SERVLETS

Atenção para a diferença entre maiúsculas e minúsculas nos nomes de arquivos e classes!

- páginas estáticas *versus* páginas dinâmicas:
  1) páginas estáticas: conteúdos escritos em HTML;
  2) páginas dinâmicas: páginas HTML geradas dinamicamente baseadas nas requisições dos usuários;
- **CGI** (1993): independente de linguagem, primeiro padrão para geração de páginas dinâmicas. Estabelece uma interface bem definida entre programas executáveis e o servidor web.
- **Servlets** (1997): primeiro padrão específico para Java que permite a criação de páginas dinâmicas;
- Crie um *Dynamic Web Project* no Eclipse conforme instruções da aula passada. Dê o nome de **`aula5`** ao projeto.
- Crie a página **`index.html`** (clique com o botão direito sobre o projeto e escolha New → HTML File). Edite o arquivo como se segue:

```
<!DOCTYPE html>
<html>
<head>
	<title>Sistema de Gestão Acadêmica</title>
</head>
<body>
	<h1>Sistema de Gestão Acadêmica da UFABC</h1>
	<form action="Login">
		<p><input type="text" name="usuario"></p>
		<p><input type="password" name="senha"></p>
		<p><input type="submit" value="Entrar"></p>
	</form>
</body>
</html>
```

- Execute o projeto (Run As → Run on Server) e abra a página http://localhost:8080/aula5 no navegador web (preferencialmente o Firefox)
- Crie a página **`admin.html`**:

```
<!DOCTYPE html>
<html>
<head>
	<title>Sistema de Gestão Acadêmica</title>
</head>
<body>
	<h1>Sistema de Gestão Acadêmica da UFABC</h1>
	<p>Bem-vindo!</p>
	<ul>
		<li><a href="ListaAlunos">Lista de Alunos</a></li>
		<li><a href="forminserealuno.html">Insere Aluno</a></li>
		<li><a href="Logout">Sair</a></li>
	</ul>
</body>
</html>
```

- Vamos criar um servlet de nome **`Login`**, que autentique o usuário "administrador" com senha "12345" e redirecione o navegador para a página **`admin.html`** se o nome de usuário e senha foram digitados corretamente, ou apresente uma mensagem de erro se houve erro na autenticação. Para criar o servlet:
  1. clique com o botão direito no nome do projeto e escolha *New* → *Servlet*
  2. Complete as informações *Java package* com **`br.edu.ufabc.prograd.servlet`** e *Class name* com **`Login`**. Deixe o resto como está e clique *Next*.
  3. Observe que em URL mappings deve haver um item */Login*. Isto indica para o Tomcat que o endereço http://localhost:8080/Login será mapeado para a classe `Login`. Clique *Finish*.
- Coloque este código no servlet **`Login`**:

```
package br.edu.ufabc.prograd.servlet;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

@WebServlet("/Login")
public class Login extends HttpServlet {
	private static final long serialVersionUID = 1L;

	protected void service(HttpServletRequest request,
				HttpServletResponse response)
	throws ServletException, IOException {
		String usuario = request.getParameter("usuario");
		String senha = request.getParameter("senha");

		if ("administrador".equals(usuario) && "12345".equals(senha))
		{
			HttpSession session = request.getSession();
			session.setAttribute("usuario", "administrador");
			response.sendRedirect("admin.html");
		}
		else
		{
			PrintWriter out = response.getWriter();
			out.println("<!DOCTYPE html>");
			out.println("<html>");
			out.println("<head>");
			out.println("  <title>Erro de Login</title>");
			out.println("</head>");
			out.println("<body>");
			out.println("  <h1>Erro de login!</h1>");
			out.println("  <p>Usuário " + usuario +
					" não existe ou senha inválida</p>");
			out.println("</body>");
			out.println("</html>");
		}
	}
}
```

