Step 1 Minimum dynamic spring web project (Spring tutorial #50 – 54) as follows:

Clone copy of this MVC dynamic web app creation guide. Delete all but steps 1 and 2. Global replace 'inv01' to the new project name.

1. File → new → dynamic web project; name=inv01. Ensure desired tomcat server is selected and Dynamic Web version=3.0. [May have to set up Tomcat to have it appear on server pick list.] Click Next → Next → tick 'Generate web.xml deployment descriptor' checkbox. Finish.
2. Convert to a Maven enabled project; right click project inv01 (project level)→ configure → convert to Maven; project group id = org.friendlytutor.dev.inv01 (top package name; inv01 = same as artifact id) artifact id = inv01 (inv01 is the project or program section name [war or jar file?]
3. If want to test current configuration: project → new JSP File; File name=inv01.jsp; Default location will be WebContent/WEB-INF/inv01.jsp. Ensure jsp header is HTML5 structure. `<!DOCTYPE html> <meta charset="UTF-8">` Edit inv01.jsp and insert hello in the <body>; right click inv01.jsp → Run As → Run on Server; select Tomcat server, check 'always use this server' if offered.
   (Can use http://validator.w3.org/ to validate any new or updated jsp files by copy/paste entire contents to the direct input text area to check the html5 syntax.)
4. Create git repository and do initial local git master commit. Copy gitignore from working project into the project directory. Instructions are for Mac OS X from terminal window.
   1. $ cd your_project_dir
   2. $ git init
   3. $ git add *
   4. $ git status
   5. $ git config --global user.name "Brent Bingham"
   6. $ git config --global user.email bvbgrad@gmail.com
   7. $ git commit -m 'Initial Commit'
   8. $ git remote add origin ssh://git@redmine.friendly-tutor.com/inv.git
5. Tell eclipse this is a git project; project → Team → Share and finish (activates git tracking for this project.) project → Refresh to ensure workspace and file space is synchronized.
6. Open pom,xml/dependencies. [may need to eventually switch from war to jar file in overview tab] (first time in a new workspace, had to build the maven index)
   Dependencies tab: type groupid=org.springframework to find and be able to select 'spring-core'. Repeat for 'spring-beans', 'spring-context', 'spring-jdbc', 'spring-web', and 'spring- webmvc'. (Springifies the project) Ensure all dependencies are the same version. Save and confirm project still deploys correctly.
7. MVC routes requests via a dispatcher servlet. Add inv01 project level → new → [other → Web] → Servlet to the project. Tick check box "Use an existing Servlet class or JSP"; Browse, use autocomplete to find - select DispatcherServlet (available if spring web & mvc jars above have been added properly). Finish.
8. Edit inv01/WebContent/WEB-INF/web.xml <servlet>. Add description if desired. Global replace 'DispatcherServlet' with 'inv01' except for <servlet-class>, which has to continue to identify the DispatcherServlet class. Add `<load-on-startup>1</load-on-startup>` in <servlet> section so inv01-servlet.xml is the default servlet that is run upon program start up. Change url pattern to just '/'.
9. Add servlet context file to WEB-INF folder by right click WEB-INF → new → other → Spring → SpringBeanConfiguration file → Next. The servlet File name must be exactly the project name, inv01-servlet.xml, i.e. test01-servlet.xml. Confirm get a 404 error when run at the project level → Run As → Run on Server. If get exception error, confirm the java build project properties Deployment Assembly includes Maven dependencies (Add Java Build Path Entries → Maven dependencies) and retest.
10. Add a controller class in a 'controllers' package [foundation structure for future controller component]; right click project level → new class; change Package=org.friendlytutor.inv01.controllers, Name=Inv01Controller (`public class inv01Controller {}`), since this will be the 'main' or 'home' controller. Notice initial capital letter 'I' to conform to class naming convention. Finish. Annotate this

controller class as a `@Controller`. Quick fix; import Controller reference. Add `public String showHome() {}` method annotated with `@RequestMapping("/")` and `return "inv01";` Import RequestMapping reference.

11. Tell Spring to use this controller as a bean by configuring inv01-servlet.xml. Click namespace tab and tick the context and mvc check boxes so these name spaces are created. The beans namespace should already be checked. In context tab; right click beans; insert component-scan with controllers fully qualified package filepath as the base package (copy controller package Qualified Name). In mvc tab; right click beans; insert "mvc:annotation-driven" element (lets javaEE 'automate' the wiring).
12. Going to start creating the views for the dispatcher servlet-mapping. First create a 'jsps' (views) folder under WEB-INF. If inv01.jsp created in step 3, move it into this folder. Otherwise, create it now.
13. Simplest view class using jsp files is the 'InternalResourceViewResolver.class' found in "inv01/Java Resources/Libraries/Maven dependencies/...webmvc/...web.servlet.view". Right click "...webmvc/servlet.view" and 'copy qualified name'. Edit inv01-servlet.xml to add new bean on the beans tab with id=jspViewResolver (name doesn't really matter). Class=Paste the servlet.view qualified name, then browse and add ".InternalResourceViewResolver". Matching items should quickly resolve to the "InternalResourceViewResolver - org.springframework.web.servlet.view" class. Finish.
14. Configure prefix and suffix properties for the jspViewResolver bean; right click jspViewResolver → beans → insert property element, name=prefix, value =/WEB-INF/jsps/ (designator where to find the jsp files). Insert a second property element, name=suffix and value =.jsp (adds .jsp to controller returned page names). Test deployment by inv01(project level) → Run As → Run on Server. From this point forward, if Project/Build Automatically is checked, Eclipse will attempt to do a continuous build and deployment every time a project element is updated and saved. If doeOccasionally this process will cause an out of memory error on the tomcat working directory. If this happens, or whenever want to ensure there is a fresh full deployment, do three step process; 1) Project/clean, and 2) right click Tomcat Server → Clean Tomcat Work Directory ... 3) Run As → Run on Server. Check console for any errors.
15. Need to override default browser styling. Establish location for style sheets. Create static resources folder structure under inv01/WebContent. Right click WebContent → new → folder; name:=resources. Right click resources folder (3 times) → new → folder; name=css, images, and scripts. Edit inv01-servlet.xml mvc tab; right click 'beans', insert mvc:resources; location:= /resources/ (relative to WebContent), mapping:=/static/** ('static' is consistent with Spring tutorial. Actually 'static' could be any desired mapping variable name. ** means scan all sub folders.) Copy reset.css, style.css, and sprites.css into resources/css folder. Add link in the header section in inv01.jsp to the reset.css, the style.css, and the sprites.css style sheets (sequence is important). (Ref: Sprint tutorial #68) In images, copy sprites.png. The sprites.css style sheet references this sprites.png file. In scripts, copy html5shiv-printshiv.js. Eventually will put other javascript external files in this folder. There is also a companion html5shiv.js link to a google file. These shiv .js files allow IE8 to render the page in spite of the html5 tags as well as print the page nicely.

Step 2 Section 4: Bare bones Html;
use jsp pages inside Spring MVC project rather than work in Tomcat directory

1. Edited initial home page inv01.jsp. Changed jsp default header to HTML5 structure. `<!DOCTYPE html>` `<meta charset="UTF-8">` Used http://validator.w3.org/ validate by direct input to check the html5 syntax.
2. For each new jsp file, i.e. web page, have to provide a handler in the inventorysController.