilterChain filterChain) **throws** IOException, ServletException {  I18NResponseWrapper i18NResponseWrapper = **new** I18NResponseWrapper(servletResponse);  filterChain.doFilter(servletRequest, i18NResponseWrapper);   Writer servletResponseWriter = servletResponse.getWriter();  **try** (BufferedReader bufferedWrapperReader = **new** BufferedReader(i18NResponseWrapper.getReader())) {  **for** (String line; (line = bufferedWrapperReader.readLine()) != **null**; ) {  servletResponseWriter.append(internationalizeLine(line));  }  }  servletResponseWriter.flush();  }   @Override  **public void** destroy() {  ***logger***.debug(**"I18N filter destroyed"**);  }   **private** String internationalizeLine(String line) {  **for** (Map.Entry<Object, Object> entry : **properties**.entrySet()) {  line = line.replace((String) entry.getKey(), (String) entry.getValue());  }  **return** line;  } } |

# I18ResponseWrapper.java

|  |
| --- |
| **public class** I18NResponseWrapper **extends** HttpServletResponseWrapper {   **private final** CharArrayWriter **writer**;   **public** I18NResponseWrapper(HttpServletResponse response) {  **super**(response);  **writer** = **new** CharArrayWriter();  }   @Override  **public** PrintWriter getWriter() {  **return new** PrintWriter(**writer**);  }   **public** Reader getReader() {  **writer**.flush();  **writer**.close();  **return new** CharArrayReader(**writer**.toCharArray());  } } |

# IndexController.java

|  |
| --- |
| @Controller @RequestMapping(**"/"**) **public class** IndexController **implements** ErrorController {   @GetMapping(**"/"**)  **public** ModelAndView showIndex() {  **return new** ModelAndView(**"redirect:questionnaires"**);  }   @GetMapping(**"/error"**)  **public** ModelAndView showError() {  **return new** ModelAndView(**"error"**);  }   @Override  **public** String getErrorPath() {  **return "/error"**;  } } |

# QuestionnaireController.java

|  |
| --- |
| @RestController @RequestMapping(**"/questionnaires"**) **public class** QuestionnaireController {   **private final** QuestionnaireRepository **questionnaireRepository**;   @Autowired  **public** QuestionnaireController(QuestionnaireRepository questionnaireRepository) {  **this**.**questionnaireRepository** = questionnaireRepository;  }   @GetMapping()  **public** ModelAndView findAll() {  ModelAndView modelAndView = **new** ModelAndView(**"list"**);  modelAndView.addObject(**"questionnaires"**, **questionnaireRepository**.findAll());  **return** modelAndView;  }   @GetMapping(**"/{id}"**)  **public** ModelAndView showQuestionnaire(@PathVariable(**"id"**) String id) {  Optional<Questionnaire> questionnaire = **questionnaireRepository**.findById(id);  **if** (questionnaire.isPresent()) {  ModelAndView modelAndView = **new** ModelAndView(**"show"**);  modelAndView.addObject(**"questionnaire"**, questionnaire.get());  **return** modelAndView;  } **else** {  **return new** ModelAndView(**"404"**);  }  }   @GetMapping(params = **"form"**)  **public** ModelAndView showCreateQuestionnairForm() {  ModelAndView modelAndView = **new** ModelAndView(**"create"**);  modelAndView.addObject(**"questionnaire"**, **new** Questionnaire());  **return** modelAndView;  }   @PostMapping()  **public** ModelAndView createQuestionnaire(@Valid Questionnaire questionnaire, BindingResult bindingResult) {  **if** (!bindingResult.hasErrors()) {  **questionnaireRepository**.save(questionnaire);  **return new** ModelAndView(**"redirect:/questionnaires"**);  } **else** {  **return new** ModelAndView(**"create"**);  }  }   @DeleteMapping(**"/{id}"**)  **public** ModelAndView deleteQuestionnaire(@PathVariable(**"id"**) String id) {  Optional<Questionnaire> questionnaire = **questionnaireRepository**.findById(id);  **if** (questionnaire.isPresent()) {  **questionnaireRepository**.deleteById(id);  **return new** ModelAndView(**"redirect:/questionnaires"**);  } **else** {  **return new** ModelAndView(**"404"**);  }  }   @GetMapping(**"/update/{id}"**)  **public** ModelAndView showUpdateQuestionnairForm(@PathVariable(**"id"**) String id) {  Optional<Questionnaire> questionnaire = **questionnaireRepository**.findById(id);  **if** (questionnaire.isPresent()) {  ModelAndView modelAndView = **new** ModelAndView(**"update"**);  modelAndView.addObject(**"questionnaire"**, questionnaire.get());  **return** modelAndView;  } **else** {  **return new** ModelAndView(**"404"**);  }  }   @PutMapping(**"/{id}"**)  **public** ModelAndView updateQuestionnaire(@PathVariable(**"id"**) String id, @Valid Questionnaire questionnaire, BindingResult bindingResult) {  **if** (bindingResult.hasErrors()) {  **return new** ModelAndView(**"update"**);  }   Optional<Questionnaire> questionnaire1 = **questionnaireRepository**.findById(id);  **if** (questionnaire1.isPresent()) {  **questionnaireRepository**.save(questionnaire);  **return new** ModelAndView(**"redirect:/questionnaires"**);  } **else** {  **return new** ModelAndView(**"404"**);  }  } } |