- Detalhes importantes sobre o servlet **`Login`** e a página **`index.html`**:
  1) Se o Eclipse reclamar que a classe `HttpServlet` não está definida, é porque a versão da JDK instalada não tem as extensões Java EE. Isso pode ser corrigido instalando-se uma JDK com essas extensões ou, alternativamente, indicando-se para o Eclipse onde é possível encontrar uma biblioteca que implemente a Servlet API (classes no pacote `javax.servlet`). Felizmente, o Apache Tomcat possui uma implementação no arquivo **`servlet-api.jar`**. Podemos indicar para o Eclipse onde ele está localizado clicando-se com o botão direito sobre o nome do projeto, escolhendo *Build Path* → *Configure Build Path* → *Add External JARs* e indo até o diretório **`lib`** do Apache Tomcat.
  2) A linha `@WebServlet("/Login")` é uma anotação (*Java annotation*), e indica ao Eclipse que ele deve informar ao Tomcat que essa classe deve sempre ser mapeada para a URL http://EnderecoDoServidor/NomeDoProjeto/**Login**. Também é fundamental que a linha `import javax.servlet.annotation.WebServlet` esteja presente no arquivo para que essa anotação seja entendida pelo Eclipse.
  3) Todo servlet deve sempre ser uma subclasse de `HttpServlet`.
  4) O campo `serialVersionUID` não é obrigatório, mas é importante colocá-lo para que o Tomcat possa recarregar corretamente a classe caso o programador resolva modificá-la. Ele deve ser incrementado a cada alteração da classe. Para mais informações, acesse http://c2.com/cgi/wiki?AlwaysDeclareSerialVersionUid.
  5) O método `service` é executado pelo Tomcat quando algum navegador acessa a URI indicada na anotação `@WebServlet`. O Tomcat passa um objeto da classe `HttpServletRequest` contendo todas as informações enviadas pelo navegador *web*, tais como endereço acessado, parâmetros passados em formulários, e muitas outras. Também é passado um objeto da classe `HttpServletResponse` que deve ser usado pelo programador para retornar informações do servlet para o navegador do usuário.
  6) O método `getSession()` da classe `HttpServletRequest` cria uma sessão de usuário, caso ela ainda não exista, e retorna um objeto representando a sessão do usuário. Para maiores informações, ver a documnentação da classe `HttpSession`.
  7) A autenticação feita neste exemplo foi simplificada para uma simples comparação de strings com o nome de usuário e senha introduzidos diretamente no código. Na prática, o que se deve fazer é: a) armazenar os nomes de usuário e as senhas num banco de dados; b) **nunca** armazenar senhas em texto puro, devendo-se usar algoritmos específicos para *hashing* seguro de senhas. Hoje em dia (nov/2013), são considerados como seguros: bcrypt, scrypt (verificação formal pendente) e PBKDF2. Já são considerados inseguros: *UNIX crypt* e qualquer hash como *MD5* ou *SHA* sem o uso de *salt*.

8) Faça login com um nome de usuário e senha incorretos e veja que ambos ficam presentes na barra de endereço do navegador, sendo armazenados junto com o histórico de navegação. Isso é uma falha de segurança inerente ao método padrão (método *GET*) de envio de parâmetros de um formulário. Na maioria dos casos, deve-se usar o método *POST*, que faz com que os parâmetros sejam enviados separadamente da URL. Para usar o método *POST*, edite o arquivo **`index.html`** e coloque `method="post"` dentro da tag `form`, como se segue: `<form action="Login" method="post">`

9) No html dinâmico gerado pelo servlet, a string `usuario` foi retornada para o navegador sem nenhum tratamento. **Isto nunca deve ser feito numa aplicação web profissional, pois a deixa vulnerável a um ataque chamado Cross Site Scripting (XSS)!** Note que, em html, caracteres como "<" e ">" possuem significado especial, e podem criar falhas de segurança. Volte para o endereço http://localhost:8080/aula5/ , digite `<h1>teste</h1>` no nome de usuário, clique "Entrar" e perceba o que aconteceu.
   - Nunca se deve confiar em nenhum dado que vem do usuário. Todo dado que vem do usuário deve ser "higienizado" antes de ser usado.
   - Existem bibliotecas e métodos específicos para transformar uma string em html seguro; essa transformação se chama *HTML escaping*. Ex.: método `escapeHTML(String str)` da classe **`org.apache.commons.lang.StringEscapeUtils`** da biblioteca Apache Commons. Apesar de *parecer* simples de se implementar uma classe para fazer escaping, **sempre** use bibliotecas consagradas para isso, pois há inúmeros detalhes a se considerar.
   - Servlets, por serem muito limitados, exigem que o programador tenha que se preocupar com *escaping*. Existem outros meios para geração de páginas dinâmicas em Java que já fazem essa e muitas outras tarefas automaticamente

- crie a página **`forminserealuno.html`** com o conteúdo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Inserção de aluno</title>
</head>
<body>
  <h1>Inserção de aluno</h1>

  <form action="InsereAluno" method="post">
    <p>Nome: <input type="text" name="nome"></p>
    <p>Email: <input type="text" name="email"></p>
    <p>Endereco: <input type="text" name="endereco"></p>
    <p>
      <input type="submit" name="confirma" value="Confirma">
      <input type="button" name="cancela" value="Cancela"
             onclick="history.go(-1)">
    </p>
  </form>
</body>
</html>
```

- acrescente o driver do banco de dados H2 ao projeto e a biblioteca **`prograd-lib.jar`** contendo as classes Aluno, AlunoDAO e ConnectionFactory:
  1) baixe da página *web* do curso, no link "H2 Database Engine," o arquivo zip contendo o H2 e descompacte-o para obter um arquivo JAR.
  2) baixe da página *web* do curso, na aula de hoje, a biblioteca **`prograd-lib.jar`**.
  3) Mova os arquivos JAR para dentro da pasta WebContent/WEB-INF/lib no projeto
  4) No Eclipse, clique no nome do projeto e aperte F5 para atualizar as bibliotecas
- crie um pacote chamado: br.edu.ufabc.prograd.servlet e dentro desse pacote crie a classe `InsereAluno` com o seguinte conteúdo:

```
package br.edu.ufabc.prograd.servlet;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

import br.edu.ufabc.prograd.dao.AlunoDAO;
import br.edu.ufabc.prograd.modelo.Aluno;

@WebServlet("/InsereAluno")
public class InsereAluno extends HttpServlet {
	private static final long serialVersionUID = 1L;

	protected void service(HttpServletRequest request,
				HttpServletResponse response)
	throws ServletException, IOException {
		HttpSession session = request.getSession();
		if (!"administrador".equals(session.getAttribute("usuario"))){
			response.sendError(403); // 403 = Acesso Negado
			return;
		}
		// Obtem parâmetros
		String nome = request.getParameter("nome");
		String email = request.getParameter("email");
		String endereco = request.getParameter("endereco");
		// Insere aluno
		Aluno aluno = new Aluno();
		aluno.setNome(nome);
		aluno.setEmail(email);
		aluno.setEndereco(endereco);
		AlunoDAO dao = new AlunoDAO();
		dao.insere(aluno);
		// Mostra página. TODO: fazer escaping do nome para evitar XSS
		PrintWriter out = response.getWriter();
		out.println("<!DOCTYPE html>");
		out.println("<html>");
		out.println("<head><title>Inserção de aluno</title></head>");
		out.println("<body>");
		out.println("  <h1>Inserção de aluno</h1>");
		out.println("  <p>Aluno " + nome + " inserido!</p>");
		out.println("</body>");
		out.println("</html>");
	}
}
```

- copie o banco de dados **`progweb.h2.db`** para a sua pasta de usuário (Linux: /home/ufabc, Windows: C:\Users\nome_de_usuario)
- procure o JAR do H2 e execute o servidor do banco de dados. No Linux, abra o gerenciador de arquivos, clique com o botão direito sobre **`h2-x.y.z.jar`** e escolha *Abrir com* → *Java Runtime* (se tiver mais de um *Java Runtime* na lista, escolha a maior versão). No Windows, abra o gerenciador de arquivos e dê um duplo clique sobre o arquivo **`h2-x.y.z.jar.`**
- Navegue para http://localhost:8080/aula5/ , faça login como administrador e insira um novo aluno.
- Vá para a interface web do H2 (http://localhost:8082) e conecte-se ao banco de dados:
  - Saved Settings: Generic H2 (Server)
  - JDBC URL: jdbc:h2:tcp://localhost/~/progweb
  - User name: admin
  - Password: admin
- Veja se o novo aluno está na tabela alunos.
- Exporte o seu projeto para poder continuar trabalhando nele em casa.
  1) Clique com o botão direito no projeto e escolha *Export* → *Export...*
  2) Na janela "Export" escolha General → Archive File
  3) Marque apenas o projeto aula5 e, em "To archive file" coloque **`projeto-aula5.zip`**.
  4) Clique "Finish"
  5) O projeto será gravado em um arquivo chamado **`projeto-aula5.zip`**
  6) Para abrir o projeto em outro computador, abra o Eclipse e vá em File → Import → General → Archive file

Exercícios para casa

1) Implemente o servlet `Logout`. Ele deve invalidar a sessão (método `invalidate()` da classe `HttpSession`) e redirecionar o navegador para a página **`index.html`**.
2) Implemente o servlet ListaAlunos. Use o método getLista() da classe AlunoDAO para obter a lista de alunos e use uma tabela HTML (ver última página) para mostrar os alunos. Lembre-se de verificar se o usuário "administrador" iniciou a sessão.
3) Continue a implementar o projeto Agenda, agora fazendo uma interface web com servlets para inserção, alteração, remoção e listagem de contatos.
   - Obs.: se o erro "*No suitable driver found for jdbc:h2:...*" insistir em ocorrer, adicione `Class.forName("org.h2.Driver");` antes de obter a conexão com `DriverManager.getConnection(...)`.

ANEXO

Exemplo de HTML para a lista de alunos:

```
<!DOCTYPE html>

<html>
<head>
  <title>Lista de alunos</title>
</head>
<body>
  <h1>Lista de alunos</h1>

  <table>
    <thead> <!-- tbody = cabeçalho da tabela (table header) -->
      <tr>
        <td>Nome</td><td>Email</td><td>Endereço</td>
      </tr>
    </thead>
    <tbody> <!-- tbody = corpo da tabela (table body) -->
      <tr> <!-- tr = linha de tabela (table row) -->
        <td>Nome do aluno 1</td>  <!-- td = célula (table division) -->
        <td>Email do aluno 1</td>
        <td>Endereço do aluno 1</td>
      </tr>
      <tr>
        <td>Nome do aluno 2</td>
        <td>Email do aluno 2</td>
        <td>Endereço do aluno 2</td>
      </tr>
      ...
      <tr>
        <td>Nome do aluno N</td>
        <td>Email do aluno N</td>
        <td>Endereço do aluno N</td>
      </tr>
    </tbody>
  </table>

</body>
</html>
```