action="/newPatient"

<c:if test="${errors ne null}">
display errors
</c:if>

patient name: john
dob: 10/13/2001
gender: M: F:
mobile: abc83
email address: john
street address:
city:
state:
zip: hhj
country:

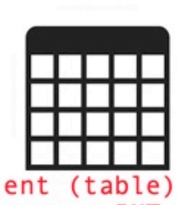
add

new-patient-form.jsp
(source page)

```
@WebServlet("/newPatient")
class PatientFormRegistrationServlet extends HttpServlet {
  public void service(req, resp) {
    String page = null;
    Map<String, Object> dataMap = new HashMap<>();

    PatientForm form = GenericWrapper.wrap(req,
PatientForm.class);

    PatientFormValidator validator = new
PatientFormValidator();
    validator.validate(form);


    if(validator.hasErrors()) {
      dataMap.put("errors", form.getErrors());
      dataMap.put("patientForm", form);
      ViewDispatcher.dispatch(req, resp, "new-patient",
dataMap);
    } else {
      PatientManagementService pmService = new
PatientManagementService();
      int patientNo = pmService.newPatient(form);

      dataMap.put("patientNo", patientNo);
      ViewDispatcher.dispatch(req, resp, "new-patient-
success", dataMap);

    }
  }
}
```

```
[module]
class PatientManagementService {

  int newPatient(PatientForm patientForm) {
    // business logic
    // invoke dao
  }
}
```

new-patient-success.jsp
(target page)



```
patient (table)
patient_no INT AI
patient_nm VARCHAR
dob DATE
gender VARCHAR
mobile_nbr VARCHAR
email_address VARCHAR
…
zip_code INT
```

```
class PatientForm {
  String patientName;
  String dob;
  String gender;
  String mobileNo;
  String emailAddress;
  String streetAddress;
  String city;
  String state;
  String zip;
  String country;
  // accessors
}
```

```
// jsp custom tags
<form action="/newPatient" method="post">
  <c:choose>
    <c:when test="${patientForm ne null">
     Patient Name: <input type="text" name="patientName" value="${patientForm.patientName}"/>
     Dob: <input type="text" name="dob" value="${patientForm.dob}"/>
     Gender: Male: <input type="radio" name="gender" value="M"/> or
             Female: <input type="radio" name="gender" value="F"/>
     Mobile No: <input type="text" name="mobileNo" value="${patientForm.mobileNo}"/>
    </c:when>
    <c:otherwise>
     Patient Name: <input type="text" name="patientName"/>
     Dob: <input type="text" name="dob"/>
     Gender: Male: <input type="radio" name="gender" value="M"/> or
             Female: <input type="radio" name="gender" value="F"/>
     Mobile No: <input type="text" name="mobileNo"/>
    </c:otherwise>
  </c:choose>
  …
  <input type="submit" value="add"/>
</form>
```

```
class PatientFormValidator extends Validator {

  public void validate(Object obj) {
    PatientForm form = (PatientForm) obj;

    SimpleDateFormat sdf = new SimpleDateForm("dd/MM/yyyy");
    try {
      sdf.parse(form.getDob());
    }catch(ParseException e) {
      rejectValue("dob", "dob must be in past");
    }

    try {
      Integer.parseInt(form.getZip());
    }catch(NumberFormatException e) {
      rejectValue("zip", "zip should be minimum of 5 digits");
    }

    if(form.getMobileNo() == null || form.getMobileNo().trim().equals("")
       || form.getMobileNo().trim().length != 10) {
      rejectValue("mobileNo", "mobile no should be 10 digits");
    }
  }
}
```

```
outcomes.properties
--------------------
patient-info=/patient-info.jsp
new-patient-added=/new-patient-success.jsp
new-patient=/new-patient-form.jsp
```

```
class GenericWrapper {
  Object wrap(HttpServletRequest req, Class<?> clazz) {
    Object obj = clazz.newInstance();

    reqMap.forEach((paramName, paramValue) -> {
    try {
        Field field = clazz.getDeclaredField(paramName);
        String setterName = "set" +
String.valueOf(paramName.charAt(0)).toUpperCase() +
paramName.substring(1);
        Method method = clazz.getDeclaredMethod(setterName,
String.class);
        method.invoke(obj, paramValue[0]);

    } catch (NoSuchFieldException e) {
        // ignore
    } catch (NoSuchMethodException e) {
        throw new RuntimeException(e);
    } catch (InvocationTargetException e) {
        throw new RuntimeException(e);
    } catch (IllegalAccessException e) {
        throw new RuntimeException(e);
    }
  });

  return obj;

  }
}
```

```
class ViewDispatcher {
  void dispatch(HttpServletRequest req, HttpServletResponse
resp, String outcome, Map<String, Object> dataMap) {
    Properties props = new Properties();
    props.load(this.getClass().getClassLoader()
         .getResourceAsStream("outcome.properties"));

    dataMap.forEach((key, val)-> {
      req.setAttribute(key, val);
    });
    String page = props.get(outcome);
    req.getRequestDispatcher(page).forward(req, resp);
  }
}
```