# Questionnaire.java

|  |
| --- |
| @Document(collection = **"questionnaires"**) **public class** Questionnaire {   @Id  **private** String **id**;   @NotNull  @Size(min = 2, max = 30)  **private** String **title**;   @NotNull  @Size(min = 10, max = 50)  **private** String **description**;   *// Dummy constructor* **public** Questionnaire() {  }   **public** Questionnaire(String title, String description) {  **this**.**title** = title;  **this**.**description** = description;  }   **public** String getId() {  **return id**;  }   **public void** setId(String id) {  **this**.**id** = id;  }   **public** String getTitle() {  **return title**;  }   **public void** setTitle(String title) {  **this**.**title** = title;  }   **public** String getDescription() {  **return description**;  }   **public void** setDescription(String description) {  **this**.**description** = description;  } } |

# layout.html

|  |
| --- |
| <!DOCTYPE **html**> <**html xmlns:th="http://www.thymeleaf.org" th:fragment="maincontent(content)"**> <**head**>  <**title**>Flashcard App</**title**>  <**link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous"**> </**head**> <**body**>  <**div class="container-fluid"**>  <**div th:replace="~{header :: header}"**>Fake footer</**div**>  <**div th:replace="${content}"**>Fake content</**div**>  <**div th:replace="~{footer :: footer}"**>Fake footer</**div**>  </**div**> </**body**> </**html**> |

# list.html

|  |
| --- |
| <!DOCTYPE **html**> <**html th:replace="~{layout :: maincontent(~{::section})}"**>  <**head**>  <**title**>flashcard</**title**>  <**link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"  crossorigin="anonymous"**> </**head**>  <**body**>  <**section**>  <**a class="btn btn-success float-right" href="?form"**>  Add Questionnaire  </**a**>  <**h3**>Questionnaires</**h3**>  <**div th:if="${questionnaires.isEmpty()}"**>  <**h2 style="text-align**: **center"**>No Questionnaires found</**h2**>  </**div**>  <**div style="padding-top**: 10**px**;**"**>  <**table class="table table-condensed table-hover"**>  <**tbody**>  <**tr th:each="questionnaire : ${questionnaires}"**>  <**td colspan="1" style="vertical-align**: **middle**;**" th:text="${questionnaire.id}"** />  <**td colspan="3" style="vertical-align**: **middle**;**" th:text="${questionnaire.title}"** />  <**td colspan="10" style="vertical-align**: **middle**;**" th:text="${questionnaire.description}"** />  <**td**>  <**div class="btn-group float-right" role="group"**>  <**div**>  <**form action="#" th:action="@{/questionnaires} + '/' + ${questionnaire.id}" method="POST"**>  <**input type="hidden" name="\_method" value="DELETE"**/>  <**button type="submit" class="btn btn-danger" style="margin-left**: 10**px"**>  Delete  </**button**>  </**form**>  </**div**>  </**div**>  <**div class="btn-group float-right" role="group"**>  <**div**>  <**form action="#" th:action="@{/questionnaires/update} + '/' + ${questionnaire.id}"**>  <**button type="submit" class="btn btn-secondary" style="margin-left**: 10**px"**>  Update  </**button**>  </**form**>  </**div**>  </**div**>  <**div class="btn-group float-right" role="group"**>  <**div**>  <**form action="#" th:action="@{/questionnaires} + '/' + ${questionnaire.id}"**>  <**button type="submit" class="btn btn-primary" style="margin-left**: 10**px"**>  Show  </**button**>  </**form**>  </**div**>  </**div**>  </**td**>  </**tr**>  </**tbody**>  </**table**>  </**div**>  </**section**>  </**body**>  </**html**> |

# create.html

|  |
| --- |
| <!DOCTYPE **html**> <**html th:replace="~{layout :: maincontent(~{::form})}"**>  <**head**>  <**title**>flashcard show</**title**>  <**link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"  crossorigin="anonymous"**> </**head**>  <**body**> <**form th:action="@{/questionnaires}" th:object="${questionnaire}" method="post"**>  <**div class="form-group row"**>  <**label class="col-md-2" for="title"**>Title:</**label**>  <**div class="col-md-10"**>  <**input type="text" id="title" class="form-control" th:field="\*{title}" autofocus="autofocus"** />  <**div th:if="${#fields.hasErrors('title')}" th:errors="\*{title}" class="text-danger"**>Title error</**div**>  </**div**>  </**div**>  <**div class="form-group row"**>  <**label class="col-md-2" for="description"**>Description:</**label**>  <**div class="col-md-10"**>  <**input type="text" id="description" class="form-control" th:field="\*{description}"** />  <**div th:if="${#fields.hasErrors('description')}" th:errors="\*{description}" class="text-danger"**>Description Error</**div**>  </**div**>  </**div**>  <**div class="form-group row"**>  <**a class="btn btn-default" th:href="@{/questionnaires}"**>Cancel</**a**>  <**button type="submit" class="btn btn-primary"**>Save</**button**>  </**div**> </**form**> </**body**>  </**html**> |

# update.html

|  |
| --- |
| <!DOCTYPE **html**> <**html th:replace="~{layout :: maincontent(~{::form})}"**>  <**head**>  <**title**>flashcard show</**title**>  <**link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"  crossorigin="anonymous"**> </**head**>  <**body**> <**form th:action="@{/questionnaires} + '/' + ${questionnaire.id}" th:object="${questionnaire}" th:method="PUT"**>  <**div class="form-group row"**>  <**label class="col-md-2" for="title"**>Title:</**label**>  <**div class="col-md-10"**>  <**input type="text" id="title" class="form-control" th:field="\*{title}" autofocus="autofocus"** />  <**div th:if="${#fields.hasErrors('title')}" th:errors="\*{title}" class="text-danger"**>Title error</**div**>  </**div**>  </**div**>  <**div class="form-group row"**>  <**label class="col-md-2" for="description"**>Description:</**label**>  <**div class="col-md-10"**>  <**input type="text" id="description" class="form-control" th:field="\*{description}"** />  <**div th:if="${#fields.hasErrors('description')}" th:errors="\*{description}" class="text-danger"**>Description Error</**div**>  </**div**>  </**div**>  <**div class="form-group row"**>  <**a class="btn btn-default" th:href="@{/questionnaires}"**>Cancel</**a**>  <**button type="submit" class="btn btn-primary"**>Save</**button**>  </**div**> </**form**> </**body**>  </**html**> |